Folke Eisterlehner
Andreas Hotho
Robert Jäschke (Eds.)

ECML PKDD Discovery Challenge 2009 (DC09)

International Workshop at
the European Conference on Machine Learning and
Principles and Practice of Knowledge Discovery in Databases
in Bled, Slovenia, September 7th, 2009

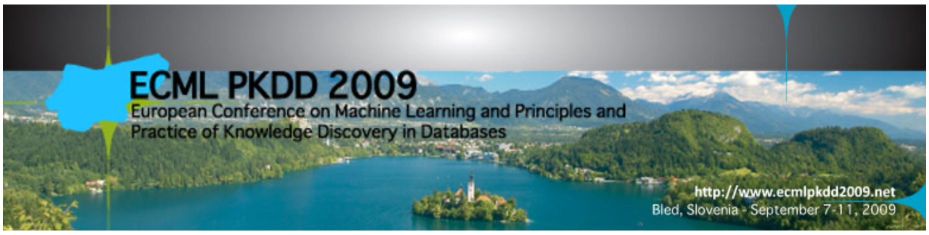

ECML PKDD 2009
European Conference on Machine Learning and Principles and
Practice of Knowledge Discovery in Databases
http://www.ecmlpkdd2009.net
Bled, Slovenia - September 7-11, 2009

Table of Contents

Preface

Since 1999 the ECML PKDD embraces the tradition of organizing a Discovery Challenge, allowing researchers to develop and test algorithms for novel and real world datasets. This year's Discovery Challenge[1] presents a dataset from the field of social bookmarking to deal with the recommendation of tags. The results submitted by the challenge's participants are presented at an ECML PKDD workshop on September 7th, 2009, in Bled, Slovenia.

The provided dataset has been created using data of the social bookmark and publication sharing system BibSonomy,[2] operated by the organizers of the challenge. The training data was released on March 25th 2009, the test data on July 6th. The participants had time until July 8th to submit their results. This gave researchers 14 weeks time to tune their algorithms on a snapshot of a real world folksonomy dataset and 48 hours to compute results on the test data.

To support the user during the tagging process and to facilitate the tagging, BibSonomy includes a tag recommender. When a user finds an interesting web page (or publication) and posts it to BibSonomy, the system offers up to five recommended tags on the posting page. The goal of the challenge is to learn a model which effectively predicts the keywords a user has in mind when describing a web page (or publication). We divided the problem into three tasks, each of which focusing on a certain aspect. All three tasks get the same dataset for training. It is a snapshot of BibSonomy until December 31st 2008. The dataset is cleaned and consists of two parts, the core part and the complete snapshot. The test dataset is different for each task.

Task 1: Content-Based Tag Recommendations. The test data for this task contains posts, whose user, resource or tags are not contained in the post-core at level 2 of the training data. Thus, methods which can't produce tag recommendations for new resources or are unable to suggest new tags very probably won't produce good results here.

Task 2: Graph-Based Recommendations. This task is especially intended for methods relying on the graph structure of the training data only. The user, resource, and tags of each post in the test data are all contained in the training data's post-core at level 2.

Task 3: Online Tag Recommendations. This is a bonus task which will take place after Tasks 1 and 2. The participants shall implement a recommendation service which can be called via HTTP by BibSonomy's recommender infrastructure when a user posts a bookmark or publication. All participating recommenders are called on each posting process, one of them is chosen to actually deliver the results to the user. We can then measure the performance of the recommenders in an online setting, where timeouts are important and where we can measure which tags the user has clicked on.

[1] `http://www.kde.cs.uni-kassel.de/ws/dc09/`

[2] `http://www.bibsonomy.org`

Results. More than 150 participants registered for the mailing list which enabled them to download the dataset. At the end, we received 42 submissions – 21 for each of the Tasks 1 & 2. Additionally, 24 participants submitted a paper that can be found in the proceedings at hand.

We used the F1-Measure common in Information Retrieval to evaluate the submitted recommendations. Therefore, we first computed for each post in the test data precision and recall by comparing the first five recommended tags against the tags the user has originally assigned to this post. Then we averaged precision and recall over all posts in the test data and used the resulting precision and recall to compute the F1-Measure as $\text{f1m} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.

The winning team of Task 1 has an f1m of 0.18740, the second and third follow with 0.18001 and 0.17975. For Task 2, the winner achieved an f1m of 0.35594, followed by 0.33185 and 0.32461. The winner of Task 3 will be announced at the conference and later on the website of the challenge.

Lipczak et al. from Dalhousie University, Halifax, Canada (cf. page 157) are the winners of Task 1. With a method based on the combination of tags from the resource's title, tags assigned to the resource by other users and tags in the user's profile they reached an f1m of 0.18740 in Task 1 and additionally achieved the third place in Task 2 with an f1m of 0.32461. The system is composed of six recommenders and the basic idea is to augment the tags from the title by related tags extracted from two tag-tag–co-occurrence graphs and from the user's profile and then rescore and merge them.

The winners of Task 2, Rendle and Schmidt-Thieme from University of Hildesheim, Germany (cf. page 235) achieved an f1m of 0.35594 with a statistical method based on factor models. Therefore, they factorize the folksonomy structure to find latent interactions between users, resources and tags. Using a variant of the stochastic gradient descent algorithm the authors optimize an adaptation of the Bayesian Personal Ranking criterion. Finally, they estimate how many tags to recommend to further improve precision.

The second of Task 1 (Mrosek et al., page 189) harvests tags from sources like Delicious, Google Scholar, and CiteULike. They also employ the full-text of web pages and PDFs. The third (Ju and Hwang, page 109) merges tags which have been earlier assigned to the resource or used by the user as well as resource descriptions by a weighting scheme. Finally, the second of Task 2 (Balby Marinho et al., page 7) uses relational classification methods in a semi-supervised learning scenario to recommend tags.

We thank all participants of the challenge for their contributions and the organizers of the ECML PKDD 2009 conference for their support. Furthermore, we want to thank our sponsors Nokia[3] and Tagora[4] for supporting the challenge by awarding prizes for the winners of each task. We are looking forward to a very exciting and interesting workshop.

Kassel, August 2009
Folke Eisterlehner, Andreas Hotho, Robert Jäschke

[3] http://www.nokia.com/

[4] http://www.tagora-project.eu/

Relational Classification for Personalized Tag Recommendation

Leandro Balby Marinho, Christine Preisach, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Samelsonplatz 1, University of Hildesheim, D-31141 Hildesheim, Germany
{marinho,preisach,schmidt-thieme}@ismll.uni-hildesheim.de
http://www.ismll.uni-hildesheim.com/

Abstract. Folksonomy data is relational by nature, and therefore methods that directly exploit these relations are prominent for the tag recommendation problem. Relational methods have been successfully applied to areas in which entities are linked in an explicit manner, like hypertext documents and scientific publications. For approaching the graph-based tag recommendation task of the ECML PKDD Discovery Challenge 2009, we propose to turn the folksonomy graph into a homogeneous post graph and use relational classification techniques for predicting tags. Our approach features adherence to multiple kinds of relations, semi-supervised learning and fast predictions.

1 Introduction

One might want tag recommendations for several reasons, as for example: simplifying the tagging process for the user, exposing different facets of a resource and helping the tag vocabulary to converge. Given that users are free to tag, i.e., the same resource can be tagged differently by different people, it is important to personalize the recommended tags for an individual user.

Tagging data forms a ternary relation between users, resources and tags, differently from typical recommender systems in which the relation is usually binary between users and resources. The best methods presented so far explore this ternary relation to compute tag predictions, either by means of tensor factorization [8] or PageRank [3], on the hypergraph induced by the ternary relational data. We, on the other hand, propose to explore the underlying relational graph between posts by means of relational classification.

In this paper we describe our approaches for addressing the graph-based tag recommendation task of the ECML PKDD Discovery Challenge 2009. We present two basic algorithms: *PWA* (probabilistic weighted average)*, an iterative relational classification algorithm enhanced with relaxation labelling, and *WA* (weighted average)*, an iterative relational classification method without relaxation labelling. These methods feature: adherence to multiple kinds of relations, training free, fast predictions, and semi-supervised classification. Semi-supervised classification is particularly appealing because it allows us to evtl. benefit from the information contained in the test dataset. Furthermore, we propose to combine these methods through unweighted voting.

The paper is organized as follows. Section 2 presents the notation used throughout the paper. In Section 3 we show how we turned the folksonomy into a post relational graph. Section 4 introduces the individual classifiers and the ensemble technique we used. In Section 5 we elaborate on the evaluation and experiments conducted for tuning the parameters of our models, and report the results obtained on the test dataset released for the challenge. The paper closes with conclusions and directions for future work.

2 Notation

Foksonomy data usually comprises a set of users U, a set of resources R, a set of tags T, and a set Y of ternary relations between them, i.e., $Y \subseteq U \times R \times T$.

Let

$$X := \{(u, r) \mid \exists t \in T : (u, r, t) \in Y\}$$

be the set of all unique user/resources combinations in the data, where each pair is called a *post*. For convenience, let $T(x = (u, r)) := \{t \in T \mid (u, r, t) \in Y\}$ be the set of all tags assigned to a given post $x \in X$. We consider train/test splits based on posts, i.e., $X_{\text{train}}, X_{\text{test}} \subset X$ disjoint and covering all of X:

$$X_{\text{train}} \dot{\cup} X_{\text{test}} = X$$

For training, the learner has access to the set X_{train} of training posts and their true tags $T|_{X_{\text{train}}}$. The tag recommendation task is then to predict, for a given $x \in X_{\text{test}}$, a set $\hat{T}(x) \subseteq T$ of tags that are most likely to be used by the resp. user for the resp. resource.

3 Relation Engineering

We propose to represent folksonomy data as a homogeneous, undirected relational graph over the post set, i.e., $G := (X, E)$ in which edges are annotated with a weight $w : X \times X \to \mathbb{R}$ denoting the strength of the relation. Besides making the input data more compact – we have only a binary relation $\mathcal{R} \subseteq X \times X$ between objects of the same type – this representation will allow us to trivially cast the problem of personalized tag recommendations as a relational classification problem.

Relational classifiers usually consider, additionally to the typical attribute-value data of objects, relational information. A scientific paper, for example, can be connected to another paper that has been written by the same author or because they share common citations. It has been shown in many classification problems that relational classifiers perform better than purely attribute-based classifiers [1, 4, 6].

In our case, we assume that posts are related to each other if they share the same user: $\mathcal{R}_{\text{user}} := \{(x, x') \in X \times X \mid user(x) = user(x')\}$, the same resource: $\mathcal{R}_{\text{res}} := \{(x, x') \in X \times X | res(x) = res(x')\}$, or either share the same user or resource: $\mathcal{R}_{\text{user}}^{\text{res}} := \mathcal{R}_{\text{user}} \cup \mathcal{R}_{\text{res}}$ (see Figure 1). For convenience, let user(x) and res(x) denote the user and resource of post x respectively. Thus, each post is connected to each other either in terms of other users that tagged the same resource, or the resources tagged by the same user. Weights are discussed in Section 4.

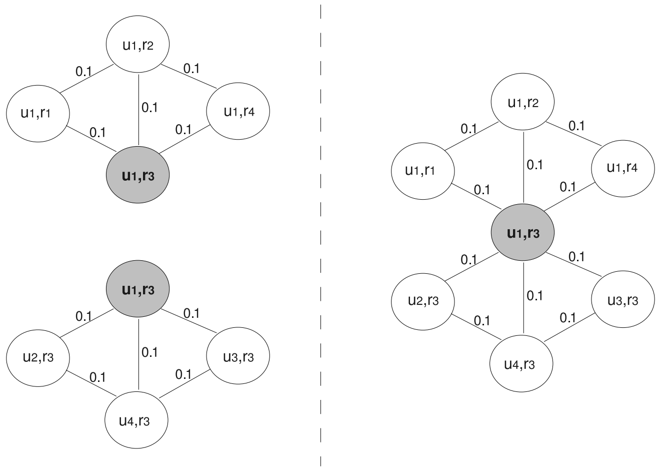


Fig. 1. $\mathcal{R}_{\text{user}}$ (top left), $\mathcal{R}_{\text{res}}$ (bottom left) and $\mathcal{R}_{\text{user}}^{\text{res}}$ (right) of a given test post (nodes in grey)

Note that it may happen that some of the related posts belong themselves to the test dataset, allowing us to evtl. profit from the unlabeled information of test nodes through, e.g., collective inference (see Section 4). Thus, differently from other approaches (e.g., [3, 8]) that are only restricted to X_{train}, we can also exploit the set X_{test} of test posts, but of course not their associated true tags.

Now, for a given $x \in X_{\text{test}}$, one can use the tagging information of related instances to estimate $\hat{T}(x)$. A simple way to do that is, e.g., through tag frequencies of related posts:

$$P(t|x) := \frac{|\{x' \in N_x | t \in T(x')\}|}{|N_x|}, \;\; x \in X, \, t \in T \tag{1}$$

while N_x is the neighborhood of x:

$$N_x := \{x' \in X \,|\, (x, x') \in \mathcal{R}, \, T(x) \neq \emptyset\} \tag{2}$$

In section 4 we will present the actual relational classifiers we have used to approach the challenge.

4 Relational Classification for Tag Recommendation

We extract the relational information by adapting simple statistical relational methods, usually used for classification of hypertext documents, scientific publications or movies, to the tag recommendation scenario. The aim is to recommend tags to users by using the neighborhood encoded in the homogeneous graph $G(X, E)$. Therefore we described a very simple method in eq. (1), where the probability for a tag $t \in T$ given a node x (post) is computed by counting the frequency of neighboring posts $x' \in N_x$ that have used the same tag t. In this case the strength of the relations is not taken into account, i.e., all considered neighbors of x have the same influence on the probability of tag t

given x. But this is not an optimal solution, the more similar posts are to each other the higher the weight of this edge should be.

Hence, a more suitable relational method for tag recommendation is the *WeightedAverage (WA)* which sums up all the weights of posts $x' \in N_x$ that share the same tag $t \in T$ and normalizes this by the sum over all weights in the neighborhood.

$$P(t|x) = \frac{\sum_{x' \in N_x | t \in T(x')} w(x, x')}{\sum_{x' \in N_x} w(x, x')} \tag{3}$$

Thus, *WA* does only consider neighbors that belong to the training set.

A more sophisticated relational method that takes probabilities into account is the *probabilistic weighted average (PWA)*, it calculates the probability of t given x by building the weighted average of the tag probabilities of neighbor nodes $x' \in N_x$:

$$P(t|x) = \frac{\sum_{x' \in N_x} w(x, x') P(t|x')}{\sum_{x' \in N_x} w(x, x')} \tag{4}$$

Where $P(t|x') = 1$ for $x' \in X_{train}$, i.e., we are only exploiting nodes contained in the training set (see eq. (2)). We will see in the next paragraph how one can exploit these probabilities in a more clever way. Both approaches have been introduced in [5] and applied to relational datasets.

Since we want to recommend more than one tag we need to cast the tag recommendation problem as a multilabel classification problem, i.e., assign one or more classes to a test node. We accomplish the multilabel problem by sorting the calculated probabilities $P(t|x)$ for all $x \in X_{\text{test}}$ and recommend the top n tags with highest probabilities.

The proposed relational methods could either be applied on $\mathcal{R}^{\text{res}}_{\text{user}}$, i.e., the union of the user and resource relation or on each relation $\mathcal{R}_{\text{user}}$, $\mathcal{R}_{\text{res}}$ individually. If applied on each relation the results could be combined by using ensemble techniques.

4.1 Semi-Supervised Learning

As mentioned before, we would like additionally, to exploit unlabeled information contained in the graph and use the test nodes that have not been tagged yet, but are related to other nodes. This can be achieved by applying collective inference methods, being iterative procedures, which classify related nodes simultaneously and exploit relational autocorrelation and unlabeled data. Relational autocorrelation is the correlation among a variable of an entity to the same variable (here the class) of a related entity, i.e., connected entities are likely to have the same classes assigned. Collective Classification is semi-supervised by nature, since one exploits the unlabeled part of the data. One of this semi-supervised methods is relaxation labeling [1], it can be formally expressed as:

$$P(t|x)^{(i+1)} = M(P(t|x')^{(i)}_{x' \in N_x}) \tag{5}$$

We first initialize the unlabeled nodes with the prior probability calculated using the train set, then compute the probability of tag t given x iteratively using a relational classification method M based on the neighborhood N_x in the inner loop. The procedure stops when the algorithm converges (i.e., the difference of the tag probability between iteration i and $i+1$ is less than a very small ϵ) or a certain number of iterations is reached.

We used eq. (4) as relational method inside the loop, where we do not require that the neighbors x' are in the training set, but are using the probabilities of unlabeled nodes. For *PWA* this means that in each iteration we use the probabilities of the neighborhood estimated in the previous iteration collectively. *PWA* combined with collective inference is denoted as *PWA** in the following.

For *WeightedAverage* we did not use relaxation labeling but applied a so called *one-shot-estimation* [5, 7]. We did only use the neighbors with known classes, i.e., in the first iteration we exploit only nodes from the training set, while in the next iteration we used also test nodes that have been classified in the previous iterations. The procedure stops when all test nodes could be classified or a specific number of iterations is reached. Hence, the tag probabilities are not being re-estimated like for the relaxation labeling but only estimated once. Thus, *WA* combined with the one-shot-estimation procedure is denoted as *WA**.

4.2 Ensemble

Ensemble classification may lead to significant improvement on classification accuracy, since uncorrelated errors made by the individual classifiers are removed by the combination of different classifiers [2, 6]. Furthermore, ensemble classification reduces variance and bias.

We have decided to combine *WA** and *PWA** through a simple unweighted voting, since voting performs particularly well when the results of individual classifiers are similar; as we will see in Section 5, *WA** and *PWA** yielded very similar results in our holdout set.

After performing the individual classifiers, we receive probability distributions for each classifier K_l as output and build the arithmetic mean of the tag-assignment probabilities for each test post and tag:

$$P(t|x) = \frac{1}{L} \cdot \sum_{l=1}^{L} P_l(t|x), \quad L := |K_l|P_l(t|x) \neq 0,\ t \in T| \tag{6}$$

4.3 Weighting Schemes

The weight w in eq. (3) and (4) is an important factor in the estimation of tag probabilities, since it describes the strength of the relation between x and x'. There are several ways to estimate these weights:

1. For two nodes $(x, x') \in \mathcal{R}_{\text{res}}$, compute their similarity by representing x and x' as user-tag profile vectors. Each component of the profile vector corresponds to the count of co-occurrences between users and tags:

$$\phi^{\text{user-tag}} := (|Y \cap (\{\text{user}(x)\} \times R \times \{t\})|)_{t \in T}$$

2. Similarly to 1, for two nodes $(x, x') \in \mathcal{R}_{\text{user}}$, the node similarity is computed by representing x and x' as resource-tag profile vectors:

$$\phi^{\text{res-tag}} := (|Y \cap (U \times \{\text{res}(x)\} \times \{t\})|)_{t \in T}$$

3. Similar to 2, but x and x' are represented as resource-user profile vectors where each component corresponds to the count of co-occurrences between resources and users:

$$\phi^{\text{res-user}} := (|Y \cap (\{u\} \times \{\text{res}(x)\} \times T)|)_{u \in U}$$

4. The same as in 1, but the node similarity is computed w.r.t. to user-resource profile vectors:

$$\phi^{\text{user-res}} := (|Y \cap (\{\text{user}(x)\} \times \{r\} \times T)|)_{r \in R}$$

The edge weight is finally computed by applying the cosine similarity over the desired profile vectors:

$$\text{sim}(\phi(x), \phi(x')) := \frac{\langle \phi(x), \phi(x') \rangle}{\|\phi(x)\| \|\phi(x')\|} \tag{7}$$

In our experiments we basically used the scheme 1, since there is no new user in the data and therefore we can always build user-tag profile vectors.

5 Evaluation

All the results presented in this section are reported in terms of F1-score, the official measure used by the graph-based tag recommendation task of the ECML PKDD Discovery Challenge 2009. For a given $x \in X_{\text{test}}$ the F1-Score is computed as follows:

$$\text{F1-score}\left(\hat{T}(x)\right) = \frac{2 \cdot \text{Recall}\left(\hat{T}(x)\right) \cdot \text{Precision}\left(\hat{T}(x)\right)}{\text{Recall}\left(\hat{T}(x)\right) + \text{Precision}\left(\hat{T}(x)\right)} \tag{8}$$

Although the methods presented in Section 4 usually do not have free parameters, we realized that $\mathcal{R}_{\text{user}}$ and $\mathcal{R}_{\text{res}}$ can have a different impact in the recommendation quality (cf. Figures 2 and 3), and thereby we introduced a parameter to reward the best relations in $\mathcal{R}^{\text{res}}_{\text{user}}$ by a factor $c \in \mathbb{N}$: if $\mathcal{R}_{\text{res}}$ yields better recommendations than $\mathcal{R}_{\text{user}}$ for example, all edge weights in $\mathcal{R}^{\text{res}}_{\text{user}}$ that refer to $\mathcal{R}_{\text{res}}$ are multiplied by c.

For searching the best c value we performed a greedy search over the factor range $\{1, ..., 4\}$ on a holdout set created by randomly selecting 800 posts from the training data. Tables 1 and 2 show the characteristics of the training and test/holdout datasets respectively. Figure 2 presents the results of *WA*-Full*[1], i.e., *WA** performed over $\mathcal{R}^{\text{res}}_{\text{user}}$, for different c values on the holdout set according to the F1-score. We also plot the results of *WA*-Res* and *WA*-Usr* (i.e., *WA** on $\mathcal{R}_{\text{res}}$ and $\mathcal{R}_{\text{user}}$ resp.).

After finding the best c value on the holdout set, we applied *WA*-Full*, *PWA*-Full*, and the ensemble (c.f. eq. 6) to the challenge test dataset (see Figure 3). Note that the

[1] Since the results of *PWA** and *WA** are very similar, we just report on *WA**.

dataset	$\|U\|$	$\|R\|$	$\|T\|$	$\|Y\|$	$\|X\|$
BibSonomy	1,185	22,389	13,276	253,615	64,406

Table 1. Characteristics of 2-core BibSonomy.

dataset	$\|U\|$	$\|R\|$	$\|X_{\text{test}}\|$
Holdout	292	788	800
Challenge test	136	667	778

Table 2. Characteristics of the holdout set and the challenge test dataset.

results on the challenge test dataset are much lower than those on the holdout set. It may indicate that either our holdout set was not a good representative of the population or that the challenge test dataset represents a concept drift. We plan to further investigate the reasons underlying this large deviation.

According to the rules of the challenge, the F1-score is measured over the Top-5 recommended tags, even though one is not forced to always recommend 5 tags. This is an important remark because if one recommends more tags than the true number of tags attached to a particular test post, one can lower precision. Therefore, we estimate the number of tags to be recommended to each test post by taking the average number of tags used by each test user to his resources. If a given test user has tagged his resources with 3 tags in average, for example, we recommend the Top-3 tags delivered by our algorithms for all test posts in which this user appears.

6 Conclusions

In this paper we proposed to approach the graph-based tag recommendation task of the ECML PKDD Discovery Challenge 2009 by means of relational classification. We first turned the usual tripartite graph of social tagging systems into a homogeneous post graph, whereby simple statistical relational methods can be easily applied. Our methods are training free and the prediction runtime only depends on the number of neighbors and tags, which is fast since the training data is sparse. The models we presented also incorporate a semi-supervised component that can evtl. benefit from test data. We presented two relational classification methods, namely *WA** and *PWA**, and one ensemble based on unweighted voting over the tag probabilities delivered by these methods.

We also introduced a parameter in order to reward more informative relations, which was learned through a greedy search in a holdout set.

In future work we want to investigate new kinds of relations between the posts (e.g. content-based), other ensemble techniques, and new methods for automatically learning more informative weights.

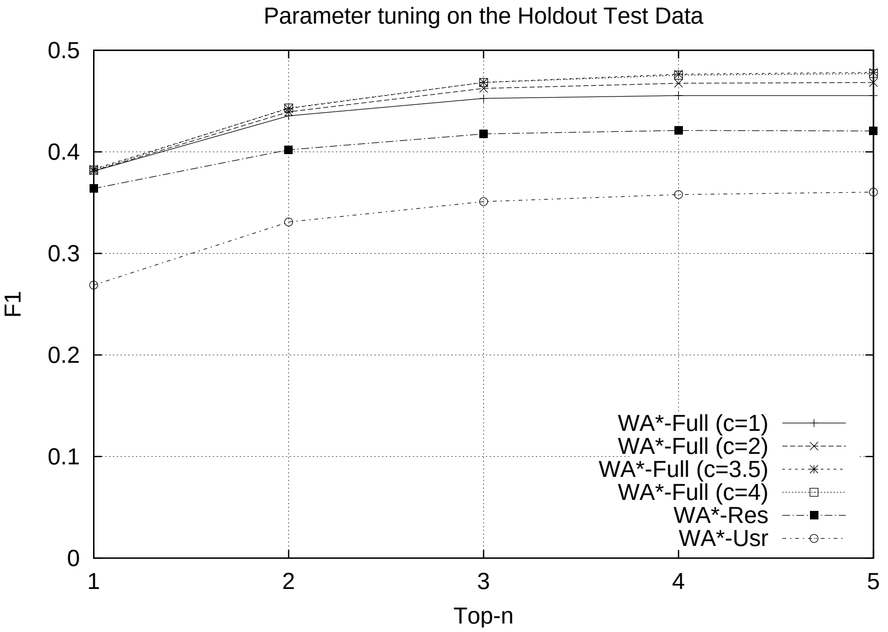


Fig. 2. Parameter search of *WA*-Full* in a holdout set. Best c value found equals 3.5

7 Acknowledgements

This work is supported by CNPq, an institution of Brazilian Government for scientific and technologic development, and the X-Media project (www.x-media-project.org) sponsored by the European Commission as part of the Information Society Technologies (IST) programme under EC grant number IST-FP6-026978. The authors also gratefully acknowledge the partial co-funding of their work through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006. For your inquiries please contact info@mymediaproject.org.

References

1. Soumen Chakrabarti, Byron E. Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of SIGMOD-98*, pages 307–318. ACM Press, New York, US, 1998.
2. Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
3. Robert Jaeschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in social bookmarking systems. *AI Communications*, pages 231–247, 2008.
4. Qing Lu and Lise Getoor. Link-based classification using labeled and unlabeled data. icml 2003 workshop on the continuum from labeled to unlabeled data. In *in Machine Learning and Data Mining*, 2003.

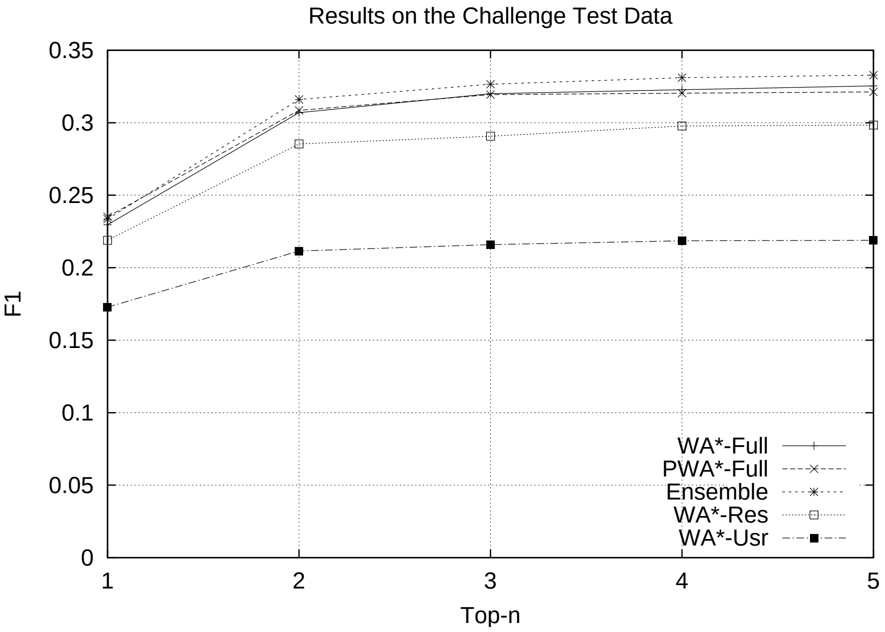


Fig. 3. Results in the challenge test datatest.

5. S. Macskassy and F. Provost. A simple relational classifier. In *Proceedings of the 2nd Workshop on Multi-Relational Data Mining, KDD2003*, pages 64–76, 2003.
6. Christine Preisach and Lars Schmidt-Thieme. Relational ensemble classification. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 499–509, Washington, DC, USA, 2006. IEEE Computer Society.
7. Christine Preisach and Lars Schmidt-Thieme. Ensembles of relational classifiers. *Knowledge and Information Systems*, 14:249–272, 2007.
8. Steffen Rendle, Leandro B. Marinho, Alexandros Nanopoulos, and Lars Schimdt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2009.

Measuring Vertex Centrality in Co-occurrence Graphs for Online Social Tag Recommendation

Iván Cantador, David Vallet, Joemon M. Jose

Department of Computing Science
University of Glasgow
Lilybank Gardens, Glasgow, G12 8QQ, Scotland, UK
{cantador, dvallet, jj}@dcs.gla.ac.uk

Abstract. We present a social tag recommendation model for collaborative bookmarking systems. This model receives as input a bookmark of a web page or scientific publication, and automatically suggests a set of social tags useful for annotating the bookmarked document. Analysing and processing the bookmark textual contents - document title, URL, abstract and descriptions - we extract a set of keywords, forming a query that is launched against an index, and retrieves a number of similar tagged bookmarks. Afterwards, we take the social tags of these bookmarks, and build their global co-occurrence sub-graph. The tags (vertices) of this reduced graph that have the highest vertex centrality constitute our recommendations, which are finally ranked based on TF-IDF and personalisation based techniques.

Keywords: social tag recommendation, co-occurrence, graph vertex centrality, collaborative bookmarking.

1 Introduction

Social tagging systems allow users to create or upload resources (web pages[1], scientific publications[2], photos[3], video clips[4], music tracks[5]), annotate them with freely chosen words – so called *tags* – and share them with others. The set of users, resources, tags and annotations (i.e., triplets user-tag-resource) is commonly known as *folksonomy*, and constitutes a collective unstructured knowledge classification. This implicit classification is then used by users to organise, explore and search for resources, and by systems to recommend users interesting resources.

These systems usually include tag recommendation mechanisms to ease the finding of relevant tags for a resource, and consolidate the tag vocabulary across users. However, as stated in [7], no algorithmic details have been published, and it is assumed that, in general, tag recommendations in current applications are based on suggesting those tags that most frequently were assigned to the resource, or to similar resources.

[1] Delicious – Social bookmarking, http://delicious.com/

[2] CiteULike – Scholarly reference management and discovery, http://www.citeulike.com/

[3] Flickr – Photo sharing, http://www.flickr.com/

[4] YouTube – Video sharing, http://www.youtube.com/

[5] Last.fm – Personal online radio, http://www.last.fm/

Recent works have proposed more sophisticated and accurate methods for tag recommendation. These methods can be roughly classified into content-based and collaborative approaches. Content-based techniques [3, 4, 10, 16] analyse the contents and/or meta-information of the resources to extract keywords, which are directly suggested to the user or matched with existing tags. Collaborative strategies [6, 7, 17], on the other hand, exploit folksonomy relations between users, resources and tags to infer which of the tags of the system folksonomy are most suitable for a particular resource. Hybrid techniques, combining content and collaborative features, have been also investigated [5, 15].

In this paper, we present a hybrid tag recommendation model for an online bookmarking system where users annotate online web pages and scientific publications. The model receives as input a bookmark, analyses and processes its textual contents – document title, URL, abstract and description – extracting a set of keywords, and forms a query that is launched against an index to retrieve a number of similar tagged bookmarks. Afterwards, its takes the social tags of these bookmarks, and builds their global co-occurrence sub-graph. The tags (vertices) of this reduced graph that have the highest vertex centrality constitute the recommendations, which are finally ranked based on TF-IDF [14] and personalisation based techniques.

Participating at the ECML PKDD 2009 Discovery Challenge[6], we have tested our approach with a dataset from BibSonomy system[7], obtaining precision values of 42% and 25% when, respectively, one and five tags are recommended per bookmark. As we explain herein, the benefits of our approach are its low computational cost, and its capability of suggesting diverse tags in comparison to selecting the most popular tags matched with each bookmark.

The rest of the paper is organised as follows. Section 2 summarises state-of-the-art tag recommendation techniques. Section 3 describes the document and index models used by our tag recommender. Section 4 explains the stages of the recommendation process. Section 5 describes the experiments conducted to evaluate the proposal. Finally, Section 6 provides some conclusions and future work.

2 Related work

Analogously to recommender systems [1], tag recommendation techniques can be roughly classified into two categories: content-based and collaborative techniques. Whereas content-based approaches focus on the suggestion of keywords extracted from resource contents and meta-data, collaborative approaches exploit the relations between users, resources and tags of the folksonomy graph to select the set of recommended tags. Continuing with the previous analogy, tag recommendation techniques that combine content-based and collaborative models can be called hybrid approaches, and techniques that make tag recommendations biased by the user's (tag-based) profile can be called personalised models.

Based on the previous classification, in this section, we describe state-of-the-art tag recommendation techniques that have been proposed for social bookmarking systems.

[6] ECML PKDD 2009 Discovery Challenge, http://www.kde.cs.uni-kassel.de/ws/dc09/

[7] BibSonomy – Social bookmark and publication sharing, http://www.bibsonomy.org/

2.1 Content-based tag recommenders

Mishne [10] presents a simple content-based tag recommender. Once a user supplies a new bookmark, bookmarks that are similar to it are identified. The tags assigned to these bookmarks are aggregated, creating a ranked list of likely tags. Then, the system filters and re-ranks the tag list. The top ranked tags are finally suggested to the user. To find similar bookmarks, the author utilises a document index, and keywords of the input bookmark to form a query that is launched against the index. The tags are scored according to their frequencies in the top results of the above query, and those tags that have been used previously by the user are boosted by a constant factor. Our approach follows the same stages, also using an index to retrieve similar bookmarks. It includes, however, more sophisticated methods of tag ranking based on tag popularity and personalisation aspects.

Byde et al. [3] present a personalised tag recommendation method on the basis of similarity metrics between a new document and documents previously tagged by the user. These metrics are derived either from tagging data, or from content analysis, and are based on the cosine similarity metric [14]. Similar metrics are used by our approach in some of its stages.

Chirita et al. [4] suggest a method called P-TAG that automatically generates personalised tags for web pages. Given a particular web page, P-TAG produces keywords relevant both to the page contents and data residing on the user's desktop, thus expressing a personalised viewpoint. A number of techniques to extract keywords from textual contents, and several metrics to compare web pages and desktop documents, are investigated. Our approach applies natural language processing techniques to extract keywords from bookmark attributes, but it can be enriched with techniques like [4] to also analyse and exploit the textual contents of the bookmarked documents.

Tatu et al. [16] propose to extract important concepts from the textual metadata associated to bookmarks, and use semantic analysis to generate normalised versions of the concepts. For instance, `European Union`, `EU` and `European Community` would be normalised to the concept `european_union`. Then, users and resources are represented in terms of the created conceptual space, and personalised tag recommendations are based on intersections between such representations. In our approach, synonym relations and lexical derivations between tags are implicitly taking into consideration through the exploitation of tag co-occurrence graphs.

2.2 Collaborative tag recommenders

Xu et al. [17] propose a collaborative tag recommender that favours tags used by a large number of users on the target resource (high authority in the HITS algorithm [8]), and minimises the overlap of concepts among the recommended tags to allow for high coverage of multiple facets. Our approach also attempts to take into account tag popularity and diversity in the recommendations through the consideration of vertex centralities in the tag co-occurrence graph.

Hotho et al. [6] present a graph-based tag recommendation approach called FolkRank, which is an adaptation of PageRank algorithm [12], and is applied in the folksonomy user-resource-tag graph. Its basis is the idea that a resource tagged with important tags by important users becomes important itself. The same holds, symmetrically, for users and tags. Having a graph whose vertices are associated to users, resources and tags, the algorithm reinforces each of them by spreading their weights through the graph edges. In this work, we restrict our study to the original folksonomy graph. As a future research goal, PageRank, HITS or other graph based techniques could be applied to enhance the identification of tags with high graph centrality values.

Jäscke et al. [7] evaluate and compare several tag recommendation algorithms: an adaptation of user-based collaborative filtering [13], FolkRank strategy [6], and methods that are based on counting tag co-occurrences. The authors show that graph-based and collaborative filtering approaches provide better results that non-personalised methods, and state that methods based on counting co-occurrences have low computational costs, thus being preferable for real time scenarios. Our approach is computationally cheap because it is based on a simple analysis of tag co-occurrence graphs, and includes a personalisation stage to better adjust the tag recommendations to the user's profile.

2.3 Hybrid tag recommenders

Heymann et al. [5] present a technique that predicts tags for a website based on page text, anchor text, surrounding hosts, and other tags assigned to the website by users. The tag predictions are based on association rules, which, as stated by the authors, may serve as a way to link disparate vocabularies among users, and may indicate synonym and polysemy cases. As a hybrid approach, our tag recommender makes use of content-based and collaborative tag information. Nonetheless, we simplify the process limiting it to the exploitation of meta-information of the contents available in the bookmarks.

Song et al. [15] suggest a tag recommendation method that combines clustering and mixture models. Tagged documents are represented as a triplet (words, documents, tags) by two bipartite graphs. These graphs are clustered into topics by a spectral recursive embedding technique [18]. The sparsity of the obtained clusters is dealt with a two-way Poisson mixture model [9], which groups documents into components and clusters words. Inference for new documents is based on the posterior probability of topic distributions, and tags recommendations are given according to the within-cluster tag rankings.

3 Document and index models

To suggest tags for an input bookmark, our recommender exploits meta-information associated to it. The text contents of bookmarked documents (web pages or scientific publications) could be also taken into account, but we decided to firstly study how accurate tag recommendations can be by only using bookmarking meta-information.

In this work, we test our approach with a dataset obtained from BibSonomy system, whose bookmarks have, among others, the attributes shown in Table 1.

Table 1. Meta-information available in BibSonomy system about two different bookmarks: a web page and a scientific publication.

URL	**http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html**
Description	Folksonomies - Cooperative Classification and Communication Through Shared Metadata
Extended	General overview of tagging and folksonomies. Difference between controlled vocabularies, author and user tagging. Advantages and shortcomings of folksonomies

Title	**Semantic Modelling of User Interests Based on Cross-Folksonomy Analysis**
Author	M. Szomszor and H. Alani and I. Cantador and K. O'hara and N. Shadbolt
Booktitle	Proceedings of the 7th International Semantic Web Conference (ISWC 2008)
Journal	The Semantic Web - ISWC 2008
Pages	632-648
URL	http://dx.doi.org/10.1007/978-3-540-88564-1_40
Year	2008
Month	October
Location	Karlsruhe, Germany
Abstract	The continued increase in Web usage, in particular participation in folksonomies, reveals a trend towards a more dynamic and interactive Web where individuals can organise and share resources. Tagging has emerged as the de-facto standard for the organisation of such resources, providing a versatile and reactive knowledge management mechanism that users find easy to use and understand. It is common nowadays for users to have multiple profiles in various folksonomies, thus distributing their tagging activities. In this paper, we present a method for the automatic consolidation of user profiles across two popular social networking sites, and subsequent semantic modelling of their interests utilising Wikipedia as a multi-domain model. We evaluate how much can be learned from such sites, and in which domains the knowledge acquired is focussed. Results show that far richer interest profiles can be generated for users when multiple tag-clouds are combined.

In our approach, for each bookmark, using a set of NLP tools [2], the text attributes title, URL, abstract and description, and extended description are processed and transformed into a weighted list of keywords. These simplified bookmark representations are then stored into an index, which will allow fast searches for bookmarks that satisfy keyword- and tag-based queries. In our implementation, we used Lucene[8], which allowed us to apply keyword stemming, stop words removal, and term TF-IDF weighting.

[8] Apache Lucene – Open-source Information Retrieval library, http://lucene.apache.org/

4 Social tag recommendation

In this section, we describe our approach to recommend social tags for a bookmark, which does not need to be already tagged. The recommendation process is divided in 5 stages, depicted in Figure 1. Each of these stages is explained in detail in the next subsections. For a better understanding, the explanations follow a common illustrative example.

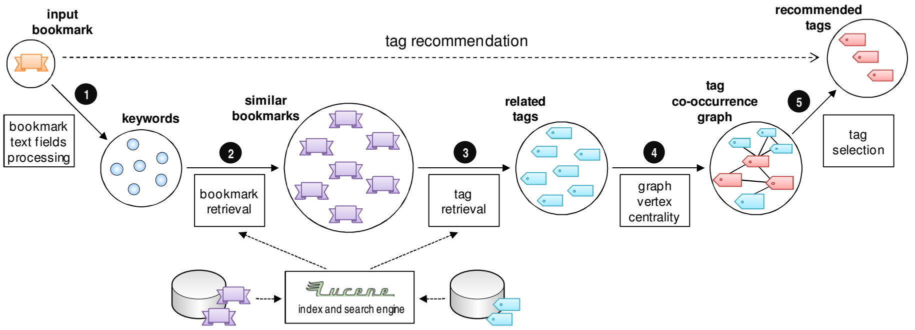


Figure 1. Tag recommendation process.

4.1 Extracting bookmark keywords

The first stage of our tag recommendation approach (identified by label 1 in Figure 1) is the extraction of keywords from some of the textual contents of the input bookmark.

According to the document model explained in Section 2, we extract such keywords from the title, URL, abstract, description and extended description of the bookmark. We made experiments processing other attributes such as authors, user comments, and book and journal titles, but we obtained worse recommendation results. The noise (in the case of personal comments) and generality (in the case of authors and book/journal titles) implied the suggestion of social tags not related to the content topics of the web page or scientific publication associated to the bookmark.

For plain text fields of the bookmark, such as title, abstract and descriptions, we filter out numeric characters and discard stop words from English, Spanish, French, German and Italian, which were identified as the predominant languages of the bookmarks available in our experimental datasets. We also carry out transformations to LATEX expressions. Finally, we remove punctuation symbols, parentheses, and exclamation and question marks, and discard special terms like `paper`, `work`, `section`, `chapter`, among others. For the URL field, we firstly remove the network protocol (`HTTP`, `FTP`, etc.), the web domain (`com`, `org`, `edu`, etc.), the file extension (`html`, `pdf`, `doc`, etc.), and possible GET arguments for CGI scripts. Next, we tokenise the remaining text removing the dots (.) and slashes (/). Finally, we discard numeric words and several special words like `index`, `main`, `default`, `home`, among others. In both cases, a natural language processing tool [2] is used to singularise the resultant keywords, and filter out those that were not nouns.

Table 2 shows the content of an example bookmark whose tag recommendations are going to be explained in the rest of this section. It also lists the keywords extracted from the bookmark in the first stage of our approach. The bookmarked document is a scientific publication. Its main research fields are *recommender systems* and *semantic web technologies*. It describes a content-based collaborative recommendation model that exploits semantic (ontology-based) descriptions of user and item profiles.

Table 2. Example of bookmark for which the tag recommendation is performed, and the set of keywords extracted from it.

Title	**A Multilayer Ontology-based Hybrid Recommendation Model**
Authors	Iván Cantador, Alejandro Bellogín, Pablo Castells
URL	http://www.configworks.com/AICOM/
Journal title	AI Communications
Abstract	We propose a novel hybrid recommendation model in which user preferences and item features are described in terms of semantic concepts defined in domain ontologies. The concept, item and user spaces are clustered in a coordinated way, and the resulting clusters are used to find similarities among individuals at multiple semantic layers. Such layers correspond to implicit Communities of Interest, and enable enhanced recommendation.
Extracted keywords	multilayer, ontology, hybrid, recommendation, configwork, aicom, ai, communication, user, preference, semantic, concept, domain, ontology, item, space, way, cluster, similarity, individual, layer, community, interest

In this stage, we performed a simple mechanism to obtain a keyword-based description of the bookmarked document (web page or scientific publication) contents. Note that more complex approaches can be performed. For example, instead of only being limited to the bookmark attributes, we could also extract additional keywords from the bookmarked document itself. Moreover, external knowledge bases could be exploited to infer new keywords related to the ones extracted from the bookmark. These are issues to be investigated in future work.

4.2 Searching for similar bookmarks

The second stage (label 2 in Figure 1) consists of searching for bookmarks that contain some of the keywords obtained in the previous stage.

The list of keywords extracted from the input bookmark are weighted based on their appearance frequency in the bookmark attributes, and are included in a weighted keyword-based query. This query represents an initial description of the input bookmark.

More specifically, in the query q_n for bookmark b_n, the weight $q_{n,k} \in [0,1]$ assigned to each keyword k is computed as the number of times the keyword appears in the bookmark attributes divided by the total number of keywords extracted from the bookmark:

$$\mathbf{q_n} = q(b_n) = \{q_{n,1}, \dots, q_{n,k}, \dots, q_{n,K}\}$$

where

$$q_{n,k} = \frac{f_{n,k}}{\sum_{i=1}^{i=K} f_{n,i}},$$

being $f_{n,k}$ the number of times keyword k appears in bookmark b_n fields.

The query is then launched against the index described in Section 2. Thus, we are not only taking into account the relevance of the keywords for the input bookmark, but also ranking the list of retrieved similar bookmarks. The searching result is a set of bookmarks that are similar to the input bookmark, assuming that "similar" bookmarks have common keywords. Using the cosine similarity measure for the vector space model [14], the retrieved bookmarks are assigned scores $w_{n,i} \in [0,1]$ that measure the similarity between the query q_n (i.e., the input bookmark b_n) and the retrieved bookmarks b_i:

$$w_{n,i} = sim(q_n, b_i) = \cos(\mathbf{q_n}, \mathbf{b_i}) = \frac{\mathbf{q_n} \cdot \mathbf{b_i}}{||\mathbf{q_n}||\,||\mathbf{b_i}||}$$

For the example input bookmark, Table 3 shows the keywords, query, and some similar bookmarks obtained in the second stage of our tag recommendation model.

Table 3. Extracted keywords, generated query, and retrieved similar bookmarks for the example input bookmark.

Input bookmark: **A Multilayer Ontology-based Hybrid Recommendation Model**

Keywords	multilayer, ontology, hybrid, recommendation, configwork, aicom, ai, communication, user, preference, semantic, concept, domain, ontology, item, space, way, cluster, similarity, individual, layer, community, interest
Query	recommendation^0.125, ontology^0.09375, concept^0.0625, hybrid^0.0625, item^0.0625, layer^0.0625, multilayer^0.0625, semantic^0.0625, user^0.0625, aicom^0.03125, cluster^0.03125, configwork^0.03125, individual^0.03125, interest^0.03125, communication^0.03125, community^0.03125, preference^0.03125, similarity^0.03125, space^0.03125, way^0.03125
Similar bookmarks	• Improving Recommendation Lists Through Topic Diversification • Item-Based Collaborative Filtering Recommendation Algorithms • Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments • Automatic Tag Recommendation for the Web 2.0 Blogosphere using Collaborative Tagging and Hybrid ANN semantic structures • PIMO - a Framework for Representing Personal Information Models

In this stage, we attempted to define and contextualise the vocabulary that is likely to describe the contents of the bookmarked document. For that purpose, the initial set of keywords extracted from the input bookmark was used to find related bookmarks, assuming that the keywords and social tags of the latter are useful to describe the content topics of the former.

4.3 Obtaining related social tags

Once the set of similar bookmarks has been retrieved, in the third stage (label 3 in Figure 1), we collect and weight all their social tags.

The weight assigned to each tag represents how much it contributes to the definition of the vocabulary that describes the input bookmark. Based on the scores $w_{n,i}$ of the bookmarks retrieved in the previous stage, the weight v_n of a tag t for the input bookmark b_n is given by:

$$v_n(t) = \sum_{i:t \in \mathrm{tags}(b_i)} w_{n,i}.$$

At this point, we could finish the recommendation process suggesting those social tags with highest weights v_n. However, doing this, we are not taking into account tag popularities and tag correlations, very important features of any collaborative tagging system. In fact, we conducted experiments evaluating recommendations based on the highest weighted tags, and we obtained worse results that the ones provided by the whole approach presented herein.

Table 4 shows a subset of the tags retrieved from the bookmarks that were retrieved in Stage 2 for the example input bookmark. The weights v_n for each tag are also given in the table.

Table 4. Weighted subset of tags retrieved from the list of bookmarks that are similar to the example input bookmark.

Input bookmark: **A Multilayer Ontology-based Hybrid Recommendation Model**

Related tag	*Weight*	*Related tag*	*Weight*	*Related tag*	*Weight*
recommender	10.538	clustering	2.013	dataset	0.871
recommendation	6.562	recommendersystems	1.669	evaluation	0.786
collaborative	5.142	web	1.669	suggestion	0.786
filtering	5.142	information	1.539	semantics	0.786
collaborativefiltering	3.585	ir	1.378	tag	0.786
ecommerce	3.138	retrieval	1.378	tagging	0.786
personalization	3.138	contentbasedfiltering	1.006	knowledgemanagement	0.290
cf	2.757	ontologies	1.006	network	0.290
semantic	2.745	ontology	1.006	neural	0.290
semanticweb	2.259	userprofileservices	1.006	neuralnetwork	0.290

In this stage, we collected the social tags that are potentially relevant for describing the input bookmarked document based on a set of related bookmarks. We assigned a weight to each tag capturing the strength of its contribution to the bookmark description. However, we realised that this measure is not enough for tag recommendation purposes, and global metrics regarding the folksonomy graph, such as tag popularities and tag correlations, have to be taken into consideration.

4.4 Building the global social tag co-occurrence sub-graph

In the fourth stage (label 4 in Figure 1), we interconnect the social tags obtained in the previous stage through the co-occurrence values of each pair of tags.

The co-occurrence of two tags t_i and t_j is usually defined in terms of the number of resources (bookmarks) that have been tagged with both t_i and t_j. In this work, we make use of the asymmetric co-occurrence metric:

$$co(t_i, t_j) = \frac{\#\{n: t_i \epsilon \text{ tags}(b_n) \wedge t_j \epsilon \text{ tags}(b_n)\}}{\#\{n: t_i \epsilon \text{ tags}(b_n)\}},$$

which assigns different values for $co(t_i, t_j)$ and $co(t_j, t_i)$ dividing the number of resources tagged with the two tags by the number of resources tagged with one of them.

Computing the co-occurrence values for each pair of tags existing in a training dataset, we build a global graph where the vertices correspond to the available tags, and the edges link tags that co-occur within at least one resource. This graph is directed and weighted: each pair of co-occurring tags is linked by two edges whose weights are the asymmetric co-occurrence values of the tags.

We propose to exploit this global graph to interconnect the tags obtained in the previous stage, and extract the ones that are more related with the input bookmark. Specifically, we create a sub-graph where the vertices are the above tags, and the edges are the same as these tags have in the global co-occurrence graph. From this sub-graph, we remove those edges whose co-occurrence values $co(t_i, t_j)$ are lower than the average co-occurrence value of the sub-graph vertices:

$$avg\_co(b_n) = \frac{\sum_{i,j} co(t_i, t_j)}{\#\{(i,j): co(t_i, t_j) > 0\}},$$

where t_i and t_j are the pairs of social tags related to the input bookmark b_n. Removing these edges, we aim to isolate (and later discard) “noise” tags that less frequently appear in bookmark annotations.

We hypothesise that vertices of the generated sub-graph that are most “strongly” connected with the rest of the vertices correspond to tags that should be recommended, assuming that high graph vertex centralities are associated to the most informative or representative vertices. In this context, it is important to note that related tags with high weights v_n do not necessarily have to be the ones with highest vertex centralities in the co-occurrence sub-graph. We hypothesise that a combination of both measures – local weights representing the bookmark content topics and global co-occurrences taking into account collaborative popularities – is an appropriate strategy for tag recommendation.

Figure 2 shows the resultant co-occurrence graph associated to the tags retrieved from the example input bookmark. The tags with highest vertex in-degree seem to be good candidates to describe the contents of the bookmarked document.

Input bookmark: **A Multilayer Ontology-based Hybrid Recommendation Model**

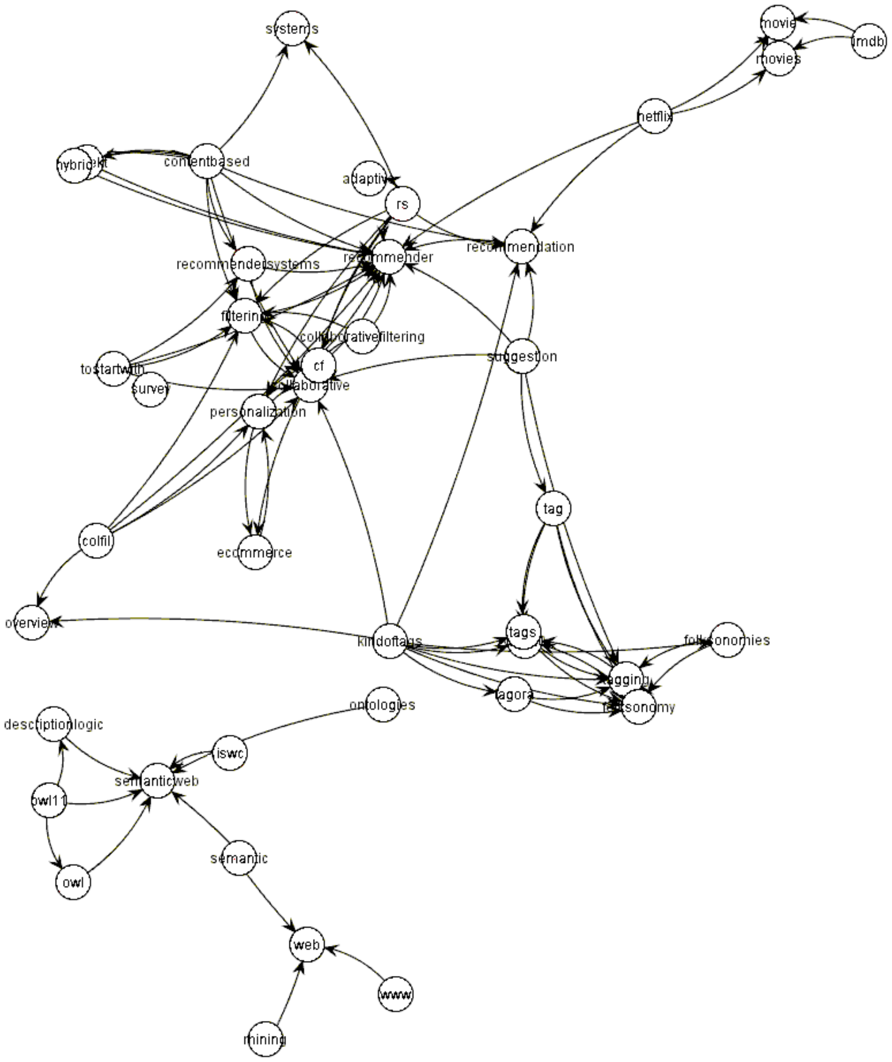


Figure 2. Filtered tag co-occurrence graph associated to the example input bookmark. Edge weights and non-connected vertices are not shown. Two main clusters can be identified in the graph, which correspond to two research areas related to the bookmarked document: recommender systems, and semantic web technologies.

The goal of this stage was to establish global relations between the social tags that are potentially useful for describing the input bookmark. Exploiting these relations, we aimed to take into account tag popularity and tag co-occurrence aspects, and expected to identify which are the most informative tags to be recommended.

4.5 Recommending social tags

In the fifth stage (label 5 in Figure 1), we select and recommend a subset of the related tags from previous stages. The selection criterion we propose is based on three aspects: the tag frequency in bookmarks similar to the input bookmark (stage 3), the tag co-occurrence graph centrality (stage 4), and a personalisation strategy that prioritises those tags that are related to the input bookmark and belong to the set of tags already used by the user to whom the recommendations are directed.

For each tag t, the first two aspects are combined as follows:

$$c_n(t) = in\_degree_n(t) \cdot (v_n(t))^2$$

where $in\_degree_n(t)$ is the number of edges that have as destination the vertex of tag t in the co-occurrence sub-graph built in stage 4 for the input bookmark b_n.

In order to penalise too generic tags we conduct a TF-IDF based reformulation of the centralities $c_n(t)$:

$$r_n(t) = c_n(t) \cdot log\left(\frac{N}{\#\{i: t \in \text{tags}(b_i)\}}\right)$$

where N is the total number of bookmarks in the repository.

Finally, to take into account information about the user's tagging activity, we increase the $r_n(t)$ values of those tags that have already been used by the user:

$$p_{n,u}(t) = r_n(t) \cdot (1 + p_u(t))$$

where $p_u(t)$ is the normalised preference of user u for tag t:

$$p_u(t) = \begin{cases} \dfrac{f_{u,t}}{\max\limits_{i \in \text{tags}(u)} f_{u,i}} & \text{if } t \in tags(u) \\ 0 & \text{otherwise} \end{cases},$$

$f_{u,i}$ being the number of times tag t has been used by user u.

The tags with highest preference values $p_{n,u}(t)$ constitute the set of final recommendations. Both the TF-IDF and personalisation based mechanisms were evaluated isolated and in conjunction with the baseline approach $c_n(t)$ improving its results.

Table 5 shows the final sorted list of tags recommended for the example input bookmark: `recommender`, `collaborative`, `filtering`, `semanticweb`, `personalization`. It is important to note that these tags are not the same as the top tags obtained in Stage 3 (see Table 4). In that case, all those tags (`recommender`, `recommendation`, `collaborative`, `filtering`, `collaborativefiltering`) were biased to vocabulary about "recommender systems", and no diversity in the suggested tags was provided.

Table 5. Final tag recommendations for the example input bookmark.

Input bookmark: **A Multilayer Ontology-based Hybrid Recommendation Model**

Tag 1	recommender
Tag 2	collaborative
Tag 3	filtering
Tag 4	semanticweb
Tag 5	personalization

In the fifth and last stage, we ranked the social tags extracted from the bookmarks similar to the input one. For that purpose, a combination of tag co-occurrence graph centrality, tag frequency, and tag-based personalisation metrics was performed. With an illustrative example, we showed that this strategy seems to offer more diversity in the recommendations than simply selecting the tags that more times were assigned to similar bookmarks.

5 Experiments

5.1 Tasks

Forming part of the ECML PKDD 2009 Discovery Challenge, two experimental tasks have been designed to evaluate the tag recommendations. Both of them get the same dataset for training, a snapshot of BibSonomy system until December 31st 2008, but different test datasets:

- **Task 1**. The test data contains bookmarks, whose user, resource or tags are not contained in the training data.
- **Task 2**. The test data contains bookmarks, whose user, resource or tags are all contained in the training data.

5.2 Datasets

Table 6 shows the statistics of the training and test datasets used in the experiments. Tag assignments (user-tag-resource) are abbreviated as *tas*.

Table 6. ECML PKDD 2009 Discovery Challenge dataset.

		Web pages	**Scientific publications**	**All bookmarks**
Training	*users*	2679	1790	3617
	resources	263004	158924	421928
	tags	56424	50855	93756
	tas	916469	484635	1401104
	tas/resource	3.48	3.05	3.32
Test (task 1)	*users*	891	1045	1591
	resources	16898	26104	43002
	tags	14395	24393	34051
	tas	64460	99603	164063
	tas/resource	3.81	3.82	3.82
Test (task 2)	*users*	91	81	136
	resources	431	347	778
	tags	587	397	862
	tas	1465	1139	3382
	tas/resource	3.40	3.28	4.35

5.3 Evaluation metrics

As evaluation metric, we use the average F-measure, computed over all the bookmarks in the test dataset as follows:

$$F\big(tags_p(u,b)\big) = \frac{2 \cdot precision\big(tags_p(u,b)\big) \cdot recall\big(tags_p(u,b)\big)}{precision\big(tags_p(u,b)\big) + recall\big(tags_p(u,b)\big)}$$

where:

$$recall\big(tags_p(u,b)\big) = \frac{\big|tags(u,b) \cap tags_p(u,b)\big|}{|tags(u,b)|}$$

$$precision\big(tags_p(u,b)\big) = \frac{\big|tags(u,b) \cap tags_p(u,b)\big|}{\big|tags_p(u,b)\big|}$$

being $tags(u,b)$ the set of tags assigned to bookmark b by user u, and $tags_p(u,b)$ the set of tags predicted by the tag recommender for bookmark b and user u.

For each bookmark in the test dataset, we compute the F-measure by comparing the recommended tags against the tags the user originally assigned to the bookmark. The comparison is done ignoring case of tags and removing all characters which are neither letters nor numbers.

5.4 Results

The tag recommendation approach presented in this work exploits training bookmark meta-information and tags, but does not analyse document contents, and does not make use of external knowledge bases, to enrich the set of suggested tags. Thus, all our recommended tags belong to the training collection, and our algorithm is only suitable for Task 2 of the ECML PKDD 2009 Discovery Challenge.

Table 7 shows recall, precision and F-measure values for the test datasets provided in the tasks. In task 2, recommending 5 tags, we reach an average F-measure value of 0.3065. We obtain a precision of 42% if we only recommend one tag, and 25% when we recommend 5 tags.

Table 7. Average recall, precision and F-measure values obtained in tasks 1 and 2 of ECML PKDD 2009 Discovery Challenge for different numbers of recommended tags.

	Number of recommended tags	**Recall**	**Precision**	**F-measure**
Task 1	*1*	0.0593	0.1810	0.0894
	2	0.0910	0.1453	0.1120
	3	0.1131	0.1233	0.1179
	4	0.1309	0.1091	0.1190
	5	0.1454	0.0991	0.1179
Task 2	*1*	0.1454	**0.4190**	0.2159
	2	0.2351	0.3477	0.2805
	3	0.2991	0.3059	0.3025
	4	0.3462	0.2716	0.3044
	5	0.3916	0.2518	**0.3065**

6 Conclusions and future work

In this work, we have presented a social tag recommendation model for a collaborative bookmarking system. Our approach receives as input a bookmark (of a web page or a research publication), analyses and processes its textual metadata (document title, URL, abstract and descriptions), and suggests tags relevant to bookmarks whose metadata are similar to those of the input bookmark.

Besides focusing on those tags that best fit the bookmark metadata, our strategy also takes into account global characteristics of the system folksonomy. More specifically, it makes use of the tag co-occurrence graph to compute vertex centralities of related tags. Assuming that tags with higher vertex centralities are more informative to describe the bookmark contents, our model weights the retrieved tags through their centrality values in a small co-occurrence sub-graph generated for the input bookmark. As additional features, the weighting mechanism also penalises tags that are too generic, and strengthens tags that have been previously used by the user to whom the tag recommendations are conducted.

Two are the main benefits of our approach: a low computational cost, and the capability of providing diversity in the recommended tag sets. On one hand, an index of keywords and tags for the available bookmarks, and the global tag co-occurrence graph, are the only information resources needed. On the other hand, the combination of exploiting content-based features, tag popularity and personalisation in the recommendation process allows suggesting tags that not only are relevant for the input bookmark, but also might belong to different domains.

A main drawback of our approach is its limitation to recommend tags that already exist in the system folksonomy. The suggestion of new terms, for example extracted from the bookmarked text contents or from external knowledge bases such as dictionaries or thesauri, is thus an open research line.

More investigation is needed to improve and evaluate the effectiveness of our tag recommender. In this context, the study of alternative graph vertex centrality measures (e.g. [11]), and the exploitation of extra folksonomic information obtained from the user and item spaces (e.g., as done in [6]), represent priority tasks to address in the future. The evaluation has to be also done comparing our approach with other state-of-the-art techniques.

Acknowledgments. This research was supported by the European Commission under contracts FP6-027122-SALERO, FP6-033715-MIAUCE and FP6-045032 SEMEDIA. The expressed content is the view of the authors but not necessarily the view of SALERO, MIAUCE and SEMEDIA projects as a whole.

References

1. Adomavicius, G., Tuzhilin, A. 2005. *Toward the Next Generation of Recommender. Systems: A Survey of the State-of-the-Art and Possible Extensions*. In IEEE Transactions on Knowledge and Data Engineering, pp. 734-749.
2. Alfonseca, E., Moreno-Sandoval, A., Guirao, J. M., Ruiz-Casado, M. 2006. *The Wraetlic NLP Suite*. In Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006).
3. Byde, A., Wan, H., Cayzer, S. 2007. *Personalized Tag Recommendations via Tagging and Content-based Similarity Metrics*. In Proceedings of the 2007 International Conference on Weblogs and Social Media.
4. Chirita, P. A., Costache, S., Handschuh, S., Nejdl, W. 2007. *P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web*. In Proceedings of the 16th International Conference on World Wide Web (WWW 2007), pp. 845-854.
5. Heymann, P., Ramage, D., Garcia-Molina, H. 2008. *Social Tag Prediction*. In Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 531-538.
6. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G. 2006. *Information Retrieval in Folksonomies*. 2006. In Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), pp. 411-426.
7. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G. 2008. *Tag Recommendations in Social Bookmarking Systems*. In AI Communications, 21, pp. 231-247.

8. Kleinberg, J. 1999. *Authoritative Sources in a Hyperlinked Environment*. In Journal of the ACM, 46(5), pp. 604–632.
9. Li, J., Zha, H. 2006. T*wo-way Poisson Mixture Models for Simultaneous Document Classification and Word Clustering*. In Computational Statistics & Data Analysis, 50(1), pp. 163-180.
10. Mishne, G. 2006. *AutoTag: A Collaborative Approach to Automated Tag Assignment for Weblog Posts*. In Proceedings of the 15th International Conference on World Wide Web (WWW 2006), pp. 953-954.
11. Newman, M. E. J. 2005. *A measure of Betweenness Centrality based on Random Walks*. In Social Networks, 27, pp. 39–54.
12. Page, L., Brin, S., Motwani, R., Winograd, T. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report. Stanford InfoLab.
13. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. 1994. *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*. 1994. In Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work (CSCW 1994), pp. 175-186.
14. Salton, G., McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York.
15. Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W. C., Giles, C. L. 2008. *Real-time Automatic Tag Recommendation*. In Proceedings of the 31st Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 515-522.
16. Tatu, M., Srikanth, M., D'Silva, T. 2008. *RSDC'08: Tag Recommendations using Bookmark Content*. In Proceedings of the ECML PKDD 2008 Discovery Challenge (RSDC 2008).
17. Xu, Z., Fu, Y., Mao, J., Su, D. 2006. *Towards the Semantic Web: Collaborative Tag Suggestions*. In Proc. of the WWWW 2006 Workshop on Collaborative Web Tagging.
18. Zha, H., He, X., Ding, C., Simon, H. 2001. *Bipartite Graph Partitioning and Data Clustering*. In Proceedings of the 10th ACM International Conference on Information and Knowledge (CIKM 2001), pp. 25-32.

Social Tag Prediction Base on Supervised Ranking Model

Hao Cao, Maoqiang Xie, Lian Xue, Chunhua Liu, Fei Teng, and Yalou Huang

College of Software, Nankai University, Tianjin, P.R.China
{caohao, xuelianlotus, hytjfxk, nktengfei}@mail.nankai.edu.cn
{xiemq,huangyl }@nankai.edu.cn

Abstract. Recently, social tag recommendation has gained more attention in web research, and many approaches were proposed, which can be classified into two types: rule-based and classification-based approaches. However, too much expert experience and manual work are needed in rule-based approaches, and its generalization is limited. Additionally, there are some essential barriers in classification-based approaches, since tag recommendation is transformed into a multi-classes classification problem, such as tag collection is not fixed. Different from them, ranking model is more suitable, in which supervised learning can be used. In additions, the whole tag recommendation task can be divided into 4 subtasks according to the existence of users and resources. In different subtasks, different features are constructed, in order that existed information can be used sufficiently. The experimental results show that the proposed supervised ranking model performs well on the training and test dataset of RSDC 2008 recovered by ourselves.

1 Introduction

Tag is a new form to index web resources, which help users to categorize and share the resources, and later search them. Also, the tags assigned by specified user reveal the user's interests, therefore, according to the tags user have already tagged, someone can find other users who have the similar interests, as well as similar interesting resources. Therefore, it is widely used in social network such as Bibsonmy, Del.icio.us, Last.fm , etc.

A tag recommendation system can suggest someone a few tags to specified web resource, thus it can save the user time and effort when them mark up resources. Further, the recommended tags and existing tags can be used to predict the profile of the user and the interesting to the web resource, for example, to predict what they like and dislike. The research of tag recommendation is also very suggestive for other applications, such as online advertisement. In the field of online advertisement, we can predict what advertisement the browser might be interested in with the help of the surrounding text and his browsing history.

Recently, social tag recommendation has gained more attention in web research. It has been a hot issue for both industry and research area. For example, tag recommendation is one of the tasks in ECML RSDC's 08. Now, in ECML

PKDD 09, tag recommendation has become the exclusive task. However, the performance of tag recommendation is not good enough to be widely used, more research work is needed and progress is essential for the practical use of tag recommendation in commercial system. In this paper, supervise ranking model is applied to tackle tag recommendation problem, and good result is achieved on test data.

The rest of paper is organized as follows: Section 2 lists the previous work on tag recommendation. Section 3 gives a description of supervised ranking model. Section 4 lists our experiment settings, experiment procedure and our analysis of the results on recovered 08's dataset. The model's performance on 09's dataset is presented in Section 5. Section 6 summarizes our work.

2 Previous Work

Much research work has been done for tag recommendation, most of which can be categorized into two types, one is rule-based, the other is classification-based.

Rule based approach is used by many researchers. Lipczak [1] proposed a three-step tag recommendation system in their paper : Basic tags are extracted from the resource title. In the next step, the set of potential recommendations is extended by related tags proposed by a lexicon based on co-occurrences of tags within resource's posts. Finally, tags are filtered by the user's personomy - a set of tags previously used by the user. Tatu, et al. [2]used document and user models derived from the textual content associated with URLs and publications by social bookmarking tool users, the textual information includes information present in a URL's title, a user's description of a document, or a bibtex field associated with a scientific publication, they used natural language understanding approach for producing tag recommendations, such as extraction of concepts, extraction of conflated tags which group tags to semantically related groups. However, too much expert experience and manual work are needed in rule-based approaches, and its generalization is limited.

Classification-based approach is also used for the tag recommendation task. Katakis et al. [3] tried to model the automated tag suggestion problem as a multilabel text classification task. Heymann et al. [4] predicted tags based on page text, anchor text, surrounding hosts, and other tags applied to the URL. They found an entropy-based metric which captures the generality of a particular tag and informs an analysis of how well that tag can be predicted. They also found that tag-based association rules can produce very high-precision predictions as well as giving deeper understanding into the relationships between tags. Their results have implications for both the study of tagging systems as potential information retrieval tools, and for the design of such systems. However , the application of classification does not suggest a good solution to the tag prediction problem: first, the tag space is fixed , all the resource can be categorized to the existed tags only, also, the amount of tags number could be very large, the traditional classification model would be rather low efficient.

Collaborative filtering is a commonly used technical for user-oriented task. Many researchers tried collaborative filtering in tag recommendation. Gilad Mishne [5] used collaborative approach to automated tag assignment for weblog posts. Robert Jaschke, et al [6] evaluated and compared user-based collaborative filtering and graph-based recommender, the result shows that both of these two methods provide better results than non-personalized baseline method, especially the graph-based recommender outperforms existing methods considerably.

Adriana Budura et al. [7] used neighborhood based tag recommendation, which make use of content similarity. Principle and simple score approach is used to select the candidate tags, however, in our paper, machine learning method is used, a ranking model is learned automatically, then the candidate tags are ranked and top-ranked tags are suggested as recommending tags.

3 Supervised Ranking Model for Tag Recommendation

3.1 Problem Statement

The tag recommendation problem can be described as follows:

For a given post P whose user is U and resource is R, a set of tags are suggested as tags for the post. Here we denote post as P, tag as T, resource as R, user as U.

A possible and most nature way to solve the tag recommendation problem is as follows: First, a set of candidate tags are selected for the post, and then tags which are most likely to be the tags for the post are selected as recommending tags. The commonly used approach to choose the tags is rule-based and classification-based methods, but both of them have defects: rule-based approach relies on expert experience and manual efforts to set up the rules and tuning the parameters; classification-based is restrict to the fix of tag space and is inefficient when it is treated as a multi-label problem. In this paper, tag recommendation is conveyed to a problem of ranking candidate tags. A ranking model is constructed to ensure tags that are most likely to be post's tags rank higher than tags that are not. Supervised learning model is used to construct the ranking model satisfying the restriction. Ranking-SVM model is the most frequently used supervised ranking model and is proofed to be a successful model, so it is used as our supervised ranking model in the experiments. All the candidate tags for one post are grouped as a ranking group and the top-ranked candidate tags are selected as recommendation tags.

3.2 Introduction to Ranking SVM

Here we briefly describe the Ranking Support Vector Machine(Ranking SVM) model for tag recommendation.

Assume that $\mathcal{X} \in \Re^m$ is the input feature space which represents feature of a candidate tag given a user and resource, and m denotes the feature number. $\mathcal{Y} = \{0, 1\}$ is the output rank space which is represented by the labels, and 1 represents the tag is labeled by user, and 0 is not. $(x, y) \in \mathcal{X} \times \mathcal{Y}$ denotes feature and label as the training instance.

Given a training set with tags $T = \{t_1, t_2, ..., t_n\}$, for each tag t_i there would be a $\{x, y\}$ associated with it, the whole training set could be formulate as $S = \{x_i, y_i\}_{i=1}^{N}$, where N represents the number of all tags.

In Ranking SVM [8], ranking model f is a linear function represented by $\langle w, x\rangle$, where w is the weight vector and $\langle\cdot,\cdot\rangle$ denotes the inner product. In RSVM we need to construct a new training set S' according to the original training set $S = \{x_i, y_i\}_{i=1}^{N}$. For every $y_i \neq y_j$ in S, construct $(x_i - x_j, z_{ij})$ and add it into S', where $z_{ij} = +1$ if $y_i \succ y_j$, and otherwise -1. Here $\succ$ denotes the preference relationship, for example, $y = 1$ is preferred to $y = 0$. For denotation consistency, we denotes S' as $\{x_i^1 - x_i^2, z_i\}_{i=1}^{D}$. The final model is formalized as the following Quadratic Programming problem:

$$\begin{aligned} & min_{w,\xi_i} \frac{1}{2C}\|w\|^2 + \sum_{i=1}^{D} \xi_i \\ s.t. \quad & \xi_i > 0, z_i\langle w, x_i^1 - x_i^2\rangle \geq 1 - \xi_i \end{aligned} \tag{1}$$

And (1) could be solved using existing Quadratic Programming methods. Figure 1 is an example of ranking SVM model.

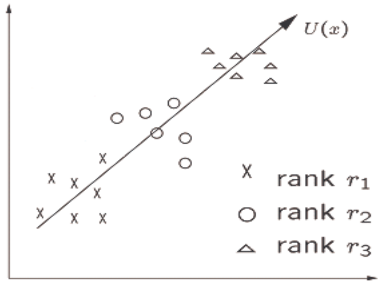


Fig. 1. Example of ranking SVM model

The ranking SVM model convey the problem of ranking into binary classification problem: for each objects to be ranked, the model compare it with all other objects in the same ranking group. For n objects, the model compares the objects C_n^2 times, and then outputs the ranking result.This is the advantage over classification model: in classification model, the existence of other candidate tags is not being considered, but in ranking model, the existence of other candidate tags is taken into consideration.

3.3 Ranking Process

For any post P_{ij} in test dataset, we denote collection of all candidate tags for post P_{ij} as $CT\{P_{ij}\}$ and $CT_k(k = 1, 2, ..., n)$ as the k-th candidate tag for the post P_{ij} , $CT\{P_{ij}\} = \{CT_1, CT_2, ..., CT_n\}$. The ranking model ranks the candidate tags to $\{CT_1{}', CT_2{}', ..., CT_n{}'\}$ from top to bottom. Then top-k tags are selected as prediction of the tags of post P_{ij}. Table 1 shows the steps to rank the candidate tags.

Table 1. Algorithm of rank the candidate tags

```
Input: candidate tags $\{CT_1, CT_2, ..., CT_n\}$
Output: top-k tags $\{CT'_1, CT'_2, ..., CT'_k\}$

1.  Extract feature $x = \{x_i\}(i = 1, 2, ..., n)$ for a sequence
    of candidate tags $CT\{P_{ij}\} = \{CT_1, CT_2, ..., CT_n\}$.
2.  Rank the features using the learned ranking model as
    $\{CT'_1, CT'_2, ..., CT'_n\}$.
3.  select top-k tags $\{CT'_1, CT'_2, ..., CT'_k\}$ as recommending tags.
```

Also, the number of recommended tags affects the performance of the system. For example, if the actual number of tags for post whose content_id=123456 is 3, a loss of precision is suffered when 4 tags are recommend to the user. So a proper number of tags to recommend should be found. The number used in our experiment is half the number of all candidate tags. If the number is bigger than 5, we cut them into 5, that means we recommend 5 tags at most.

3.4 Training Process

For all the post in the test dataset, candidate tags $CT\{P_{ij}\}$ for each post P_{ij} are extracted. Then they are grouped by the post, and features are extracted for each of them in the post content. For those $CT_k \in T\{P_{ij}\}$, we label them '1', else label them with '0'. Then we use SVM-light tool to train a ranking-SVM model. When predicting the tags of the post in test dataset, the model learned on the training dataset is applied to rank the candidate tags, and top ranked tags are selected as recommending tags.

4 Experiments on 08's recovered dataset

4.1 Experiment settings

2008's dataset recovery In order to compare our experiments' performance with that of the 08's teams, we try to get the 08's dataset (both training and test data) and test our model's performance on the recovered dataset. Though the 08's test data can be downloaded from the web, we found that user IDs have been changed between the datasets. However, the content_id field in 08's test data is consistent with 09's data, so we try to recover the 08's dataset on the 09's dataset using the content_id field and date time field. The 08's real training data and test data are subset of 09's data, so it is possible to recover 08's data on 09's data. After observing 08's real test data, we found that all posts in 08's test data are between Mar. 31, 2008 and May. 15, 2009, so we use the posts during this period on 09's training data as recovered 08's test data and posts before Mar. 31, 2008 as our recovered 08's training data. There are still slight difference between our recovered data and the 08's real data. We assume that the difference won't affect our performance seriously, so the result is comparable with 08's results.

Some statistics have been made on our recovered 08's dataset. Table 2 shows the statistics of posts on this recovered dataset. Table 3 shows the statistics of posts according to the existence of their user and resource in the recovered training data. In following part in section 4, the training data refers to the recovered training data, the test data refers to the recovered test data.

Table 2. Statistics of posts on recovered 08's dataset

Post in recovered training data	234,134	BOOKMARK	184,655
		BIBTEX	49,479
Post in recovered test data	63,192	BOOKMARK	20,647
		BIBTEX	42,545

Table 3. Statistics of posts according to their user and resource status

Users in recovered test data appear in recovered training data	265
Users in recovered test data do not appear in recovered training data	225
Resources in recovered test data appear in recovered training data	1230
Resources in recovered test data do not appear in recovered training data	61970

Data format description The dataset used in experiments is released by ECML. The data consists of three tables: TAS table, BOOKMARK table and BIBTEX table.

Table 4 is a description of the fields of the three tables. Only the fields we used in experiments are listed in the table.

Data preprocess Firstly, the terms are converted into lowercase. Then the stop words are removed, such as "a, the, is, an", these terms are not likely to be the tags of the post. Finally, the punctuations as ':', ',', etc are removed. Latex symbols such as '{' and '}' is also removed using regular expressions.

Table 5 shows example results of data preprocess.

4.2 Post Division

It can be observed from data distribution that some users of posts exist in the training data (54%) and some do not exist in the training data (46%). Also

Table 4. Data fields of TAS, BOOKMARK and BIBTEX

Table name	Fields name
TAS	user, tag, content_id, content_type, date
BOOKMARK	content_id (matches tas.content_id) ,url description ,extended ,description ,date ,bibtex
BIBTEX	content_id (matches tas.content_id) ,simhash1 (hash for duplicate detection among users) ,title

Table 5. Example results of data preprocess

Before data preprocess	After data preprocess
Ben Mezrich: the telling of a true story	ben mezrich telling true story
{XQ}uery 1.0: An {XML} Query Language, {W3C} Working Draft	xquery 1.0 xml query language w3c working draft

some resources of posts exist in the training data (2%) and others do not exist in the training data (98%).

In the analysis above, we divide the posts in test dataset into two categories according to the existence of their users in the training data: existed user posts, non-existed user posts. Also, the posts in test dataset can be divided into two categories according to the existence of their resource in the training data: existed resource posts, non-existed resource posts.

The posts can be divided into four different categories according to their user status and resource status in the training data: existed user existed resource post, existed user non-existed resource post, non-existed user existed resource post, non-existed user non-existed resource post.

We denote symbols as shown in Table 6 to simplify the language.

Table 7 and Table 8 show statistics after our post division on our recovered 08's data.

Table 6. Simplified symbols

EUER post	Existed user existed resource post
EUNR post	Existed user non-existed resource post
NUER post	Non-existed user existed resource post
NUNR post	Non-existed user non-existed resource post

It can be observed from statistics that not every category of posts occupies the same ratio of the posts. In BOOKMARK, EUNR posts occupied about 82.80% of all BOOKMARK posts. In BIBTEX, NUNR posts occupied about 93.43% of all BIBTEX posts. In order to promote our model's performance on

Table 7. Distribution of different categories of BOOKMARK posts in test dataset

Category	Posts number	ratio
EUER post	621	3.01%
EUNR post	17099	82.80%
NUER post	346	1.68%
NUNR post	2585	12.52%

Table 8. Distribution of different categories of BIBTEX posts in test dataset

Category	Posts number	ratio
EUER post	164	0.39%
EUNR post	2532	5.95%
NUER post	99	0.23%
NUNR post	39754	93.43%

the test dataset, we should focus on those data which occupy high proportion of the posts, that is: EUNR posts of BOOKMARK and NUNR posts in BIBTEX.

After data division, the following steps are carried out for our tag recommendation task.

1. Extract candidate tags by different methods according to the category of post.

2. Rank the candidate tags, and select top ranked tags as recommendation tags.

4.3 Candidate tags extraction

According to the statistics of the sources of the tags on the dataset, we can find that tags can be retrieved from three sources mainly: 1.The content information of the post, such as 'description' field in BOOKMARK and 'title' field in BIBTEX. 2. $T\{R_j\}$: The tags being assigned to the same resource previously. 3.$T\{U_i\}$: The tags assigned by the same user previously. Statistics of tags from different sources for BOOKMARK and BIBTEX posts are listed in Table 9 and Table 10.

Table 9. Statistics of the tags from 3 sources of BOOKMARK Post

Total tags	56267
Tags from terms of description	5253
Tags from terms of URL	1353
Tags from user's previous tags	29672

Table 10. Statistics of the tags from 3 sources of BIBTEX Post

Total tags	95782
Tags from terms of title	41801
Tags from terms of URL	547
Tags from user's previous tags	5377

The four different categories of test dataset have different characters, for example, we can explore the tags assigned by user previously and the tags assigned to the resource previously for EUER posts. But for NUNR posts, we lack this information. So we should explore different features for the four different categories of posts individually, in order that existed information can be used sufficiently. In the following part, while using the supervised ranking model, we train four models to handle these four categories of posts individually.

The candidate tags extraction strategies for different categories of posts:

For EUER post and NUER post, $CT\{P_{ij}\} = \{$ terms in post $(P_{ij}) \bigcup T\{R_j\}\}$.

For EUNR post and NUNR post, $CT\{P_{ij}\} = \{$ terms in post $(P_{ij})\}$.

We denote the candidate tags for post whose user_id=i and resource is j as $CT\{P_{ij}\}$. $\{$ terms in post $(P_{ij})\}$ denotes the remaining set of words after trimming and removing of the stop words in the text information of post P_{ij}.

Notice should be paid here that we do not take $T\{U_i\}$ (the user's pervious tags) as candidate tags because we find the tags are too massive. When they are added, the precision of the system drops down and the F-1 value on the whole dataset also declines dramatically. However, in the ranking procedure, we will use $T\{U_i\}$ as one of the features in SVM model to rank the candidate tags.

4.4 SVM Features construction

While using SVM, we select features that discern high ranked tags and low ranked tags well and add the features according to our experience. For example, the term frequency in the post content: those words which have high term frequency within the post content tend to rank higher than those which have low term frequency. Also, whether the candidate words have been used as tags for other post in the training data is an excellent feature.

Table 11 is a brief description of features of ranking SVM model for BOOKMARK posts. The features for BIBTEX posts are almost the same except for the different data fields:

4.5 Analysis of Model

Table 12 and Table 13 show the results of our supervised Ranking SVM model on the recovered 08's data.

Combing different types and category of data together, we can get the overall performance on the recovered 08's test data, as shown in Table 14.

Table 11. Some of the features for ranking SVM model for BOOKMARK

Feature1	Candidate tag's TF (term frequency) in post's description terms.
Feature2	Candidate tag's TF in post's URL terms.
Feature3	Candidate tag's TF in post's extended description terms.
Feature4	Candidate tag's TF in $T\{R_j\}$ (tags assigned to the post of the same URL in the training data).
Feature5	Candidate tag's TF in $T\{U_i\}$ (tags assigned previously by user in the training data.)
Feature6	Times of candidate tag being assigned as a tag in the training data.

Table 12. Individual and overall Performance on BOOKMARK posts

Post category	Recall	Precision	F1-value	ratio
EUER Post	0.369699	0.394973	0.381918	3.01%
EUNR Post	0.046591	0.053739	0.04991	82.80%
NUER Post	0.160883	0.255652	0.197487	1.68%
NUNR Post	0.069158	0.106366	0.083819	12.52%
overall-performance on BOOKMARK	0.061067	0.073997	0.066633	

Table 13. Individual and overall Performance on BIBTEX posts

Post category	Recall	Precision	F1-value	ratio
EUER Post	0.4219356	0.3472393	0.3809605	0.39%
EUNR Post	0.2250226	0.1628605	0.1889605	5.95%
NUER Post	0.5667162	0.3715986	0.4488706	0.23%
NUNR Post	0.3561221	0.1603686	0.2211494	93.43%
overall-performance on BIBTEX	0.349063	0.161732	0.220381	

Table 14. Overall performance on test dataset using ranking SVM model

Recall	Precision	F1-value
0.153	0.185	0.167

The F1-value is 0.167, less than the F1-value 0.193 of the team ranked first in 08's competition.

It can be observed from the results that the performance of the model is poor on EUNR posts, which occupied most of the BOOKMARK posts. However, the model performs well on EUER posts. When comparing the two types of data, we find that the only difference is that the candidate tags of EUER posts are not only come from the post content but also from the tags of the same resource in the training data, however, the candidate tags for EUNR posts come from post content only. In order to overcome the weakness of lacking candidate tags, we relax restriction on the definition of the same resource. For those posts whose resources have not appeared in the training data, the role of the same post is substituted by the similar post. This method is based on the assumption that users tend to tag the similar posts with the same tags.

We try to use post content similarity to measure the similarity of posts. For those EUNR posts, which have no same resources in the training data, we add the tags of those posts whose content similarity with the current post content is above a certain threshold to the candidate tags set of the post.

4.6 Post content similarity based KNN model

For EUNR post, the candidate tags come from text of the post content only, that is $CT\{P_{ij}\} = \{$ terms in post $(P_{ij})\}$. We attribute the poor performance of the model on such kind of data to the sparse of candidate tags. So we use content similarity to expand the candidate tags set. For any EUNR post P_{ij}, we set a similarity threshold t, and find in the training dataset content P_{mn}, whose $sim(text(P_{ij}), text(p_{mn}) > t)$. Then the tags of post P_{mn} are added to the candidate tags ofP_{ij}: $CT\{P_{ij}\} = \{$ terms in post $(P_{ij})\} \bigcup T\{P_{mn}\}$.

Post content P_{ij} and P_{mn} are mapped into vector space:

$text(P_{ij}) = \{W_1, W_2, ..., W_n\}$, $text(P_{mn}) = \{W_1{}', W_2{}', ..., W_n{}'\}$,Then we use vector space model to calculate the similarity between two posts P_{ij} and P_{mn}.

$$sim(text(P_{ij}), text(P_{mn})) = \frac{text(P_{ij}) * text(P_{mn})}{|text(P_{ij})| * |text(P_{mn})|} \tag{2}$$

W_i means the weigh of word i in the content. The simplest way to define W_i is as following:$W_i = 0$,word i in post content, $W_i \neq 0$,word i not in post content.

In our experiment, we define the W_i as TF(Term Frequency) multiply IDF (Inverted document frequency) :$W_i = TF_i * IDF_i$.We applied open source software Lucene to calculate the similarity of two content , the scoring function of Lucene is a derivation of vector space model formula using TF/IDF weighing schema.

The modification of threshold value T and the corresponding performance on EUNR content in BOOKMARK are shown in Figure 2.

It can be observed that the value of recall, precision and F1 value reach highest when threshold T=0.5. So, in the further experiment settings, we set threshold value T to 0.5.

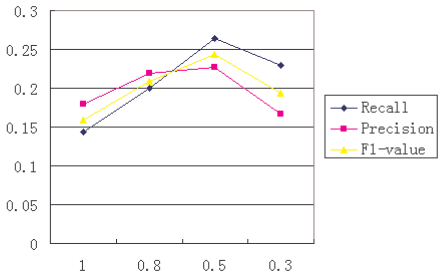


Fig. 2. KNN performance on various threshold t on BOOKMARK EUNR posts, k=5

However, we find that the application of content similarity based KNN model works for BOOKMARK posts but not for BIBTEX posts. After investigation, we attribute it to the uneven distribution of the dataset in training datasets and test datasets. In training datasets, the number of BOOKMARK posts is 184,655 and the number of BIBTEX posts is 20,647. But in test dataset, the number of BOOKMARK post is 20,647 and the number of BIBTEX post is 49,479, it is easy for 20,647 BOOKMARK posts to find similar posts in 184,655 BOOKMARK posts, but difficult for 42,545 BIBTEX posts in only 20,647 posts. So this method is especially useful for BOOKMARK posts but not for BIBTEX posts.

After applying content similarity based KNN model on BOOKMARK EUNR posts, the performance on overall test dataset is as listed in Table 15.

Table 15. Overall performance on test dataset adding content similarity based KNN model

Recall	Precision	F1-value
0.323828	0.200926	0.238803

The F1-value is 0.238, higher than the F1-value 0.193 of the team ranked first in 08s competition.

5 Experiment on 09's dataset

5.1 Statistics of 09's dataset

Table 16 and Table 17 show the distribution of different categories of posts on 09's dataset after data division according to the existence of their user and resource in the training data. In our experiment settings on 09's test data, clean-dump dataset is used as training dataset in Task 1, Post-core dataset is used as training dataset in Task 2.

Table 16. Different categories of BOOKMARK posts in 09s test dataset for Task 1

Category	Posts number	ratio
EUER Post	821	4.86%
EUNR Post	10622	62.86%
NUER Post	872	5.16%
NUNR Post	4583	27.12%

Table 17. Different categories of BIBTEX posts in 09s test dataset for Task 1

Category	Posts number	ratio
EUER Post	365	1.40%
EUNR Post	9287	35.71%
NUER Post	591	2.27%
NUNR Post	15761	60.61%

It can be observed from the statistics of the distribution of categories in 09's test data for Task 1 agrees with the recovered 08's dataset: EUER posts occupied most of the BOOKMARK post and NUNR post occupied large proportion of BIBTEX posts, so we can expect our model a good result on such data. The whole posts in 09's test dataset for Task2 can be classified to EUER posts. Since the good performance of our model on EUER posts, we can also expect a good result on task 2.

Eight different models are trained on 09's clean-dump training data and applied in 09's test data for Task 1. For Task 2, we apply the BOOKMARK EUER post model and the BIBTEX EUER post model trained on 09's post-core dataset.

5.2 Experiment results on 09's test dataset

The performance on the whole 09's test data of both task 1 and task 2 is shown in Table 18.

Table 18. Performance on 09's dataset @5

Task No.	Submission ID	Precision	Recall	F1-value
1	67797	0.162478	0.146582	0.154121
2	13651	0.31622	0.222065	0.260908

6 Conclusion

In this paper, we briefly describe an approach utilizes supervised ranking model for tag recommendations. Our tag prediction contains three steps. First, posts are divided into four categories according to the existence of the user and the resource in the training data and then candidate tags are extracted for the different categories with different strategies. Second, features are decided according to categories. Then we rank the candidate tags, using the supervised ranking model, and pick the top tags as recommendation tags.

For the existed user non-existed resource post, we use post content similarity based KNN model to expand the candidate tags set. Performance of this experiment for the corresponding module is promoted after adding this model on 08's dataset. Our tag recommendation system is generated from the combination of these two models and applied to the 09's tags recommendation task 1 and task 2.

Acknowledgement

Thanks to Zhen Liao for his helpful discussions and suggestions for this paper.

This paper is supported by the National Natural Science Foundation of China under the grant 60673009 and China National Hanban under the grant 2007-433.

References

1. Marek Lipczak, Tag Recommendation for Folksonomies Oriented towards Individual Users, *ECML 2008*
2. Marta Tatu, Munirathnam Srikanth, and Thomas D'Silva, RSDC'08: Tag Recommendations using Bookmark Content, *Proceedings of ECML PKDD Discovery Challenge 2008 (RSDC 2008)*
3. Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas, Multilabel Text Classifcation for Automated Tag Suggestion, *Proceedings of ECML PKDD Discovery Challenge 2008 (RSDC 2008)*
4. Paul Heymann, Daniel Ramage, and Hector Garcia-Molina, Social Tag Prediction, SIGIR'08, July 20–24, 2008, Singapore.pages 531∼538
5. Gilad Mishne,AutoTag: A Collaborative Approach to Automated Tag Assignment for Weblog Posts. *WWW 2006,* May 22–26, 2006, Edinburgh, Scotland. pages 953 954
6. Robert J¨aschke, LeandroMarinho, Andreas Hotho,Lars Schmidt-Thieme, and Gerd Stumme, Tag Recommendations in Folksonomies J.N. Kok et al. (Eds.): *PKDD 2007,* LNAI 4702, pp. 506–514, 2007.
7. Adriana Budura, Sebastian Michel, Philippe Cudre-Mauroux, and Karl Aberer, Neighborhood-based Tag Prediction, *6th Annual European Semantic Web Conference (ESWC2009)*
8. R Herbrich, T Graepel, K Obermayer ,Large margin rank boundaries for ordinal regression, textitKDD'02: Proceedings of the eighth ACM SIGKDD international
9. Yunbo CAO, Jun XU, Tie-Yan LIU, Hang LI, Yalou HUANG, Hsiao-Wuen HON, Adapting Ranking SVM to Document Retrieval, *SIGIR'06,* August 6-11,2006, Seattle, Washington, USA. pp.186∼193

Tag Recommendations Based on Tracking Social Bookmarking Systems

Szymon Chojnacki

Department of Artificial Intelligence, Institute of Computer Science,
Polish Academy of Sciences

Abstract. The purpose of this paper is to describe our approach to Tag Recommendation Task during ECML PKDD 2009 Challenge. The organizers supplied a training set of tagged webpages and publications from BibSonomy portal. Our goal was to build a model which can predict tags for new users bookmarking digital resources. Our strategy was based on an assumption that users tend to tag the same resources in various systems. Therefore, have we developed a tracking engine, which was adjusted to the profile of BibSonomy users in selection of RSS feeds and utilized the training data to optimize the list of tracked URLs. We had over 90 days to collect the data from the feeds, but this period did not overlap with the dates of posts from the training set. As a result we had to set manually parameters responsible for a trade-off between recall and accuracy of the model. We stored all downloaded feed entries in a searching engine. The recommendation was based on tags attached to the documents retrieved from the engine by means of typical information retrieval query.

Keywords: Information Retrieval, Searching Engines, Tag Recommendations.

1 Introduction

The development of collaborative society that we experience in recent years can be characterized by four principles: being open, peering, sharing and acting globally [6]. These principles determine the way we exchange information and organize the knowledge. Very important part of this phenomenon is the popularity of social classification, indexing and tagging. Attaching labels to common resources (webpages, blogs, music, videos, photos) can on one hand shed a new light on information retrieval problems, on the other hand poses new challenges concerning uncontrolled explosion of folksonomy size and its usability. The goal of our research is to build a tag recommendation system that would influence user’s selection of tags and as a result enable us to reuse folksonomy entries in more efficient way than we observe currently

This paper describes our attempt to predict tags already chosen by BibSonomy users. This was the Task 1 in ECML PKDD 2009 Challenge. However, we believe that our system is better suited for the third Task, in which the Teams have an opportunity to deliver recommendations online.

2 Related work

The growing interest of research community in the field of social bookmarking was fueled by last year's ECML challenge, during which the evaluation measures were standardized and benchmark data sets prepared. Thirteen solutions were submitted to the tag spam detection task and only five to tag recommendation task. We were inspired by the best teams in the Challenge, which relied on several external resources [7] and used only data available in title/description fields [5]. The team from the Aristotle University of Thessaloniki [3] reformulated the task as a multilabel classification problem.

3 Examined datasets

We used cleaned dump dataset which consisted of three tables: bibtex (158 924 records), bookmark (263 004 records) and tas (1 401 104 records). The dump contained all public bookmarks and publication posts of BibSonomy until (but not including) 2009-01-01. Posts from the user dblp (a mirror of the DBLP Computer Science Bibliography) as well as all posts from users which have been flagged as spammers have been excluded. Furthermore, the tags were cleaned. Java method was used to remove all characters which were neither numbers nor letters and removed those tags, which were empty after cleansing or matched one of the tags imported, public, systemimported, nn, systemunfiled.

The tas table (Tag Assignments) was a fact table with information about who attached which tag to which resource/content. The bookmark table consisted of following columns (content_id, url_hash, url, description, extended description and date). The bibtex table was described by following dimensions (content_id, journal, volume, chapter, edition, month, day, booktitle, howPublished, institution, organization, publisher, address, school, series, bibteXKey, url, type, description, annote, note, pages, key, number, crossref, misc, bibtexAbstract, simhash0, simhash1, simhash2, entrytype, title, author, edition, year).

4 Our approach

In this section we describe three main parts of our system. Firstly we focus on a selection of RSS feeds and the problems we encountered while downloading the posts. In the second part we define the vector space in which the posts were stored as well as main characteristics of deployed database. Finally we present the details of the tag recommendation algorithm. The algorithm is divided into four steps: searching of matching resources based on URL address, retrieval of the most similar cluster, selection of the post with highest overlap score and ranking of suggested tags.

4.1 RSS Feeds selection

Our strategy was to optimize a set of keywords that we were going to track in popular bookmarking systems as well as in a variety of domain portals. We analyzed distribution of most common tags in BibSonomy and Delicious and decided that tracking only the most recent posts would be biased (Table 1). We decided to enrich the most recent posts with a set of 100 most popular tags (out of 93 757 unique tags) in BibSonomy training data.

Table 1. The most common tags in Delicious and Bibsonomy

	Tag	Delicious[1]	BibSonomy[2]
1	design	1.69%	27
2	blog	1.29%	13
3	tools	1.05%	10
4	software	0.96%	4
5	webdesign	0.92%	54
6	programming	0.89%	5
7	tutorial	0.85%	44
8	art	0.75%	83
9	reference	0.72%	33
10	video	0.72%	3
11	inspiration	0.71%	587
12	music	0.66%	25
13	web2.0	0.65%	7
14	education	0.63%	17
15	photography	0.52%	166

We had to face different problems in case of bookmarking systems and domain portals. We used Google Reader to search for top 10 domain portals and their RSS URLs for each chosen keyword. Because some feeds appeared in different searching results we end up with 734 feeds.

An example of feeds recommended by Google Reader for a keyword “linux” is presented in Table 2. Even though numerous feeds use the most recent RSS or Atom standard and we could easily parse the content of XML files, it is uncommon to fill in the category field by feed editors. We can see in the Table 2, that out of 10 sources: one did not contain proper URL, four did not deliver information about category, one marked each feed entry with the same category.

[1] Relative frequency of a tag in a random collection of 603 750 downloaded from the Delicious.

[2] Rank of a corresponding tag in the BibSonomy.

Table 2. Results of searching first ten feeds for "Linux" keyword in Google Reader.

RSS Feed	Categories of updated entries
Linux Insider[3]	| Community | Community | Distros | Licensing | Financial News | Mobile | Community | Community | Mobile
Linux Magazine[4]	No Categories
DistroWatch.com: News[5]	No Categories
Linux Today[6]	No Categories
Slashdot: Linux[7]	| programming | xwindows | google | gui | software | microsoft | portables | os | storage | linuxbusiness | security | gnu | education | caldera | portables
Linux.com :: Features[8]	URL broken
HowtoForge - Linux Howtos and Tutorials[9]	| Ubuntu | Debian | Ubuntu | Desktop | Debian | Lighttpd | Ubuntu | Desktop | Virtualization | Ubuntu | Desktop | Security | Ubuntu | CentOS | Samba | Ubuntu | Desktop | Linux | Ubuntu | Security | Ubuntu | Desktop | Fedora | Security
Linux and Open Source - RSS Feeds[10]	No Categories
LinuxQuestions.org[11]	| Linux - Newbie | Linux - Newbie | Linux - Newbie | Linux - Software | Programming | Red Hat | Linux - General | Linux - General | Linux - Laptop and Netbook | Puppy | Ubuntu | Linux - Desktop | Ubuntu | Ubuntu | Linux - Security
LXer Linux News[12]	| linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux | linux

On the other hand the problem with typical bookmarking systems is the fact that when we subscribe most recent posts for a given keyword we get only tags of a particular user who bookmarked the resource. As a consequence we need to crawl a service in order to find out about most typical tags for a given resource. The problem of

[3] http://www.linuxinsider.com/perl/syndication/rssfull.pl
[4] http://www.linux-mag.com/cache/rss20.xml
[5] http://distrowatch.com/news/dw.xml
[6] http://linuxtoday.com/backend/biglt.rss
[7] http://rss.slashdot.org/Slashdot/slashdotLinux
[8] http://www.linux.com/index.rss
[9] http://www.howtoforge.com/node/feed
[10] http://rssnewsapps.ziffdavis.com/eweeklinux.xml
[11] http://www.linuxquestions.org/syndicate/lqlatest.xml
[12] http://lxer.com/module/newswire/headlines.rss

connection limits arises when we want to crawl every out of 100 entries downloaded for a given keyword. Because of this, we decided to verify if we can cluster tags based on their cooccurence score.

Table 3 contains 20 pairs of tags with highest symmetric Jaccard cooccurance coefficient calculated as a division of number of posts with both tags by a number of all posts with any of the tags.

Table 3. Co-occurances of pairs of tags, occurances and normalized Jaccard coefficient.

Tag 1	Tag 2	$\lvert t_1 \cap t_2 \rvert$	$\lvert t_1 \rvert$	$\lvert t_2 \rvert$	$\frac{\lvert t_1 \cap t_2 \rvert}{\lvert t_1 \cup t_2 \rvert}$
ccp	jrr	4294	4294	4294	1.00
algorithms	genetic	5775	6220	5888	0.91
aaaemulation-topgames	emulation-videogames	3653	3653	4576	0.80
emulationgames	emulation-videogames	4576	6055	4576	0.76
aaaemulation-topgames	classicemulated-remakeretrogames	2472	3653	2472	0.68
aaaemulation-topgames	emulationgames	3653	3653	6055	0.60
classicemulated-remakeretrogames	emulation-videogames	2472	2472	4576	0.54
genetic	programming	5262	5888	9491	0.52
journal	medical	1693	2566	2448	0.51
algorithms	programming	5303	6220	9491	0.51
classicemulated-remakeretrogames	emulationgames	2472	2472	6055	0.41
book	nlp	1230	2614	2027	0.36
education	learning	2143	5021	4751	0.28
media	texts	1998	7149	2012	0.28
analysis	data	1187	3352	2589	0.25
folksonomy	tagging	1027	2561	3083	0.22
emulationgames	zzztosort	2844	6055	11839	0.19
audio	music	919	1857	4142	0.18
howto	tutorial	850	2876	2798	0.18
bookmarks	indexforum	9164	52795	9183	0.17

We can see that “ccp” and “jrr” always appear together. Also “genetic”, “algorithms” and “programming” create a cloud of tags. Four tags “emulationgames”, “emulationvideogames”, “aaaemulationgames”, “classicemulatedremakeretrogames” create another cloud. However, the Jaccard coefficient drops very fast below 20% level and therefore we decided not to abandon the idea of tag clustering.

4.2 Data storage

In order to recommend tags online we needed a fast engine that does not need to be taught every time we get need posts from a scratch. The Beatca system (developed in our Institute [1,4]) is an example of such engine. It performs online incremental hierarchical clustering of documents and proved very effective in the field of intelligent Information Retrieval. Soft classification of documents and construction of conceptual closeness graph is based on large-scale Bayesian networks. Optimal document map search and document clustering is based on SOM (self-organizing maps), AIS (artificial immune systems), and GNG (growing neural gas).

Each post is defined as a point in a multidimensional space in which coordinates represent frequency of a token appearing in a post's title or description. Because some tokens are very common and others are present in only few posts we selected only the most informative tokens as coordinates in our vector space. The dictionary optimization was based on a entropy-like quality measure $Q(t_i)$ of a token t_i:

$$Q(t_i) = \frac{N_i}{N} \cdot \frac{-\sum_{j=1}^{N} \frac{N_{ij}}{N_i} \cdot \log \frac{N_{ij}}{N_i}}{\log N_i} \tag{1}$$

where N_{ij} is the number of occurrences of term t_i in document d_j, N_j is the number of documents that contains term t_i and N is the total number of documents. We removed tokens with $Q(t_i)$ measure below 0.01 or above 0.95.

We implemented term frequency inverse document frequency weighting scheme. According to the scheme we divided term frequency in a single document by the number of documents in which the term appears.

4.3 Tag recommendations

Our tag recommendation consisted of four steps. If we had a positive result in the first step then we went directly to the final fourth step.

Step One

In the first step we checked if a post is present in the BibSonomy training set or an URL of the post is among downloaded RSS entries. If the answer was true then we selected all tags attached to these resources and moved to the Step Four.

Step Two

In the second step we retrieved a group (cluster) of documents that was the most similar to the post's description or title field. The similarity was measured as a cosine of an angel between vectors $x=\{x_1,\ldots,x_n\}$ and $y=\{y_1,\ldots,y_n\}$ representing the resources in our database and the post (Eq. 2). For example, one of the posts had following title: "Attribute Grammar Based Programming and its Environment". The query consisting of the first five informative tokens from the above title returned a cluster of four documents:

1. “A purely functional high order attribute grammar based system” from Deliciuos.com; tagged with [lisp;rag;compilers]
2. “AspectAG 0.1.1 strong typed attribute grammar implemented using type-level programming” from Reedit.com; tagged with [haskell]
3. “A 2D game programming environment based around the Ruby programming language” from Dzone.com; tagged with [frameworks,games,ruby]
4. “Attribute grammar based language extension for Java” from CiteULike.org tagged with [metaprogramming]

$$\cos(\alpha) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2}$$

A cluster of all the retrieved posts was transferred to the next step.

Step Three

For all the posts retrieved in the second step we calculated normalized overlap score and chosen the post with the highest score. The overlap was defined as a maximum length of n-gram appearing in both posts. In order to compute the score we used all the words from title/description fields (not only the most informative tokens). The overlap score was divided by the length of title/description field of the candidate posts. For example, normalized overlap score between “Attribute Grammar Based Programming and its Environment” and “Attribute grammar based language extension for Java” equals to 3/7=0.42. The post with highest score was transferred to the final fourth step if the value of a score was greater than 0.6 threshold.

Step four

In the last step we ordered the tags of selected post according to their count in BibSonomy training set. Top five tags were selected as predictions in the Challenge.

5 Evaluation

The F1-Measure common in Information Retrieval was used to evaluate the recommendations. The precision and recall were first computed for each post in the test data by comparing the recommended tags against the tags the user has originally assigned to this post [2]. Then the average precision and recall over all posts in the test data was used to calculate the F1-Measure as f1 = (2 * precision * recall) / (precision + recall). The number of tags one can recommend was not restricted. However, the organizers regarded the first five tags only. We computed both precision and recall measures for various levels of a threshold parameter from step three in our recommendation algorithm (Fig. 1). According to these simulations optimum level of the threshold is approximately 0.6 and yields F1-measure between 3% and 4%.

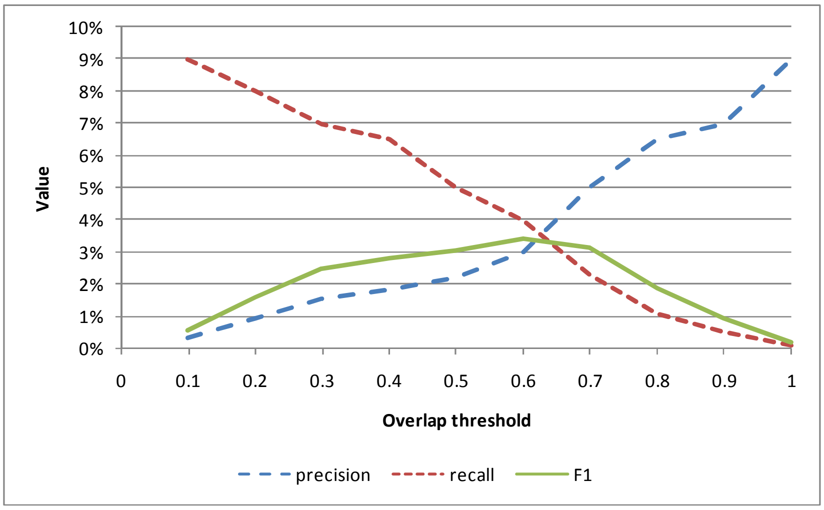


Fig. 1. Values of precision, recall and F1 measures for different levels of overlap threshold.

During the challenge we obtained overall F1-measure of 4,6%, which was slightly better than in our simulations, but incomparable to the results of the best teams.

6 Conclusions

We must admit that the way we approached the problem needs substantial computing power and disc space. Unfortunately the quality of our tag recommendations was below an average and probably this direction of research in the field of tag recommending systems is not a promising one. However we believe that there are certain situations in which best tags are not a function of words contained in title of a post and in our future research we would like to focus on such examples. Despite of unsatisfactory result in the first Task of ECML PKDD 2009 Challenge we are going to verify our recommendations within the third online recommendation task.

References

1. Ciesielski, Draminski, Klopotek, Kujawiak, Wierzchon, "On some clustering algorithms for document maps creation", in: Proceedings of the Intelligent Information Processing and Web Mining Conference (IIS:IIPWM-2005), Gdansk, Advances in Soft Computing, Springer-Verlag, (2005).
2. Jäschke, Marinho, Hotho, Schmidt-Thieme, Stumme, "Tag Recommendations in Social Bookmarking Systems", in: AI Communications , Amsterdam: IOS Press (2008)
3. Katakis, Tsoumakas, Vlahavas, "Multilabel Text Classification for Automated Tag Suggestion", in: Proceedings of ECML PKDD Discovery Challenge (RSDC08) (2008).
4. Klopotek, Wierzchon, Ciesielski, Draminski, Kujawiak, "Coexistence of Fuzzy and Crisp Concepts in Document Maps", in: Proceedings of the International Conference on Artificial Neural Networks (ICANN 2005), Lecture Notes in Artificial Intelligence, LNAI 3697, Springer-Verlag, 2005
5. Lipczak, "Tag Recommendation for Folksonomies Oriented towards Individual Users", in: Proceedings of ECML PKDD Discovery Challenge (RSDC08) (2008).
6. Tapscott, Williams, "Wikinomics", Atlantic Books, London, (2008).
7. Tatu, Srikanth, D'Silva, RSDC'08: "Tag Recommendations using Bookmark Content", in: Proceedings of ECML PKDD Discovery Challenge (RSDC08) (2008).

A Fast Effective Multi-Channeled Tag Recommender

Jonathan Gemmell, Maryam Ramezani, Thomas Schimoler,
Laura Christiansen, and Bamshad Mobasher

Center for Web Intelligence
School of Computing, DePaul University
Chicago, Illinois, USA
{jgemmell,mramezani,tschimoler,lchris,mobasher}@cti.depaul.edu

Abstract. Collaborative tagging applications allow users to annotate online resources, resulting in a complex three dimensional network of interrelated users, resources and tags often called a folksonom A pivotal challenge of these systems remains the inclusion of the varied information channels introduced by the multi-dimensional folksonomy into recommendation techniques. In this paper we propose a composite tag recommender based upon popularity and collaborative filtering. These recommenders were chosen based on their speed, memory requirements and ability to cover complimentary channels of the folksonomy. Alone these recommenders perform poorly; together they achieve a synergy which proves to be as effective as state of the art tag recommenders.

Key words: Folksonomies, Tag Recommenders, Hybrid Recommenders

1 Introduction

Collaborative tagging has emerged as a popular method for organizing and sharing online content with user-defined keywords. Delicious[1], Flickr[2] and Last.fm[3] are among the most popular destinations on the Web allowing users to annotate bookmarks, digital photographs and music. Other less popular tagging applications serve niche communities enabling users to tag blogs, business documents or scholarly articles.

At the heart of collaborative tagging is the post; a user describes a resource with a set of tags. A collection of posts results in a complex network of interrelated users, resources and tags commonly referred to as a folksonomy [10].

The rich tapestry of a folksonomy presents an enticing target for data mining techniques such as recommenders. Recommenders reduce a burdensome number of items to a manageable size correlated to the user's interests. Recommendation in folksonomies can include resources, tags or even other users. In this work

[1] delicious.com

[2] www.flickr.com

[3] www.last.fm

we focus on tag recommendation, the suggestion of tags during the annotation process.

Tag recommendation reduces the cognitive effort from generation to recognition. Users are therefore encouraged to tag more frequently, apply more tags to a resource, reuse common tags and perhaps use tags the user had not previously considered. User error is reduced by eliminating capitalization inconsistencies, punctuation errors, misspellings and other discrepancies. The final result is a cleaner denser dataset that is useful in its own right or for further data mining techniques.

Despite the richness folksonomies offer, they present unique challenges for tag recommenders. Traditional recommendation strategies, often developed to work with two dimensional data, must be adapted to work with the three dimensional nature of folksonomies. Otherwise they risk disregarding potentially useful information. To date the most successful tag recommenders are graph-based models, which exploit the links between users, resources and tags. However, this approach is computationally intense and ill suited for large scale implementation.

In this work we propose a composite tag recommender incorporating several distinct recommendation strategies. These recommenders are combined to generate a new hybrid. As such no single recommender is required to fully exploit the data structure of the folksonomy. Instead the recommenders may specialize in a single channel. The aggregation of these recommenders, none of which performs well on its own, produce a synergy allowing the composite recommender to outperform its constituent parts.

Our hybrid includes popularity models and item-based collaborative filtering techniques. Popularity based approaches include information garnered from the crowd with little computational cost. Item-based collaborative filtering focuses more closely on the user's profile incorporating a degree of personalization.

We provide a through evaluation of the composite recommender and its constituent parts. Our experiments reveal that the composite model produces results far superior to the capabilities of their individual components. We further include a comparison with the highly effective but computationally inefficient graph-based approach. We show that a low cost alternative can be constructed from less time consuming recommenders and perform nearly as well as the state or the art graph based approaches.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we offer a model of folksonomies and describe tag recommendation. We further describe four recommendation algorithms. Informational channels in folksonomies are discussed in Section 4. We design a hybrid recommender in Section 5. Our experimental evaluation is presented in Section 6 including a discussion of the dataset, methodology and results. Finally we end the paper with a discussion of our conclusions and directions for future work in Section 7.

2 Related Work

As collaborative tagging applications have gained in popularity researchers have begun to explore and characterize the tagging phenomenon. In [9] and [4] the authors studied the information dynamics of Delicious, one of the most popular folksonomies. The authors discussed how tags have been used by individual users over time and how tags for an individual resource stabilize over time. They also explored two semantic difficulties: tag redundancy, when multiple tags have the same meaning, and tag ambiguity, when a single tag has multiple meanings. In [9] the authors provide an overview of the phenomenon and explore reasons why both folksonomies and ontologies will have a place in the future of information access.

There have been several recent research investigations into recommendation within folksonomies. Unlike traditional recommender systems which have a two-dimensional relation between users and items, tagging systems have a three dimensional relation between users, tags and resources. Recommender systems can be used to recommend each of the dimensions based on one or two of the other dimensions. In [17] the authors apply user-based and item-based collaborative filtering to recommend resources in a tagging system and uses tags as an extension to the user-item matrices. Tags are used as context information to recommend resources in [13] and [12].

Other researchers have studied tag recommendation in folksonomies. In [7] user-based collaborative filtering is compared to a graph-based recommender based on the Pagerank algorithm for tag recommendation. The authors in [5] use association rules to recommend tags and introduce an entropy-based metric to define how predictable a tag is. In [8] the title of a resource, the posts of a resource and the user's vocabulary are used to recommend tags.

General criteria for a good tagging system including high coverage of multiple channels, high popularity and least-effort are presented in [18]. They categorize tags as content-based tags, context-based tags, attribute tags, subjective tags, and organizational tags and use a probabilistic method to recommend tags. In [2] the authors propose a classification algorithm for tag recommendation. The authors in [15] use a co-occurrence-based technique to recommend tags for photos in Flickr. The assumption is that the user has already assigned a set of tags to a photo and the recommender uses those tags to recommend more tags. Semantic tag recommendation systems in the context of a semantic desktop are explored in [1]. Clustering to make real-time tag recommendation is developed in [16].

3 Tag Recommendation

Here we first provide a model of folksonomies, then review several common recommendation techniques which we employ in our evaluation. A folksonomy can be described as a four-tuple $D = \langle U, R, T, A \rangle$, where, U is a set of users; R is a set of resources; T is a set of tags; and A is a set of annotations, represented as user-tag-resource triples: $A \subseteq \{\langle u, r, t\rangle : u \in U, r \in R, t \in T\}$. A folksonomy can, therefore, be viewed as a tripartite hyper-graph [11] with users, tags, and resources represented as nodes and the annotations represented as hyper-edges connecting a user, a tag and a resource.

Aggregate projections of the data can be constructed, reducing the dimensionality but sacrificing information [14]. The relation between resources and tags, RT, can be formulated such that each entry, $RT(r, t)$, is the weight associated with the resource, r, and the tag, t. This weight may be binary, merely showing that one or more users have applied that tag to the resource. In this work we assume $RT(r, t)$ to be the number of users that have applied t to the r: $RT_{tf}(r, t) = |\{a = \langle u, r, t\rangle \in A : u \in U\}|$. Analogous two-dimensional projections can be constructed for UT in which the weights correspond to users and tags, and UR in which the weights correspond to users and resources.

Many authors have attempted to exploit the data model for recommendation in folksonomies. In traditional recommendation algorithms the input is often a user, u, and the output is a set of items, I. Tag recommendation differs in that the input is both a user and a resource. The output remains a set of items, in this case a set of recommended tags, T_r. Given a user-resource pair, the recommendation set is constructed by calculating a weight for each tag, $w(u, r, t)$, and recommending the top n tags.

3.1 Popularity Based Approaches

We consider two popularity based models which rely on the frequency a tag is used. **PopRes** ignores the user and relies on the popularity of a tag within the context of a particular resource. We define the resource based popularity measure as:

$$w(u, r, t) = \frac{|\{a = \langle u, r, t\rangle \in A : u \in U\}|}{|\{a = \langle u, r, t\rangle \in A : u \in U, t \in T\}|} \tag{1}$$

PopUser, on the other hand, ignores the resource and focuses on the frequency of a tag within the user profile. We define the user based popularity measure as:

$$w(u, r, t) = \frac{|\{a = \langle u, r, t\rangle \in A : r \in R\}|}{|\{a = \langle u, r, t\rangle \in A : r \in R, t \in T\}|} \tag{2}$$

Popularity based recommenders require little online computation. Models are built offline and can be incrementally updated. However both these models focus on a single channel of the folksonomy and may not incorporate otherwise relevant information into the recommendation.

3.2 Item-Based Collaborative Filtering

KNN_RT models resources as a vector over the tag space. As before the weights of the vectors may be calculated through a variety of means. Given a resource and a tag, we define the weight as the entry of the two dimensional projection, $RT(r, t)$, the number of times r has been tagged with t.

When a user selects a resource to annotate, the similarity between it and every resource in the user profile is calculated. A neighborhood of the k most similar resources, S, is thus constructed. We then define the item-based collaborative filtering measure as:

$$w(u, r, t) = \frac{\sum_s^S sim(s, r) * d(u, s, t)}{k} \tag{3}$$

where $d(u, s, t)$ is 1 if the user has applied t to s and 0 otherwise. Like *popUser*, this recommender focuses strongly on the user's tagging practice. However this recommender includes an additional informational channel, identifying resources in the user profile that are similar to the query resource. This technique therefore includes resource-to-resource information.

If the system waits to compute the similarity between resources until query time, this recommender will also scale well to larger datasets so long as user profiles remain small. Alternatively similarities between resources can be computed offline. Consequently the computation at query time is dramatically reduced and the algorithm becomes viable for large collaborative tagging implementations.

3.3 Folkrank

Folkrank was proposed in [6]. It computes a Pagerank vector from the tripartite graph of the folksonomy. This graph is generated by regarding $U \cup R \cup T$ as the set of vertices. Edges are defined by the three two-dimensional projections of the hypergraph, RT, UR and UT.

If we regard the adjacency matrix of this graph, W, (normalized to be column-stochastic), a damping factor, d, and a preference vector, p, then we iteratively compute the Pagerank vector, w, in the usual manner: $w = dAw+(1-d)p$.

However due to the symmetry inherent in the graph, this basic Pagerank may focus too heavily on the most popular elements. The Folkrank vector is taken as a difference between two computations of Pagerank: one with and one without a preference vector. Tag recommendations are generated by biasing the preference vector towards the query user and resource [7]. These elements are given a substantial weight while all other elements have uniformly small weights.

We include this method as a benchmark as it has demonstrated to be an effective method of generating tag recommendations. However, it imposes steep computational costs.

4 Informational Channels of Folksonomies

The model of a folksonomy suggests several informational channels which may be exploited by data mining applications such as tag recommenders. The relation between users, resources and tags generate a complex network of interrelated items as shown in Figure 1.

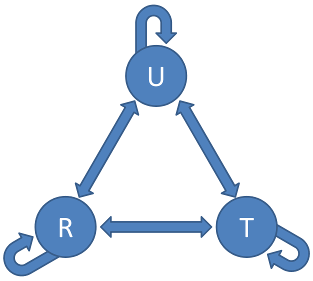


Fig. 1. Informational channels of a folksonomy.

The channel between resources and tags reveals a highly descriptive model of the resources. The accumulation of many users' opinions (often numbered in the thousands or millions) results in a richness which taxonomies are unable to approximate. Conversely the tags themselves are characterized by the resources to which they have been assigned.

As users annotate resource with tags they define their interests in as much as they describe a resource. The user-tag channel therefore reveals the users interests and provides opportunities for data mining algorithms to offer a high degree of personalization. Likewise a user may be defined by the resources which he has annotated as in the user-resource channel.

These primary channels can be used to produce secondary informational channels. The user-user channel can be constructed by modeling users as a vector of tags or as a vector of resources and applying a similarity measure such as cosine similarity. Many variations exist. However the result reveals a network of users that can be explored directly or incorporated into further data mining approaches. The resource-resource and tag-tag channels provide similar utility, presenting navigational opportunities for users to explore similar resources or tags.

5 A Multi-Channeled Tag Recommender

The most successful tag recommenders to date have included multiple informational channels. Folkrank explicitly includes the user-resource, user-tag and resource-tag channels in the graph model. Moreover since the algorithm calculates the Pagerank vector of the graph it implicitly includes the secondary channels of the folksonomy. The success Folkrank has achieved is due to its ability to incorporate multiple informational channels into a single tag recommender.

However the success it has achieved is blunted by the computational effort required to produce a recommendation; a new Pagerank vector is computed for each query.

Here we construct a hybrid recommender. The constituent parts by themselves perform poorly when compared to Folkrank. However, when aggregated into a single recommender they achieve a synergy which exploits several channels of the folksonomy while retaining their modest computational needs.

Our model includes *PopRes*, *PopUser* and *KNN_RT*. We employ a weighted approach to combine the recommenders. First in order to ensure that weight assignments are on the same scale for each recommendation approach, we normalize the weights given to the tags by $w(u, r, t)$ to 1 producing $w'(u, r, t)$. We then combine the weights in a linear combination:

$$w(u, r, t) = \alpha w'_{PopRes}(u, r, t) + \beta w'_{PopUser}(u, r, t) + \gamma w'_{KNN\_RT}(u, r, t) \qquad (4)$$

such that weights $\alpha + \beta + \gamma = 1$ and all values are positive. If α is set near 1 then hybrid would rely mostly on *PopRes*.

Tags promoted by *PopRes* will have a strong relevance to the resource, while tags promoted by *PopUser* will include tags in the user's profile. *PopRes* alone will ignore personal tags that the user often users. *PopUser*, on the other hand, will ignore tags related to the context of the query resource. Together these recommenders can include both aspects in the recommendation set. Moreover by including *KNN_RT* tags which the user has applied to resources similar to the query resource are promoted.

PopRes explicitly includes the resource-tag information. *PopUser*, on the other hand, includes user-tag information. Both these models are based on popularity and are single-minded in their approach ignoring all data except the informational channel to which they are employed. We use *KNN_RT* to introduce more subtlety into the hybrid. It focuses heavily on the user-tag channel, but gives more weight to tags that have been applied to similar resources. Hence it also includes resource-tag information. Moreover by focusing exclusively on resources in the user profile it includes the user-resource channel. Finally, *KNN_RT* includes resource-resource information when it calculates the neighborhood of similar resources.

This hybrid does not include user-user information or tag-tag information. Additional recommenders could be included to cover these informational channels. However, we have built this hybrid with the goals of speed and simplicity. The two popularity based approaches are among the fastest and simplest recommendation algorithms. The item-based collaborative filtering recommender is used to tie together these approaches incorporating similarities among resources into the model while retaining its speed.

	Complete	PostCore(2)
Users	3,617	253,615
URLs	235,328	41,268
BibTeXs	143,050	22,852
Tags	93,756	1,185
Tag Assignments	1,401,104	14,443
Bookmark Posts	263,004	7,946
BibTeX Posts	158,924	13,276

Table 1. Bibsonomy datasets.

6 Experimental Evaluation

In this section we describe the dataset used for experimentation. We then describe our experimental methodology and metrics. Finally we discuss the results of our experiments.

6.1 Data Set

The dataset was provided by Bibsonomy[4] for the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) 2009 Challenge. BibSonomy was originally launched as a collaborative tagging application allowing users to organize and share scholarly references. It has since expanded its scope allowing users to annotate URLs.

The data includes all public bookmarks and publication posts of BibSonomy until 2009-01-01. The data was cleaned by removing all characters which are neither numbers nor letters from tags. Additionally the system tags *imported*, *public*, *systemimported*, *nn* and *systemunfiled* where removed.

Task 1 for the 2009 Challenge utilizes the complete dataset. Task 2 however focuses on the post-core at level 2 geared toward graph based approaches. For the post-core all users, tags, and resources which appear in only one post were removed. This process was repeated until convergence and produced a core in which each user, tag, and resource occurs in at least two posts. Reducing a dataset to its core was first proposed in [3]. In [6] it was adapted for folksonomies. The experiments for this work rely on post-core at level 2.

6.2 Experimental Methodologies

We employ the leave one post out methodology as described in [7]. One post from each user was placed in the testing set consisting of a user, u, a resource, r, and all the tags the user has applied to that resource. These tags, T_h, are analogous to the holdout set commonly used in Information Retrieval evaluation. The remaining posts are used to generate the recommendation models.

[4] www.bibsonomy.org

The tag recommendation algorithms accepts the user-resource pair and returns an ordered set of recommended tags, T_r. From the holdout set and recommendation set utility metrics were calculated. For each metric the average value was calculated across all test cases.

6.3 Experimental Metrics

Recall is a common metric of recommendation algorithms that measures coverage. It measures the percentage of items in the holdout set, T_h, that appear in the recommendation set T_r. It is defined as:

$$r = (|T_h \cap T_r|)/|T_h| \tag{5}$$

Precision is another common metric that measures specificity and is defined as:

$$p = (|T_h \cap T_r|)/|T_r| \tag{6}$$

In order to conform to the evaluation methods of the ECML-PKDD 2009 Challenge, we use the F1-Measure common in Information Retrieval to evaluate the recommendations. We compute for each post the recall and precision for a recommendation set of five tags. Then we average precision and recall over all posts in the test data and use the resulting precision and recall to compute the F1-Measure as:

$$f1 = (2 * p * r)/(p + r) \tag{7}$$

6.4 Experimental Results

Our approach required that several variables be tuned. For *KNN_RT*, after extensive experimentation of k in increments of 1 we set k equal to 15. We observed that as k increased from 0 to 15 recall and precision both increased rapidly until it suffers from diminishing returns.

We evaluated the weights α, β and γ in .05 increments attempting every possible combination. Best results were found when $\alpha = 0.35$, $\beta = 0.15$ and$\gamma = 0.50$. As such *KNN_RT* accounts for 50% of the model, *PopRes* acounts for 35% and $PopUser$ acounts for 15%.

KNN_RT identifies resources in the user profile most similar to the query resource and promotes the tags applied to these resources. This approach is most effective when the user has generated a large user profile. Since users often employ tags as an organizational tool they often reuse tags. Hence the success of *KNN_RT* stems from its ability to identify which previously used tags are most appropiate given the context of the query resource.

PopRes, on the other hand, ignores the user profile and concentrates on the popularity of a tag given the query resource. When the tags provided by *KNN_RT* are insufficient, perhaps because the user has yet to build a deep user profile or

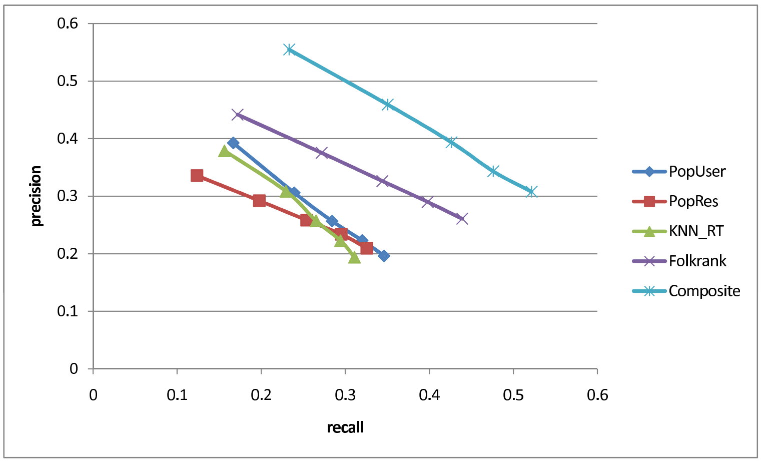


Fig. 2. Evaluation of recommendation techniques: recall vs. precision.

is tagging a resource dissimilar to items in the profile, *PopRes* is able to provide relevant suggestions.

Finally *PopUser* promotes tags in the user profile regardless of the similarity to the query resource. It may promote idiosyncratic, subjective or organizational tags that do not necessarily relate to the context of the query resource but are often applied by the user.

Our evaluation of the composite recommenders in Figures 2 and 3 reveals that *PopRes*, *PopUser* and *KNN_RT* achieve only modest success when used alone. However when combined together as a hybrid recommender the three are able to cover multiple informational channels and produce a synergy allowing the hybrid to produce superior results.

Not only is the hybrid recommender able to outperform the baseline recommenders it is also able to outperform Folkrank, a highly effective tag recommender. Moreover the hybrid retains the computationally efficiency of its parts making it suitable for deployment in large real work collaborative filtering applications.

7 Conclusions and Future Work

In this paper we have introduced the idea of informational channels in folksonomies and have proposed a fast yet effective tag recommender composed of three separate algorithms. The constituent recommenders were chosen for their speed and simplicity as well as their ability to cover complimentary informational channels. We have demonstrated that these recommenders while performing poorly alone, create a synergy when combined in a linear combination. The hybrid recommender is able to surpass the effective graph based approaches while

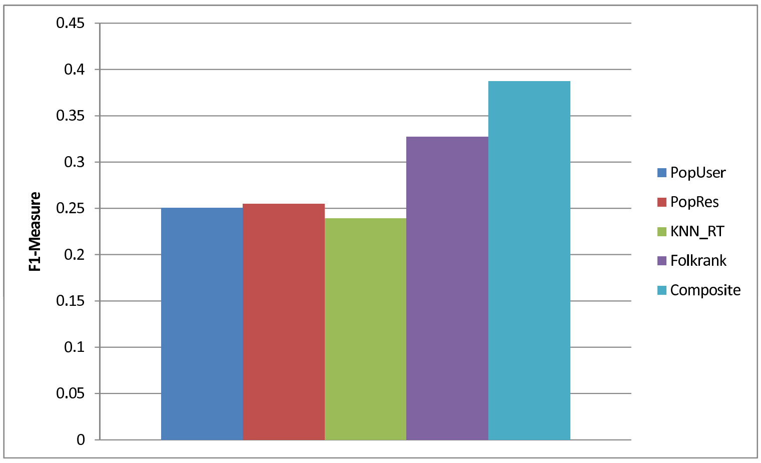


Fig. 3. Evaluation of recommendation techniques: F1-measure.

retaining the efficiency of its parts. Future work will include an examination of alternative hybrid recommenders and present work on other datasets.

8 Acknowledgments

This work was supported in part by the National Science Foundation Cyber Trust program under Grant IIS-0430303 and a grant from the Department of Education, Graduate Assistance in the Area of National Need, P200A070536.

References

1. B. Adrian, L. Sauermann, and T. Roth-Berghofer. Contag: A semantic tag recommendation system. In T. Pellegrini and S. Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 297–304. JUCS, 2007.
2. P. Basile, D. Gendarmi, F. Lanubile, and G. Semeraro. Recommending smart tags in a social bookmarking system. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, pages 22–29, 2007.
3. V. Batagelj and M. Zaveršnik. Generalized cores. *Arxiv preprint cs/0202039*, 2002.
4. S. A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198, 2006.
5. P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, New York, NY, USA, 2008. ACM.
6. A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. *Lecture Notes in Computer Science*, 4011:411, 2006.
7. R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag Recommendations in Folksonomies. *LECTURE NOTES IN COMPUTER SCIENCE*, 4702:506, 2007.

8. M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.
9. G. Macgregor and E. McCulloch. Collaborative tagging as a knowledge organisation and resource discovery tool. *Library Review*, 55(5):291–300, 2006.
10. A. Mathes. Folksonomies-Cooperative Classification and Communication Through Shared Metadata. *Computer Mediated Communication, (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December*, 2004.
11. P. Mika. Ontologies are us: A unified model of social networks and semantics. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(1):5–15, 2007.
12. R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, and H. Kato. Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation. In *Advances in Communication Systems and Electrical Engineering*, pages 309–318. Springerlink, 2008.
13. R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and Y. Inagaki. Reasonable tag-based collaborative filtering for social tagging systems. In *WICOW '08: Proceeding of the 2nd ACM workshop on Information credibility on the web*, pages 11–18, New York, NY, USA, 2008. ACM.
14. C. Schmitz, A. Hotho, R. Jaschke, and G. Stumme. Mining association rules in folksonomies. In *Proc. IFCS 2006 Conference*, pages 261–270. Springer, 2006.
15. B. Sigurbjörnsson and R. van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.
16. Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522, New York, NY, USA, 2008. ACM.
17. K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999, New York, NY, USA, 2008. ACM.
18. Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland, May*, 2006.

A modified and fast Perceptron learning rule and its use for Tag Recommendations in Social Bookmarking Systems

Anestis Gkanogiannis and Theodore Kalamboukis

Department of Informatics
Athens University of Economics and Business, Athens, Greece
utumno@aueb.gr tzk@aueb.gr

Abstract. A modified and fast to converge Perceptron learning rule algorithm is proposed as a general classification algorithm for linearly separable data. The strategy of the algorithm takes advantage of training errors to successively refine an initial Perceptron Classifier.

Original Perceptron learning rule uses training errors along with a parameter α (learning rate parameter that has to be determined) to define a better classifier. The proposed modification does not need such a parameter (in fact it is automatically determined during execution of the algorithm).

Experimental evaluation of the proposed algorithm on standard text classification collections, show that results compared favorably to those from state of the art algorithms such as SVMs. Experiments also show a significant improvement of the convergence rate of the proposed Perceptron algorithm compared to the original one.

Seeing the problem of this year's Discovery Challenge (Tag Recommendation), as an automatic text classification problem, where tags play the role of categories and posts play the role of text documents, we applied the proposed algorithm on the datasets for Task 2. In this paper we briefly present the proposed algorithm and its experimental results when applied on the Challenge's data.

1 Introduction

Text categorization is the process of making binary decisions about related or non-related documents to a given set of predefined thematic topics or categories. This task is an important component in many information management organizations. In our participation on the ECML/PKDD challenge 2009, we treat Task 2 as a standard text classification problem and try to solve it using a machine learning, supervised, automatic classification method.

The rest of the paper is organized as follows. Section 2 provides a description of the algorithm that we used. Section 3 briefly present the tasks of this year's Challenge. Section 4 presents the experimental setup, data processing and results and finally in section 5 we conclude on the results.

2 The Learning Algorithm

The algorithm that we used is an evolution of the algorithm that appeared in [1] as a text classification algorithm and then a revised version in [2] won last year's ECML PKDD Discovery Challenge on Spam Detection. The proposed algorithm is a binary linear classifier and it combines a centroid with a batch perceptron classifier and a modified perceptron learning rule that does not need any parameter estimation. Details on this modified algorithm, its experimental evaluation, theoretical investigation etc, have already submitted and are under review for publication at the time this paper was written. In the following paragraphs we will briefly describe this method that we used for solving the problem of ECML PKDD 2009 Discovery Challenge, Task 2.

2.1 Linear Classifiers

Linear Classifiers is a family of classifiers whose trained model is a linear combination of features. In another perspective linear classifiers train a model which is a hyperplane in a high dimensional feature space. In this space each instance, either of the train set or an unseen, is a point. The goal of a linear classifier is then to find such a hyperplane that splits the space into two subspaces, where one contains all the points of the positive class and the other contains all the points of the negative class.

Assuming that feature space is of n dimensions, each instance x_i will be represented by an n dimensions vector

$$\overrightarrow{x}_i = (w_{i1}, w_{i2}, \cdots, w_{in}) \tag{1}$$

where w_{ik} is a real value of the kth feature for instance x_i.

Apart of each vector representation $\overrightarrow{x}_i$, each instance x_i may bears information about being a member of a class or not. For example a document is known to be spam or an image is known that shows a benign tumor. This information can be coded using a variable y_i for each instance x_i which takes values as:

$$y_i = \begin{cases} 1 & \text{if } x_i \in C_+ \\ -1 & \text{if } x_i \in C_- \end{cases} \tag{2}$$

That is $y_i = 1$ when x_i is member of the positive class C_+ and $y_i = -1$ when it is member of the negative class C_-. So each instance x_i is represented by a tuple $(\overrightarrow{x}_i, y_i)$. A training set Tr would be

$$Tr = \{(\overrightarrow{x}_1, y_1), (\overrightarrow{x}_2, y_2), \cdots, (\overrightarrow{x}_m, y_m)\} \tag{3}$$

A linear classifier then is defined by a model $\left\langle \overrightarrow{W}, b \right\rangle$ where $\overrightarrow{W}$ is a vector in the same n-dimensional space and b is a scalar bias (threshold) value. This model defines a hyperplane h

$$h : \overrightarrow{W} \cdot x + b = 0 \tag{4}$$

This is the equation of a hyperplane h in the n-dimensional space. This hyperplane is of $n-1$ dimensions. $\overrightarrow{W}$ is a linear combination of n features (dimensions). Hyperplane h splits space into two subspaces, the one where for every vector $\overrightarrow{x}_i : \overrightarrow{W} \cdot \overrightarrow{x}_i + b > 0$ and the other where $\overrightarrow{W} \cdot \overrightarrow{x}_i + b < 0$. Every vector for which $\overrightarrow{W} \cdot \overrightarrow{x}_i + b = 0$ lies on hyperplane h. The objective of each linear classifier is to define such $h : \left\langle \overrightarrow{W}, b \right\rangle$. Different linear classifiers have different ways to define model vector $\overrightarrow{W}$ and bias b.

2.2 Perceptron

Perceptron is a flavor of Linear Classifiers. It starts with an initial model and iteratively refines this model using the classifications errors during training. It is the elementary particle of neural networks and it has been investigated and studied since the 1950s [3]. It has been shown that when trained on a linearly separable set of instances, it converges (it finds a separating hyperplane) in a finite number of steps [4] (which depends on the geometric characteristics of the instances on their feature space).

The Perceptron is a Linear Binary Classifier that maps its input $\overrightarrow{x}$ (a real-valued vector) to an output value $f(\overrightarrow{x})$ (a single binary value) as:

$$f(\overrightarrow{x}) = \begin{cases} 1 & \text{if } \overrightarrow{W} \cdot \overrightarrow{x} + b > 0 \\ -1 & \text{else} \end{cases} \tag{5}$$

where $\overrightarrow{W}$ is a vector of real-valued weights and $\overrightarrow{W} \cdot \overrightarrow{x}$ is the dot product (which computes a weighted sum). b is the bias, a constant term that does not depend on any input value. The value of $f(\overrightarrow{x})$ (1 or -1) is used to classify instance x as either a positive or a negative instance, in the case of a binary classification problem. The bias b can be thought of as offsetting the activation function, or giving the output neuron a "base" level of activity. If b is negative, then the weighted combination of inputs must produce a positive value greater than $-b$ in order to push the classifier neuron over the 0 threshold. Spatially, the bias alters the position (though not the orientation) of the decision boundary (separating hyperplane h).

We can always assume for convenience that the bias term b is zero. This is not a restriction since an extra dimension $n+1$ can be added to all the input vectors $\overrightarrow{x}_i$ with $\overrightarrow{x}_i(n+1) = 1$, in which case $\overrightarrow{W}(n+1)$ replaces the bias term.

Learning is modeled as the weight vector $\overrightarrow{W}$ being updated for multiple iterations over all training instances. Let

$$Tr = \{(\overrightarrow{x}_1, y_1), (\overrightarrow{x}_2, y_2), \cdots, (\overrightarrow{x}_m, y_m)\}$$

denote a training set of m training examples (instances). At each iteration k the weight vector is updated as follows. For each $(\overrightarrow{x}_i, y_i)$ pair in Tr

$$\overrightarrow{W}^{(k)} = \overrightarrow{W}^{(k-1)} + \frac{\alpha^{(k-1)}}{2} \left(y_i - f^{(k-1)}\left(\overrightarrow{x}_i\right) \right) \overrightarrow{x}_i \tag{6}$$

where α is a constant real value in the range $0 < \alpha \leq 1$ and is called the learning rate. Note that equation 6 means that a change in the weight vector $\overrightarrow{W}$ will only take place for a given training example $(\overrightarrow{x}_i, y_i)$ if its output $f(\overrightarrow{x}_i)$ is different from the desired output y_i. In other words the weight vector will change only in the case where the model has made an error. The initialization of $\overrightarrow{W}$ is usually performed simply by setting $\overrightarrow{W}^{(0)} = 0$.

The training set Tr is said to be linearly separable if there exists a positive constant γ and a weight vector $\overrightarrow{W}$ such that

$$y_i \left(\overrightarrow{W} \cdot \overrightarrow{x}_i + b\right) > \gamma, \forall (\overrightarrow{x}_i, y_i) \in Tr \tag{7}$$

Novikoff [4] proved that the perceptron algorithm converges after a finite number of iterations k if the train data set is linearly separable. The number of mistakes (iterations) is bounded then by

$$k \leq \left(\frac{2R}{\gamma}\right)^2 \tag{8}$$

where $R = \max\{||\overrightarrow{x}_i||\}$ is the maximum norm of an input train vector.

2.3 Batch Perceptron

Equation 6 defines a single sample fixed increment perceptron learning rule. It is called fixed increment because parameter α is constant throughout training. In the case where this parameter changes at each iteration, we say that it is a variable increment perceptron. It is also called single sample because this rule applies at each instance x_i which was misclassified during iteration k. In other words, at iteration k each $(\overrightarrow{x}_i, y_i) \in Tr$ is presented to model $\overrightarrow{W}^{(k-1)}$ and if it is misclassified by it $(f^{(k-1)}(\overrightarrow{x}_i) \neq y_i)$ then this single instance $\overrightarrow{x}_i$ is used (along with parameter $\alpha^{(k-1)}$) to alter $\overrightarrow{W}^{(k-1)}$ into $\overrightarrow{W}^{(k)}$.

A modification of this perceptron can be made defining a set of instances $Err \subset Tr$

$$Err = \{(\overrightarrow{x}_i, y_i)\}, f^{(k-1)}(\overrightarrow{x}_i) \neq y_i \tag{9}$$

that contains all the misclassified examples at iteration k and then modifying weight vector as:

$$\overrightarrow{W}^{(k)} = \overrightarrow{W}^{(k-1)} + \alpha^{(k-1)} \sum_{(\overrightarrow{x}_i, y_i) \in Err} y_i \overrightarrow{x}_i \tag{10}$$

In the case where bias value is not incorporated into example and weight vectors (via an additional n+1 dimension), then bias value is modified as:

$$b^{(k)} = b^{(k-1)} + \alpha^{(k-1)} \sum_{(\overrightarrow{x}_i, y_i) \in Err} y_i \tag{11}$$

Equations 10 and 11 are called a Batch Perceptron learning rule and as the single sample perceptron, parameter $\alpha^{(k-1)}$ can be constant (fixed increment) or varying at each iteration (variable increment).

2.4 Centroid Classifier

A Centroid classifier is a simple linear classifier, that will help us understand the notion behind our modification presented in the next Section. In the simple binary case there are two classes, the positive and the negative one. We define set C_+ and C_- containing instances from the positive and respectively the negative class. We call Centroid of the positive class and respectively the Centroid of the negative class as

$$\overrightarrow{C}_+ = \frac{1}{|C_+|} \sum_{\overrightarrow{x_i} \in C_+} \overrightarrow{x_i} \tag{12}$$

$$\overrightarrow{C}_- = \frac{1}{|C_-|} \sum_{\overrightarrow{x_i} \in C_-} \overrightarrow{x_i} \tag{13}$$

We then define a linear classifier as

$$h : \overrightarrow{W} \cdot \overrightarrow{x} + b = 0 \tag{14}$$

where

$$\overrightarrow{W} = \overrightarrow{C}_+ - \overrightarrow{C}_- \tag{15}$$

and bias value b is defined by some technique we discuss in the following paragraphs.

Figure 1 illustrates a simple case of a centroid classifier in a 2-dimensional space. Sets C_+ of the positive class and C_- of the negative class are shown along with their centroid vectors $\overrightarrow{C}_+$ and $\overrightarrow{C}_-$ respectively. We note that in this simple example, these two classes are linearly separable and therefore it is possible to find a value for bias b such that h is a perfect separating hyperplane.

A method for finding such a value is Scut [5], where we iteratively choose values for bias b and then keep the one that lead to the best classifier (as measured by some evaluation measurement). Bias takes values as

$$b_i = \overrightarrow{W} \cdot \overrightarrow{x}_i, \forall \overrightarrow{x}_i \in Tr \tag{16}$$

and then an evaluation measure (for example the F_1 measure) is computed for classifier $h : \overrightarrow{W} \cdot \overrightarrow{x} + b_i = 0$. Finally as bias value is chosen the one that gave the maximum evaluation measure. It is clear that the instance x_i that corresponds to the chosen b_i lies on hyperplane h. In the shown 2-dimensional example of Figure 1 this instance is marked by point $\overrightarrow{x}_{Scut}$.

This simple algorithm has previously investigated and
methods have been proposed for altering initial centroids or weights in order to achieve a better classifier [6–8].

In the next subsection we present how ideas from Centroid Classifier and Perceptron are combined to our modified version of Perceptron.

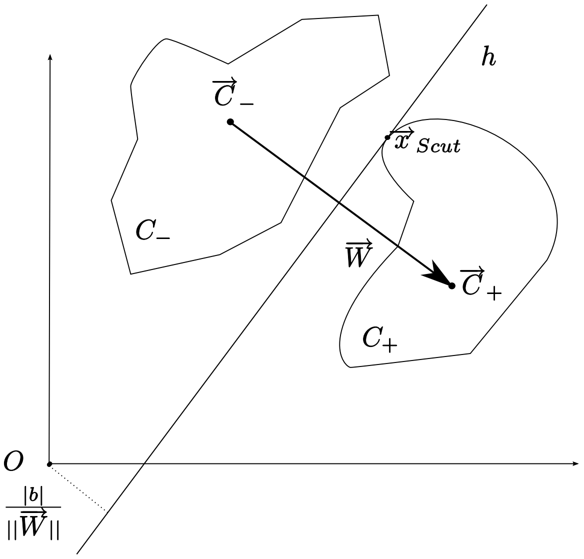


Fig. 1. A simple Centroid Classifier in 2 dimensions. The positive class C_+ is linearly separable from the negative one C_-.

2.5 The proposed modification to Perceptron

Centroid Classifier of the previous subsection can be seen as a perceptron with initial weight vector $\overrightarrow{W}^{(0)} = \overrightarrow{C}_+ - \overrightarrow{C}_-$, bias value b as defined by an Scut method and no other training adjustments at all. The case shown in Figure 1 is an ideal case for a Centroid Classifier, meaning that it is possible to find a value for b resulting to a perfect separating hyperplane $h : \overrightarrow{W} \cdot \overrightarrow{x} + b = 0$.

This is not however true in all cases. Figure 2 shows such a case where finding a perfect separating hyperplane is not possible for a simple Centroid Classifier. Dark regions contains misclassified instances that cannot correctly classified. A Simple Sample or a Batch Perceptron would use these errors to modify the weight vector $\overrightarrow{W}$.

If we define sets FP and FN as:

$$FP = \{(\overrightarrow{x}_i, y_i)\} \forall x_i \in C_-, f(\overrightarrow{x}_i) \neq y_i \quad (17)$$

$$FN = \{(\overrightarrow{x}_i, y_i)\} \forall x_i \in C_+, f(\overrightarrow{x}_i) \neq y_i \quad (18)$$

in other words set FP contains negative instances that were misclassified as positive (False Positive), whereas set FN contains positive instances that were misclassified as negative (False Negative). A Batch Perceptron then using mis-

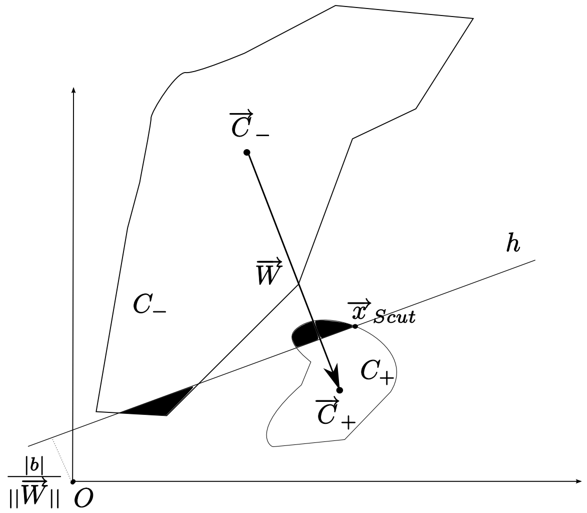


Fig. 2. No perfect separating hyperplane exists for this Centroid Classifier. Dark regions are misclassified.

classified instances modifies weight vector as Equation 10 or equivalently as:

$$\overrightarrow{W}^{(k+1)} = \overrightarrow{W}^{(k)} + \alpha^{(k)} \left(\sum_{\overrightarrow{x}_i \in FN^{(k)}} \overrightarrow{x}_i - \sum_{\overrightarrow{x}_i \in FP^{(k)}} \overrightarrow{x}_i \right) \tag{19}$$

However there is a parameter α, either constant or variable that needs to be estimated. This learning rate parameter is strongly related to the field on which perceptron learning is applied and train data itself. A way to estimate it is using a validation set of instances and selecting a value for α that leads to maximum performance. But this operation must be repeated whenever field of operation or data is switched and costs very much in terms of time.

Another approach is to use a fixed value for the learning rate like $\alpha = 1$ or $\alpha = 0.5$ for example, without attempting to find a optimal value. However this could result to very unwanted effects because learning rate is too small or too large for the specific field of operation and training instances.

The key idea of our approach is illustrated in Figure 3 where we concentrate on the misclassified regions. Positive class and a portion of negative class are shown. Initial weight vector $\overrightarrow{W}^{(0)}$ and hyperplane $h^{(0)}$ are defined by a simple Centroid Classifier. The idea is, at the next iteration 1, to modify weight vector and bias into $\overrightarrow{W}^{(1)}$ and $b^{(1)}$ such that the resulting hyperplane $h^{(1)}$ passes through the points defined by centroid vectors of the misclassified regions FP and FN.

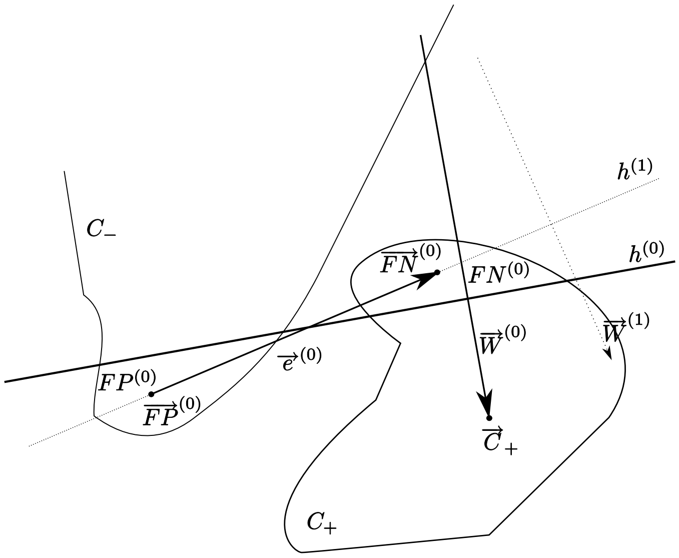

$h^{(1)}$
C_-
$\overrightarrow{FN}^{(0)}$
$h^{(0)}$
$FN^{(0)}$
$\overrightarrow{W}^{(1)}$
$\overrightarrow{W}^{(0)}$
$\overrightarrow{e}^{(0)}$
$FP^{(0)}$
$\overrightarrow{FP}^{(0)}$
$\overrightarrow{C}_+$
C_+

Fig. 3. The proposed modification to batch perceptron.

We define these misclassified centroids at each iteration as

$$\overrightarrow{FP}^{(k)} = \frac{1}{|FP^{(k)}|} \sum_{\overrightarrow{x_i} \in FP^{(k)}} \overrightarrow{x_i} \tag{20}$$

$$\overrightarrow{FN}^{(k)} = \frac{1}{|FN^{(k)}|} \sum_{\overrightarrow{x_i} \in FN^{(k)}} \overrightarrow{x_i} \tag{21}$$

where sets FP and FN are defined in Equations 17 and 18. We then define the error vector at each iteration as

$$\overrightarrow{e}^{(k)} = \overrightarrow{FN}^{(k)} - \overrightarrow{FP}^{(k)} \tag{22}$$

Batch Perceptron learning rule of Equation 19 is then modified to:

$$\overrightarrow{W}^{(k+1)} = \overrightarrow{W}^{(k)} + \alpha'^{(k)} \overrightarrow{e}^{(k)} \tag{23}$$

We can easily compute the value of this modified learning rate $\alpha'^{(k)}$ if we note that misclassified centroids $\overrightarrow{FN}^{(k)}$ and $\overrightarrow{FP}^{(k)}$ lie by construction on the new hyperplane $h^{(k+1)}$. As a result error vector $\overrightarrow{e}^{(k)}$ is vertical to the new normal vector $\overrightarrow{W}^{(k+1)}$. So

$$\overrightarrow{W}^{(k+1)} \cdot \overrightarrow{e}^{(k)} = 0$$

$$\left(\overrightarrow{W}^{(k)} + \alpha'^{(k)}\overrightarrow{e}^{(k)}\right) \cdot \overrightarrow{e}^{(k)} = 0$$
$$\overrightarrow{W}^{(k)} \cdot \overrightarrow{e}^{(k)} + \alpha'^{(k)}||\overrightarrow{e}^{(k)}||^2 = 0$$
$$\alpha'^{(k)} = -\frac{\overrightarrow{W}^{(k)} \cdot \overrightarrow{e}^{(k)}}{||\overrightarrow{e}^{(k)}||^2}$$

And then the modified learning rule of Equation 23 is

$$\overrightarrow{W}^{(k+1)} = \overrightarrow{W}^{(k)} - \frac{\overrightarrow{W}^{(k)} \cdot \overrightarrow{e}^{(k)}}{||\overrightarrow{e}^{(k)}||^2}\overrightarrow{e}^{(k)} \tag{24}$$

This is the normal vector defining the direction of the next hyperplane $h^{(k+1)}$. The actual position of it is determined by the new bias value which is easily computed (bringing in mind that misclassified centroids lie on the new hyperplane):

$$b^{(k+1)} = -\overrightarrow{W}^{(k+1)} \cdot \overrightarrow{FP}^{(k)} = -\overrightarrow{W}^{(k+1)} \cdot \overrightarrow{FN}^{(k)} \tag{25}$$

Equations 24 and 25 define the new hyperplane

$$h^{(k+1)} : \overrightarrow{W}^{(k+1)} \cdot \overrightarrow{x} + b^{(k+1)} = 0$$

3 Task Description

As last year's, this year's ECML PKDD Discovery Challenge deals with the well known social bookmarking system called Bibsonomy [1]. In such systems, users can share with everyone links to web pages or scientific publications. The former are called bookmark posts, where the later are called bibtex posts. Apart from posting the link to the page or the publication, users can assign tags (labels) to their posts. Users are free to choose their own tags or the system can assist them by suggesting them the appropriate tags.

This year's Discovery Challenge problem is about generating methods that would assist users of social bookmarking systems by recommending them tags for their posts. There are two distinct task for this problem. Task 1 is about recommending tags to posts over an unknown set of tags. That means that the methods developed for Task 1 must be able to suggest tags that are unknown (in other words suggest new tags). Task 2, on the other hand, is about recommending tags that have been already known to be ones. [2]

3.1 Data Description

Data provided for these tasks was extracted from Bibsonomy databases. Two datasets where provided, one for training participant's methods, and the other

[1] http://www.bibsonomy.org

[2] More details about tasks can be found on Challenge's site at http://www.kde.cs.uni-kassel.de/ws/dc09/#tasks

for evaluating their performance. Both of them where provided as a set of 3 files (tas, bookmark, bibtex). Files bookmark and bibtex contain textual data of the corresponding posts. File tas contains which user assigned which tags to which bookmark or bibtex resource. Each triplet (user,tags,resource) defines a post. Train and test files where of the same tabular format, except test tas file which of course did not contain tag information, as this was Challenge's goal. [3]

More details about preprocessing of the datasets will be given on the following section 4.

4 Experimental Setup and Results

Challenge's organizers had suggest that graph method would fit better to task 2, whereas content based method would fit to task 1. In our work we concentrated on task 2, and from this point on whenever we mention a task, we mean task 2. Although organizers suggested graph method for the task, we choose to use our modified perceptron rule for solving this problem. We made this decision because we wanted to test the performance and robustness of the proposed algorithm on a domain with a large category set. As we are going to present in our under review paper, we have evaluated the proposed algorithm on standard text classification datasets as well as on artificially generated (and linearly separable) datasets. Although feature spaces of these datasets are of tens or hundreds of thousands features, their categories sets are of few to at most a thousand categories. We wanted to investigate how this method is going to perform when both feature and category spaces are large.

So, Task 2 can be seen as a standard text classification problem, and the proposed algorithm as a machine learning, supervised, automatic classification method that applies on it. In this problem, tags (labels) that assigned on posts can be seen as categories. On the other hand, posts can be seen as text documents, where category labels (tags) are assigned on them.

4.1 Data Preprocessing

Viewing task 2 as a supervised text classification problem, implies that datasets must transformed to a vector space, where the proposed linear classifier can be used. For every post (user,tags,resource), we construct a text document and then transform it to the vector space.

We choose to discard user information from the posts, so the only textual information for each post came from the assigned tags and the resource. Furthermore for each bookmark post we kept *url*, *description* and *extended_description* fields. For each bibtex post we kept *journal*, *booktitle*, *url*, *description*, *bibtexAbstract*, *title* and *author* fields.

Fore every post, and using those field, we construct a text document. We then transform document dataset to a vector space. First tokenization of the

[3] More details about datasets can be found at http://www.kde.cs.uni-kassel.de/ws/dc09/dataset

text, then stop word removal, then stemming (using Porter's stemmer [9], then term and feature extraction and finally feature weighting using *tf*idf* statistics.

The following table 1 presents some statistics about categories (tags) and documents (posts) in the train and the test dataset.

	Train dataset	**Test dataset**
Number of Documents	64,120	778
Number of Categories	13,276	-

Table 1. Statistics for categories and documents in datasets

The following diagram 4 presents the distribution of the sizes of categories in the train dataset. Axis ***x*** denotes the number of categories that are of a certain size. Axis ***y*** denotes the number of documents that a certain sized category contains.

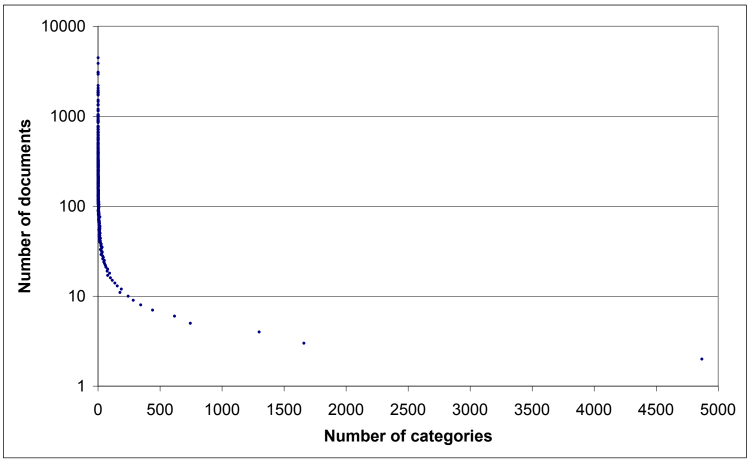


Fig. 4. Distribution of the sizes of categories in the train dataset

We note that categories sizes are small in general. In fact 10,500 out of 13,276 categories have at most 10 documents. The average size of categories is 1.97 (average posts per tag).

4.2 Experimental Setup and Train phase

After converting documents (posts) into vectors in a high-dimensional space, we can apply the proposed text classification method for solving the multilabel classification problem. Since the method trains a binary linear classifier, the problem must be transformed into binary classification. This is done by cracking the problem into multiple binary classification problems. So, at the end we have to solve $13,276$ binary classification problems.

The number of problems is quite large and therefore the used method must be as much fast as possible. After the train phase (which finishes after the reasonable time of 2 hours in a mainstream laptop), the final classification system consists of $13,276$ binary classifiers.

4.3 Test phase and Results

Test phase consists of presenting each document of the test dataset (778 in total) to every binary classifier resulted from training phase ($13,276$ in total). Each classifier decides whether the presented document (post) belongs or not to the corresponding category (tag). Time needed fore presenting all document to all classifiers on a mainstream laptop was about 10 minutes (that is about 0.8 seconds for a document to pass through all classifiers).

We produced 2 types of results. The ones that come from binary classification and the ones that come from ranking. During binary classification a document could be assigned or not into a category. Therefore a document, after been presented to every binary classifier, could be assigned to zero, one, or more categories (max is $13,276$ of course).

On the ranking mode, a classifier gives a score to each presented document (higher score mean higher confidence of the classifier that this document belongs to the corresponding category). Therefore at this mode, a document can be assigned to any number z of categories we select (simply by selecting the z categories which gave the higher scores).

We chose our submission to the Challenge, to contain results of the ranking mode (by selecting the 5 higher scored categories for each document).

After releasing the original tag assignments of the test dataset, our results of the ranking mode achieved a performance of $F_1 = 0.1008$. The results of the first mode (binary mode), that where never submitted, achieved a performance of $F_1 = 0.1622$. Of course, those results could not have been known prior releasing original test tas file, but we had a belief that the ranking mode (suggesting 5 tags for every post, instead of less or even zero) would had better results. Unfortunately this belief was false.

5 Concluding Remarks

In this paper we described the application of a modified version of the Perceptron learning rule on Task 2 of ECML PKDD Discovery Challenge 2009. This algorithm acts as a supervised machine learning, automatic text classification algorithm on the data of the task. Task 2 is transformed to a supervised text classification problem by treating users' posts ass text documents and assigned tags as thematic categories.

This algorithm has been prior tested on various text classification datasets and artificially generated linearly separable datasets, and it has shown a robust performance and efficiency. Compared with the original Batch Perceptron learning algorithm, it shows a significant improvement on the convergence rate.

Its fast training phase made it feasible to be used on Task 2 dataset, which consists of a large categories set (more than $13,000$ categories) and a linear classifier had to be trained for each category.

Although its results on Task 2 test dataset where not so well, we think that its fast training phase and fast evaluation (since it is just a dot product for each category-document tuple) allow for further investigation.

Acknowledgments

This paper is part of the 03ED316/8.3.1. research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (20% from the Greek Ministry of Development-General Secretariat of Research and Technology and 80% from E.U.-European Social Fund).

References

1. Gkanogiannis, A., Kalampoukis, T.: An algorithm for text categorization. In: 31st ACM International Conference on Research and Development in Information Retrieval SIGIR-2008. (2008) 869–870
2. Gkanogiannis, A., Kalamboukis, T.: A novel supervised learning algorithm and its use for spam detection in social bookmarking systems. In: ECML PKDD Discovery Challenge '08. (2008)
3. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review **65**(6) (November 1958) 386–408
4. Novikoff, A.B.: On convergence proofs for perceptrons. In: Proceedings of the Symposium on the Mathematical Theory of Automata. Volume 12. (1963) 615–622
5. Yang, Y.: A study on thresholding strategies for text categorization (2001)
6. Karypis, G., Shankar, S.: Weight adjustment schemes for a centroid based classifier (2000)
7. Harman, D.: Relevance feedback and other query modification techniques. (1992) 241–263
8. Buckley, C., Salton, G.: Optimization of relevance feedback weights. In: SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (1995) 351–357
9. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137

Discriminative Clustering for Content-Based Tag Recommendation in Social Bookmarking Systems

Malik Tahir Hassan[1], Asim Karim[1], Suresh Manandhar[2], and James Cussens[2]

[1]Dept. of Computer Science
LUMS School of Science and Engineering
Lahore, Pakistan
[2]Dept. of Computer Science
The University of York
York, UK
{mhassan, akarim}@lums.edu.pk; {suresh, jc}@cs.york.ac.uk

Abstract. We describe and evaluate a discriminative clustering approach for content-based tag recommendation in social bookmarking systems. Our approach uses a novel and efficient discriminative clustering method that groups posts based on the textual contents of the posts. The method also generates a ranked list of discriminating terms for each cluster. We apply the clustering method to build two clustering models – one based on the tags assigned to posts and the other based on the content terms of posts. Given a new posting, a ranked list of tags and content terms is determined from the clustering models. The final tag recommendation is based on these ranked lists. If the poster's tagging history is available then this is also utilized in the final tag recommendation. The approach is evaluated on data from BibSonomy, a social bookmarking system. Prediction results show that the tag-based clustering model is more accurate than the term-based clustering model. Combining the predictions from both models is better than either model's predictions. Significant improvement in recommendation is obtained over the baseline method of recommending the most frequent tags for all posts.

1 Introduction

Social bookmarking systems have become popular in recent years for organizing and sharing resources on the Web. Such systems allow users to build a database of resources, typically Web pages and publications, by adding basic information (such as URLs and titles) about them and by assigning one or more keywords or tags describing them. The tags serve to organize the resources and help improve recall in searches. Individual users' databases are shared among all users of the system enabling the development of an information repository which is commonly referred to as a folksonomy [1]. A folksonomy is a collection of users, resources, and tags assigned by a user to a resource posted by him or her. Tag recommendation for new posts by users is desirable for two reasons. First, it ensures uniformity of tagging enabling better searches, and second, it eases the task of users in selecting the most descriptive keywords for tagging the resource.

Tag recommendation can have one of two goals: (1) to suggest tags tailored to individual users' preferences (the 'local' goal). and (2) to suggest tags that promote uniformity in tagging of resources (the 'global' goal). Tag recommendation can benefit from the tagging history of users and resources. However, when a user posts for the first time and/or the posted resource is new this historical information is less useful. In such cases, content-based tag recommendation is necessary, in which the contents of the resource are relied upon for tag recommendation.

This paper addresses task 1 of the ECML PKDD Discovery Challenge 2009 [2]. This task deals with content-based tag recommendation in BibSonomy, a social bookmarking system. The goal of tag recommendation is 'local', that is, to suggest tags tailored to individual users' preferences. Historical data of users, resources, and tags is available; however, the tag recommendation system must be able to provide good recommendations for unseen users and/or resources. Thus, the contents of resources must be utilized for tag recommendation.

Our solution to task 1 of the ECML PKDD Discovery Challenge 2009 relies on a novel discriminative clustering and term ranking method for textual data. We cluster the historical data of posted resources and develop a ranked list of discriminating tags and content terms for each cluster. Given a new posting, based on its contents, we find the best 3 clusters and develop a weighted list of tags and terms appropriate for tagging the post. If the poster's tagging history is available, then this provides a third ranked list of tags appropriate for the post. The final tag recommendation for the post is done by rules that select terms from the weighted lists. These rules also decide on the number of tags to recommend for each known poster. Extensive performance results are presented for the post-core training data provided by the challenge organizers.

The rest of the paper is organized as follows. We present the related work and motivation in Section 2. Section 3 presents details of our content-based tag recommendation approach, including description of the discriminative clustering and method. Data preprocessing and analysis is discussed in Section 4. The results of our approach are presented and discussed in Section 5. We conclude in Section 6.

2 Related Work and Motivation

Tagging resources with one or more words or terms is a common way of organizing, sharing, and indexing information. Tagging has been popularized by Web applications like image (e.g flickr), video (e.g. YouTube), bookmark (e.g. dec.icio.us), and publication (e.g. BibSonomy) sharing/organizing systems. Automatic tag recommendation for these applications can improve the organization of the information through 'purposeful' tag recommendations. Moreover, automatic tag recommendations ease the task of users while posting new resources.

The majority of the approaches proposed for tag recommendation assume that either the user posting the resource and/or the resource itself has been seen in the historical data available to the system [3–6]. If this is not the case, then only the contents of the posted resource can be relied upon. For social bookmarking systems, contents of resources are textual in nature requiring appropriate text and natural language processing techniques.

Content-based tag recommenders for social bookmarking systems have been proposed by [7, 8]. Lipczak's method extracts the terms in the title of a post, expands this set by using a tag co-occurrence database, and then filters the result by the poster's tagging history [7]. He reports significant improvements in performance after each step of this three step process. Tatu et al.'s method utilizes terms from several fields including URL and title to build post and user based models [8] . It relies on natural language processing to normalize terms from various sets before recommending them. We use terms from several fields of the posts including URL and title. We also study the impact of filling in missing and augmenting fields from information crawled from the Web.

A key challenge in tag recommendation is dealing with sparsity of information. In a typical collaborative tagging system, the vast majority of tags are used very infrequently making learning tagging behavior very difficult. This issue is often sidestepped in evaluation of tag recommenders when they are evaluated on post-core data with a high level of duplication (e.g. in [4, 6] post-core at level 5 is used). Our evaluation is done on post-core at level 2 data provided by the ECML PKDD Discovery Challenge 2009 [2].

Document clustering has been used extensively for organizing and summarizing large document collections [9, 10]. A useful characteristic of clustering is that it can handle sparse document spaces by identifying cohesive groups. However, clustering is generally computationally expensive. In the domain of collaborative tagging systems, clustering has been explored for information retrieval and post recommendation [11, 12]. In this paper, we explore the use of clustering for content-based tag recommendation. We use an efficient method that is practical for large data sets.

3 Discriminative Clustering for Content Based Tag Recommendation

Our approach for content-based tag recommendation in social bookmarking systems is based on discriminative clustering, content terms and tags rankings, and rules for final recommendations. We use a novel and efficient discriminative clustering method to group posts based on the tags assigned to them and based on their contents' terms. This method maximizes the sum of the discrimination information provided by posts and outputs a weighted list of discriminating tags and terms for each cluster. We also maintain a ranked list of tags for seen users. Tags are suggested from these three rankings by intuitive rules that fuse the information from the lists. The rest of this section presents our approach in detail.

3.1 Problem Definition and Notation

A social bookmarking system, such as BibSonomy [13], allows users to post and tag two kind of resources: Web bookmarks and publications. Each resource type is described by a fixed set of textual fields. A bookmark is described by fields like URL, title, and description, while a publication is described by fields in the standard bibtex record. Some of these fields (like title for bookmarks) are mandatory while others are optional. This textual information forms the content of the resource. Each user who posts a resource must also assign one or more tags for describing the resource.

Let $p_i = \{u_i, \mathbf{x}_i, \mathbf{t}_i\}$ denotes the ith post, where u_i is the unique user/poster ID, and $\mathbf{x}_i$ and $\mathbf{t}_i$ are the vector space representations of the post's contents and tags, respectively. If T is the size of the vocabulary then the ith post's contents and tags can be written as $\mathbf{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iT}\}$ and $\mathbf{t}_i = \{t_{i1}, t_{i2}, \ldots, t_{iT}\}$, respectively, where x_{ij} (t_{ij}) denotes the frequency of term j (tag j) in post i. Note that an identical vector space model is used to represent both content terms and tags, $t_{ij} \in \{0, 1\}, \forall i, j$, and $x_{ij} \geq 0, \forall i, j$. The historical data contain N posts. The tag recommender suggests tags for a new post i described by u_i and $\mathbf{x}_i$. The user u_i and resource described by content $\mathbf{x}_i$ may or may not appear in the historical data.

Let $TG(i)$, $TM(i)$, and $TU(i)$ be the ranked list of tags from clustering, terms from clustering, and user tags, respectively, corresponding to the ith post. The actual tags recommended for post i, denoted by $TR(i)$, are determined from these ranked lists by intuitive rules.

Given a test data containing M posts, the performance of the tag recommender is evaluated by averaging F1-score of each prediction over the entire test data.

3.2 Discriminative Clustering for Tag and Term Ranking

The historical data of N posts is clustered into $K \ll N$ groups using a novel discriminative clustering method. This method is motivated from the recently proposed DTWC algorithm for text classification [14]. It is an iterative partitioning method that maximizes the sum of discrimination information provided by each textual content (a post, in our setting) between its assigned cluster and the remaining clusters. The key ideas include discriminative term weighting, discrimination information pooling, and discriminative assignment. Unlike other partitioning clustering methods, this method does not require the explicit definition of a similarity measure and a cluster representative. Furthermore, it builds a ranked list of discriminating terms for each cluster implicitly. The method is computationally more efficient than popular methods like the k-means clustering algorithm. We perform two clusterings of the historical data – one based on the content terms $\mathbf{x}$ and the other based on the tags $\mathbf{t}$ of the posts in the data. In the following description, we develop the method for content terms only; the method as applied to tags will be similar.

First, an initial clustering of the data is done. This can be done randomly or, less efficiently especially for large collections, by a single iteration of the k-means algorithm with the cosine similarity measure. Given this clustering, a discriminative term weight w_j^k is computed for each term j in the vocabulary and for each cluster k as [14]

$$w_j^k = \begin{cases} p(x_j|k)/p(x_j|\neg k) & \text{when } p(x_j|k) > p(x_j|\neg k) \\ p(x_j|\neg k)/p(x_j|k) & \text{otherwise} \end{cases}$$

where $p(x_j|k)$ and $p(x_j|\neg k)$ are the probabilities that term j belongs to cluster k and the remaining clusters ($\neg k$), respectively. The discriminative term weight quantifies the discrimination information that term j provides for cluster k over the remaining clusters. Note that this weight is expressed as a probability ratio and is always greater than or equal to 1. The probabilities are computed by maximum likelihood estimation from the historical data.

Having computed the discriminative term weights for the current clustering, two discrimination scores can be computed for each post i. One score, denoted as $Score^k(\mathbf{x}_i)$, expresses the discrimination information provided by post i for cluster k, whereas the other score, denoted as $Score^{\neg k}(\mathbf{x}_i)$, expresses the discrimination information provided by post i for clusters $\neg k$. These scores are computed by linearly pooling the discrimination information provided by each term x_j in post i as [14]

$$Score^k(\mathbf{x}_i) = \frac{\sum_{j \in Z^k} x_j w_j^k}{\sum_j x_j} \text{ and}$$

$$Score^{\neg k}(\mathbf{x}_i) = \frac{\sum_{j \in Z^{\neg k}} x_j w_j^k}{\sum_j x_j}$$

In these equations, $Z^k = \{j | p(x_j|k) > p(x_j|\neg k)\}$ and $Z^{\neg k} = \{j | p(x_j|\neg k) > p(x_j|k)\}$ are sets of term indices that vouch for clusters k and $\neg k$, respectively. Each post, described by its contents $\mathbf{x}$, is then reassigned to the cluster k for which the cluster score $f^k = Score^k(\mathbf{x}) - Score^{\neg k}(\mathbf{x})$ is maximum. This is the cluster that makes each post most discriminating among all the clusters.

The overall clustering objective is to maximize the sum of discrimination information, or cluster scores, of all posts. Mathematically, this is written as

$$\text{Maximize } J = \sum_{i=1}^{N} \sum_{k=1}^{K} I^k(\mathbf{x}_i) \cdot f^k$$

where $I^k(\mathbf{x}_i) = 1$ if post i is assigned to cluster k and zero otherwise. Iterative reassignment is continued until the change in the clustering objective becomes less than a specified small value. Typically, the method converges satisfactorily in fewer than 15 iterations.

The discriminative term weights for the terms in the index set Z^k are ranked to obtain the weighted and ranked list of terms for cluster k. As mentioned earlier, clustering is also performed based on the tags assigned to posts. This clustering yields another weighted and ranked list of tags for each cluster.

It is worthwhile to point out that the term-based clustering is done on both the training and testing data sets. This approach allows the terms that exist only in the test data to be included in the vocabulary space, and for such terms to be available for recommendation as tags.

Given a new post i described by $\mathbf{x}_i$, the best cluster for it is the cluster k for which the cluster score f^k is a maximum. The corresponding ranked list of terms and tags for post i are denoted by $TM(i)$ and $TG(i)$, respectively. These ranked lists contain the most discriminating tags for post i based on its contents.

3.3 Final Tag Recommendation

Given a new post, and based on the contents $\mathbf{x}$ of the post, two ranked lists of terms appropriate for tagging are generated by the procedures described in the previous section.

If the user of the post appears in the historical data, then an additional list of potential tags can be generated. This is the ranked list of tags $TU(i)$ used by the user of post i The ranking is done based on frequency. Moreover, the average number of tags per user is computed and used while recommending tags for seen users.

The final list of tags for post i is made by simple and intuitive rules that combine information from all the lists. Let S be the number of tags to recommend for post i. Then, the final list of tags for the post is given by the following algorithm:

$$TR(i) = TG(i)[1 : P] \cap TM(i)[1 : Q]$$

$$\text{IF } TU(i)| \neq \oslash \text{ THEN } TR(i) = TR(i) \cap TU(i)[1 : R]$$

$$\text{IF } |TR(i)| < S \text{ THEN add top terms from } TG(i), TM(i) in TR(i)$$

In the above algorithm, P, Q, and R are integer parameters that define how many top terms to include from each list. If after taking the set intersections $|TR(i)| < S$ then the remaining tags are obtained from the top tags and terms in $TG(i)$ and $TM(i)$, respectively. In general, as seen from our evaluations, $R \leq Q \leq P$, indicating that $TG(i)$ is the least noisy source and $TU(i)$ the most noisy source for tags.

4 Evaluation Setup

4.1 Data and their Characteristics

We evaluate our approach on data sets made available by the ECML PKDD Discovery Challenge 2009 [2]. These data sets are obtained from dumps of public bookmark and publication posts on BibSonomy [13]. The dumps are cleaned by removing spammers' posts and posts from the user dblp (a mirror of the DBLP Computer Science Bibliography). Furthermore, all characters from tags that are neither numbers nor letters are removed. UTF-8 encoding and unicode normalization to normal form KC are also performed.

The post-core at level 2 data is obtained from the cleaned dump (until 31 December 2008) and contain all posts whose user, resource, and tags appear in at least one more post in the post-core data. The post-core at level 2 contain 64,120 posts (41,268 bookmarks and 22,852 publications), 1,185 distinct users, and 13,276 distinct tags. We use the first 57,000 posts (in content ID order) for training and the remaining 7,120 posts for testing.

We also present results on the test data released as part of task 1 of the ECML PKDD Discovery Challenge 2009. This data is cleaned and processed as described above, but it contain only those posts whose user, resource, or tags do not appear in the post-core at level 2 data. This data contain 43,002 posts (16,898 bookmarks and 26,104 publications) and 1,591 distinct users. For this evaluation, we use the entire 64,120 posts in the post-core at level 2 for training and test on the 43,002 posts in the test data.

These data sets are available in the form of 3 tables – tas, bookmark, and bibtex – as described below. The content of a post is defined by the fields in the bookmark and bibtex tables, while the tags appear in the tas table.

tas fact table; who attached which tag to which post/content. Fields include: user (number; user names are anonymized), tag, content id (matches bookmark.content id or bibtex.content id), content type (1 = bookmark, 2 = bibtex), date

bookmark dimension table for bookmark data. Fields include: content id (matches tas.content id), url hash (the URL as md5 hash), url, description, extended description, date

bibtex dimension table for BibTeX data. Fields include: content id (matches tas.content id), journal, volume, chapter, edition, month, day, booktitle, howPublished, institution, organization, publisher, address, school, series, bibtexKey (the bibtex key (in the @... line)), url, type, description, annote, note, pages, bKey (the "key" field), number, crossref, misc, bibtexAbstract, simhash0 (hash for duplicate detection within a user – strict – (obsolete)), simhash1 (hash for duplicate detection among users – sloppy –), simhash2 (hash for duplicate detection within a user – strict –), entrytype, title, author, editor, year

A few tagging statistics from the post-core data are given in Table 1 and Figure 1. These statistics are used to fix the parameter S (number of recommended tags) for known users. For unseen users, S is set at 5.

Table 1. Post-core at level 2 data statistics

	Avg	**Min**	**Max**	**Std. Deviation**
No. of tags per post	4	1	81	3.3
No. of posts per user	54	2	2031	162.9
No. of tags per user	62	1	4711	214.5
Frequency of tags	19	2	4474	106.9

4.2 Data Preparation

We explore tag recommendation performance on original contents, contents that have been augmented by crawled information, and contents that have been augmented and lemmatized.

The vocabulary for the vector space representation is formed from the tags and content terms in the training and testing sets. Selected content fields are used for gathering the content terms. For bookmark posts, the selected fields are url, description, and extended. For publication posts, the selected bibtex fields are booktitle, journal, howpublished, publisher, series, bibtexkey, url, description, annote, note, bkey, crossref, misc, bibtexAbstract, entrytype, title, and author. As mentioned earlier, the tags, which appear in the tas table, are also included in the vocabulary.

We remove all the non-letter and non-digit characters, but retain umlauts and other non-Latin characters due to UTF-8 encoding. All processed terms of length greater than or equal to three are retained. The tags are processed similarly, but without considering the token length constraint.

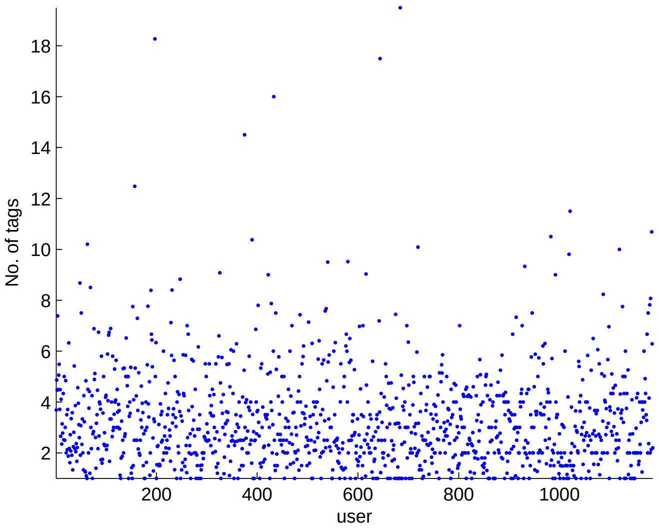


Fig. 1. Number of tags assigned to posts by users

Crawling Crawling is done to fill in and augment important fields. For bookmark posts, the extended description field is appended with textual information from <TITLE>, <H1> and <H2> HTML fields of the URL provided in the posts.

For publication posts, missing abstract field are filled using online search. We use the publication title to search for its abstract on CiteULike [15]. If the article is found, and its abstract is available on CiteULike, the bibtexAbstract field of the post is updated. CiteULike is selected because its structure is simpler and it does not have any restrictions on the number of queries (in a day for example).

Lemmatization We also explore lemmatization of the vocabulary while developing the vector space representation. Lemmatization is different from stemming as lemmatization returns the base form of a word rather than truncating it. We do lemmatization using TreeTagger [16]. TreeTagger is capable of handling multiple languages besides English. We lemmatize the vocabulary using English, French, German, Italian, Spanish and Dutch languages. The procedure, in brief, is as below:

1. TreeTagger is run on the vocabulary file once for each language: English, French, German, Italian, Spanish and Dutch.
2. TreeTagger returns the output file containing token, pos, lemma. The lemma is "<unknown>" if a token is not recognized in that language.
3. Using this "<unknown>" word, we combine the output of all six lemmatized files. If a term is not recognized by any language, the term itself is used as lemma.

4. If a word is lemmatized by more than one language, then lemmas are prioritized in the sequence: English, French, German, Italian, Spanish, Dutch. The first lemma for the word is selected.

4.3 Evaluation Criteria

The performance of tag recommendation systems is typically evaluated using precision, recall, and F1 score, where the F1 score is a single value obtained by combining both precision and recall. We report the precision, recall, and F1 score averaged over all the posts in the testing set.

5 Results

In this section, we present and discuss the results of our discriminative clustering approach for content based tag recommendation. We start off by evaluating the performance of the clustering method.

5.1 Clustering Performance

The performance of the discriminative clustering method is evaluated on the entire 64,120 posts of the post-core at level 2 data. We cluster these posts based on the tags assigned to them. After clustering and ranking of tags for each cluster, we recommend the top 5 tags from the ranked list for all posts in each cluster. The average precision, recall, and F1 score percentages obtained for different values of K (number of desired clusters) is shown in Table 2.

The top 5 tags become increasingly accurate recommendations as the number of clusters is increased, with the maximum recall of 48.7% and F1 score of 30.6% obtained when $K = 300$. These results simulate the scenario when the entire tag space (containing 13,276 tags) is known. Furthermore, there is no separation between training and testing data. Nonetheless, the results do highlight the worth of clustering in grouping related posts that can be tagged similarly.

Table 2. Performance of discriminative clustering of posts using the tags assigned to them (post-core at level 2 data)

K	10	50	100	200	300
Act. Clusters	10	48	95	189	274
Av. Precision (%)	12.5	19.2	22.3	25.2	26.9
Av. Recall (%)	21.0	32.8	38.6	45.9	48.7
Av. F1-score (%)	13.7	21.4	25.0	28.7	30.6

Table 3 shows the top ranked tags for selected clusters. It is seen that the discriminative clustering method is capable of grouping posts and identifying descriptive tags for

Table 3. Top tags for selected clusters (K = 200)

/ No.	Top Discriminating Tags
1	svm, ki2007webmining, mining, kernels, textmining, dm, textclassification
2	windows, freeware, utility, download, utilities, win, shareware
3	fun, flash, games, game, microfiction, flashfiction, sudden
4	tag, cloud, tagcloud, tags, folksonomia, tagging, vortragmnchen2008
5	library, books, archive, bibliothek, catalog, digital, opac
6	voip, mobile, skype, phone, im, messaging, hones
7	rss, feeds, aggregator, feed, atom, syndication, opml
8	bookmarks, bookmark, tags, bookmarking, delicious, diigo, socialbookmarking

each group of posts. Noisy tags are not ranked high in the lists. It is even able to discriminate and group posts of different languages (not shown in this table), especially when clustering is based on content terms. Two valuable characteristics of the discriminative clustering method are its stability and efficiency. The method converges smoothly (Figure 2) usually within 15 iteration. More importantly, especially considering the large post by vocabulary sizes involved, is the efficiency of the method. Each iteration of the method completes within 3 minutes, even for the large $107,122 \times 317,283$ data for the content-based clustering of the post-core plus task 1 test data.

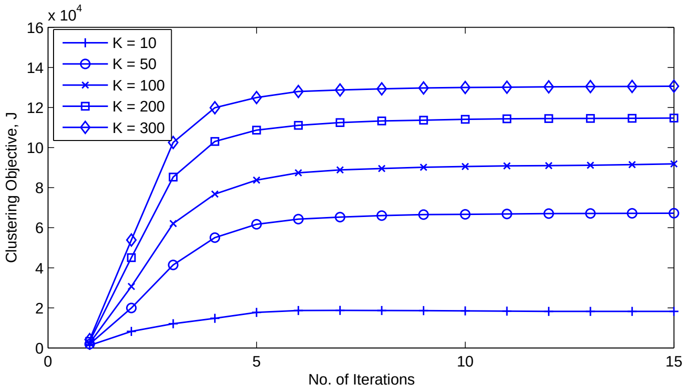


Fig. 2. Discriminative clustering convergence curves (clustering posts based on tags)

5.2 Tag Recommendation Using *TG* and *TM* Only

In this section, we discuss the performance of recommending the top 5 tags from the $TG(i)$ or $TM(i)$ list of each post i. This evaluation is done on the testing data of 7,120 posts held out from the post-core at level 2 data. The clustering model is based on the first 57,000 posts (in content ID order) from the data. In this evaluation, the original data, without augmentation with crawled information, is used for creating the vector space representation.

The recommendation results for different K values are given in Table 4. Results are shown for the case when only the top cluster for each post is considered, and for the case when the top three clusters of each post are merged in a weighted manner (using cluster score and discriminative term weights). It is observed that merging the lists of the top three clusters always gives better performance. Moreover, recommendations based on $TG(i)$ are always better than those based on $TM(i)$ indicating that the term-based clustering is more noisy than that based on tags. We also find out that $K = 200$ yields the highest recommendation performances.

Table 4. Tag recommendation performance (average F1-score percentages) using *TG* or *TM* only for original data

K	10	50	100	200	300
***TG* Only (Best Cluster)**	6.6	7.4	8.7	8.7	7.2
***TG* Only (Top 3 Clusters)**	7.3	8.2	9.5	10.6	9.1
***TM* Only (Best Cluster)**				6.3	
***TM* Only (Top 3 Clusters)**				7.8	

5.3 Tag Recommendation Using All Lists

In this section, we evaluate the performance of our approach when utilizing information from all lists. We also evaluate performance on original, crawled, and crawled plus lemmatized data. These results are shown in Table 5. For this evaluation, we fix $K = 200$ and use the top three clusters for building $TG(i)$ and $TM(i)$.

The first column (identified by the heading *TF*) shows the baseline result of recommending the top 5 most frequent tags in the training data (57,000 posts from post-core data). It is seen that our clustering based recommendation improves performance beyond the baseline performance. The second and third columns show the performance of recommending the top 5 terms from $TG(i)$ and $TM(i)$, respectively. The predictions of the tag-based clustering always outperform the predictions of the term-based clustering. In the fourth column, we report results for the case when the top 5 recommended tags are obtained by combining $TG(i)$ and $TM(i)$, as described in Section 3.3. These results are significantly better than those produced by each list independently.

The fifth column shows the results of combining all lists, including the user list $TU(i)$ when known. This strategy produces the best F1 score of 15.5% for the crawled data. This is a significant improvement over the baseline F1 score of 7.0%.

Table 5 also shows that filling in missing fields and augmenting the fields with crawled information improves performance. Lemmatization does not help, probably because users do not necessarily assign base forms of words as tags.

Table 5. Tag recommendation performance (average F1-score percentages) for processed data (K = 200; prediction based on top 3 clusters). The bottom line shows performance on task 1 test data

Data / Lists)	***TF***	***TG***	***TM***	***TG, TM***	***TG, TM, TU***
Original Contents	7.0	10.6	7.8	11.5	12.8
Crawled Contents	7.0	12.3	10.4	14.3	15.5
Crawled+Lemmatized Contents	7.0	11.7	9.7	13.3	14.6
Task 1 Test Data (Crawled)	1.1	4.9	3.2	5.2	5.4

5.4 Tag Recommendation for Task 1 Test Data

We report the performance of our approach on task 1 test data released by the challenge organizers on the bottom line of Table 5. We filled in missing and augmented other fields by crawled information. No lemmatization is done. The final vocabulary size is equal to 317,283 terms making the tag recommendation problem very sparse. The baseline performance of using the 5 most frequent tags from the post-core at level 2 (the training data for this evaluation) is the F1 score of 1.1% only. By using our discriminative clustering approach, the average F1 score reaches up to 5.4%. This low value is attributable to the sparseness of the data, and it is unlikely that other methods can cope better without extensive semantic normalization and micro modeling of the tagging process.

6 Conclusion

In this paper, we explore a discriminative clustering approach for content-based tag recommendation in social bookmarking systems. We perform two clusterings of the posts: one based on the tags assigned to the posts and the second based on the content terms of the posts. The clustering method produces ranked lists of tags and terms for each cluster. The final recommendation is done by using both lists, together with the user's tagging history if available. Our approach produces significantly better recommendations than the baseline recommendation of most frequent tags.

In the future, we would like to explore language specific models, incorporation of a tag extractor method, and semantic relatedness and normalization.

References

1. Golder, S., Huberman, B.A.: The structure of collaborative tagging systems. http://arxiv.org/abs/cs.DL/0508082 (Aug 2005)
2. Eisterlehner, F., Hotho, A., Jäschke, R.: Ecml pkdd discovery challenge 2009. http://www.kde.cs.uni-kassel.de/ws/dc09 (2009)
3. Sigurbjörnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: WWW '08: Proceeding of the 17th international conference on World Wide Web, New York, NY, USA, ACM (2008) 327–336
4. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. (2007) 506–514

5. Jschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in social bookmarking systems. AI Communications **21**(4) (2008) 231–247
6. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems, New York, NY, USA, ACM (2008) 43–50
7. Lipczak, M.: Tag recommendation for folksonomies oriented towards individual users. In: ECML PKDD Discovery Challenge 2008. (2008) 84–95
8. Tatu, M., Srikanth, M., D'Silva, T.: Rsdc 08: Tag recommendation using bookmark content. In: ECML PKDD Discovery Challenge 2008. (2008) 96–107
9. Andrews, N.O., Fox, E.A.: Recent developments in document clustering. Technical report, Computer Science, Virginia Tech (2007)
10. Kogan, J., Nicholas, C., Teboulle, M.: A Survey of Clustering Data Mining Techniques. In: Grouping Multidimensional Data. Springer (2006) 25–71
11. Begelman, G.: Automated tag clustering: Improving search and exploration in the tag space. In: In Proc. of the Collaborative Web Tagging Workshop at WWW06. (2006)
12. Shepitsen, A., Gemmell, J., Mobasher, B., Burke, R.: Personalized recommendation in social tagging systems using hierarchical clustering. In: RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems, New York, NY, USA, ACM (2008) 259–266
13. BibSonomy: Bibsonomy: A blue social bookmark and publication sharing system. http://www.bibsonomy.org/ (2009)
14. Junejo, K., Karim, A.: A robust discriminative term weighting based linear discriminant method for text classification. In: Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on. (Dec. 2008) 323–332
15. CiteULike: Citeulike website. http://www.citeulike.org/ (2009)
16. TreeTagger: Treetagger – a language independent part-of-speech tagger. http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/ (2009)

Time based Tag Recommendation using Direct and Extended Users Sets

Tereza Iofciu and Gianluca Demartini

L3S Research Center
Leibniz Universität Hannover
Appelstrasse 9a D-30167 Hannover, Germany
{iofciu,demartini}@L3S.de

Abstract. Tagging resources on the Web is a popular activity of standard users. Tag recommendations can help such users assign proper tags and automatically extend the number of annotations available in order to improve, for example, retrieval effectiveness for annotated resources. In this paper we focus on the application of an algorithm designed for Entity Retrieval in the Wikipedia setting. We show how it is possible to map the hyperlink and category structure of Wikipedia to the social tagging setting. The main contribution is a time-based methodology for recommending tags exploiting the structure in the dataset without knowledge about the content of the resources.

1 Introduction

Tagging Web resources has become a popular activity mainly due to the availability of tools and systems making it easy to tag and also due to the advantage users see in tagging their resources. People can for example get better search results, or they can get new resources recommended based on tags other people assigned. One particular problem is the one of recommending relevant tags to users for resources they have introduced in the system.

Being able to effectively recommend tags would, firstly, simplify the tasks of the users on the web who want to tag resources (e.g., bookmarks, pictures, . . .), and, secondly, would allow an automatic annotation of resources that enables, for example, a better search for resources or an improved resource recommendation.

When we want to assign a tag to a resource (or, to predict which tag a user would assign to a resource) a possible approach is to use the most popular tags for the given resource of the given user. Of course, this is not working well because users can tag resources which are different and people tag the same resource in different ways. For this reason most effective approaches look at the content of the resources and perform more complex analysis of the structure connecting users, resources, and tags.

Previous approaches focus on the content for resources (e.g., textual content of a web page) or on the structure of the tripartite graph composed of users, resources, and tags. The approaches we propose in this paper do not take into account the content of the resources but only the connection structure in the graph. Additionally, we put more importance on more recent tags with the assumption that users' interests might change over time.

We adapt an algorithm proposed for ranking entities in Wikipedia [1] based on a set of initial relevant examples (e.g., already tagged resources) and on the structure of hyperlinks connecting pages and categories containing them. As we defined hard links between documents and categories they belong to and soft links between documents and categories containing linked documents, so we define these types of links between resources/users and tags in the tag recommendation setting.

The rest of the paper is structured as follows. In Section 2 we describe the proposed algorithms also showing the correspondence to the Wikipedia setting. In Section 3 we describe the experimental setting and results. In Section 4 we compare our work with previously proposed approaches and, finally, in Section 5 we conclude the paper.

2 Graph Based Algorithms

In this section we describe the algorithms we designed and used for the graph based task that have been run at Discovery Challenge (DC) 2009.

2.1 Using the Resource-User Graph

In both submitted approaches, starting from the input *query post* (i.e., the posts from the test file) we retrieve the resource it refers to. We call this resource the *query resource*. For the query resource we retrieve, using the train data, all the users that have annotated it in different posts. We call this set of users the *direct user set*. We then use this set of user as an input for the algorithm and retrieve all tags the users have assigned. In the second algorithm, in addition to the set of direct users, we also retrieve the user neighborhood (i.e., users that used at least once a tag in common with the given user). We then use the reunion of the two user sets as input for recommending tags. We call the reunion of the two user sets, the *extended user set*. As a third approach we have also retrieved just the tags that have previously been assigned to the resource as baseline for comparison.

As seen in Figure 1, by traversing the post - resource - users graph, we obtain the set of direct users that have annotated the resource given in the query post. The extended user set is obtained by adding also the neighborhood users to the direct user set, see Figure 2. We considered two users as being neighbors if they had common tags.

As a baseline approach we considered the recommendation of the most popular tags for a resource, where we only kept the tags assigned by the *direct users* to the resource of the query post.

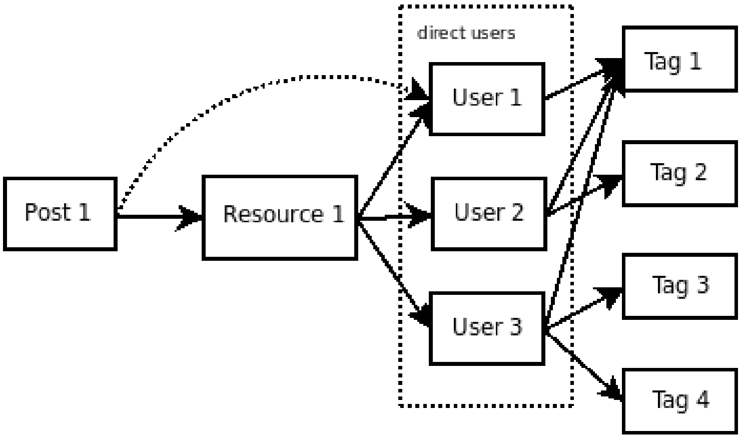


Fig. 1. Tags recommended based on the set of users who have annotated the query resource.

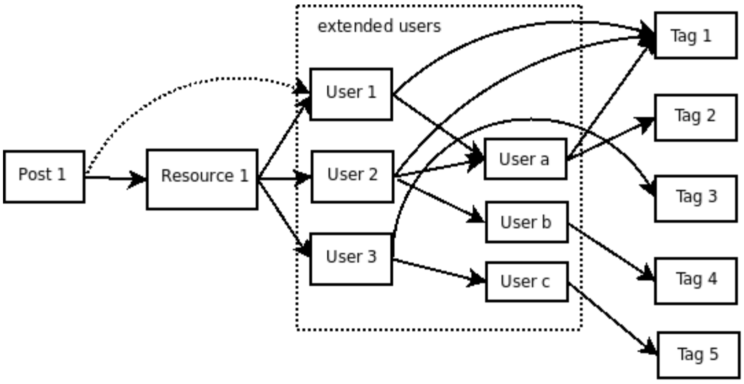


Fig. 2. Tags recommended based on the set of users who have annotated the query resource and users in the immediate neighborhood of the *direct user set*.

2.2 Comparison to the Wikipedia scenario

The algorithms described in this paper are adapted from those developed for finding relevant results for Entity Retrieval queries in the Wikipedia Setting [1]. This work was performed in the context of the Entity Ranking track at the evaluation initiative INEX 2008 [2]. In the following we describe how we can map the Entity Ranking setting with the tag recommendation one.

In the Wikipedia setting we have as input a set of example entities. The goal is to extend such set with other relevant entities. If, for example, the initial set for the query "European Countries" contains Italy, Germany, and France, then the goal is to extend this list with entities such as Spain, Slovenia, Portugal, ... Our approach is to retrieve other entities based on common assigned Wikipedia categories. We extract two sets of categories, hard categories as direct categories (similarly to the *direct user set*) and soft categories from the neighboring entities (i.e., following hyperlinks between Wikipedia articles). As neighboring entities we considered the most frequent entities the example entities linked to (similarly to the *extended user set*). In the Wikipedia setting entities link to entities via hyperlinks, and each entity has several categories assigned to it.

2.3 Time dependent tag ranking

Following the intuition that tags can get outdated over the years, and, thus, older assigned tags should be weighted less for recommendation, we introduced a time decaying function of posts. Scores are assigned to posts based on the time when they have been issued compared to the time the latest test post has been issued. The time decaying function is defined by the following formula:

$$postScore_i = \lambda^{\Delta Time_i} \tag{1}$$

with the decaying factor lambda being smaller than 1 and the time difference being calculated in years. The tag scores are computed based on the tag specificity (i.e., how often they have been assigned) defined as:

$$tagSpecificity_i = \log(50 + tagCount_i) \tag{2}$$

Given the different user sets for a query post, we extract from the training data the most frequent common tags the users have assigned. The tag score is computed based on the formula:

$$tagScore_i = \frac{\sum_j (postScore_j)}{tagSpecificity_i} \tag{3}$$

where a post j was considered only if it was posted by one of the users from the *direct user set* for the first approach and from the *extended user set* for the second approach. The tags are sorted based on this score and the top five tags are kept and recommended.

As a baseline, we ranked the tags based on popularity within the resource (i.e., how often a tag has been assigned to a resource) also keeping into account when they had been assigned to the resource, based on the formula:

$$tagScore_i = \sum_j (postScore_j) \tag{4}$$

3 Experiments

Experiments were performed on the DC 2009 benchmark[1] in order to evaluate the proposed algorithms.

Starting from the query posts in the test file we recommended for each post the top five tags using the two described approaches and the baseline. In Figure 3 it is possible to see effectiveness values for the two approaches when a different number of retrieved tags is considered. We can see that the direct user approach performs better. Figure 4 shows the same result with Precision/Recall curves of the two proposed approaches.

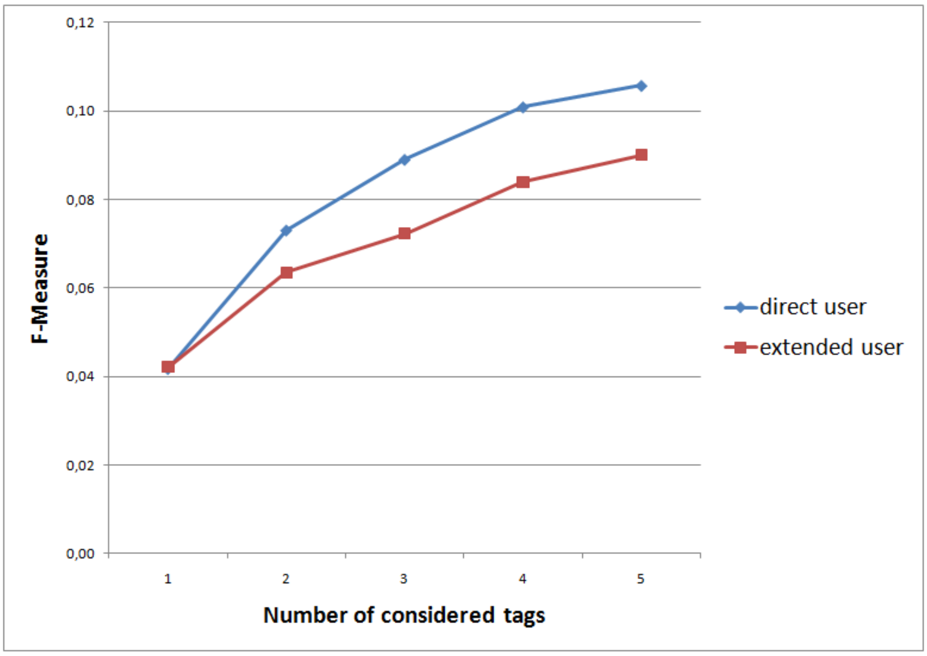


Fig. 3. F-Measure values for *Direct user* approach and *Extended user* approach (λ=0.9)

[1] http://www.kde.cs.uni-kassel.de/ws/dc09

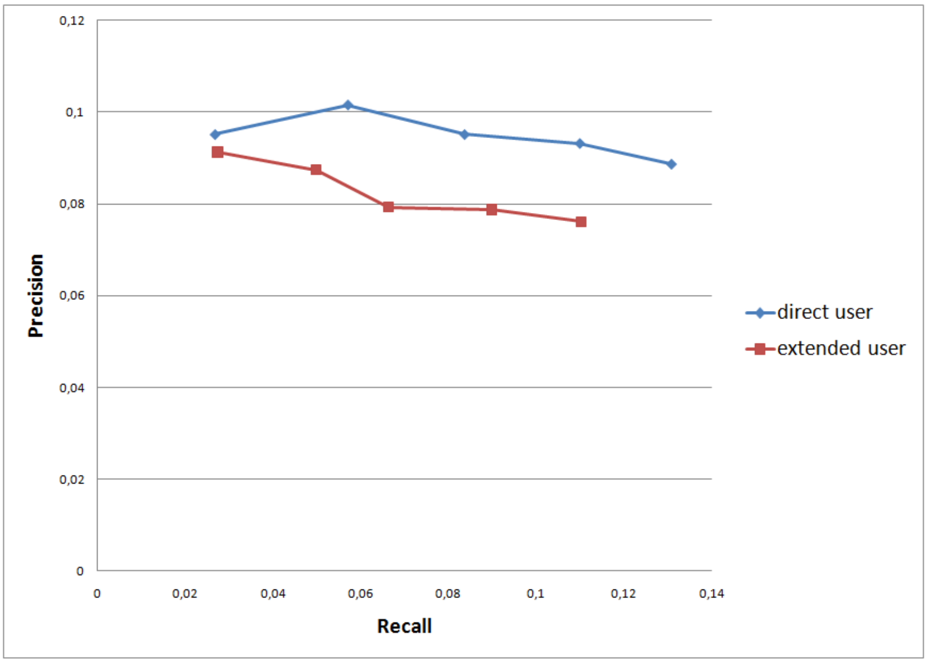


Fig. 4. Precision/Recall curves for *Direct user* approach and *Extended user* approach (λ=0.9)

In Figures 5 and 6 we measure the impact of using the time information when recommending the most popular tags for a resource. With a value of 0.9 for λ, in the time decaying function, the scores were slightly lower than when using just the popularity information (Figure 5). When using a value of 0.95 for λ, there is a small improvement over the baseline when considering 4 and 5 tags (see Figure 6). We ran experiments also with values smaller than 0.9 for λ which have shown that Precision and F-measure decrease quite a lot (3% for F-measure with $\lambda = 0.1$).

4 Related Work

Previous work on tag recommendation mainly distinguish between those looking at the content of the resources and those looking at the structure connecting users, resources, and tags.

Approaches looking at content of resources for tag recommendations are, for example, [5] which looks at content-based filtering techniques. In [6] the authors also look at collaborative tag suggestion in order to identify most appropriate tags.

A specific area of this field looks at recommending tags focusing on an individual user rather than providing general recommendation for a resource. In [4] they first create a set of candidate tags to be recommended and then they

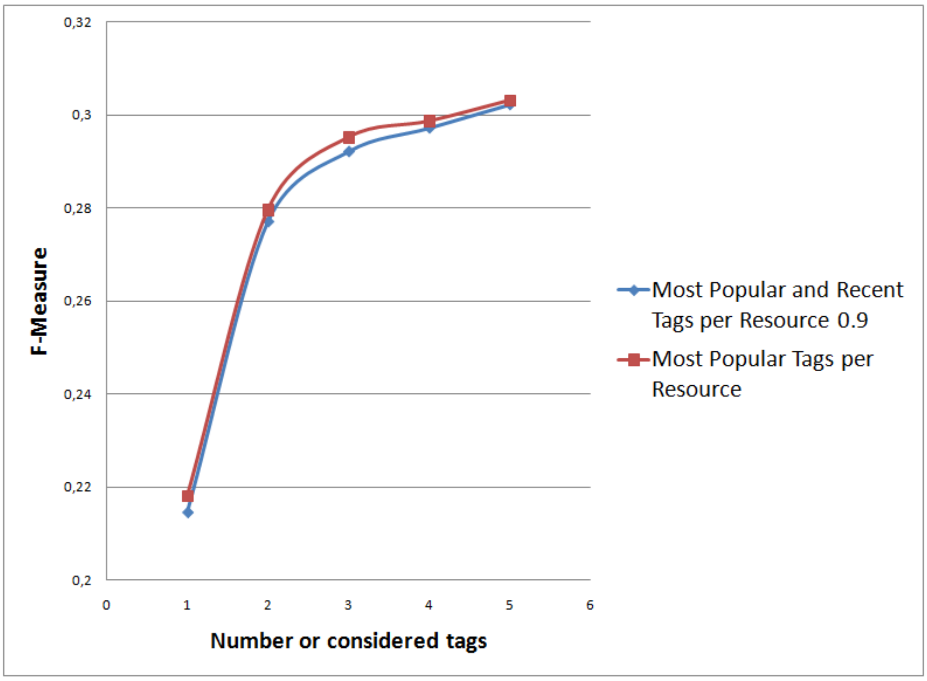


Fig. 5. F-Measure values for *Most Popular Tags per Resource* approach and *Most Popular and Recent Tags per Resource* approach (λ=0.9)

filer it based on the previous tag a particular user has assigned in the past. In [3] the FolkRank algorithm is evaluated and compared with simpler approaches. This is a graph based approach that computes popularity scores for resources, users, and tags based on the well-known PageRank algorithm exploiting the link structure. The assumption is that resources which are tagged with important tags by important users becomes important themself. Similarly to FolkRank, our approach exploits the link structure between users, resources, and tags, but rather looks at the vicinity of a post (i.e., a [resources,user] pair) in order to compute a weight for the most appropriate tags.

5 Conclusions and Further Work

In this paper we presented our first approaches for tag recommendation using graph information. We proposed two approaches, where, given a query post, we retrieve two sets of users. Based on the tags assigned by users in these sets we recommend new tags. The first set of users, the direct user set, consists of the users that have tagged the resource referred to by the query post. The second set of user, the extended user set, consists of the direct user set as well as the users who are neighbours based on commonly assigned tags to the users in the direct set. The tag scores have been computed keeping into account also the time when they have been assigned.

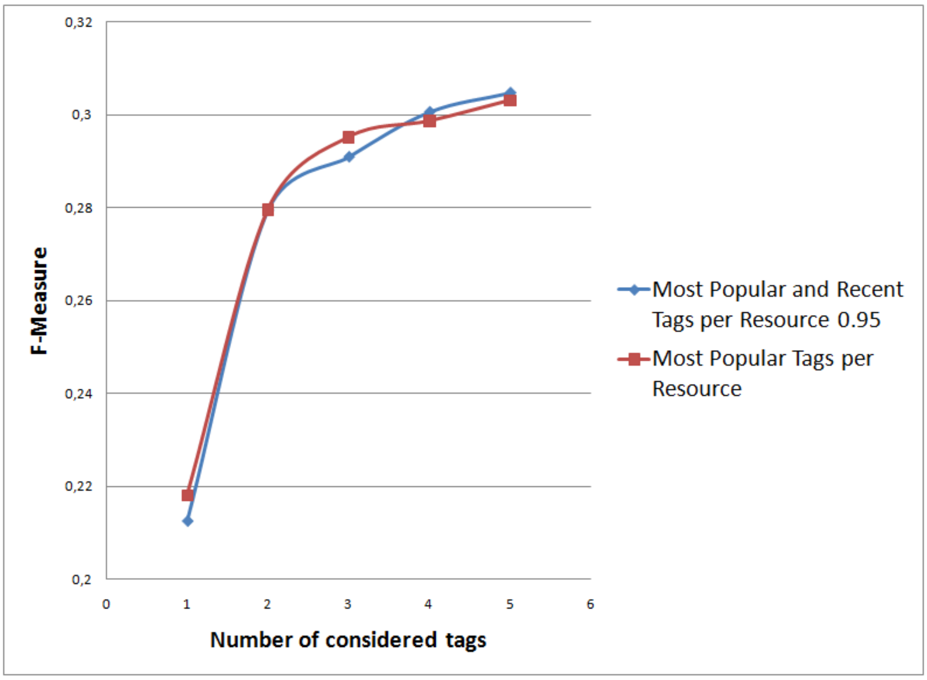


Fig. 6. F-Measure values for *Most Popular Tags per Resource* approach and *Most Popular and Recent Tags per Resource* approach (λ=0.95)

With the proposed approaches, we evaluated the effect of the tag posting time. We compared a time dependent ranking to a tag popularity. In the future, we aim at giving a higher importance to the user given in the query post than to the rest of the direct users.

Acknowledgments. This work is partially supported by the EU Large-scale Integrating Projects OKKAM[2] - Enabling a Web of Entities (contract no. ICT-215032), and LivingKnowledge[3] (contract no. 231126)

References

1. Nick Craswell, Gianluca Demartini, Julien Gaugaz, and Tereza Iofciu. L3s at inex 2008: Retrieving entities using structured information. In *INEX*, 2008.
2. Gianluca Demartini, Arjen P. de Vries, Tereza Iofciu, and Jianhan Zhu. Overview of the inex 2008 entity ranking track. In *INEX*, 2008.
3. Robert Jäschke, Leandro Balby Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in social bookmarking systems. *AI Commun.*, 21(4):231–247, 2008.

[2] `http://fp7.okkam.org/`
[3] `http://livingknowledge-project.eu/`

4. Marek Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 84–95, 2008.
5. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM.
6. Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006.

A Weighting Scheme for Tag Recommendation in Social Bookmarking Systems

Sanghun Ju and Kyu-Baek Hwang

School of Computing, Soongsil University, Seoul 156-743, Korea
shju@ml.ssu.ac.kr, kbhwang@ssu.ac.kr

Abstract. Social bookmarking is an effective way for sharing knowledge about a vast amount of resources on the World Wide Web. In many social bookmarking systems, users bookmark Web resources with a set of informal tags which they think are appropriate for describing them. Hence, automatic tag recommendation for social bookmarking systems could facilitate and boost the annotation process. For the tag recommendation task, we exploited three kinds of information sources, i.e., resource descriptions, previously annotated tags on the same resource, and previously annotated tags by the same person. A filtering method for removing inappropriate candidates and a weighting scheme for combining information from multiple sources were devised and deployed for ECML PKDD Discovery Challenge 2009. F-measure values of the proposed approach are 0.17975 for task #1 and 0.32039 for task #2, respectively.

Keywords: social bookmarking, folksonomy, tag recommendation

1 Introduction

Social bookmarking systems such as BibSonomy[1] and Delicious[2] have increasingly been used for sharing bookmarking information on the Web resource. Such systems are generally built on a set of collectively-annotated informal tags, comprising a folksonomy. A tag recommendation system could guide users during the bookmarking procedure by providing a suitable set of tags for a given resource. In this paper, we propose a simple but effective approach for tackling the tag recommendation problem. The gist of our method is to appropriately combine different information sources with pre-elimination of barely-used tags.

The candidate tags for recommendation can be extracted from the following information sources. First, resources themselves may have the annotated tags. For example, the title of a journal article is likely to include some of the annotated keywords. Second, the tags previously annotated by other users for the same resource

[1] http://www.bibsonomy.org/

[2] formerly del.icio.us, http://delicious.com/

could be a good candidate set. Third, previously annotated tags for other resources by the same user could also provide some information.[3]

The paper is organized as follows. In Section 2, the proposed tag recommendation method is detailed. Then, Section 3 shows the results of experimental evaluation on the training dataset, confirming the effectiveness of the proposed method. Performance of our method on the test dataset is briefly described in Section 4. Finally, concluding remarks are drawn in Section 5.

2 The Method

In this section, we detail the proposed tag recommendation method. First, the procedure for keyword extraction from resource descriptions with importance estimation and filtering is explained. Then, the keyword extraction and importance estimation method from previously annotated information is described. Finally, tag recommendation by combining multiple information sources is explained.

2.1 Keyword Extraction from Documents (Resource Descriptions)

In our approach, candidate keywords are extracted from the columns *url*, *description*, and *extended description* of the table *bookmark* as well as the columns *journal*, *booktitle*, *description*, and *title* of the table *bibtex*. It should be noted here that the candidates extracted from different fields are processed separately. This means that even the same keywords could have multiple importance values according to the columns from which they are extracted.[4]

In order to estimate the importance of each keyword, its accuracy and frequency ratios are calculated as follows.

D: set of all documents (resources) such as bookmarks or BibTex references.
EC(k, d): extraction count of keyword k in document d.
MC(k, d): matching count of keyword k with one of the tags of document d.[5]
TEC(d): extraction count of all the keywords in document d.

$$\text{Accuracy Ratio, AR}(k) = \sum_{d\in\mathbf{D}} \mathrm{MC}(k,d) \;/\; \sum_{d\in\mathbf{D}} \mathrm{EC}(k,d). \qquad (1)$$

$$\text{Frequency Ratio, FR}(k) = \sum_{d\in\mathbf{D}} \mathrm{EC}(k,d) \;/\; \sum_{d\in\mathbf{D}} \mathrm{TEC}(d). \qquad (2)$$

The accuracy and frequency ratios of each keyword are calculated across all the documents.

[3] [1] also exploited these kinds of information sources for tag recommendation. We extend this approach by extracting keywords from not only resource title but also other resource descriptions.

[4] It is because average importance values of keywords are different according to extracted columns.

[5] MC(k, d) is equal to EC(k, d) if d is tagged with k, 0 otherwise.

The keywords whose accuracy is lower than average are not considered for recommendation. This elimination procedure is implemented by the following criterion, which also penalizes frequent words.

TMC(d): sum of MC(k, d) across all the keywords in document d.

$$\text{Limit Condition: AR}(k) \,/\, (1 + \text{FR}(k)) > \textstyle\sum_{d\in\mathbf{D}} \text{TMC}(d) \;/\; \sum_{d\in\mathbf{D}} \text{TEC}(d). \tag{3}$$

Some keywords with high accuracy ratio values are shown in Table 1. It should be noted that there exist a large amount of keywords having high AR(k) values and the keywords in Table 1 are a sample from them.

Table 1. Example keywords with high accuracy ratios.

Keywords	Extracted columns	Accuracy ratio, AR(k)	Frequency ratio, FR(k)
nejm	*extended description*	1.0000	0.0002579
medscape	*extended description*	1.0000	0.0001146
freebox	*description*	1.0000	0.0000533
harum	*description*	0.9800	0.0000556
ldap	*url*	0.9354	0.0000403
shipyard	*description*	0.9146	0.0002734

The keywords in Table 1 have accuracy ratios much higher than the average, satisfying Equation (3). In Table 2, we present some keywords on the border with respect to Limit Condition.

Table 2. Example keywords on the border in terms of Limit Condition.

Keywords	Extracted columns	Accuracy ratio, AR(k)	Frequency ratio, FR(k)	Difference in Limit Condition	Limit Condition satisfied
netbib	*url*	0.0789	0.0002468	0.0004281	Yes
guide	*url*	0.0778	0.0006510	-0.0007060	No
media	*url*	0.0781	0.0008810	-0.0003974	No
daily	*extended description*	0.0602	0.0002744	0.0005867	Yes
list	*extended description*	0.0601	0.0008056	0.0005053	Yes
engine	*extended description*	0.0590	0.0005598	-0.0006156	No
tool	*description*	0.1279	0.0007647	0.0006509	Yes
ontologies	*description*	0.1271	0.0001312	-0.0000749	No
corpus	*description*	0.1264	0.0000967	-0.0007337	No

The average AR(k) values in *url*, *description*, and *extended description* are 0.07849973, 0.12715830, and 0.05963773, respectively. We also show some example keywords with low accuracy ratios, which do not satisfy Equation (3) as follows.

url: org, co, ac, au, default, main, details, welcome.
description: feeds, economy, review, images, help.
extended description: a, have, that, one, other, are, person, its.

Finally, each extracted keyword, satisfying Equation (3), is stored in ***d-keyword set*** (**DS**). The accuracy weight of each candidate is calculated by multiplying its accuracy ratio and extraction count from the present document as follows.

$$\text{Accuracy Weight from Document Set, } \mathrm{AW}_{\mathrm{DS}}(k) = \mathrm{EC}(k, d) \times \mathrm{AR}(k). \quad (4)$$

The accuracy weight, $\mathrm{AW}_{\mathrm{DS}}(k)$, is calculated when recommending tags for a given document (resource) *d*.

2.2 Keyword Extraction from Previously-Annotated Information

Candidate keywords could be extracted from the previously annotated tags for the same resource. For the BibTex references, the field *simhash1* of the table *bibtex* is adopted for the semantically-same resource detection. For the bookmarks, a pruning function, which has similar effect of the approach used in [2], was implemented and deployed in our experiments. These candidate keywords are stored in ***r-keyword set*** (**RS**). Their accuracy weight is calculated as follows.

D: set of all documents (resources) satisfying the same document condition with the present document *d*.
TC(*k*, *d*): 1 if document *d* has keyword *k*; 0 otherwise.

$$\text{Accuracy Weight from Resource Set, } \mathrm{AW}_{\mathrm{RS}}(k) = \sum_{d \in \mathbf{D}} \mathrm{TC}(k, d). \quad (5)$$

Candidate keywords are also extracted from the previously annotated tags by the same person. These candidate keywords are stored in ***u-keyword set*** (**US**). Their accuracy weight is obtained as follows.

D: set of all documents (resources) which are previously tagged by user *u*.
UC(*k*, *d*): 1 if document *d* has keyword *k*; 0 otherwise.

$$\text{Accuracy Weight from User Set, } \mathrm{AW}_{\mathrm{US}}(k) = \sum_{d \in \mathbf{D}} \mathrm{UC}(k, d). \quad (6)$$

2.3 Tag Recommendation by Combining Multiple Information Sources

The last step is to recommend appropriate tags from the three candidate keyword sets, i.e., ***d-keyword set*** (**DS**), ***r-keyword set*** (**RS**), and ***u-keyword set*** (**US**). Given a specific user and a document (resource) for tagging, these three candidate keyword sets are specified with accuracy weight for each candidate. Before unifying these candidates, the accuracy weights are normalized into [0, 1] as follows.

$\mathbf{EK} = \mathbf{DS} \cup \mathbf{RS} \cup \mathbf{US}$ ($ek \in \mathbf{EK}$).
$\mathrm{NW}_{\mathrm{DS}}(ek) = \mathrm{AW}_{\mathrm{DS}}(ek) / \sum_{k \in \mathbf{DS}} \mathrm{AW}_{\mathrm{DS}}(k)$.
$\mathrm{NW}_{\mathrm{RS}}(ek) = \mathrm{AW}_{\mathrm{RS}}(ek) / \sum_{k \in \mathbf{RS}} \mathrm{AW}_{\mathrm{RS}}(k)$.
$\mathrm{NW}_{\mathrm{US}}(ek) = \mathrm{AW}_{\mathrm{US}}(ek) / \sum_{k \in \mathbf{US}} \mathrm{AW}_{\mathrm{US}}(k)$.

We also added tag frequency information, denoting how many times a tag was annotated during the training period. This tag frequency rate is calculated as follows.

$\mathrm{TFR}(ek) = \sum_{d \in \mathbf{D}} \mathrm{TagCount}(ek, d) \;/\; \sum_{t \in \mathbf{T}} \sum_{d \in \mathbf{D}} \mathrm{TagCount}(t, d);\; 0 \le \mathrm{TFR}(ek) \le 1,$

where TagCount(t, d) denotes the number of occurrences of a tag t annotated for a document d. **T** and **D** denote the set of all tags and the set of all documents (resources), respectively.

The above four factors are linearly combined with appropriate coefficients. We have experimented with different coefficient values, trying to obtain nearly optimal results. First, we focused on the fact that the performance of extracted keywords from ***d-keyword set*** (**DS**) is higher than that from ***r-keyword*** or ***u-keyword sets*** (**RS** or **US**). Figure 1 compares the performance using each keyword set on the training dataset when the number of recommended tags is five.

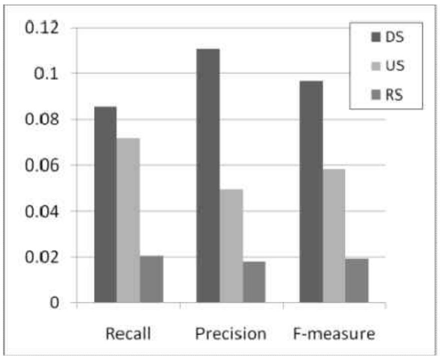


Figure 1. Performance comparison of extracted keywords from different information sources.

Accordingly, we tried high coefficient values on $\mathrm{NW}_{\mathrm{DS}}(ek)$ and relatively low coefficient values on $\mathrm{NW}_{\mathrm{RS}}(ek)$ and $\mathrm{NW}_{\mathrm{US}}(ek)$. However, this scheme does not produce better results than other schemes as shown in Figure 2.

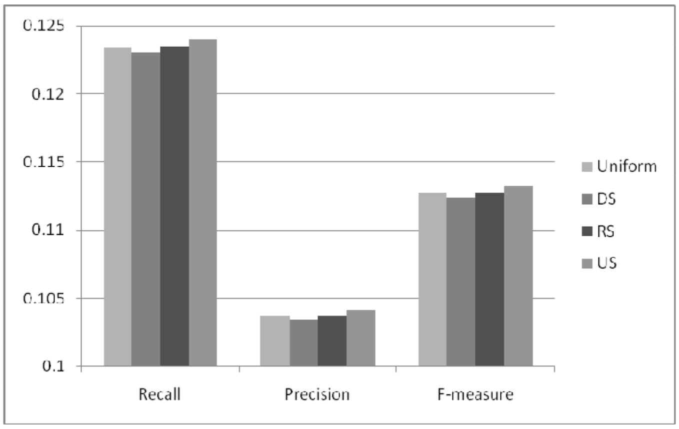


Figure 2. Performance comparison among different weighting schemes.

In Figure 2, Uniform denotes the case of assigning an equal coefficient (0.3) to each keyword set and DS (RS or US) denotes the case of assigning 0.45 to **DS** (**RS** or **US**) and 0.25 to the other keyword sets. TFR(*ek*) was assigned 0.1 or 0.05 in the above cases. On the contrary to our expectation, the weighting scheme assigning high coefficient value to **US** showed the best performance.

The reason for this phenomenon is not clear but one possible clue is that performance of the candidates extracted from **DS** varies much according to extracted data columns. Such keywords are illustrated in Table 3.

Table 3. Variation in accuracy ratio, AR(*k*), of the same keywords extracted from different data columns. Accuracy values higher than the average are represented in bold.

Keywords	*url*	*description*	*extended description*
portal	0.0439	0.1080	**0.1250**
tag	0.0410	0.1194	**0.1237**
tech	0.0318	0.0598	**0.1406**
template	0.0620	**0.1911**	**0.2023**
time	**0.1560**	0.0951	0.0322
youtube	**0.3217**	0.0877	0.0319

In Table 3, it is observed that even the same keyword from **DS** could have extremely different accuracy ratio values. For example, the keyword portal from *extended description* has much higher AR(*k*) value than the average, i.e., about 0.05964. However, the accuracy ratios of the same keyword from *url* or *description* are lower than the averages.

After several trials, we applied the following formula for the recommendation, which has shown fine results on the training dataset.

$$\mathrm{NW}_{\mathrm{DS}}(ek) \times .2 + \mathrm{NW}_{\mathrm{RS}}(ek) \times .35 + \mathrm{NW}_{\mathrm{US}}(ek) \times .4 + \mathrm{TFR}(ek) \times .05. \quad (7)$$

3 Experimental Evaluation

To evaluate the proposed approach, we reserved the postings spanning the latest six months from the given training dataset like the real challenge. Hence, the training period is from January 1995 to June 2008 and the validation period is from July to December of 2008. The numbers of postings, resources, and users during these periods are shown in Tables 4 and 5.

Table 4. The Post-Core dataset size

	bookmark	*bibtex*	*tas*	# of users
Training	37037	17267	218682	982
Validation	4231	5585	34933	433

Table 5. The Cleaned Dump dataset size

	bookmark	*bibtex*	*tas*	# of users
Training	212373	122115	1101387	2689
Validation	50631	36809	299717	1292

3.1 Effectiveness of Candidate Elimination

In this subsection, we present the effect of our keyword elimination method (Equation (3)). Note that Limit Condition is applied to the candidate keywords whose accuracy ratio is lower than average with some penalizing effect on frequently-occurred keywords. Figures 3 and 4 show the effect of candidate elimination on the Post-Core and Cleaned Dump datasets, respectively. The results are obtained when the number of recommended tags is five. On the both validation datasets (i.e., Post-Core and Cleaned Dump), the proposed elimination method increases precision and F-measure values regardless of the number of recommended tags (from one to ten, although the results are not shown here). In the case of the Cleaned Dump dataset, recall is also improved by our filtering method.

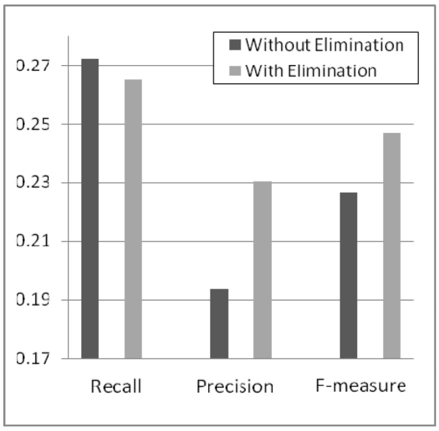


Figure 3. Effect of candidate elimination on the Post-Core dataset

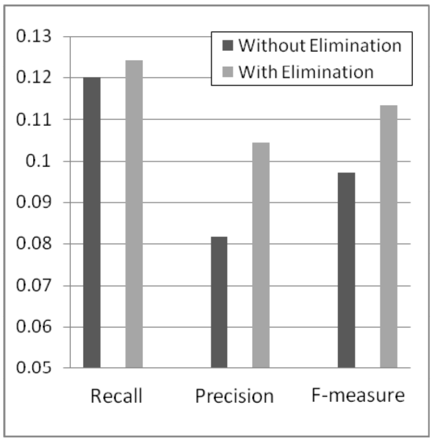


Figure 4. Effect of candidate elimination on the Cleaned Dump dataset

4 Final Results

Here, we append the final results of our method on the test dataset of ECML PKDD Discovery Challenge 2009.

Table 6. The test dataset size

	bookmark	*bibtex*	# of users
Cleaned Dump	16898	26104	1591
Post-Core	431	347	136

Table 7. Final results on the test dataset (Post-Core, Task #1)

# of tags	Recall	Precision	F-measure
1	0.074695721	0.243523557	0.114324737
2	0.121408237	0.213594717	0.154817489
3	0.152896044	0.193533634	0.170831363
4	0.175617505	0.179512968	0.177543872
5	**0.191311486**	**0.169508783**	**0.179751416**
6	0.203439061	0.162068819	0.180412681
7	0.213460494	0.156249045	0.180428119
8	0.22072531	0.151207336	0.179469517
9	0.227309809	0.147113534	0.178623208
10	0.232596191	0.143564862	0.177544378

Table 8. Final results on the test dataset (Cleaned Dump, Task #2)

# of tags	Recall	Precision	F-measure
1	0.142522512	0.42159383	0.213029148
2	0.241682971	0.367609254	0.291633121
3	0.315328224	0.331191088	0.323065052
4	0.367734647	0.295308483	0.327565903
5	**0.406172737**	**0.264524422**	**0.320390826**
6	0.443927734	0.242502142	0.313661833
7	0.47018359	0.221477537	0.301115964
8	0.49385481	0.204859836	0.289591798
9	0.509440246	0.190310422	0.27710381
10	0.520841594	0.176357265	0.263494978

5 Conclusion

We applied a simple weighting scheme for combining different information sources and a candidate filtering method for tag recommendation. The proposed filtering method was shown to improve precision and F-measure for the tag recommendation task in all the cases of our experiments. It has also shown to be effective for improving recall in some cases. Future works include finding more optimal scheme for combining multiple information sources. Evolutionary algorithms would be a suitable methodology for this task.

Acknowledgements

This work was supported in part by the Seoul Development Institute through Seoul R&BD Program (GS070167C093112) and in part by the Ministry of Culture, Sports and Tourism of Korea through CT R&D Program (20912050011098503004).

References

1. Marek Lipczak: Tag Recommendation for Folksonomies Oriented towards Individual Users. Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2008)

2. Marta Tatu, Munirathnam Srikanth, and Thomas D'Silva: RSDC'08: Tag Recommendations using Bookmark Content. Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (2008)

ARKTiS - A Fast Tag Recommender System Based On Heuristics

Thomas Kleinbauer and Sebastian Germesin

German Research Center for Artificial Intelligence (DFKI)
66123 Saarbrücken
Germany
firstname.lastname@dfki.de

Abstract. This paper describes the tag recommender system ARKTiS, our contribution to the 2009 ECML PKDD tag discovery challenge. ARKTiS consists of two separate modules for BibTeX entries and for bookmarked web pages. For generating tags, we distinguish between so-called *internal* and *external* methods, depending on whether a tag was extracted from the given information about a resource or whether additional resources were employed.

1 Introduction

The role of the end-user in the world wide web (WWW) has undergone a substantial change in recent years from a passive consumer of relatively static web pages to a central content producer. The addition of backchannels from WWW clients to internet servers empowers non-expert users to actively participate in the generation of web content. This has led to a new paradigm of usage, colloquially coined “Web 2.0” [1].

These novel kinds of interactions can be divided into two categories: producing or making accessible of new information (e.g., web logs, forums, wiki wikis, etc.) and enriching already existing contents (e.g., consumer reviews, recommendations, tagging, etc.). One interpretation of the second type of interaction is that it provides means to cope with one of the problems generated by the first type of interaction, namely the massive growth of available content and the increasing difficulty for traditional information retrieval approaches to support efficient access to the contained information. In this sense, the *meta-content* produced in the second kind of interaction can be construed as mainly serving as a navigation aid in an increasingly complex, but weakly structured online-world.

In particular, the possibility for users to attach keywords to web resources in order to describe or classify their contents bares an enormous potential for structuring information which facilitates subsequent access by both the original user and other users. This task, commonly referred to as *tagging* is simple enough not to scare users away, yet the benefit of web resources annotated in such a way is obvious enough to keep the motivation to supply tags high. The process of attaching tags to web resource must therefore show a fine balance between simplicity and quality. Both of these properties could be greatly improved if an automatic system could support a user by recommending tags for a given resource.

In this paper, we describe the ARKTiS system, developed to recommend tags to a user for two specific types of web resources. The system was developed as a contribution to an international challenge as part of the 2009 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2009).

2 Task and Data

The goal of this challenge is the implementation of a system that can automatically generate tag recommendations[1] for a given resource. Here, a resource is either a bookmark of a web page or a BibTEX entry for different kinds of documents. Tags typically are short English words although they may also be artificially created terms, generated e.g. by concatenating words ("indexforum") or by using abbreviations ("langfr", for "language french"). The number of tags a system may generate is restricted to a maximum of five.

To prepare for the challenge, a training set of manually tagged resources was provided. The data set consists of web page bookmarks and BibTEX entries taken from the BibSonomy project[2]. Each entry has a unique id and was tagged by at least two users. Thus a point in the data set can be viewed as a triple `<resource-id, user-id, tags>`.

For each resource, the corpus contains meta-data describing the resource with a number of different fields. These fields are different for BibTEX entries and for bookmark entries. For instance, the meta-data for BibTEX entries contain fields describing the title of an article, the authors, the year of publication, or the number of pages. For bookmarks, one field gives a short description of the resource while another one contains the web pages' URL. A full list of all available fields can be found on the homepage of the challenge.

In total, the training corpus contains 41,268 entries for bookmarks and 22,852 BibTEX entries, annotated with 1,3276 unique tags by 1,185 different users.

The data of the actual challenge (the *eval set*), was provided 48 hours before the submission deadline. This set consists of unseen data that each system had to tag and all results that are presented and analyzed in section 5 were achieved on this set. The evaluation data set contains 43,002 entries in total, with 26,104 BibTEX- and 16,898 bookmark-entries - circa two thirds of the amount of the training data.

[1] For the remainder of this paper, we will refer to this process as "(automatic) tagging"

[2] http://www.bibsonomy.org

3 Approach

One key observation for our participation in the ECML PKDD challenge was that time played a central role in two different senses. First, the task required each system to suggest tags for quite a large number of resources in a rather short period of time: 43,002 entries in only 48 hours, including retrieval and parsing of the test data as well as formatting and uploading of the final result data. Second, since the challenge had a fixed deadline, the time to develop a running system faced a naturally limit with the release of the 48 hour evaluation period.

Both points had a direct influence on the conceptualization and realization of our system `ARKTiS`, in that a number of more sophisticated ideas had to be sacrificed. As a result, `ARKTiS` can be seen as an exercise in software engineering rather than thorough science. The final system implements straight-forward strategies with a focus on robustness and processing speed.

3.1 Motivation

As outlined above, the training corpus contains 13276 different tags for over 41268 data points, a proportion that indicates a potential data sparseness problem for classical machine learning approaches. We therefore opted for an algorithmic approach instead, based on heuristic considerations.

Since the desired system output is (English) words, we can distinguish two potential sources for output: from within the resource itself (internal words) or from outside material (external words). This distinction is in so far blurred, as the system input actually consists not of the resources themselves, but rather of meta-data. In so far, even tags taken from the resources themselves could be argued to be external. We take the view that words stemming from meta-data or the resources referred to by the meta-data are considered internal.

Since we could not hope to implement a competitive system, we were mainly interested in how useful such a distinction would be in terms of recommending tags. Although we concentrated on internal methods as described below, we explored using document similarity measures in the BibTeX module to re-use tags that were manually assigned to documents similar to the current system input. Also, some of our implemented techniques, such as translating German words from the original resource to English, can be considered borderline between internal and external.

For the internal approaches, we looked at the task of tagging a resource as an analogy to automatic text summarization, somewhat taken to an extreme where a "summary" consists only of five words. In *extractive summarization*, summaries of documents are generated by identifying those sentences in a document which when concatenated serve as surrogate for the original document. In that spirit, tagging becomes the task of identifying those words from a resource that together describe the "aboutness" of the resource.

3.2 Related Work

A number of researchers in recent years have engaged in the task of developing an automatic tagging system. [2] use a natural language resources to suggest tags for BibTEX entries and web pages. They include conceptual information external resources, such as WordNet [3], to create the notion of a "concept space". On top of this notion, they exploit the textual content that can be associated with bookmarks, documents and users and generate models within the concept space or the tag space derived from the training data.

[4] model the problem of automated tag suggestion as a multi-label text classification problem with tags as categories.

In [5], the TagAssist system focuses on the task of the generation of tags for web-log entries. They access tags of similar documents in a similar spirit to our own method described in section 4.1.

In addition to these concrete systems, we find automatic tagging to bare some similarities with research in automatic extractive summarization. In both task, the identification of salient portions of a resource's text is a central consideration. For tagging which reduces extraction single words, we call such a method an *internal* approach. (s. section 3.1).

For instance, in [6], the author conducted a wide range of tests to find predictive features for relevant sentences. Despite relying on manual experiments, the general results from this early research were later confirmed by machine learning approaches, e.g., [7]. For the bookmark modules, both the *title* and the *first sentence* heuristic (see 4.2) were inspired by these findings.

4 Taggers

As hinted by the data set, the task can be viewed as two different sub-tasks, the tagging of BibTEX entries and the tagging of bookmarked web pages. Consequently, the `ARKTiS` system consists of two independent modules which are both instances of a common framework architecture, depicted in Figure 1. The modules can be run in two distinct processes.

Efficient processing of the input data is an important requirement for this challenge. In a sequential architecture, processing 43002 data points in 48 hours would leave a tagging system about 4 seconds per data point on average. Given that the data points contain only metadata and that the actual documents, if needed, have to be retrieved through the internet and parsed, a time span of 4 seconds poses quite a strong limitation on the complexity of the performed computations. Running multiple instances of taggers concurrently relaxes this limitation. In our current setup, both modules for BibTEX and bookmark tagging internally run ten tagging threads in parallel which increases the maximum average processing time to 80 seconds per data point.

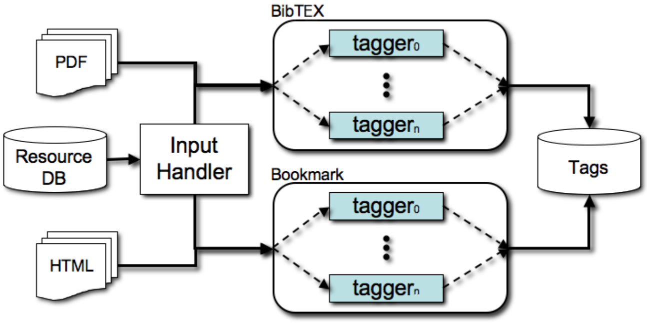


Fig. 1. The parallel architecture of `ARKTiS`.

4.1 The BibTEXTagger

The tagging system responsible for the BibTEX entries uses a combination of internal and external techniques. A thorough investigation of the provided training material showed that most of the entries (95.4%) do not contain a valid link to the actual PDF document. This is unfortunate, as it limits *internal* approaches which draw tags from the contents of the document in question.

Internal approach To compensate the cases in which the PDF document is unavailable, we use the remaining information from the meta-data, namely the *title* of the document, its *description* and its *abstract*. The employed approach analyzes these fields and extracts tags directly out of their textual information. Before that, we lowercase all words in the text of each field and remove all punctuation and symbols. After that, we apply the POS tagging system described in [8] to extract content-words - nouns, adjectives and verbs - out of the text. The sequential processing of the text is shown below:

Pre-processing

Original title:	`The PageRank Citation Ranking: Bringing Order to the Web`
Lowercased:	`the pagerank citation ranking: bringing order to the web`
No symbols:	`the pagerank citation ranking bringing order to the web`

Extraction of Tags

POS-tagged:	`the/DT pagerank/NN citation/NN ranking/NN bringing/VBG order/NN to/TO the/DT web/NN`
Content words:	`pagerank citation ranking order web`

Fig. 2. Sequential processing of textual contents

After removing stop-words, the remaining words are directly used as tags, giving preference to tags stemming from the *title* field over those from *description* field over those from the *abstract* field. Only the first five tags are returned after filtering out duplicates.

External approach For the remaining entries - where the source documents were available - we use a corpus-based approach inspired by standard information retrieval techniques. The idea here is that if a new document is similar to a document from the training corpus, we may re-use the tags that have been added manually to the training document.

Hence, this part of the tagger first ranks all documents from the training data by similarity to the current document. A second step then takes tags from the documents in rank order and returns the first five of them, discarding duplicates.

To do so, all documents in the corpus are first transformed from PDF format to plain text by the PDFBox toolkit[3]. After that, the whole text is segmented into sentences using punctuation information (`.!?;:\n`) and then pre-processed in the same way as described in the internal approach. After removing non-content words, we calculate tf.idf values for each word in the document, resulting in the following mapping:

`<list of tags>` → `<vector of TF/IDF-values>`

A tf.idf value is a value that calculates the relative importance of the word w_i for the current document j, in relation to a set of documents D (see equation 1, where $n_{i,j}$ is the number of occurrence of the word i in document j).

$$tfidf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} * \log \frac{|D|}{|\{d \in D : w_i \in d\}|} \tag{1}$$

This procedure is, of course, carried out only once and the resulting mapping is stored offline. In the actual tagging process, we generate the vector of tf.idf values in the same way for the document to tag and compare the resulting vector to with all document vectors in the corpus. We have experimented with two different similarity measures.

The first variant compares of two documents by the normalized distance between their tf.idf vectors, as shown in equation 2.

$$sim(\boldsymbol{t_0}, \boldsymbol{t_1}) = \frac{\sum_{i=0}^{N-1} |t_0[i] - t_1[i]|}{\sum_{i=0}^{N-1} t_0[i] + t_1[i]} \tag{2}$$

In addition, we implemented *cosine* similarity that measures similarity by the cosine of the angle Θ between the two vectors that the two documents describe (equation 3).

$$sim(\boldsymbol{t_0}, \boldsymbol{t_1}) = \cos \Theta = \frac{\sum_{i=0}^{N-1} t_0[i] t_1[i]}{\sqrt{\sum_{i=0}^{N-1} t_0[i]} \sqrt{\sum_{i=0}^{N-1} t_1[i]}} \tag{3}$$

In our experiments, the normalized distance measure yielded better performance than cosine similarity and consequently we used only the former in the final system.

[3] http://incubator.apache.org/pdfbox/

4.2 The Bookmark Tagger

As in the case for the BibTeX tagger, the bookmark tagger relies on relatively simple heuristics to determine the keywords to recommend. The input data provides two kinds of information, the URL of the web page to tag and a short description which in some cases is identical to the web page's title string.

Processing the URL field In our system, the URL is used to fetch the contents of the actual web page, but since the domain name and path may already contain candidate terms, the URL string is also processed itself, in three sequential steps: *tokenizing*, *filtering*, and *dict/split*.

For the tokenization, the URL is split up at every non-letter non-digit character, such as a forward slash. By matching against a manually crafted blacklist of terms generally considered uninformative, typical artifacts such as "www" or "html" that result from the tokenization process are filtered out. The following examples illustrate these two steps:

Original URL:	`http://www.example.com/new-example/de/bibtex.htm`
Tokenizing:	`http www example com new example de bibtex htm`
Filtering:	`example new example de bibtex`

Original URL:	`http://www.coloradoboomerangs.com`
Tokenizing:	`http www coloradoboomerangs com`
Filtering:	`coloradoboomerangs`

A dictionary of American English together with a list of the names of all articles in the English Wikipedia[4] of 2007 are used to check if the resulting tokens are actual words. The rationale for incorporating Wikipedia is that it gives additional terms from article titles which often are not found in a dictionary, such as, e.g. technical terms ("bibtex"). If a token cannot be found in either list, we try to split the token up into two sub-tokens which, in case they are both contained in the dictionary, are then used instead of the original tokens. This idea is based on the observation that domain names in particular are sometimes a concatenation of two terms.

Applied to the above example, this step generates the following keyword lists:

Keywords:	`example new example de bibtex`
Dict/Split:	`example new example bibtex`

Keywords:	`coloradoboomerangs`
Dict/Split:	`colorado boomerangs`

[4] http://en.wikipedia.org

Processing the description field The description that is part of the input data is tokenized in the same way. However, no further attempts are made to filter out tokenization artifacts or to split the resulting tokens into sub-parts in case they are not contained in the dictionary. In other words, of the above three steps, only *tokenizing* is performed on the description of the bookmark.

Processing the bookmarked web page With the provided URL, the content of the given web page is retrieved at run-time. We do not attempt to detect whether the server returns an actual content page or a specialized message, such as a `HTTP 404 Not Found` error page. After the HTML content of a web page has been downloaded, three different extraction methods are applied: *HTML-meta*, *title*, and *first sentence.*

The first method operates on the head section of the document where it locates and parses the `<meta>` elements "keywords" and "description". The contents of these elements are provided by the author of the HTML document and may contain valuable hints on what the document actually is about. The contents are extracted and then undergo the same tokenizing procedure as described above.

```
             <html>
               <head>
                 <meta name=keywords content="example, sample">
                 <meta name="description" content="A made-up
HTML:                  example webpage">
                 ...
               </head>
               ...
             </html>

Extracting:  example, sample
             a made-up example webpage
Tokenizing:  example sample a made up example webpage
```

Also in the head section is the declaration of the title of the document. This is not only intuitively a good source for relevant keywords, but research in the field of automatic text summarization has also shown in the past that headings contain informative content [9].

```
HTML:        <title>Hello, world - again, an example</title>
Extracting:  hello, world - again, an example
Tokenizing:  hello world again an example
```

Another finding from summarization research is that locational cues work well for determining relevant content words. To apply this insight to the task at hand, the third document-based method tokenizes the first sentence of each HTML document in the same manner described above.

The result of these steps is a set of basic terms. For the final recommendation, two more processing steps are performed, a ranking step and a normalization step.

Ranking keywords For the ranking, each of the previously extracted keywords is described according to four predefined dimensions: Source, InDict, POS, and Navigational.

The values for these dimensions are floating point numbers that represent how valuable a keyword is with respect to being among the recommended tags. For instance, analog to the first step described above, the Source dimension may receive one of the following values:

- URL (= 0.4)
- Description (= 1)
- HTML-Meta (= 1)
- Title (= 0.8)
- First sentence (= 0.9)

The other dimensions describe whether a keyword is found in the English dictionary (and/or list of Wikipedia articles), its part of speech (NN = 1, NNS = 0.9, VBG = 0.8, VERB = 0.5, OTHER = −4) and whether it is found on a blacklist of navigational terms, such as “impress”, “home”, etc. which was created manually by the authors. As with other heuristics, the idea of using such *stigma* word lists can also be tracked back to early summarization research, see e.g. [6].

To rank the keywords, a weighted sum of the four values is computed for each keyword. Since a training corpus was available, good practical weights could have been determined with a machine learning approach. Unfortunately, since time was scarce, we had to estimate sensible weights by hand–inspecting the performance of the tagger on selected samples from the training corpus helped in this part of the development.

Normalization In a final normalization step, a German-English dictionary is used to translate German keywords to English ones and to re-weight those keywords that contain other keywords as sub-strings. In such a case it was speculated that the keyword contained as a sub-string would likely be the more general term and thus its final score was slightly increased.

5 Results

In the evaluation, the results of our tagging system `ARKTiS` had to be compared against the tags that were annotated by a human. The test data were provided 48 hours before the submission deadline. Considering at most five tags per entry, the evaluation uses *precision, recall* and *f-score* values as measurements. In the following, we will present our results, that were achieved on this data and compare them against a baseline system. The baseline system predicts the five most common tags from the training data (Figure 3) to each input entry.

BibTEX:	`ccp jrr programming genetic algorithms`
Bookmarks:	`software indexforum video zzztosort bookmarks`

Fig. 3. Most common tags in the data

The results of the baseline system are presented in Table 1 where we can see a maximum f-score of 0.55%. Comparing this to the results of `ARKTiS` (Table 2), we can see that our system clearly outperforms the baseline with an f-score of almost 11%.

#(tags)	recall	precision	f-score
1	0.0025	0.0114	0.0041
2	0.0025	0.0057	0.0035
3	0.0039	0.0057	0.0046
4	0.0041	0.0046	0.0043
5	0.0053	0.0058	0.0055

Table 1. Baseline

#(tags)	recall	precision	f-score
1	0.0305	0.1072	0.0475
2	0.0595	0.1082	0.0768
3	0.0839	0.1064	0.0938
4	0.1032	0.1032	0.1032
5	0.1179	0.0995	0.1079

Table 2. Results of the `ARKTiS` system

6 Conclusion and Future Work

Our work shows that is is possible to design and implement a basic tag recommender system even with a very limited development time. The two tracks, BibTEX and bookmark tagging, were designed and realized independently but on top of a common, concurrent framework.

The overall task can be considered challenging, especially if results are evaluated on the basis of recall and precision: our final system scored a rather low 11% f-score. The large number of different gold-standard tags makes this number difficult to interpret; however, it is clear that it leaves room for improvement. The winning entry of the 2009 challenged reached an f-score of 19 percent.

In a more detailed analysis, we found that the bookmark module outperformed the BibTEX module to some degree. As described above, the two modules employ rather different approaches, thus a next logical step will be to combine the best ideas from both modules.

The biggest drawback for `ARKTiS` as described in this paper was the fact that we entered the ECML PKDD challenge at a late point. As a consequence, a number of interesting and more sophisticated ideas had to be left out of the system purely due to the lack of implementation time. For instance, the use of tf.idfscores in the BibTeX module is very limited, as is the use of content terms beyond the first sentence in the bookmark module.

At the same time, the ARKTiS system has proven its robustness and will be a good starting point for further research in the area of automatic tagging.

References

1. Oreilly, T.: What is web 2.0: Design patterns and business models for the next generation of software. Social Science Research Network Working Paper Series (2005)
2. Tatu, M., Srikanth, M., D'Silva, T.: Rsdc'08: Tag recommendations using bookmark content. In: ECML PKDD Discovery Challenge 2008. (2008)
3. Fellbaum, C., ed.: WordNet–An Electronic Lexical Database. MIT Press (1998)
4. Vlahavas, I.K.G.T.I.: Multilabel text classification for automated tag suggestion. In: ECML PKDD Discovery Challenge 2008. (2008)
5. Mishne, G.: Autotag: A collaborative approach to automated tag assignment to weblog posts. In: Proceedings of WWW'06, Edinburgh, Scotland (2006)
6. Edmundson, H.P.: New methods in automatic extracting. Journal of the ACM **16**(2) (1969) 264–285
7. Kupiec, J., Pedersen, J., Chen, F.: A trainable document summarizer. In: SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (1995) 68–73
8. Phan, X.: CRFTagger: CRF English POS Tagger. (2006)
9. Mani, I., Maybury, M.T., eds.: Advances in Automatic Text Summarization. The MIT Press, Cambridge, MA (1998)

Tag Recommendation using Probabilistic Topic Models

Ralf Krestel and Peter Fankhauser

L3S Research Center
Leibniz Universität
Hannover, Germany

Abstract. Tagging systems have become major infrastructures on the Web. They allow users to create tags that annotate and categorize content and share them with other users, very helpful in particular for searching multimedia content. However, as tagging is not constrained by a controlled vocabulary and annotation guidelines, tags tend to be noisy and sparse. Especially new resources annotated by only a few users have often rather idiosyncratic tags that do not reflect a common perspective useful for search. In this paper we introduce an approach based on Latent Dirichlet Allocation (LDA) for recommending tags of resources. Resources annotated by many users and thus equipped with a fairly stable and complete tag set are used to elicit latent topics represented as a mixture of description tokens and tags. Based on this, new resources are mapped to latent topics based on their content in order to recommend the most likely tags from the latent topics. We evaluate recall and precision for the bibsonomy benchmark provided within the ECML PKDD Discovery Challenge 2009.

1 Introduction

Tagging systems [1] like Flickr, Last.fm, Delicious or Bibsonomy have become major infrastructures on the Web. These systems allow users to create and manage tags to annotate and categorize content. In *social* tagging systems like Delicious the user can not only annotate his own content but also content of others. The service offered by these systems is twofold: They allow users to publish content and to search for content, thus *tagging* also serves two purposes for the user:

1. Tags help to organize and manage own content, and
2. Find relevant content shared by other users.

Tag recommendation can focus on one of the two aspects. Personalized tag recommendation helps individual users to annotate their content in order to manage and retrieve their own resources. Collective tag recommendation aims at making resources more visible to other users by recommending tags that facilitate browsing and search.

However, since tags are not restricted to a certain vocabulary, users can pick any tags they like to describe resources. Thus, these tags can be inconsistent and idiosyncratic, both due to users' personal terminology as well as due to the different purposes tags fulfill [2]. This reduces the usefulness of tags in particular for resources annotated by only a few users (aka cold start problem in tagging), whereas for popular resources collaborative tagging typically saturates at some point, i.e., the rate of new descriptive tags quickly decreases with the number of users annotating a resource [3].

The main goal of the approach presented in this paper is to overcome the cold start problem for tagging new resources. To this end, we use Latent Dirichlet Allocation (LDA) to elicit latent topics from resources with a fairly stable and complete tag set. The latent topics are represented as a mixture of description tokens like URL, title, and other metadata, and tags, which typically co-occur. Based on this, new resources are mapped to latent topics based on their description in order to recommend the most likely tags from the latent topics.

The remainder of this paper is organized as follows. In Section 2, we define the problem of tag recommendation more formally, and introduce the approach based on LDA. In Section 3 we present our evaluation results. In Section 4 we discuss related work, and in Section 5 we summarize and outline possible future research directions.

2 Tag Recommendation

2.1 Problem Definition

Given a set of resources R, tags T, and users U, the ternary relation $X \subseteq R \times T \times U$ represents the user specific assignment of tags to resources. T consists of two disjoint sets T_{tag} and T_{desc}. T_{tag} contains all user assigned tags, T_{desc} contains the vocabulary of content and meta information, such as abstract or resource description, which is represented as tag assignment by a special "user". A post $b(r_i, u_j)$ for resource $r_i \in R$ and a user $u_j \in U$ comprises all tags assigned by u_j to r_i: $b(r_i, u_j) = \pi_t \sigma_{r_i,u_j} X$[1]. The goal of collective tag recommendation is to suggest tags to a user u_j for a resource r_i based on tag assignments to other resources by other users collected in $Y = \sigma_{r \neq r_i \vee u \neq u_j} \pi_{r,t} X \subseteq R \times T$.

2.2 Latent Dirichlet Allocation

The general idea of Latent Dirichlet Allocation (LDA) is based on the hypothesis that a person writing a document has certain topics in mind. To write about a topic then means to pick a word with a certain probability from the pool of words of that topic. A whole document can then be represented as a mixture of different topics. When the author of a document is one person, these topics reflect the person's view of a document and her particular vocabulary. In the context of tagging systems where multiple users are annotating resources, the

[1] projection π and selection σ operate on multisets without removing duplicate tuples

resulting topics reflect a collaborative shared view of the document and the tags of the topics reflect a common vocabulary to describe the document.

More generally, LDA helps to explain the similarity of data by grouping features of this data into unobserved sets. A mixture of these sets then constitutes the observable data. The method was first introduced by Blei, et. al. [4] and applied to solve various tasks including topic identification [5], entity resolution [6], and Web spam classification [7].

The modeling process of LDA can be described as finding a mixture of topics for each resource, i.e., $P(z \mid d)$, with each topic described by terms following another probability distribution, i.e., $P(t \mid z)$. This can be formalized as

$$P(t_i \mid d) = \sum_{j=1}^{N} P(t_i | z_i = j) P(z_i = j \mid d), \tag{1}$$

where $P(t_i)$ is the probability of the ith term for a given document and z_i is the latent topic. $P(t_i|z_i = j)$ is the probability of t_i within topic j. $P(z_i = j)$ is the probability of picking a term from topic j in the document. These probability distributions are specified by LDA using Dirichlet distributions. The number of latent topics N has to be defined in advance and allows to adjust the degree of specialization of the latent topics. The algorithm has to estimate the parameters of an LDA model from an unlabeled corpus of documents given the two Dirichlet priors and a fixed number of topics. Gibbs sampling [5] is one possible approach to this end: It iterates multiple times over each tag t, and samples a new topic j for the tag based on the probability $P(z_i = j|t, z_{-i})$, where z_{-i} represents all topic-word and document-topic assignments except the current assignment z_i for tag t, until the LDA model parameters converge.

Application to Tagging Systems LDA assigns to each document latent topics together with a probability value that each topic contributes to the overall document. For tagging systems the documents are resources $r \in R$, and each resource in addition to its description from T_{desc} is described by tags $t \in T_{tag}$ assigned by users $u \in U$. Instead of documents composed of terms, we have resources composed of tags. To build an LDA model we need resources and associated tags previously assigned by users. For each resource r we need some posts $b(r, u_i)$ assigned by users $u_i, i \in \{1 \ldots n\}$. Note that for each resource, at least the tag assignments from its description is available. Then we can represent each resource in the system not with its actual tags but with the tags from topics discovered by LDA.

For a new resource r_{new} with few or no posts, we can expand the latent topic representation of this resource with the top tags of each latent topic. To accomodate the fact of some tags being added by multiple users whereas others are only added by one or two users we can use the probabilities that LDA assigns. As formalized in Equation 1 this is a two level process. Probabilities are assigned not only to the latent topics for a single resource but also to each tag within a latent topic to indicate the probability of this tag being part of that particular

Table 1. Top terms composing the latent topic "images" and "tutorial"

Tag	Count	Prob.	Tag	Count	Prob.
images(tag)	243	0.064	tutorial(tag)	640	0.185
photo(tag)	218	0.057	howto(tag)	484	0.140
photography(tag)	205	0.054	tutorial(desc)	204	0.059
image(tag)	188	0.049	tutorials(tag)	184	0.053
photos(tag)	164	0.043	tutorials(desc)	173	0.050
photo(desc)	138	0.036	tips(tag)	126	0.037
images(desc)	106	0.028	reference(tag)	118	0.034
photos(desc)	98	0.026	guide(tag)	79	0.023
flickr(tag)	93	0.024	lessons(tag)	50	0.014
pictures(desc)	61	0.016	tips(desc)	48	0.014
graphics(tag)	49	0.013	wschools(desc)	45	0.013
media(tag)	48	0.013	tutoriel(tag)	33	0.010
art(tag)	48	0.013	comment(tag)	29	0.008

topic. We represent each resource r_i as the probabilities $P(z_j|r_i)$ for each latent topic $z_j \in Z$. Every topic z_j is represented as the probabilities $P(t_n|z_j)$ for each tag $t_n \in T$. By combining these two probabilities for each tag for r_{new}, we get a probability value for each tag that can be interpreted similarly as the tag frequency of a resource. Setting a threshold allows to adjust the number of recommended tags and emphasis can be shifted from recall to precision.

Imagine a resource with the following tags: "photo", "photography", and "howto". Table 1 shows the top terms for two topics related with the assigned tags. The latent topics comprise a broad notion of (digital) photography and the various aspects of tutorial material. Given these topics we can easily extend the current tag set or recommend new tags to users by looking at the latent topics. If LDA assumes that our resource in question belongs to 66% to the "photo"-topic and to 33% to the "howto"-topic, these probabilities are multiplied with the individual topic/tag probabilities, and the top five tags recommended are "tutorial", "howto", "images", "photo", and "photography".

3 Evaluation

We used the data provided by the ECML PKDD Discovery Challenge 2009 to evaluate our approach and fine-tune our parameters. For assessing precision, recall, and f-measure we used the supplied evaluation script.

3.1 Dataset

Our dataset consists of the provided training data for the Discovery Challenge. All experiments were performed on the post-core at Level 2, where all tags, users, and resources occur at least in two posts. To measure the performance of our system, we split the training data into a 90% training set and a 10% test set based on posts (called content IDs in the dataset).

Table 2. Fields parsed to represent a resource

Bibtex		Bookmark
Author	Title	URL
Editor	Description	Description
Booktitle	Journal	Extended
Abstract		

Table 3. Actual tags and recommended tags with computed probablity for URL http://jo.irisson.free.fr/bstdatabase/

Real Tag	LDA Tag	LDA Prob.
latex	bibtex(tag)	0.017
bibtex	latex(tag)	0.017
bibliography	bibtex(desc)	0.014
database	latex(desc)	0.008
engine	theory(desc)	0.005
style	citeulike(desc)	0.005
tex	bibliography(tag)	0.004
reference	database(tag)	0.003
academic	styles(desc)	0.003

For each resource, as defined by the hash values, we build up a textual representation. This representation contains all the tags that were assigned by users in the training set to a particular resource. In addition, we add terms extracted from the description of the resource. More precisely, we tokenized different fields describing a bookmark or bibtex entry. An overview of the fields can be seen in Table 2. Afterwards, we removed stopwords and punctuation marks. Using also the description ensures that we have some terms related to a resource even if no other user before tagged it.

3.2 Results

The tag recommendation algorithm is implemented in Java. We used Mallet [8], which provides an efficient SparseLDA implementation [9], to perform the Latent Dirichlet Allocation with Gibbs sampling. The LDA algorithm takes three input parameters: the number of terms to represent a latent topic, the number of latent topics to represent a document, and the overall number of latent topics to be identified in the given corpus.

Table 3 shows the actual tag distribution for a randomly selected resource (`http://jo.irisson.free.fr/bstdatabase/`), the top tags recommended by LDA with aggregated probabilities, and all the tags provided by a sample user. As the actual tags indicate, the url is a database/latex related site. The tags recommended by LDA come from six latent topics, comprising latex, databases, academia, references, bibliography, and style. These tags characterize the resource quite well.

Table 4. F-measure for different number of recommended tags and different number of LDA topics compared with recommending the most frequent tags (mf)

No. Tags	# LDA topics										
	50	100	200	400	600	800	1000	2500	5000	10000	mf
1	0.170	0.191	0.214	0.229	0.229	0.230	0.229	0.238	0.240	0.235	0.270
2	0.200	0.225	0.248	0.266	0.271	0.271	0.274	0.289	0.288	0.283	0.335
3	0.209	0.233	0.257	0.277	0.282	0.285	0.287	0.302	0.303	0.300	0.362
4	0.209	0.237	0.257	0.279	0.287	0.289	0.292	0.305	0.307	0.303	0.379
5	0.209	0.238	0.258	0.280	0.286	0.291	0.293	0.307	0.307	0.304	0.388

Table 4 compares the f-measure reached for various numbers of latent topics and the baseline which simply recommends the top most frequent tags for each resource (mf) [2]. As can be seen, the best f-measure for LDA is reached between 2500 and 5000 latent topics, but it does not reach the baseline by far. The main reason for this seems to be that the average number of tags per resource is just 10.3 (7.4 distinct tags). This is significantly smaller than the number of (distinct) tokens in a full-text abstract or document, to which LDA has been applied traditionally. Moreover, there are only about 2.8 posts per resource. Thus, there is on the one hand too little co-occurrence evidence for eliciting latent topics, on the other hand there is too little overlap between users on a resource to effectively predict tags via the latent topics of a resource for a new post.

However, to deal with resources that have only few tags associated it makes sense to combine tag recommendations based on most frequent tags with tag recommendations based on latent topics. With $freq(t, r)$ the frequency of tag t annotated for resource r, one estimate of the probability of tag t given resource r is as follows:

$$P_1(t \mid r) = \frac{freq(t, r)}{\sum_{t_i \in r} freq(t_i, r)} \tag{2}$$

This estimate can be combined with the estimate $P_2(t \mid r)$ via latent topics in Equation 1 by means of a mixture:

$$P(t \mid r) = \lambda P_1(t \mid r) + (1 - \lambda) P_2(t \mid r). \tag{3}$$

Table 5 shows that this combination achieves consistently better recall and precision than the individual approaches. The largest gain is achieved for the first recommended tag. Similar accuracies are achieved when varying the mixture parameter λ between 0.3 and 0.9, and for a number of latent topics ≥ 1000.

[2] Unless stated explicitly otherwise, we recommend at least one tag and at most the number of tags annotated to a resource

Table 5. Evaluation results for tag recommendation based on most frequent tags, based on 5000 latent topics, and their combination with $\lambda = 0.5$.

No. Tags	Most Frequent Tags			Latent Topics			Combination		
	Recall	Prec.	F-Meas.	Recall	Prec.	F-Meas.	Recall	Prec.	F-Meas.
1	0.190	0.467	0.270	0.165	0.437	0.240	0.214	0.537	0.306
2	0.274	0.430	0.335	0.232	0.380	0.288	0.302	0.479	0.370
3	0.329	0.403	0.362	0.271	0.343	0.302	0.357	0.441	0.394
4	0.370	0.388	0.379	0.298	0.316	0.307	0.393	0.415	0.404
5	0.400	0.377	0.388	0.316	0.299	0.307	0.421	0.398	0.409

Table 6. Evaluation results DC09 challenge Task 1 based on most frequent tags, based on 5000 latent topics, and their combination with $\lambda = 0.5$.

No. Tags	Most Frequent Tags			Latent Topics			Combination		
	Recall	Prec.	F-Meas.	Recall	Prec.	F-Meas.	Recall	Prec.	F-Meas.
1	0.010	0.032	0.015	0.045	0.158	0.070	0.049	0.169	0.076
2	0.018	0.031	0.022	0.073	0.131	0.094	0.078	0.140	0.100
3	0.022	0.029	0.025	0.092	0.114	0.102	0.099	0.122	0.110
4	0.026	0.028	0.027	0.094	0.112	0.103	0.102	0.120	0.111
5	0.028	0.027	0.028	0.096	0.112	0.103	0.105	0.120	0.112

3.3 Setups and Results for the Challenge Submission

We have submitted tag recommendations for Task 1 and Task 2 in the ECML PKDD Discovery Challenge 2009. Task 1 aims at recommending tags for arbitrary users annotating a resource in 2009 based on tag assignments until 2008. Thus the test data contain tags, resources, and users which are not available in the training data. The topic models have been trained on the full dataset, comprising about 9.3 Mio tokens for 415 K resources. The test set consists of 43002 posts. Table 6 compares the results for 5000 latent topics with the results using the most frequent tags, and the combination of the two approaches [3]. Because for the posts in the test set there are only about 0.3 posts per resource in the training set, recommending only the most frequent tags does not recommend any tags for most of the resources. Consequently, recall and precision are significantly lower than for the approach based on latent topics. The combination of the two approaches achieves slightly but consistently better recall and precision.

Task 2 operates on the post-core at Level 2, where all tags, users, and resources occur at least twice in the training data, which comprises about 750 K tokens for 22389 resources. The test set consists of 778 posts, for which there exist on average 5.8 posts in the training set. Table 7 again compares the results

[3] Our submission to the DC09 challenge was based on 2500 latent topics without combination with most frequent tags, which achieved an F-measure of 0.098.

Table 7. Evaluation results DC09 challenge Task 2 based on most frequent tags, based on 5000 latent topics, and their combination with $\lambda = 0.5$.

No. Tags	Most Frequent Tags			Latent Topics			Combination		
	Recall	**Prec.**	**F-Meas.**	**Recall**	**Prec.**	**F-Meas.**	**Recall**	**Prec.**	**F-Meas.**
1	0.147	0.411	0.216	0.133	0.404	0.200	0.156	0.450	0.232
2	0.223	0.341	0.270	0.204	0.326	0.251	0.252	0.386	0.305
3	0.284	0.305	0.294	0.258	0.281	0.269	0.313	0.339	0.326
4	0.325	0.275	0.298	0.298	0.251	0.272	0.352	0.300	0.324
5	0.357	0.256	0.298	0.319	0.224	0.263	0.386	0.276	0.322

Table 8. Evaluation results for DC09 challenge Task 2 for 5000 latent topics without content

No. Tags	Recall	Precision	F-Measure
1	0.128	0.362	0.189
2	0.191	0.293	0.232
3	0.236	0.254	0.245
4	0.267	0.225	0.244
5	0.299	0.207	0.245

for the two individual approaches and their combination [4]. As is to be expected, recall and precision are much better than for Task 1, because there is more knowledge available about the tagging practices of users. Like in our internal tests tag recommendation based on most frequent tags outperforms the approach based on LDA, and the combination outperforms the individual approaches.

Table 8 shows the results when only tags are used to elicit latent topics. Recall and precision are consistently lower. Thus taking into account the content of resources leads to more effective latent topics for tag recommendation. However, this does not hold for tag recommendation based on most frequent tags. Recommending the most frequent content terms or tags consistently leads to lower precision and recall.

4 Related Work

Tag recommendation has received considerable interest in recent years. Most work has focused on personalized tag recommendation, suggesting tags to the user bookmarking a new resource: This is often done using collaborative filtering, taking into account similarities between users, resources, and tags. [10] introduces an approach to recommend tags for weblogs, based on similar weblogs tagged by the same user. Chirita et al. [11] realize this idea for the personal desktop, recommending tags for web resources by retrieving and ranking tags from

[4] Our submission to the DC09 challenge was based on 5000 topics without combination with the most frequent tags and no limit on the number of recommended tags. This achieved an F-measure of 0.258.

similar documents on the desktop. [12] aims at recommending a few descriptive tags to users by rewarding co-occuring tags that have been assigned by the same user, penalizing co-occuring tags that have been assigned by different users, and boosting tags with high descriptiveness (TFIDF).

Sigurbjörnsson and van Zwol [13] also look at co-occurence of tags to recommend tags based on a user defined set of tags. The co-occuring tags are then ranked and promoted based on e.g. descriptiveness. Jaeschke et al. [14] compare two variants of collaborative filtering and Folkrank, a graph based algorithm for personalized tag recommendation. For collaborative filtering, once the similarity between users on tags, and once the similarity between users on resources is used for recommendation. Folkrank uses random walk techniques on the user-resource-tag (URT) graph based on the idea that popular users, resources, and tags can reinforce each other. These algorithms take co-occurrence of tags into account only indirectly, via the URT graph. Symeonidis et al. [15] employ dimensionality reduction to personalized tag recommendation. Whereas [14] operate on the URT graph directly, [15] use generalized techniques of SVD (Singular Value Decomposition) for n-dimensional tensors. The 3 dimensional tensor corresponding to the URT graph is unfolded into 3 matrices, which are reduced by means of SVD individually, and combined again to arrive at a more dense URT tensor approximating the original graph. Tag recommendation then suggests tags to users, if their weight is above some threshold.

An interactive approach is presented in [16]. After the user enters a tag for a new resource, the algorithm recommends tags based on co-occurence of tags for resources which the user or others used together in the past. After each tag the user assigns or selects, the set is narrowed down to make the tags more specific. In [17], Shepitsen et al. propose a recommendation system based on hierarchical clustering of the tag space. The recommended resources are identified using user profiles and tag clusters to personalize the recommendation results. Note that they use tag clusters to recommened resources whereas we use LDA topics, which can be considered clusters, to recommend tags.

[3] introduce an approach to tag recommendation using association rules. Resources are regarded as baskets consisting of tags, from which association rules of the form $T_1 \rightarrow T_2$ are mined. On this basis tags in T_2 are recommended whenever the resource contains all tags in T_1. A comparison of this approach with the approach presented in this paper can be found in [18].

When content of resources is available, tag recommendation can also be approached as a classification problem, predicting tags from content. A recent approach in this direction is presented in [19]. They cluster the document-term-tag matrix after an approximate dimensionality reduction, and obtain a ranked membership of tags to clusters. Tags for new resources are recommended by classifying the resources into clusters, and ranking the cluster tags accordingly.

5 Conclusions and Future Work

In this paper we have presented and evaluated the use of Latent Dirichlet Allocation for collective tag recommendation. Using selected features from the content of resources, tags, and users, we elicit latent topics that comprise typically co-occuring tags and users. On this basis we can recommend tags for new users and resources by mapping them to the latent topics and choosing the most likely tags from the topics. The approach complements simple tag recommendation based on most frequent tags especially for new resources with only few posts. Consequently, combining tag recommendations based on latent topics with tag recommendations based on most frequent tags outperforms the individual approaches.

For future work we want to investigate approaches that take into account individual tagging practices for personalized tag recommendation.

Regarding data sets, we also want to experiment with datasets from different domains, to check whether photo, video, or music tagging sites show different system behavior influencing our algorithms. Another interesting direction we want to follow is to apply LDA not only for tag recommendation but to employ it in the context of recommending resources.

6 Acknowledgments

This work was supported in part by the EU project IST 45035 - Platform for searcH of Audiovisual Resources across Online Spaces (PHAROS).

References

1. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Ht06, tagging paper, taxonomy, flickr, academic article, to read. In Wiil, U.K., Nürnberg, P.J., Rubart, J., eds.: HYPERTEXT 2006, Proceedings of the 17th ACM Conference on Hypertext and Hypermedia, August 22-25, 2006, Odense, Denmark, New York, NY, USA, ACM (2006) 31–40
2. Golder, S., Huberman, B.A.: Usage patterns of collaborative tagging systems. Journal of Information Science **32**(2) (April 2006) 198–208
3. Heymann, P., Ramage, D., Garcia-Molina, H.: Social tag prediction. In: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2008) 531–538
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3** (January 2003) 993–1022
5. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proc Natl Acad Sci U S A **101 Suppl 1** (April 2004) 5228–5235
6. Bhattacharya, I., Getoor, L.: A latent dirichlet model for unsupervised entity resolution. In: SIAM Conference on Data Mining (SDM). (April 2006) 47–58
7. Bíró, I., Siklósi, D., Szabó, J., Benczúr, A.A.: Linked latent dirichlet allocation in web spam filtering. In: AIRWeb '09: Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web, New York, NY, USA, ACM (2009) 37–40

8. McCallum, A.K.: Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu (2002)
9. Yao, L., Mimno, D., McCallum, A.: Efficient methods for topic model inference on streaming document collections. In: KDD '09: Proceedings of the 15th ACM SIGKDD conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM (2009)
10. Mishne, G.: Autotag: a collaborative approach to automated tag assignment for weblog posts. In: WWW '06: Proceedings of the 15th international conference on World Wide Web, New York, NY, USA, ACM (2006) 953–954
11. Chirita, P.A., Costache, S., Nejdl, W., Handschuh, S.: P-tag: large scale automatic generation of personalized annotation tags for the web. In: WWW '07: Proceedings of the 16th international conference on World Wide Web, New York, NY, USA, ACM (2007) 845–854
12. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. In: Proceedings of Collaborative Web Tagging Workshop at 15th International World Wide Web Conference. (2006)
13. Sigurbjörnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: WWW '08: Proceeding of the 17th international conference on World Wide Web, New York, NY, USA, ACM (2008) 327–336
14. Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In Kok, J.N., Koronacki, J., de Mántaras, R.L., Matwin, S., Mladenic, D., Skowron, A., eds.: Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007, Proceedings. Volume 4702 of Lecture Notes in Computer Science., Heidelberg, Germany, Springer (2007) 506–514
15. Symeonidis, P., Nanopoulos, A., Manolopoulos, Y.: Tag recommendations based on tensor dimensionality reduction. In: RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems, New York, NY, USA, ACM (2008) 43–50
16. Garg, N., Weber, I.: Personalized, interactive tag recommendation for flickr. In: RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems, New York, NY, USA, ACM (2008) 67–74
17. Shepitsen, A., Gemmell, J., Mobasher, B., Burke, R.D.: Personalized recommendation in social tagging systems using hierarchical clustering. In Pu, P., Bridge, D.G., Mobasher, B., Ricci, F., eds.: Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008, New York, NY, USA, ACM (2008) 259–266
18. Krestel, R., Fankhauser, P., Nejdl, W.: Latent dirichlet allocation for tag recommendation. In: 3rd ACM Conference on Recommender Systems, New York City, USA, October 22-25, 2009, to appear. (2009)
19. Song, Y., Zhuang, Z., Li, H., Zhao, Q., Li, J., Lee, W.C., Giles, C.L.: Real-time automatic tag recommendation. In: SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2008) 515–522

A Probabilistic Ranking Approach for Tag Recommendation

Zhen Liao[1], Maoqiang Xie[2], Hao Cao[2], Yalou Huang[2]

[1] College of Information Technology Science, Nankai University, Tianjin, China
{liaozhen, caohao}@mail.nankai.edu.cn
[2] College of Software, Nankai University, Tianjin, China
{xiemq, huangyl}@nankai.edu.cn

Abstract. Social Tagging is a typical Web 2.0 application for users to share knowledge and organize the massive web resources. Choosing appropriate words as tags might be time consuming for users, thus a tag recommendation system is needed for accelerating this procedure. In this paper we formulate tag recommendation as a probabilistic ranking process, especially we propose a hybrid probabilistic approach which combines language model and statistical machine translation model. Experimental results validate the effectiveness of our method.

1 Introduction

Folksonomy is a way to categorize Web resources via utilizing the "wisdom" of web users, nowadays it is existing in many web applications such as Delicious[3], Filckr[4], Bibsonomy[5]. One user could create and share her knowledge during the tagging on resources that are interesting to her. Web resources come in many forms, for example, one resource could be a Web pages, a published paper, or a book. To tag a resource with appropriate words is not so easy and might cost lots of time. Thus a tag recommendation system is needed for easing the time-consuming step. Typically a recommendation system would suggest 5 or 10 tags to the user for a given resource. Those suggested tags would help one user to think about eligible words and to realize the interesting aspects concerned by others. To solve the problems, ECML PKDD holds the second round discovery challenge[6] of tag recommendation. This paper presents a probabilistic ranking approach submitted to the challenge.

Given a resource, users choose tags by different aspects of the resource and their specific interests. To pick up a tag from the entire tag set and assign it to the resource could be formulated as following process: given a resource and a user, ranking the tags by their *relevance* to the resource and user. Here relevance denotes the 'value' of how likely the user would label this tag on this resource.

[3] http://del.icio.us
[4] http://www.flickr.com/
[5] http://www.bibsonomy.org/
[6] http://www.kde.cs.uni-kassel.de/ws/dc09

We suppose a tag recommendation system works best while recommending tags are sorted by the relevance and then suggested to the user.

In this paper, the datasets provided by Bibsonomy is a set of *post*. Each *post* denotes a triple {user, resource, a set of tag}. A resource type could be *bookmark* or *bibtex*, where bookmark is Web page and bibtex is publication. Both bookmark and bibtex resources contain many fields: URL, description, etc. The textural information in the fields could be merged as a pseudo document.

A natural way of choosing tags is to select words from the pseudo document of given resource. A TF-like maximum likelihood method could reach the goal. The important problem is that maximum likelihood model could not generate tags which are meaningful but not existing in the document. To incorporate previously popular tags and tags preferred by a user, a tag recommendation model could be formulate into language model smoothed via Jelinek-Mercer method as described in Section 3.2. However, the language modeling approach could not learn the word-tag relateness which reflects how other users choose tags for those words in the document. Since the textural information existing in a post could be considered as a parallel corpus - {words in document, tags}, we propose to use the statistical machine translation approach to learn the translation probability from words to tags.

Finally, we propose a candidate set based tag recommendation algorithm which generates candidate tags from the textual fields of a resource using maximum likelihood and statistical machine translation model. The effectiveness of our approach is validated on the bookmark and bibtex tagging test datasets provided by Bibsonomy. While textural content of a bookmark resource is inadequate, we utilize the tags used within same Domain to extend the candidate set. We also found simple co-occurrence based translation probability estimation performs as good as IBM Model 1 [6] which uses the EM algorithm to learn the translation probability. An advantage of co-occurrence based approach is its convenience for handling with new training data, since training the model is just counting the co-occurrence of words and tags. However, EM-based approach needs to re-train translation model though iterations which might be time consuming for large scale dataset.

The rest of this paper is organized as follows. In Section 2 the related work is surveyed. In Section 3 our content based tag recommendation models are presented, and the recommendation algorithm is described in Section 4. In Section 5 we descrbe the data format and preprocessing step, and experimental results are reported in Section 6. Finally in Section 7 we conclude this paper and give out some possible future research issues.

2 Related Work

Most of existing tag recommendation approaches are based on the textual information of the resource and previous interests of users. Up to now, the information retrieval, data mining and natural language processing techniques have been used for solving the tag recommendation problem.

Heymann et al. [1] use one of the largest crawls from the social bookmarking system Delicious and presents studies of the factors which could impact the performance of tag prediction. The predictability of tags is measured by some method such as entropy based metric. The tag-based association rule is proposed to assist tag predictions. The method of learning the word-tag relateness via association rule needs to tune the confidence and support to find meaningful rules, but we transfer it into the translation probability which could get the converged solution without tuning.

Tatu et al [2] uses document and user models derived from the textual content associated with URLs and publications by social bookmarking tool users. The natural language processing techniques are used to extract the concept(Part of Speech, etc.) from the textual information. WordNet[7] are used to stem the concepts and link synonyms. The difference between our work and theirs is that they expand the concept via WordNet, but do not have the word-to-tag translation probability such as from 'eclipse' to 'java'.

Lipczak [3] focus on the folksomomies towards individual users, and proposed a three step tag recommendation system which conducts the Personmony based filtering using previously used tags of users after the extraction and retrieving of tags. The recommendation approach in [3] is similar with our work, but the scores of candidate tags are computed differently. They use the multiply strategy for different factors, but we conduct a weighted sums in which the weight could be set to prefer different components. Besides, we use the statistical machine translation approach to learn the word-tag relateness which is different from model proposed in [3].

Language modeling approach [4] has been applied in Information Retrieval with lots of smoothing strategies [5]. The statistical machine translation approaches [6] shows its theoretical soundness and effectiveness in translation, and Berger et al [7] and Xue et al [8] incorporate the statistical translation approaches into information retrieval and automatic question answering fields. The theoretical soundness and effectiveness make it stable to adopt the language modeling and statistical machine translation approach into tag recommendation. The statistical machine translation approach also naturally solve the problem of learning the word-tag relateness of sharing the common tagging knowledge among users.

3 Content Based Tag Recommendation Models

3.1 Problem Definition

In this paper, a tag set is denoted as $\mathbf{t} = \{t_i\}_{i=1}^{Q}$ where t_i is a single word or term and Q is the number of tags in $\mathbf{t}$.

The tag recommendation task is to suggest a tag set $\mathbf{t}$ for a user U_k while given a bookmark/publication resource R_j which might be a web page, a book or paper etc. The resource R_j contains several fields such as URL, title, description and we denote the resource content as a pseudo document D_j.

[7] http://wordnet.princeton.edu

Suppose the recommendation system is required to suggest N tags, it is to find N tags $\{t_i\}_{i=1}^N$ from the entire tag sets with the biggest probability $p(t_i|U_k, D_j)$.

For solving the task, a training set $\mathbf{S} = \{S^i\}_{i=1}^K$ is given, where S^i specifies a triple $\{\mathbf{t}^i, U^i, D^i\}$. The $\mathbf{t}^i$ is a tag set, $U^i \in \mathcal{U} = \{U_1, ..., U_M\}$ is a user and $D^i \in \mathcal{D} = \{D_1, ..., D_N\}$ is a resource . Then we can learn a tag recommendation model $\mathcal{M}$ from $\mathbf{S}$.

At the testing stage, a testing set $\mathbf{T} = \{T^j\}_{j=1}^P$ where $T^j = \{U^j, D^j\}$ is given. The model $\mathcal{M}$ is asked to suggest tag set $\mathbf{t}^j$ for each T^j. After that a groudtruth tag sets $G = \{\mathbf{g}^j\}_{j=1}^P$ is used to judge the recommendations $\{\mathbf{t}^j\}_{j=1}^P$, and the performance is get via some evaluation measures such as Precision, Recall and F-measure.

For a specific user U_k, she would have her preference in choosing a word t_i as a tag, and if we have this user's information in the training set $\mathbf{S}$, we can formulate this preference as $P(t_i|U_k) = \frac{c(t_i;U_k)}{|U_k|}$ where $c(t_i; U_k)$ is frequency of t_i be used by user U_k, and $|U_k|$ is total frequency of all tags used by U_k.

We define the tag generating probability a tag t_i for a given user and document tuple $\{U_k, D_j\}$ as:

$$P(t_i|D_j, U_k) = (1 - \beta)P(t_i|D_j) + \beta P(t_i|U_k) \tag{1}$$

Where β is a trade-off parameter between the resource content and user.

Following we will introduce language model and statistical machine translation approaches for estimating $P(t_i|D_j)$, and then we will combine them into our final model.

3.2 Language Modeling Approach

A natural and simple way to estimate $P(t_i|D_j)$ is to use the maximum likelihood approach as:

$$P_{ml}(t_i|D_j) = \frac{c(t_i; D_j)}{|D_j|} \tag{2}$$

Where $c(t_i; D_j)$ is occurrence of t_i in D_j, and $|D_j|$ is document length of D_j. The shortcoming of the maximum likelihood estimation is that it could not generate tag which does not exist in D_j, thus we introduce language model smoothed via Jelinek-Mercer method [5] as:

$$P_{lm}(t_i|D_j) = (1 - \lambda)P_{ml}(t_i|D_j) + \lambda P_{ml}(t_i|C) \tag{3}$$

Where λ is the smoothing parameter, and C corresponds to the entire corpus. Actually the smoothing term $P(t_i|C)$ could be formulated as the probability of the word t_i be used as a tag. We define $P(t_i|C)$ as $\frac{c(t_i)}{\#tags}$ where $\#tags$ is the total number of tags in the training set $\mathbf{S}$. The language modeling approach (3) could be considered as the incorporation of words in the document and previously popular tags of all users.

3.3 Statistical Machine Translation Approach

However, the language modeling approach has not considered word-tag relateness which would be important for tag recommendation. For solving the problem, we further introduce the Statistical Machine Translation(SMT) approach [6] [7] [8] for estimating the probability $P(t_i|D_j)$:

$$P_{smt}(t_i|D_j) = \frac{|D_j|}{|D_j|+1} P_{tr}(t_i|D_j) + \frac{1}{|D_j|+1} P(t_i|null) \tag{4}$$

Where $P(t_i|null)$ could be regarded as the background smoothing model $P(t_i|C)$, and a more detailed comparison them could be found in [8]. $P_{tr}(t_i|D_j)$ is the translation probability from D_j to ti as following:

$$P_{tr}(t_i|D_j) = \sum_{w \in D_j} P_{tr}(t_i|w) P_{ml}(w|D) \tag{5}$$

To learn the word-word transition probability $P_{tr}(t_i|w)$, the EM algorithm could be used. The detail of EM algorithm of learning the word-tag relateness $P(t_i|w)$ in Statistical Machine Translation(SMT) Model is described in [6]. In the training set $\mathbf{S} = \{S^j\}_{j=1}^{K}$, the parallel corpus of tag and document as $S^j = \{\mathbf{t}^j, D^j\}$ is utilized, and the EM step for learning $P(t_i|w)$ can be formulated as:

E-Step:

$$P^1_{tr}(t_i|w) = \delta_w^{-1} \sum_{j=1}^{K} c(t_i, w; \mathbf{t}^j, D^j) \tag{6}$$

M-Step:

$$c(t_i, w; \mathbf{t}^j, D^j) = \frac{P(t_i|w)}{P(t_i|w_1) + ... + P(t_i|w_o)} \#(t_i, \mathbf{t}^j) \#(w, D^j) \tag{7}$$

In Equation (6) $\delta_w^{-1} = \sum_{t_i} \sum_{j=1}^{K} c(t_i, w; \mathbf{t}^j, D^j)$ is the normalization factor. In Equation (7) $\{w_1, ..., w_o\}$ is words contained in D^j, $\#(t_i, \mathbf{t}^j)$ and $\#(w, D^j)$ is the number of t_i in $\mathbf{t}^j$ and number of w in D^j. The convergency of this EM algorithm is proved in [6].

In this paper, we also find that the co-occurrence based translation probability could be helpful in tag recommendation, and we denote it as:

$$P^2_{tr}(t_i|w) = \frac{\sum_{j=1}^{K} \#(t_i; \mathbf{t}^j) \cdot \#(w; D^j)}{\sum_{j=1}^{K} \#(w; \mathbf{t}^j, D^j)} \tag{8}$$

Where $\#(t_i; \mathbf{t}^j)$ denotes the number of tag t_i exists in $\mathbf{t}^j$ and the same to $\#(w; D^j)$. This model could be regarded as a simple approximation of the EM based translation model, and it is also effective. Note that the EM based translation probability is denoted as $P^1_{tr}(t_i|w)$ whereas the co-occurrence based translation probability is denoted as $P^2_{tr}(t_i|w)$ hereafter.

3.4 Final Model

Now we combine above methods together to get our final model:

$$\begin{aligned} P_{final}(t_i|D_j, U_k) =& \lambda P(t_i|C) + \beta P(t_i|U_k) \\ &+ \alpha P_{ml}(t_i|D_j) + \gamma \sum_w P_{tr}(t_i|w) P_{ml}(w|D) \end{aligned} \tag{9}$$

Where $\lambda + \beta + \alpha + \gamma = 1$ and P_{tr} could be P_{tr}^1 or P_{tr}^2. Tuning these four parameters is not easy, and thus we split both Cleaned Dump and Post Core dataset into a training set and a validation set respectively, train the model on the training set and set parameters empirically several times for choosing one with better performance on the validation set. We do not illustrate the detail due to space restriction, and in the experiments we found the performance is relatively well while $\lambda = 0.15$, $\beta = 0.1$, $\alpha = 0.05$, $\gamma = 0.7$. We use these parameters with Cleaned Dump dataset as our final training set for the challenge.

4 Candidate Set based Tag Recommendation Algorithm

Since the task of tag recommendation is to suggest tags for given document and user, it is different from the task of Information Retrieval [7] or Question Answering [8] where the query/question is given for finding the relevant documents/answers.

Given a document D_j and user U_k, we firstly find a recommendation tag candidate set CS from the words in D_j, and we also add the top L related words by $P_{tr}(t|w)$ for every word w in D^j. Then we compute the $P(t_i|D_j, U_k)$ for each tag $t_i \in CS$. Finally we sort the tags descending according to $P(t_i|D_j, U_k)$, and return the top N tags as required by the application system. The L is set to be 20 and N is set to 5 in the experiments. In summary, we get this algorithm in Table 1.

5 Data Preparing and Preprocessing

The dataset we used is download from ECML PKDD Discovery Challenge 2009[8] which is provided by BibSonomy[9]. There are two datasets: **Cleaned Dump** and **Post Core**. The Cleaned Dump contains all public bookmarks and publication posts of BibSonomy until (but not including) 2009-01-01. The Post Core is a subset of the Cleaned Dump, it removes all users, tags, and resources which appear in only one post from Cleaned Dump. Brief statistics of Cleaned Dump and Post Core could be found in Table 2. One tag assignment means one user choose a tag for a resource, and thus one posts could have several tag assignments. The number of posts are shown for bookmark, bibtex, and entire set. The bookmark and bibtex are seperated by '/', and the entire set are illustrated after ':'.

[8] http://www.kde.cs.uni-kassel.de/ws/dc09

[9] http://www.bibsonomy.org/

Table 1. Candidate Set based Tag Recommendation Algorithm

```
Input: testing sample: $T^j = \{D^j, U^j\}$, threshold $N$ and $L$
Output: top $N$ tags $\mathbf{t} = \{t_1, ..., t_N\}$

candidate set $CS \leftarrow \emptyset$
for $w$ in $D^j$
    add $w$ into $CS$
    add top $L$ tags $t$ into $CS$ according to $P(t|w)$
end for
for each word $t_k \in C$
    compute $P(t_k|D^j)$ using (9)
end for
sort $t_k \in CS$ with $P(t_k|D^j)$ in descending order
return top $N$ tags in $C$ as $\mathbf{t}$
```

Table 2. Statistics of Cleaned Dump & Post Core datasets

	tag assignments	number of posts	number of users
Cleaned Dump	1,401,104	263, 004 / 158, 924 : 421, 928	3, 617
Post Core	253,615	41,268 / 22,852 : 64, 120	1, 185

There are three tables *tas*, *bookmark*, and *bibtex* in the dataset. The fields of these tables are list in Table 3. For bookmark resource the field 'content_type' is 1 and that of bibtex resource is 2. The fields in bold are used to generate the pseudo document D_j and the tags $\mathbf{t}^j$ in the training process.

Table 3. Fields of Three Dataset Tables

table	fields
tas	user, **tag**, content_type, content_id, date
bookmark	content_id, url_hash, URL, **description**, **extended description**, date
bibtex	content_id, journal, chapter, edition, month, day, **booktitle**, howPublished, institution, organization, publisher, address, school, series, bibtexKey, url, type, **description**, annote, note, pages, bKey, number, crossref, misc, bibtexAbstract, simhash0, simhash1, simhash2, entrytype, **title**, author, editor, year

We firstly remove the stop words in the bookmark and bibtex table since they are seldom used as tags and usually meaningless. The stop word list are download from Lextek[10]. Note that we do not remove stop words in the tas file, and the top 5 stop words exist in Post Core and their frequency could be found in Table 4. There are totally 19, 647 and 2, 513 stop word tag assignments in Cleaned Dump and Post Core, corresponds to 1.39% and 0.99% respectively.

[10] http://www.lextek.com/manuals/onix/stopwords1.html

In contrast, the total frequency of stop words in pseudo documents of Cleaned Dump and Post Core are over 588, 907 and 61, 113, which suggest not to consider stop words as tags in most cases.

Table 4. Top 5 stop words in tags of Cleaned Dump & Post Core

dataset	top 5 stop words and their frequency in tags
Cleaned Dump	all:3105 of:1414 and:1227 best:1124 three:1081 c:806
Post Core	all:655 open:211 c:165 best:152 work:77

In Table 5 we list out the top 10 tags in Cleaned Dump and Post Core. We could see later that the co-occurrence based translation model are likely to generate words which appear more times.

Table 5. Top 10 Tags and their Frequency

Cleaned Dump	bookmarks:52795 → zzztosort:11839 → video:10788→ software:10171 → programming:9491 → indexforum:9183 → web20:8777 → books:7934 → media:7149 → tools:6903
Post Core	web20:4474 → software:3867 → juergen:3092 → tools:3058 → web:2930 → tagging:2196 → semanticweb:2055 → folksonomy:1944 → search:1896 → bookmarks:1840

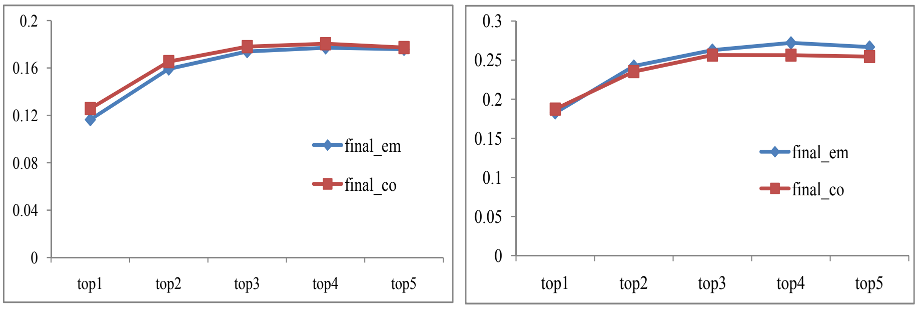
6 Experimental Result

6.1 Tagging Performance

The evaluation measure in following experiments are widely used Precision, Recall, and F1-measure. The testing datasets are released by ECML-PKDD challenge in tasks. There are 2 tasks: task 1 and task 2, where task 1 is for content based tag recommendation, and task 2 is for graph based tag recommendation[11]. In task 1 the user, resource of a post might not exist before, so the content information of the resource would be critical for tag recommendation. In task 2 user, resource, and tags of each post in the test data are all contained in the Post Core dataset, thus it intends for methods relying on the graph structure of the training data only.

We use the whole Cleaned Dump dataset as the training set to train the model and test the performance of our model on both tasks. For choosing the parameters, we set $\alpha = 0.15, \lambda = 0.05, \beta = 0.1, \gamma = 0.7$ as mentioned before in Section 3.4. The results are shown in Figure 1. The final_em denotes final model with P_{tr}^1(EM-based), and final_co denotes final model with P_{tr}^2(Co-occurrence based). The x-axis is the top position and y-axis is the f-measure.

[11] http://www.kde.cs.uni-kassel.de/ws/dc09



(a) On Task 1 Testing Set (b) On Task 2 Testing Set

Fig. 1. Performance of Selected Models

The results indicates that although P_{tr}^{2}(Co-occurrence) is more simpler, it is comparable to P_{tr}^{1}. In our previous experiment, we also found sometimes the textual information from the bookmark resource are not adequate enough to generate some tags in the post and it needs to be expanded. Instead of using extrinsic resource such as WordNet, we aggregate the tags in the same web site domain for bookmark resource, and use them to expand the recommendations. The reason we don't expand the term in bibtex is because resources in bibtex are publication and the web site provide less information about tags. Also, trying other tag expansion methods would be our future work. We formulate this expansion as $P(t_i|Site)$, and the recommendation model for bookmark would become:

$$\begin{aligned} P_{final\_ex}(t_i|D_j, U_k) = &\lambda P(t_i|C) + \beta P(t_i|U_k) + \alpha P_{ml}(t_i|D_j) \\ &+ \gamma \sum_{w} P_{tr}(t_i|w) P_{ml}(w|D) + \theta P(t_i|Site) \end{aligned} \tag{10}$$

For illustrate the expansions of different domains, we sample some domains and their top used tags with the probability in Table 6.

Table 6. Sample Domains with Top 5 used tags

domain	tags and their previously used probability
www.apple.com	apple:0.17 mac:0.13 software:0.09 osx:0.07 bookmarks:0.07
answers.yahoo.com	knowledge:0.14 yahoo:0.14 web20:0.07 all:0.07 answer:0.07
ant.apache.org	java:0.19 ant:0.17 programming:0.07 apache:0.07 tool:0.07
picasa.google.com	google:0.21 image:0.14 download:0.14 linux:0.14 picasa:0.14
research.microsoft.com	microsoft:0.10 research:0.09 people:0.04 social:0.04 award:0.03
www.research.ibm.com	ibm:0.11 datamining:0.07 software:0.04 machinelearning:0.04 journal:0.04

After the tag expansion via the URL domain, the candidates set CS for the recommendation will have top used tags in the same domain of D_j. The performance of (10) with the expansions on the testing set are shown in Table 7 and 8. The performance are shown for only bookmark, only bibtex, and on entire set. The bookmark and bibtex are seperated by '/', and the entire set are illustrated after ':'. We choose the co-occurrence based model P^2_{tr} in the competition, and actually the performance in terms of F-measure at 5 is also good when using EM-based model P^1_{tr}. The F-measure of EM-based model with the same parameters as Table 7 for task 1 and task 2 are shown in Table 9. We can find that the P^2_{tr} and P^1_{tr} are comparable once again, on F-measure at 1, the Co-occurrence based model are better, but on F-measure at 5, the EM-based model are better.

Table 7. Performance for Task 1 ($\alpha = 0.15, \lambda = 0.05, \beta = 0.05, \gamma = 0.5, \theta = 0.25$ for bookmark, $\alpha = 0.15, \lambda = 0.05, \beta = 0.1, \gamma = 0.7$ for bibtex with P^2_{tr})

TOP N	Recall	Precision	F-Measure
1	0.0702 / 0.0975 : 0.0809	0.2232 / 0.3056 : 0.2556	0.1067 / 0.1477 : 0.1229
2	0.1116 / 0.1584 : 0.1300	0.1905 / 0.2584 : 0.2172	0.1406 / 0.1961 : 0.1624
3	0.1412 / 0.2011 : 0.1648	0.1664 / 0.2251 : 0.1895	0.1525 / 0.2120 : 0.1760
4	0.1636 / 0.2318 : 0.1904	0.1489 / 0.2000 : 0.1690	0.1556 / 0.2143 : 0.1787
5	0.1810 / 0.2563 : 0.2106	0.1339 / 0.1802 : 0.1521	0.1536 / 0.2111 : 0.1762

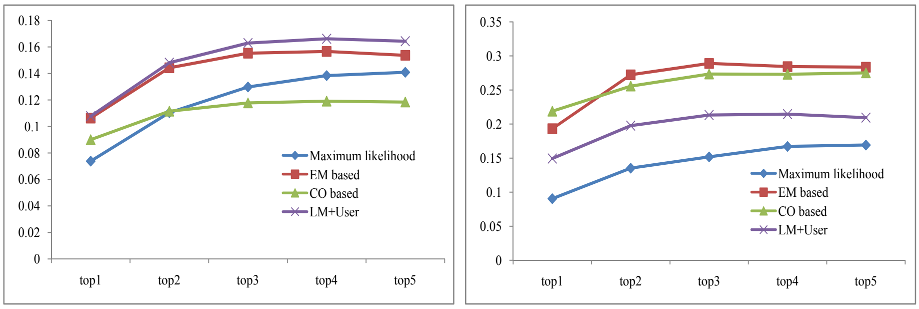
Table 8. Performance on Task 2 data($\alpha = 0.15, \lambda = 0.05, \beta = 0.05, \gamma = 0.5, \theta = 0.25$ for bookmark, $\alpha = 0.15, \lambda = 0.05, \beta = 0.1, \gamma = 0.7$ for bibtex with P^2_{tr})

TOP N	Recall	Precision	F-Measure
1	0.1399 / 0.1215 : 0.1297	0.4063 / 0.3666 : 0.3843	0.2073 / 0.1823 : 0.1938
2	0.2136 / 0.1919 : 0.2016	0.3444 / 0.3086 : 0.3246	0.2625 / 0.2365 : 0.2485
3	0.2887 / 0.2379 : 0.2605	0.3093 / 0.2676 : 0.2862	0.2977 / 0.2517 : 0.2726
4	0.3212 / 0.2848 : 0.3010	0.2630 / 0.2454 : 0.2532	0.2883 / 0.2636 : 0.2749
5	0.3532 / 0.3220 : 0.3359	0.2346 / 0.2237 : 0.2285	0.2812 / 0.2639 : 0.2718

Next we conduct the experiment on each component of our final model (9), the document maximum likelihood method, language model('LM + User Model'), the EM-based translation model $P^1_{tr}(t_i|w)$, and co-occurrence based translation model $P^2_{tr}(t_i|w)$ are chosen. In the 'LM + User Model' we set the parameters $\alpha = 0.5, \lambda = 0.3, \beta = 0.2, \gamma = 0$. It could be considered as the language model which incorporates the maximum likelihood, the previously tag probability in the whole corpus, and the user's preference model. The performance on both testing datasets of task 1 and task 2 are illustrated in Figure 2. The x-axis is the top position from top1 to top5 and the y-axis is the value of F-Measure. We only list out the F1 measure because it reflects both precision and recall.

Table 9. Performance of ($\alpha = 0.15, \lambda = 0.05, \beta = 0.05, \gamma = 0.5, \theta = 0.25$ for bookmark, $\alpha = 0.15, \lambda = 0.05, \beta = 0.1, \gamma = 0.7$ for bibtex with P^1_{tr})

TOP N	task 1 F-Measure	task 2 F-measure
1	0.1167	0.1909
2	0.1593	0.2548
3	0.1745	0.2790
4	0.1778	0.2866
5	0.1770	0.2833



(a) On Task 1 Testing Set

(b) On Task 2 Testing Set

Fig. 2. Performance of Selected Models

From the experimental results we can see the translation based models are better than maximum likelihood method and 'LM + User Model' in task 2. The co-occurrence based model are worst in task 1, and the EM-based model is better than co-occurrence based model on both task. We analyze the results of co-occurrence based model on task 1 and find many recommendations are common used tags, because the co-occurrence based model would prefer to generate those tags occurred more times before. This suggest that if the resource/users have been seen before, thus the co-occurrence based model would perform well, if not, then it is better to choose EM based model. The 'LM + User Model' perform best on task 1, but the performance is still lower than that in Table 7, and also, 'LM + User Model' performs worse than translation models on task 2.

For comparison between EM-based and co-occurrence based model, we pick out several words w with their top translating words t_i in both $P^1_{tr}(t_i|w)$(EM-based) and $P^2_{tr}(t_i|w)$(Co-occurrence based). The sampling words could be found in Table 10. We could find that in EM-based translation model, the words are most likely to translate into itself. It indicates that we could consider the EM-based translation model as the combination of the maximum likelihood which only generates the word it self and the co-occurrence based translation model which has higher probability to generate other words as tags. The co-occurrence model are likely to generate those popular tags in the corpus, such as 'tools', 'software', 'social'.

Table 10. Sampled Words with their top tags t_i: $P^1_{tr}(t_i|w)$(EM); P^2_{tr}(CO)

w	model	Top tags t_i with highest probability $P_{tr}(t_i\|w)$
web	EM	web:0.36 web20:0.26 semanticweb:0.12 semantic:0.01970 ajax:0.02
	CO	web20:0.05 semanticweb:0.04 web:0.04 semantic:0.02 tools:0.01
wiki	EM	wiki:0.85 web20:0.01 semantic:0.01 wikipedia:0.01 collaboration:0.01
	CO	wiki:0.15 semantic:0.03 semanticweb:0.03 web20:0.02 software:0.02
dynamics	EM	dynamics:0.18 loreto:0.06 tagging:0.05 rmpcfl:0.04 analysis:0.04
	CO	tagging:0.07 dynamics:0.04 folksonomy:0.03 juergen:0.03 social:0.02
eclipse	EM	eclipse:0.55 java:0.23 development:0.05 ide:0.03 plugin:0.02
	CO	eclipse:0.18 java:0.13 plugin:0.06 develop:0.04 tools:0.04
yahoo	EM	yahoo:0.52 search:0.09 news:0.04 bookmarks:0.03 email:0.02
	CO	yahoo:0.09 search:0.04 web20:0.02 web:0.02 news:0.02

7 Conclusion and Future Work

In this paper we propose a probabilistic ranking approach for tag recommendation. The textual information from the resources and the parallel textual corpus from previously posts are used to learn the language and statistical translation model. Our hybrid probabilistic approach incorporates both the content based textural model and graph structure existing in posts for sharing the common tagging knowledge among users.

As our future work, we intent to study how to choose parameters via machine learning approaches to avoid heuristic setting. Further more, increasing the extra information of the resources, for example, using the citations(references) of a publication to augment the information of bookmark resource; using other tag expansion techniques; conducting the natural language understanding of the tag concept as well as studying the evaluation measures for tag recommendation are all possible future research work.

Acknowledgement

This paper is supported by the National Natural Science Foundation of China under the grant 60673009 and China National Hanban under the grant 2007-433. The authors thank Chin-Yew Lin at Microsoft Research Asia for his valuable comments to this paper. Thanks also to Jie Liu, Yang Wang and Min Lu for their helpful discussions and suggestions.

References

1. Heymann, P. and Ramage, D. and Garcia-Molina, H. Social Tag Prediction. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2008),* pages 531-538.
2. Tatu, M., Srikanth, M. and D'Silva, T. RSDC'08: Tag Recommendations using Bookmark Content. In *Proceedings of ECML PKDD Discovery Challenge 2008 (RSDC 2008),* pages 96-107.

3. Lipczak, M. Tag Recommendation for Folksonomies Oriented towards Individual Users. In *Proceedings of ECML PKDD Discovery Challenge (RSDC 2008),* pages 84-95.
4. Ponte, J. M. and Croft, W.-B. A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1998),* pages 275-281.
5. Zhai, C.-X. and Lafferty, J. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Transaction of Information System 2004,* pages 179-214.
6. Brown, P.-F., Pietra, V. J. D., Pietra, S. A. D. and Mercer, R.-L. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Journal of Computational Linguist 1993,* pages 263-311.
7. Berger, A. and Lafferty, J. Information Retrieval as Statistical Translation. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 1999),* pages 222-229.
8. Xue, X., Jeon, J. and Croft., W.-B. Retrieval Models for Question and Answer Archives. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2008),* pages 475-482.

Tag Sources for Recommendation in Collaborative Tagging Systems

Marek Lipczak, Yeming Hu, Yael Kollet, Evangelos Milios

Faculty of Computer Science, Dalhousie University, Halifax, Canada, B3H 1W5
lipczak@cs.dal.ca

Abstract. Collaborative tagging systems are social data repositories, in which users manage resources using descriptive keywords (tags). An important element of collaborative tagging systems is the tag recommender, which proposes a set of tags to each newly posted resource. In this paper we discuss the potential role of three tag sources: resource content as well as resource and user profiles in the tag recommendation system. Our system compiles a set of resource specific tags, which includes tags related to the title and tags previously used to describe the same resource (resource profile). These tags are checked against user profile tags – a rich, but imprecise source of information about user interests. The result is a set of tags related both to the resource and user. Depending on the character of processed posts this set can be an extension of the common tag recommendation sources, namely resource title and resource profile. The system was submitted to ECML PKDD Discovery Challenge 2009 for "content-based" and "graph-based" recommendation tasks, in which it took the first and third place respectively.

1 Introduction

The emergence of social data repositories made a fundamental change in the way information is created, stored and perceived. Instead of a rigid hierarchy of folders, collaborative tagging systems (e.g., BibSonomy[1], del.icio.us[2], Flickr[3], Technorati[4]) use a flexible folksonomy of tags. The folksonomy is created collaboratively by system users. While adding a resource to the system, users are asked to define a set of tags – keywords which describe it and relate it to other resources gathered in the system. To ease this process, some folksonomy services recommend a set of potentially appropriate tags. Proposing a tag recommendation system was a task of ECML PKDD Discovery Challenge 2009[5]. This paper presents a tag recommendation system submitted to the challenge.

[1] http://bibsonomy.org/help/about/
[2] http://del.icio.us/about/
[3] http://flickr.com/about/
[4] http://technorati.com/about/
[5] http://www.kde.cs.uni-kassel.de/ws/dc09/

1.1 Definitions

Collaborative tagging systems allow *users* ($u_i \in U$) to store *resources* ($r_j \in R$) in the form of *posts* ($p_{ij} \in P$). A post is a triple $p_{ij} = (u_i, r_j, T_{ij})$, where $T_{ij} = \{t_k\}$ is a *set of tags* assigned by the user to the resource. The data structure constructed by the collaborative tagging system (referred to as *folksonomy* [5]) is simply a set of posts. However, relations between three basic elements of the post allow us to represent the folksonomy as a tripartite graph of resources, users and tags. Each post can be then understood as a set of edges that form triangles connecting resource, user and tag. Projections of this tripartite graph can be used to examine the relations between folksonomy elements (e.g., two tags can be considered as similar when they are both linked to a large number of common resources, two users are similar when they are linked to the same tags).

Tag recommendation s is a pair (t, l), where t is a tag and l is a *recommendation score*, which is supposed to reflect the likelihood of the tag t being chosen by a user as a proper tag. A tag recommendation system returns a set of tag recommendations S. In this paper we use the term *tag recommendation set* (or simply *recommendation*) not only to refer to the final set of tags returned to the user, but also to denote the results of intermediate tag recommendation steps. In section 5 we define a set of operations on tag recommendation sets, which are used by our tag recommendation system.

User profile is a set of tags used by the user prior to the post that is being currently added to the system, $\mathbb{P}_u = \{t_k : u_i = u, r_j \in R, p_{ij} \in P, t_k \in T_{ij}\}$. The user profile is usually referred to as *personomy* [5]. We use a more general term, because it does not imply that the profile is personal. By analogy we can define a *resource profile*, which contains all tags that were attached to the resource (e.g., a scientific publication) by all users prior to the current post, $\mathbb{P}_r = \{t_k : u_i \in U, r_j = r, p_{ij} \in P, t_k \in T_{ij}\}$. Both user and resource profiles can serve as a simple tag recommendation set. For example, resource profile recommendation $S_{\mathbb{P}_r}$ is a set of tags from resource profile of r. Their score is the ratio of posts in which the tag was used to all posts of the resource (Eq. 1). The intuition behind this formula is that tags frequently used to describe a resource are likely to be used again, hence they are good recommendations.

$$l(t_k, r) = \frac{|\{p_{ij} : u_i \in U, r_j = r, t_k \in T_{ij}\}|}{|\{p_{ij} : u_i \in U, r_j = r\}|} \tag{1}$$

1.2 Tag recommendation tasks

The off-line evaluation of a tag recommendation system for challenge purposes is a complex task. Tags added to the resource are highly dependent on the state of the system and previous decisions of the user. It is not possible to create a large, realistic test dataset of posts, hiding at the same time the tags used in these posts. A test dataset which is large enough to objectively measure the quality of a recommendation system must cover a long period of time. If the tags in test data are hidden we lose access to the information about the state of the system, especially newly joined users, which make the dataset not representative. To ease this problem, the organizers of ECML PKDD Discovery Challenge 2009 divided the recommendation task into two subtasks which simulate two complementary recommendation approaches.

The first task "content-based recommendation" focuses on the content of a resource that is tagged. In this task we assume that information about the resource and user profile is in most cases not available in the folksonomy. A recommender based on resource content is especially important for new users, which are in the early stage of building their profile. Although, as shown in Section 3.1, the need of creating the recommendation based only on the content is rare, the content based recommender can be a valuable starting point for more complex recommenders that use information gathered in the folksonomy. Such more complex recommenders are evaluated in the second task – "graph-based recommendation". The test set in this task contains only users, resources and tags that were present at least twice in the training data. To obtain this set the organizers extracted k-core of order 2 [2] of tripartite graph of users, resources and tags created from training data. The test set contained only posts for which user, resource and all tags can be found in the k-core. It is important to notice that the second task neglects the disproportion between the number of unique resources and users. It also greatly simplifies the recommendation task by removing posts with unique tags which are hardest to recommend in real systems. To improve the results for this task the system must follow some unrealistic assumptions. Although this paper describes an entry to the challenge, we aimed to present a general system which can be applied to a real folksonomy based repository of bookmarks or scientific publications. Each modification that was made to match the specific constraints created by the dataset and the second task of the challenge is clearly stated.

2 Related work

Most of the tag recommendation systems presented in the literature are graph-based methods. It is a natural choice for folksonomies in which textual content is hard to access. For example, a system by Sigurbjörnsson and van Zwol [9] uses co-occurrence of tags to propose tags that complement user-defined tags of photographs in Flickr. Jäschke et al. [6] proposed a graph-based recommendation system for social bookmarking services. The method is based on FolkRank, a modification of PageRank, which is suited for folksonomies. The evaluation on a dense core of folksonomy showed that the FolkRank based recommender outperforms PageRank and collaborative filtering methods.

Even if a tag recommendation system extracts tags from the resource content, usually it also uses the graph information. An example of a content-based recommender is presented by Lee and Chun [7]. The system recommends tags retrieved from the content of a blog, using an artificial neural network. The network is trained based on statistical information about word frequencies and lexical information about word semantics extracted from WordNet. Another system designed to recommend tags for blog posts is TagAssist [10]. The recommendation is built on tags previously attached to similar resources. Meaning disambiguation is performed based on co-occurrence of tags in the complete repository.

Finally, we would like to mention two somewhat similar systems which took the first and second place in the ECML PKDD Discovery Challenge 2008. The winning system was proposed by Tatu et al. [11], while the second place was taken by our submission [8]. Both systems utilize information from resource content and the folksonomy graph. The graph is used to create a set of tags related to the resource and a set of tags related to the user who is adding the resource to the system. The winning system bases these sets on tags gathered in the profile of resource or user. Natural language processing techniques are later used to extend the set of tags related to resource or user (i.e., WordNet based search for words that represent the same concept). Our system bases the resource related tags on the resource title, the set is extended by finding tags that co-occur with the base tags in the system. The user related tags are simply the tags from the user profile. The intersection of both sets creates a set of tags that are related to both resource and user. Our system tries to extend this set by finding more related tags in user profile. Finally, both systems extract tags from resource content and join the content tags with the resource and user related tags to create the final recommendation.

3 BibSonomy dataset

All presented experiments and the evaluation of proposed tag recommendation system were performed on a snapshot of BibSonomy [4], a collaborative tagging system, which is a repository of website bookmarks and scientific publications (represented by BibTeX entries). The training dataset contained posts entered to the system before January 1, 2009. The test data contained posts entered between January 1, 2009 and June 30, 2009. The snapshot was provided by the organizers of the ECML PKDD Discovery Challenge 2009. The preprocessing steps, applied prior to the release of the dataset, included removing useless tags (e.g., *system:unfiled*), changing all letters to lower case and removing non-alphabetical and non-numerical characters from tags. We decided to clean the dataset further by removing sets of posts that were imported from an external source. This preprocessing step involved posts for which one set of tags, defined by user or system, was assigned to a large number of imported resources. An example of such a set consists of $9,183$ posts tagged with tag *indexforum* by one user. Leaving that tag in the system would result in a biased profile of its author. Unfortunately, this cleaning step could not detect another type of imported posts, for which the system automatically defines tags and timestamps based on the information from an external source. An example of such posts is a set of bookmarks imported from a web browser, for which the collaborative tagging system can use the names of bookmark folders to automatically define tags. The second preprocessing step applied to the released data was separation of bookmark and BibTeX posts. We observed that the vocabulary used for both types of resources is different, even for individual users. Some of the tags (e.g., *free*) have different meaning when tagging websites or scientific publications. Finally, content based recommendation can be based on different metadata fields in both resource types.

3.1 General characteristics

According to the statistical information about the dataset presented on the Discovery Challenge website[6] the BibSonomy snapshot matches the usual characteristics of folksonomies, including large disproportion between the number of unique resources and users (Table 1). Among the posts in the BibSonomy snapshot 90% contained unique resources. These resources cannot be found in any other post, hence it is not possible to deduce tag recommendation based on resource profile. At the same time 0.8% of the posts, corresponding to 3,167 posts, were entered by users with no previous posts in the system. Except those posts, every time a post is added, the system is able to use the user profile to recommend tags. Similar proportions can be observed for the CiteULike[7] dataset.

	BibSonomy		CiteULike	
number of tags	1,401,104		4,927,383	
number of unique tags	93,756	(7% of tags)	206,911	(4% of tags)
number of bookmark posts	263,004		N/A	
number of unique bookmarks	235,328	(89% of posts)	N/A	
number of BibTeX posts	158,924		1,610,011	
number of unique BibTeX entries	143,050	(90% of posts)	1,390,747	(86% of posts)
number of users	3,617	(1% of posts)	42,452	(3% of posts)

Table 1. Statistics of BibSonomy training data compared to CiteULike dataset (complete dump up to February 27, 2009). Both datasets have similar proportion of unique tags, posts and users.

The disproportion between unique resources and users is ignored in the test data of "graph-based recommendation" task. All users and resources present in the dataset can be found in the training data at least twice. Despite this fact the differences in statistical characteristics of resource and user profiles should be taken into consideration while proposing a recommendation system for this task. The cumulative frequency distribution of resources shows that both for bookmark and BibTeX entries, even if we remove elements that occurred twice or less, most of the remaining elements still have a very small profile (Fig. 1). Looking at the same statistic for users we see that a significant fraction of them have over 100 posts in their profiles. Hence user profiles are likely to contain more

[6] http://www.kde.cs.uni-kassel.de/ws/dc09/dataset

[7] http://www.citeulike.org/

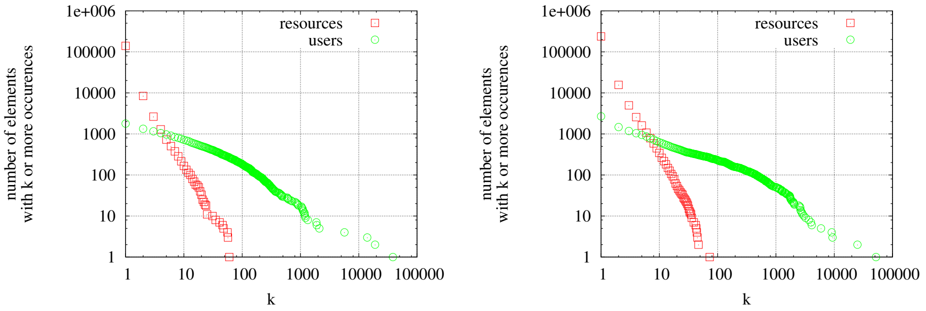


Fig. 1. Cummulative frequency distribution of resources and users for BibTeX (left) and bookmark (right) data. Much steeper curve for resources shows that we are much less likely to find a rich resource profile, comparing to the profiles of users.

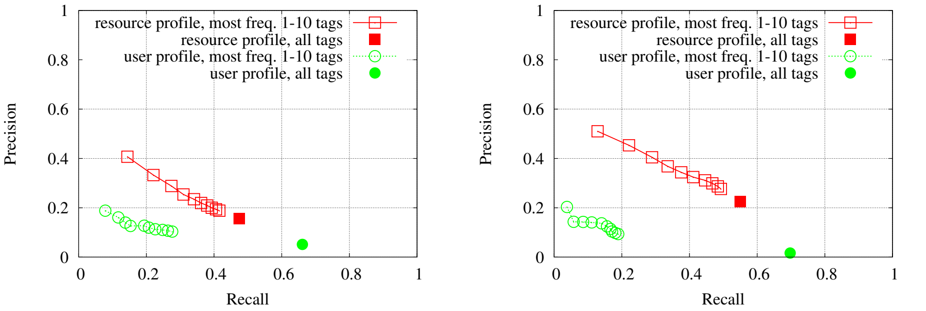


Fig. 2. Precision and recall of most frequent tags from resource and user profiles for BibTeX (left) and bookmark (right) “graph-based recommendation” task data.

potentially useful tags. To confirm this hypothesis we ran another experiment in which we simulated the test data of “graph-based recommendation” task and checked what is the precision and recall of basic recommenders that propose tags from resource/user profile sorted by frequency against real tags. To obtain a test set we divided the training data into training posts (entered before September 1, 2008) and test posts (entered later). We pruned them to be sure that all resources, users and tags occurred in the remaining part of the training set at least twice. Although this setting favours resource profiles, their overall recall is still lower than recall of the user profiles (Fig. 2). The fact that resource profiles are smaller makes them, however, a more precise source of tags. High recall of user profiles was observed by us repeatedly in many experiments. This is the reason why in our work we focused on user profiles, trying to increase the precision of this source of tags, while preserving reasonably high recall.

4 Tag recommendation sources

The presented recommendation system is the evolution of the work on the system [8] submitted to the ECML PKDD Discovery Challenge 2008[8]. In this section we summarize the results of experiments conducted during the work on the previous version of the system. Their main objective was to evaluate the quality of three basic sources of tags – words from resource title, tags assigned to the resource by other users (resource profile) and tags in user profile.

Resource title We tested most of the metadata fields looking for potential tags. Among them the resource title appears to be the most robust source of tag recommendations. The title is a natural summarization of web page or scientific publication, which means it plays a similar role as tags. In addition, the title is present on the resource posting page, which means it can possibly suggest tags to the user. It is easy to notice the evidence for this observation in the example posts of *User B* and *User C* shown in Table 2. Both of them used the tags *prediction* and *social* for "Social tag prediction" paper, which became the only occurrence of these tags in their profiles, unlike tag *recommender* which was used by them around fifty times, probably to describe the general area of interests. The number of words in the title is comparable to the number of tags, hence no additional cleaning steps are needed the achieve fairly high precision comparing to other examined tag sources (around 0.1). The drawback of this source is low recall (around 0.2), which makes the title inappropriate as a stand-alone tag recommender. For bookmark posts the web page URL appears to be another valuable source of tags. Although URL tags are less precise than title tags, their union can increase the recall of recommendation.

Posts:	*Social tag prediction*	*Towards the Semantic Web: Collaborative Tag Suggestions*
User A	Heymann 08 *tag recommendation*	Xu 06 *tag recommendation*
User B	**prediction** *tag recommender* **social** tagging	*tag recommender* tagging
User C	folksonomy **prediction** *recommender* **social** tag *tagging* toread	folksonomy *recommender* summerschool *tagging*

Table 2. Example posts of three users tagging two publications related to the tag recommendation problem (two tags were removed to increase anonymity of posts). **Bold** tags seem to be suggested by the title. Tags in *italics* likely represent the concept of tag recommendation problem in users' profiles.

Resource profile Tags assigned to the resource by other folksonomy users are not a good source of tag recommendations. One of the reasons is the sparsity of

[8] http://www.kde.cs.uni-kassel.de/ws/rsdc08/

data; 90% of resources were added to the system only once. This fact significantly limits the possible recall of this source of tags. The other issue is the personal character of posts and tags, which hurts the precision of retrieved tags. Given the example of two resources about the same concept, we see that users cannot agree on tags describing it: *tag recommendation*, *tag recommender*, *tagging recommender* (Table 2). The variety of tags attached by users creates, however, another application of resource tag sets. Mining relations between tags attached to the same resource can result in a graph of relations between tags. Using a relationship graph the system can identify tags which are also potential recomendations. The graph consists of general relations between tags and can be used independently of the resources, which reduces the negative impact of data sparsity. In our work we use two types of graphs. *TagToTag graph* is a directed graph which captures the co-occurence of tags. The weight of an edge is analogous to the confidence score (Eq. 2) in association rule mining [1], where $support(\{t_1 \cap t_2\})$ is the number of co-occurrences of tags t_1 and t_2 and $support(\{t_1\})$ is the number of occurrences of tag t_1. The second graph (*TitleToTag*) is created specifically for the resource title as the base of the recommendation. Using the same model it captures the relations between words from resource title and its tags.

$$confidence(t_1,\ t_2) = \frac{support(\{t_1 \cap t_2\})}{support(\{t_1\})} \tag{2}$$

User profile For cognitive simplicity and effcient retrieval, a typical user employs the same limited set of tags to describe resources of the same topic (Table 2). This pattern is the reason for high recall of user tags. On the other hand the user profile is a combination of tags related to many user interests and activities, which makes it a very imprecise source of tags. The most frequent tags from the user profile are likely to be related to the most central interests of the user. In our system we try to utilize the potential of user profile tags to extract user's tags that are related to the interests specific to the posted resource.

5 Tag recommendation system

Our tag recommendation system is a composition of six basic tag recommenders (Fig. 3). The result of each recommender is a tag recommendation set with scores in the range $[0, 1]$. The recommender makes a decision based on the resource content, resource related tags and user profile tags. However, its design makes it applicable to all posts even if the resource or user profile cannot be found in the system database. In such cases, the corresponding basic recommenders are not active. The following sections and Algorithm 1 give the detailed description of each basic recommender and the data flow in the system.

5.1 Recommendation based on resource content

The process starts with the extraction of potential tags from the content of resource. For BibTeX posts the **title** of publication is used, for bookmarks the

Algorithm 1: Tag recommendation system

```
Data: a resource $r$ and user $u$
Result: a tag recommendation set $S_{final}$
begin
    /*Step 1 – Extraction of content based tags*/
    $Words_{title} \longleftarrow extractTitleWords(r)$
    $S_{title} \longleftarrow \emptyset$
    foreach $w \in Words_{title}$ do
        $S_{title}$ add $makeTag(w, getPriorUsefullness(w))$
    $removeLowQualityTags(S_{title}, 0.05)$
    if $isBookmark(r)$ then
        $Words_{URL} \longleftarrow extractUrlWords(r)$
        $S_{URL} \longleftarrow \emptyset$
        foreach $w \in Words_{URL}$ do
            $S_{URL}$ add $makeTag(w, getPriorUsefullness(w))$
        $removeLowQualityTags(S_{URL}, 0.05)$
        $rescoreLeadingPrecision(S_{title}, 0.2)$
        $rescoreLeadingPrecision(S_{URL}, 0.1)$
    $S_{content} \longleftarrow mergeSumProb(S_{title}, S_{URL})$
    /*Step 2 – Retrieval of resource related tags*/
    $S_{TitleToTag} \longleftarrow \emptyset$// related tags from TitleToTag graph
    $S_{TagToTag} \longleftarrow \emptyset$// related tags from TagToTag graph
    $S_{\mathbb{P}_r} \longleftarrow getProfileRecommendationBasic(\mathbb{P}_r)$
    foreach $s_k \in S_{title}$ do
        $S_{s_k,TitleToTag} \longleftarrow \emptyset$
        foreach $t \in getRelated(g_{TitleToTag}, s_k)$ do
            $S_{s_k,TitleToTag}$ add $makeTag(t, s_k.l * confidenceTitleToTag(s_k.t, t))$
    foreach $s_k \in S_{content}$ do
        $S_{s_k,TagToTag} \longleftarrow \emptyset$
        foreach $t \in getRelated(g_{TagToTag}, s_k)$ do
            $S_{s_k,TagToTag}$ add $makeTag(t, s_k.l * confidenceTagToTag(s_k.t, t))$
    $S_{TitleToTag} \longleftarrow unionProb(T_{s_1,TitleToTag}, \ldots, T_{s_n,TitleToTag})$
    $S_{TagToTag} \longleftarrow unionProb(T_{s_1,TagToTag}, \ldots, T_{s_n,TagToTag})$
    $S_{r\ Related} \longleftarrow unionProb(S_{TitleToTag}, S_{TagToTag}, S_{\mathbb{P}_r})$
    /*Step 3 – Retrieval of resource and user related tags*/
    $S_{\mathbb{P}_u} \longleftarrow getProfileRecommendationByDay(\mathbb{P}_u)$
    $S_{r,u\ Related} \longleftarrow indersectionProb(S_{r\ Related}, S_{\mathbb{P}_u})$
    /*Final recommendation*/
    $rescoreLeadingPrecision(S_{title}, 0.3)$
    $rescoreLeadingPrecision(S_{\mathbb{P}_r}, 0.3)$
    $rescoreLeadingPrecision(S_{r,u\ Related}, 0.45)$
    $S_{final} \longleftarrow unionProb(S_{title}, S_{\mathbb{P}_r}, S_{r,u\ Related})$
end
```

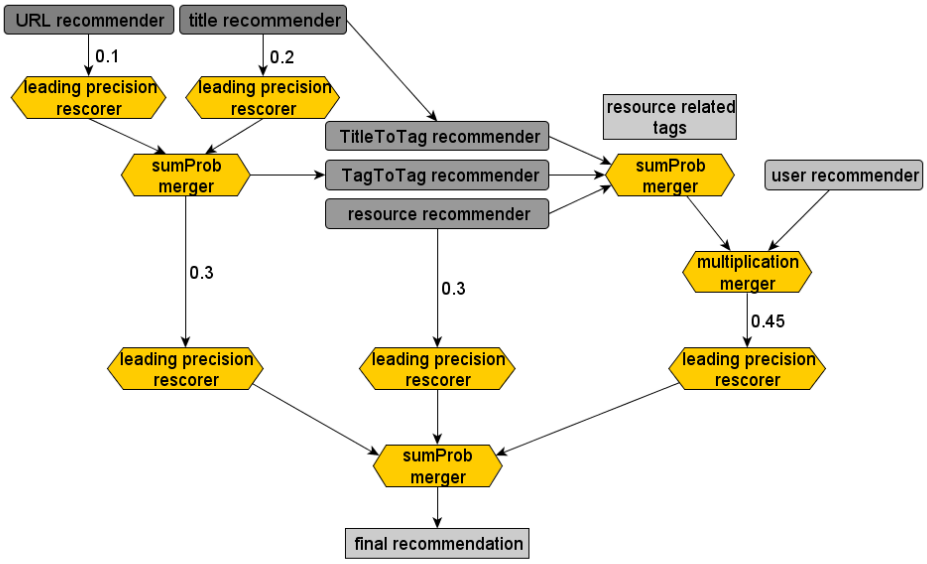


Fig. 3. Data flow in proposed tag recommendation system.

title recommendation is combined with tags extracted from the resource **URL**. Each word extracted from the title (or URL) is scored based on the usage of this word in previous posts. The score is the ratio of the number of times the word was used in the title (or URL) and as a tag to the total number of occurrences of the word in the title (or URL). Low-frequency words (i.e., words that were used in the title less than 50 times) are assigned an arbitrary score 0.1 which is the estimated probability of using a low-frequency word as a tag. To improve precision, content based recommender tags with score lower than 0.05 are removed from the recommendation set. This step serves also as a language independent stop-words remover. Preliminary experiments indicated that the bookmark title is more precise source of tag recommendation than its URL. This observation should be reflected in the way both tag recommendation sets are merged for bookmark posts. We tested a few rescoring functions, the best results were observed for the *leading precision* rescorer (Eq. 3), which sets the average precision (based on training data) as the score of first tag l_1 and modifies the scores of following tags l_i to preserve the proportion between all tag scores. Based on the tests on training data, the average precision of the title tag with the highest score is 0.2, while for URL it is 0.1.

$$l'_i = \frac{avgPrecisionAt1 * l_i}{l_1} \qquad (3)$$

5.2 Extraction of resource related tags

The result of title recommender is later used to propose title related tags in **TitleToTag recommender**. The related tags are extracted for each title word independently. The relation score, multiplied by the score of the word from the title recommender, becomes the score of the tag. This process produces a set of related tags for each title word. These sets are later merged, the scores of tags that can be found in more than one set are summed as they were probabilities of independent probabilistic events (Eq. 4). **TagToTag recommender** processes tags analogously, however, the input of this recommender is a complete content based tag recommendation set (title and URL for bookmarks). The aim of these recommenders is to produce a large, but likely not precise set of tags related to the resource. The third recommender that is able to produce a similar set is the **resource recommender**, which returns a set of tags from resource profile. The score of resource tag is the number of its occurrences divided by the number of occurrences of the resource. Although for most real posts this recommender would not return any tags, it plays a significant role in the "graph-based recommendation" task, where the resource of each tested post can be found in the system database at least twice. The scores of the results of three recommenders are summed in a probabilistic way (Eq. 4). This union of tags represents all the tags that are somehow related to the resource, and we refer to them as *resource related* tags.

$$l_{merged} = 1 - \prod_{i:t_i = t_{merged}} (1 - l_i) \tag{4}$$

5.3 Recommendation based on user profile

The **user recommender** produces a set of tags that were used by the user prior to the current post. Issues related to the construction of user profiles (i.e., import of posts, possible change of user interests) make a simple frequency value not a good score for user profile based recommendation. Tags most likely to be reused are the ones that were steadily assigned to posts while the user profile was built. To capture these tags we counted the number of separate days in which a tag was used by the user. To obtain the tag score we divided the number of days the tag was used by the total number of days in which the user was adding posts to the system. This approach allows a decrease in the importance of tags that were assigned by the user in a short period of time only; however, it only partially solves the problem of imported posts. For some of imported posts the system automatically produces low-quality tags and assigns time stamps copied from an external repository (e.g., importing web browser bookmarks, the system copies the time they were created). The combination of artificial tags and real time-stamps makes these posts very hard to detect. Removing such artificial posts is likely to improve the accuracy of the user profile recommender in a real recommendation system; however, it can have undesired consequences when applied to the challenge datasets. If the user imported posts before both training and test data were collected, it is possible that some of them can be found in both datasets. Hence we should train the system for tags from these posts, because it is possible that they can be found in test data as well. Even if we modify the frequency score the representation of user profile still contains tags related to various user interests. Checking the tags extracted from user profile against resource related tags allows us to extract tags that are particularly important for the processed posts. The intersection of both sets of tags produces tags related both to user as well as resource. The score of a tag is the product of scores from both source sets.

Finally the results of title recommender, resource recommender and the intersection of resource related tags and user profile are merged. As all three sets are results of independent recommenders, tags must be rescored to ensure that tags from more accurate recommenders will have higher score in the final tag recommendation set. Again the *leading precision* rescorer was used for the three input tag recommendation sets. The top ten tags of this set create the final recommendation set. The challenge organizers proposed to limit the recommendation set size to five tags, which seems to be a good number to be presented to a user, however, for evaluation purposes it is interesting to observe more tags.

6 Evaluation

This section presents the results of the off-line system evaluation based on the available BibSonomy snapshot. The evaluation approach assumed that all and only relevant tags were given by the user. Although this method simplifies the problem, it is robust and objective. The quality metrics were precision and recall, commonly used in recommender system evaluations [3].

6.1 Methodology

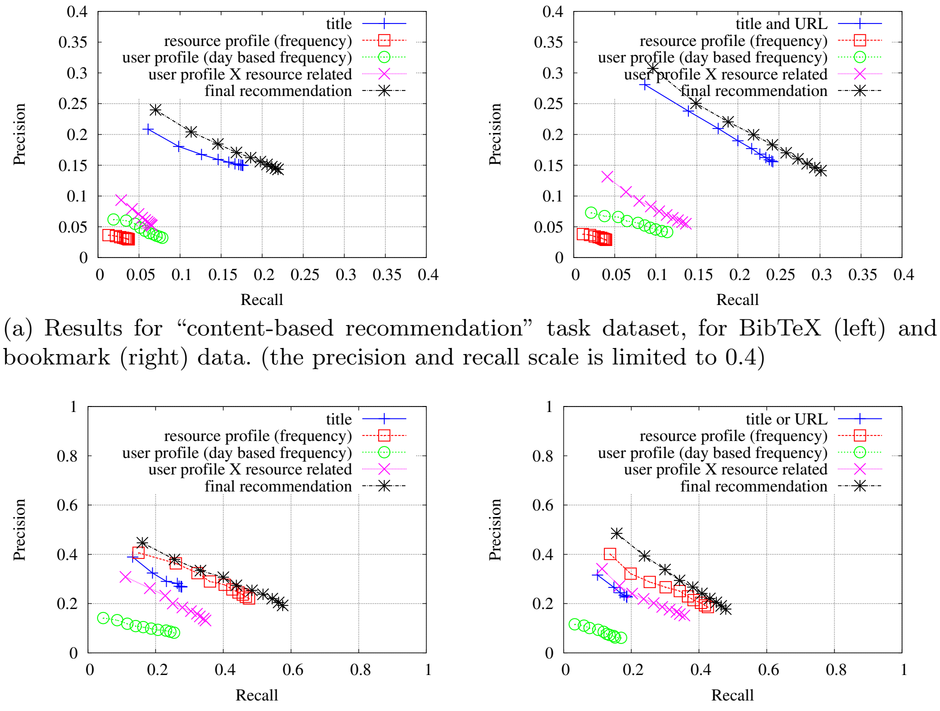
To keep the list of correct tags secret during the contest the organizers kept strict division between training and test set. The test data contained posts entered to to BibSonomy between January 1, 2009 and June 30, 2009. Each post of which user, resource and all tags could be found in k-core of order 2 of training data was used as test post for the "graph-based recommendation" task. The remaining posts were used for the "content-based recommendation" task. Comparison of training and test data for both tasks is presented in Table 3.

As we decided to separate the processing of BibTeX and bookmark posts we present the results for two post types separately. The final recommendation is presented together with the intermediate steps of the system: tags extracted from the resource title (and URL), the most frequent tags from resource profile and user profile and the combination of resource related tags and user profile tags. As each tag from the tag recommendation set can be ranked by its score it is straightforward to present any selected number of recommended tags. The plots (Fig. 6.1) present consecutive results for the top n tags, where $1 \leq n \leq 10$. For the "graph-based recommendation" task the tags that could not be found in

	training	test - Task 1	test - Task 2	test total
BibTex	158,924	26,104 (98.7% of test total)	347 (1.3% of test total)	26,451
bookmark	263,004	16,898 (97.5% of test total)	431 (2.5% of test total)	17,329
posts total	421,928	43,002 (98.2% of test total)	778 (1.8% of test total)	43,780

Table 3. Number of posts in training and test dataset. Sparsity of folksonomy graph causes large disproportion between test set for “content-based recommendation” task (Task 1) and “graph-based recommendation” task (Task 2). Another interesting fact is a different ratio of BibTeX and bookmark posts in training and test data.

the k-core of training data were removed from each recommendation set before calculating precision and recall.



(a) Results for “content-based recommendation” task dataset, for BibTeX (left) and bookmark (right) data. (the precision and recall scale is limited to 0.4)

(b) Results for “graph-based recommendation” task dataset, for BibTeX (left) and bookmark (right) data.

Fig. 4. Precision and recall of proposed tag recommendation system and intermediate steps. Test data was divided into BibTeX and bookmark part.

6.2 Results

As expected, precision and recall of the recommendation results in the "content-based recommendation" task are mostly driven by the content tags. Low score of user profile recommenders for BibTeX data is likely caused by a large number of posts by users who started to use the system after the training set was built. According to the rules set by the organizers the precision score was averaged over all posts in the test set, even if a recommender returned no tags for some of them. Whenever a user profile was available the user based recommender obtained significantly better results than content based recommender only.

The results for the "graph-based recommendation" task show surprisingly high accuracy of resource profile tags (which was not observed to such a degree on training data). For the test dataset in this task the intersection of resource related tags and user profile has lower precision than resource profile tags. This is an unexpected result, comparing to the previous results on training dataset, where the intersection of resource related tags and user profile had comparable or higher precision and recall to resource profile. Despite this unexpected behaviour the tags from the user profile are able to increase the f1 score by 0.02 for tag recommendation set of size 5. The open question is how representative the results of this dataset are, considering the fact that less than 2% of test posts matched the conditions of this task.

For both tasks there is a noticeable difference between the results for both types of data. However, it is not clear if it is caused by some fundamental differences between BibTeX and bookmark posts, or the differences between the two particular test datasets used. It is important to notice that the high number of tested posts has no impact on the statistical validity of results. The way the test data was prepared makes it very dependent on the behavior of users in the period of time the data was collected.

Finally we present the results of the final recommendation for combined BibTeX and bookmark posts, which were submitted to the challenge (Table 4). The systems were ranked based on the f1 score (Eq. 5) for the tag recommendation set of size 5. Based on that criterion the presented tag recommendation system took the first place in the "content-based recommendation" task (out of 21 participants) and the third place in the "graph-based recommendation" task (again, out of 21 participants).

$$f1 = \frac{2 * precision * recall}{precision + recall} \tag{5}$$

7 Conclusions and future work

In creating the presented tag recommendation system we considered the title of a resource as a natural starting point of the recommendation process. We tried to extend the set by tags related to the title as well as tags present in the profiles of resource and user. Our main aim was to extract valuable tags

Table 4. The results of the presented tag recommendation system. In the challenge the systems were ranked based on the f1 score for the tag recommendation set of size 5.

	content-based recommendation			graph-based recommendation		
#result tags	recall	precision	f1	recall	precision	f1
1	0.0805	0.2664	0.1236	0.1587	0.4679	0.2370
2	0.1275	0.2224	0.1621	0.2465	0.3869	0.3012
3	0.1626	0.1987	0.1788	0.3143	0.3361	0.3248
4	0.1885	0.1821	0.1852	0.3682	0.2998	0.3305
5	0.2080	0.1705	**0.1874**	0.4070	0.2700	**0.3246**
6	0.2218	0.1616	0.1869	0.4425	0.2468	0.3169
7	0.2323	0.1549	0.1858	0.4701	0.2282	0.3072
8	0.2403	0.1495	0.1843	0.4929	0.2116	0.2961
9	0.2467	0.1457	0.1832	0.5092	0.1967	0.2838
10	0.2515	0.1424	0.1819	0.5220	0.1842	0.2723

from user profile which is a very rich but imprecise source of tags. Designing the system we mostly focused on the precision of the recommended tags. To avoid the risk of recommending tags less precise than tags extracted from the title we decided to leave it as the only recommendation whenever the user profile was unavailable. This was a frequent case in "content-based recommendation" task, which gives us hope that the system will be able to achieve even better results for the final "on-line recommendation" task. The system is now connected to BibSonomy and recommends tags to each newly added post in real time. This evaluation setting will give a realistic assessment of system quality.

In our future work on this project we plan to focus on tagging patterns of individual users which would allow us to tune the recommendation for each specific user. Discovering strong patterns, like user who uses author name and year of publication for each BibTeX post, can greatly increase the accuracy of recommender for this specific user. Another interesting issue is handling of multiword concepts (e.g., is a user going to use two tags "information" "retrieval" or one "information.retrieval"?). Finally, we hope that evaluation settings like "on-line recommendation" task would allow us to investigate short temporal patterns when a user adds a sequence of posts related to the same problem.

References

1. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.
2. V. Batagelj and M. Zaveršnik. Generalized cores, 2002. cite arxiv:cs.DS/0202039.
3. Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
4. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. BibSonomy: A social bookmark and publication sharing system. In *Proc. the First*

Conceptual Structures Tool Interoperability Workshop at the 14th Int. Conf. on Conceptual Structures, pages 87–102, Aalborg, 2006. Aalborg Universitetsforlag.

5. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Trend detection in folksonomies. In *Proc. First International Conference on Semantics And Digital Media Technology (SAMT)*, volume 4306 of *LNCS*, pages 56–70, Heidelberg, dec 2006. Springer.
6. Robert Jäschke, Leandro Balby Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In *Knowledge Discovery in Databases: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007, Proceedings*, volume 4702 of *LNCS*, pages 506–514. Springer, 2007.
7. Sigma On Kee Lee and Andy Hon Wai Chun. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ANN semantic structures. In *ACOS'07: Proc. the 6th Conf. on WSEAS Int. Conf. on Applied Computer Science*, pages 88–93, Stevens Point, Wisconsin, USA, 2007. WSEAS.
8. Marek Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.
9. Börkur Sigurbjörnsson and Roelof van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proc. the 17th international conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.
10. S.C. Sood, K.J. Hammond, S.H. Owsley, and L. Birnbaum. TagAssist: Automatic tag suggestion for blog posts. In *Proc. the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.
11. Marta Tatu, Munirathnam Srikanth, and Thomas DSilva. RSDC08: Tag recommendations using bookmark content. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.

Collaborative Tag Recommendation System based on Logistic Regression ⋆

E. Montañés[1], J. R. Quevedo[2], I. Díaz[1], and J. Ranilla[1]
{montaneselena,quevedo,sirene,ranilla}@uniovi.es

[1] Computer Science Department, University of Oviedo (Spain)
[2] Artificial Intelligence Center, University of Oviedo (Spain)

Abstract. This work proposes an approach to collaborative tag recommendation based on a machine learning system for probabilistic regression. The goal of the method is to support users of current social network systems by providing a rank of new meaningful tags for a resource. This system provides a ranked tag set and it feeds on different posts depending on the resource for which the recommendation is requested and on the user who requests the recommendation. Different kinds of collaboration among users and resources are introduced. That collaboration adds to the training set additional posts carefully selected according to the interaction among users and/or resources. Furthermore, a selection of post using scoring measures is also proposed including a penalization of oldest post. The performance of these approaches is tested according to F_1 but just considering at most the first five tags of the ranking, which is the evaluation measure proposed in ECML PKDD Discovery Challenge 2009. The experiments were carried out over two different kind of data sets of Bibsonomy folksonomy, core and no core, reaching a performance of 26.25% for the former and 6.98% for the latter.

1 Introduction

Recently, tag recommendation has been gained popularity as a result of the interest of social networks. This task can be defined as the process of providing promising keywords to the users of a social network in the presence of resources of the network itself. These keywords are called tags and the users can assign them to the resources [12]. Tagging resources present several advantages: they facilitate other users a later search and browsing, they consolidate the vocabulary of the users, they provide annotated resources and they build user profiles. An option to perform such task could be to provide tags manually to each user, but this time-consuming and tedious task could be avoided using a Tag Recommender System (TRS).

Folksonomies are examples of large-scale systems that take advantage of a TRS. A Folksonomy [9] is a set of posts included by a user who has attached

⋆ This research has been partially supported by the MICINN grants TIN2007-61273 and TIN2008-06247.

a resource through a tag. Generally, each resource is specific to the user who added it to the system, as `Flickr`, which shares photos, or `BibSonomy`, which shares bookmarks and bibtex entries. However, for some types of networks identical resources can be added to the system by different users, as is the case of `Del.icio.us` which shares bookmarks.

This paper proposes an approach to collaborative tag recommendation based on a logistic regression learning process. The work starts from the hypothesis that a learning process improves the performance of the recommendation task. It explores several information the learner feeds on. In this sense, the training set depends on each test post and it is specifically built for each of them. In addition, a set of additional posts carefully selected are added to the training set according to the collaboration among users and/or resources.

The remainder of the paper is structured as follows. Section 2 presents background information about tag recommendation in social networks. Our approach is put in context in Section 3 while the proposed method is provided in Sections 4, 5 and 6. Section 7 describes the performance evaluation metric. The results conducted on public data sets are presented and analyzed in Section 8. Finally, Section 9 draws conclusions and points out some possible challenges to address in the near future.

2 Related Work

Different approaches have been proposed to support the users during the tagging process depending on the purpose they were built for. Some of them makes recommendations by analyzing content [1], analyzing tag co-occurrences [23] or studying graph-based approaches [10].

Brooks et al. [4] analyze the effectiveness of tags for classifying blog entries by measuring the similarity of all articles that share a tag. Jäschke et al. [10] adapt a user-based collaborative filtering as well as a graph-based recommender built on top of FolkRank. TagAssist [24] recommends tags of blog posts relying upon tags previously attached to similar resources.

Lee and Chun [14] propose an approach based on a hybrid artificial neural network. ConTag [1] is an approach based on Semantic Web ontologies and Web 2.0 services. CoolRank [2] utilizes the quantitative value of the tags that users provide for ranking bookmarked web resources. Vojnovic et al. [27] keep in view collaborative tagging systems where users can attach tags to information objects.

Basile et al.[3] propose a smart TRS able to learn from past user interaction as well as from the content of the resources to annotate. Krestel and Chen [13] raise TRP-Rank (Tag-Resource Pair Rank), an algorithm to measure the quality of tags by manually assessing a seed set and propagating the quality through a graph. Zhao et al. [29] propose a collaborative filtering approach based on the semantic distance among tags assigned by different users to improve the effectiveness of neighbor selection.

Katakis et al. [12] model the automated tag suggestion problem as a multilabel text classification task. If the item to tag exists in the training set, then it suggests the most popular tags for the item. Tatu et al. [25] use textual content associated with bookmarks to model users and documents.

Sigurbjornsson et al. [23] present the results by means of a tag characterization focusing on how users tags photos of `Flickr` and what information is contained in the tagging.

Most of these systems require information associated with the content of the resource itself [3]. Others simply suggest a set of tags as a consequence of a classification rather than providing a ranking of them [12]. Some of them require a large quantity of supporting data [23]. The purpose of this work is to avoid these drawbacks using a novel approach which establishes a tag ranking through a machine learning approach based on logistic regression.

3 Tag Recommender Systems (TRS)

A folksonomy is a tuple $F := (\mathcal{U}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$ where $\mathcal{U}$, $\mathcal{T}$ and $\mathcal{R}$ are finite sets, whose elements are respectively called users, tags and resources, and $\mathcal{Y}$ is a ternary relation between them, i. e., $\mathcal{Y} \subseteq \mathcal{U} \times \mathcal{T} \times \mathcal{R}$, whose elements are tag assignments (posts). When a user adds a new or existing resource to a folksonomy, it could be helpful to recommend him/her some relevant tags.

TRS usually take the users, resources and the ratings of tags into account to suggest a list of tags to the user. According to [15], a TRS can briefly be formulated as a system that takes a given user $u \in \mathcal{U}$ and a resource $r \in \mathcal{R}$ as input and produces a set $\mathcal{T}(u, r) \subset \mathcal{T}$ of tags as output.

Jäschke et al. in [10] define a post of a folksonomy as a user, a resource and all tags that this user has assigned to that resource. This work slightly modifies this definition in the sense that it restricts the set of tags to the tags used simultaneously by a user in order to tag a resource.

There are some simple but frequently used TRS [10] based on providing a list of ranked tags extracted from the set of posts connected with the current annotation.

- MPT (Most Popular Tags): For each tag t_i, the posts with t_i are counted and the top tags (ranked by occurrence count) are utilized as recommendations.
- MPTR (Most Popular Tags by Resource): The number of posts in which a tag occurs together with r_i is counted for each tag. The tags occurring most often together with r_i are then proposed as recommendations.
- MPTU (Most Popular Tags by User): The number of posts in which a tag occurs together with u_i is counted for each tag. The tags occurring most often together with u_i are then proposed as recommendations.
- MPTRU (Most Popular Tags by Resource or User): The number of posts in which a tag occurs together either with r_i or u_i is counted for each tag. The tags occurring most often together with either r_i or u_i are taken as recommendations.

Our hypothesis is that the introduction of a learning system is expected to improve their performance of these systems. These are the key points of the system:

- The training set depends on each test post and it is specifically built for each of them. Section 4 explains the way of building the initial training set and the example representation.
- Several training sets are built according to different kinds of collaboration among users and resources, performing post selection adapting several scoring measures and penalizing oldest posts. Afterwards all of them are compared and evaluated. This approaches are detailed in Section 5.
- The learning system adopted was LIBLINEAR [5], which provides a probabilistic distribution before the classification. This probability distribution is exerted to rank the tags taking as the most suitable tag the one with highest probability value. The tags of the ranking will be all that appear in the categories of each training set. This entails that some positive tags of a test post might not be ranked. This issue is exposed in depth in Section 6.

4 Test and Training Data Representation

This section depicts the whole procedure followed in order to provide a user and a resource with a set of ranked tags. These recommendations are based on a learning process that learns how the users have previously tagged the resources. The core of the method is a supervised learning algorithm based on logistic regression [5].

The traditional approach splits the data into training and test sets at the beginning. Afterwards, a model is inferred using the training set and it is validated thanks to the test set [12]. In this paper, the methodology used is quite different in the sense that the training and test sets are not fixed. The test set is randomly selected and afterwards an *ad hoc* training set is provided for each test post. This paper studies different training sets built according to the resource and the user for whom the recommendations are provided.

4.1 Definition of the Test Set

According to the definition of a folksonomy in Section 3, it is composed by a set of posts. Each post is formed by a user, a resource and a set of tags, i.e.,

$$p_i = (u_i, r_i, \{t_{i_1}, \ldots, t_{i_k}\})$$

Each post of a folksonomy is candidate to become a test post. Each test post is then turned into as many examples as tags used to label the resource of this post. Therefore, post p_i is split into k test examples

$$\begin{aligned} e_1 &= (u_i, r_i, t_{i_1}) \\ &\vdots \\ e_k &= (u_i, r_i, t_{i_k}) \end{aligned} \tag{1}$$

Example 1 Let the following folksonomy be

post	*date*	*User*	*Resource*	*Tags*
p_1	d_1	u_1	r_1	t_1
p_2	d_2	u_1	r_2	t_2
p_3	d_3	u_2	r_1	t_1
p_4	d_4	u_3	r_1	t_3
p_5	d_5	u_2	r_2	t_4
p_6	d_6	u_2	r_1	t_2, t_3
p_7	d_7	u_2	r_2	t_2, t_5
p_8	d_8	u_3	r_2	t_1

(2)

Let $p_7 = (u_2, r_2, \{t_2, t_5\})$ be a randomly selected test post at instant d_7. Therefore the test set is formed by

example	*date*	*User*	*Resource*	*Tags*
e_1	d_7	u_2	r_2	t_2
e_2	d_7	u_2	r_2	t_5

(3)

4.2 Definition of the Initial Training Set

Whichever learning system strongly depends on the training set used to learn. In fact, in order to guarantee a better learning, it would be ideal for the distribution of the categories in both training and test sets to be as similar as possible. Therefore, the selection of an adequate training set is not a trivial task that must be carefully carried out.

Once the test set is randomly selected, an *ad hoc* training set is dynamically chosen from the posts posted *before* the test post.

The point of departure for building the training set is the set of posts concerning with the resource or the user for which the recommendations are demanded. Once the posts are converted into examples, those examples whose tags have been previously assigned to the resource by the user to whom the recommendations are provided are removed because it has no sense to recommend a user the tags he/she had previously used to label the resource. This section deals with the way of building the initial training set. Next section will explain in depth the way of selecting promising posts through a collaborative approach, using relevance measures for post selection and penalizing oldest posts.

Let $p_i = (u_i, r_i, \{t_{i_1}, \ldots, t_{i_k}\})$ be a test post.

Let $\mathcal{R}_{r_i}$ be the subset of posts associated to a resource r_i and

$$\mathcal{R}^t_{r_i} = \{p_i/p_i \in \mathcal{R}_{r_i} \; and \; it \; was \; posted \; before \; t\}$$

Let $\mathcal{P}_{u_i}$ be the personomy (the subset of posts posted by a user constitutes the so-called personomy) associated to a user u_i and

$$\mathcal{P}^t_{u_i} = \{p_i/p_i \in \mathcal{P}_{u_i} \; and \; it \; was \; posted \; before \; t\}$$

Therefore, the training set associated to p_i is formed by

$$UR^{d_i}_{u_i,r_i} = \{\mathcal{P}^{d}_{u_i} \cup \mathcal{R}^{d}_{r_i}\}\backslash\{p_j/p_j = (u_i, r_i, \{t_{i_1}, ..., t_{i_n}\})\}$$

Example 2 Let us show an example of each training set for the test set of Example 1.

In this case the training set is computed as follows.

$$UR^{d_7}_{u_2,r_2} =$$

$$\{\mathcal{P}^{d_7}_{u_2} \cup \mathcal{R}^{d_7}_{r_2}\}\backslash\{p_j/p_j = (u_i, r_i, \{t_{i_1}, ..., t_{i_n}\})\} =$$

$$\{\{p_3, p_5, p_6\} \cup \{p_2, p_5\}\}\backslash\{p_5\} =$$

$$\{p_2, p_3, p_6\}$$

Therefore the training set is defined as follows.

$example$	$date$	$User$	$Resource$	$Tags$
e_2	d_2	u_1	r_2	t_2
e_3	d_3	u_2	r_1	t_1
e_{6_1}	d_6	u_2	r_1	t_2
e_{6_2}	d_6	u_2	r_1	t_3

(4)

4.3 Example Representation

Now we will explain the way of transforming the post into a computable form understandable for a machine learning system. Therefore, we have to define the features which characterize the examples as well as the class of each example.

The features which characterize the examples are the tags previously used to tag the resource in the folksonomy. Hence, each example will be represented by a vector V of size M (the number of tags of the folksonomy) where $v_j \geq 1$ if and only if t_j was used to tag the resource before and 0 otherwise, where $j \in 1, \ldots, M$. The class of an example will be the tag the user has tagged the resource with at this moment.

Let us represent the training set of Example 2.

Example 3 As an illustration of how to represent a example, let us represent example e_{6_1} of Example 2. The class of e_{6_1} is t_2, which is its corresponding tag. The features are t_1 and t_3, since the resource r_1 of e_{6_1} was also tagged before by t_1 in p_1 and p_3 and by t_3 in p_4. The representation of example e_{6_1} is then $\{1, 0, 1, 0, 0\}$.

4.4 Simple Feature Selection

An additional proposal to improve the representation is also adopted, since removing redundant or non-useful features which add noise to the system is usually helpful to increase both the effectiveness and efficiency of the classifiers. The example representation based on tags as features makes possible a simple feature selection in the training set. This selection consists of keeping just those tags which represent the test set.

Obviously, this is possible just in case the information about the resource of the test post is considered for building the training set, which is the case here. This approach is based on the fact that in a linear system, as the one adopted here, the weights of the features that neither represent the test post nor contribute to obtain the ranking for this post. Therefore, they could be considered as irrelevant features beforehand. This fact can be assumed only for a particular test post. Thus, this is another advantage of building a training set particularly for each test post.

Let us consider the test post of Example 1 and the training set of Example 2.

Example 4 The features for the test post are t_2 and t_4, hence, the training set of Approach 3 in Example 2 will be reduced to be represented at most with these two tags. Originally, that training set has the following representation:

example	*date*	*resource*	*features*	*category*
e_2	d_2	r_2	$\emptyset$	t_2
e_3	d_3	r_1	t_1	t_1
e_{6_1}	d_6	r_1	t_1, t_3	t_2
e_{6_2}	d_6	r_1	t_1, t_2, t_3	t_3

(5)

In the folksonomy represented in Example 1, resource r_2 does not have any tag assigned before instant d_2, then its representation is an empty set of features. Analogously, resource r_1 has only been tagged before instant d_3 with t_1, particularly in instant d_1 by user u_1, then it is represented only by feature t_1. The instant d_6 in which the resource r_1 was tagged deserves special attention. Since this resource has been tagged before d_6 with t_1 and t_3, then both tags are included in its representation. Besides, in example e_{6_1} when the category is t_2, the tag t_3 is also added because it is a tag assigned in the same instant. In the same way, in example e_{6_2} when the category is t_3, the tag t_2 is included, since it is a tag assigned in the same instant.

Reducing such representation to the tags of the test post, the results of this new approach is

example	*date*	*resource*	*features*	*category*
e_2	d_2	r_2	$\emptyset$	t_2
e_3	d_3	r_1	$\emptyset$	t_1
e_{6_1}	d_6	r_1	$\emptyset$	t_2
e_{6_2}	d_6	r_1	t_2	t_3

(6)

5 Post Selection

This section copes with the way of selecting promising posts to be included as examples for the machine learning system. The selection of such posts is carefully carried out taking into account several issues. Let us expose the outline of the process now that will be discussed in depth later. Firstly, only posts that satisfy certain collaborative conditions will be the candidates to add to the initial training set. Secondly, every candidate is scored according to certain measure of relevance. Thirdly, such relevance is penalized depending on the time that the post was posted with regard to the test post. Finally, once a ranking of the candidates is established according to such scoring measure with the correspondent penalization, the most relevance ones will be the posts that will form the final training set. Therefore, the choice of the training set for a given test post is reduced to define the criteria the posts must satisfy to be included in the training set.

5.1 Collaborative conditions

Several approaches are proposed to introduce collaboration among users and resources. The effect over the training set is the presence of additional posts carefully selected according to the collaboration among users and/or resources. The collaborative conditions can be the following:

- Collaboration using resources
  - Take the tags in the posts (contained in the training set described in Section 4.2) that were assigned to the **resource** r_i of the test post p_i. Let be this set T_{r_i}.
  - Take the posts (contained in the training set described in Section 4.2) that contain the tags of T_{r_i}.
  - Add such posts to the training set described in Section 4.2 ($UR^{d_i}_{u_i,r_i}$). Hence, the training set is formed by the posts of $UR^{d_i}_{u_i,r_i} \cup T_{r_i}$.
- Collaboration using users
  - Take the tags in the posts (contained in the training set described in Section 4.2) that were assigned by the **user** u_i of the test post p_i. Let be this set T_{u_i}.
  - Take the posts (contained in the training set described in Section 4.2) that contain the tags T_{u_i}.
  - Add such posts to the training set described in Section 4.2 ($UR^{d_i}_{u_i,r_i}$). Hence, the training set is formed by the posts of $UR^{d_i}_{u_i,r_i} \cup T_{u_i}$.
- Collaboration using both resources and users by union
  - Take the tags in the posts (contained in the training set described in Section 4.2) that were assigned to the **resource** r_i of the test post p_i, that is the set T_{r_i}, and that were assigned by the **user** u_i of the test post p_i, that is the set T_{u_i}.
  - Take the posts (contained in the training set described in Section 4.2) that contain the tags of $T_{r_i} \cup T_{u_i}$

    - Add such posts to the training set described in Section 4.2 ($UR^{d_i}_{u_i,r_i}$). Hence, the training set is formed by the posts of $UR^{d_i}_{u_i,r_i} \cup T_{r_i} \cup T_{u_i}$.
- Collaboration using both resources and users by intersection
    - Take the tags in the posts (contained in the training set described in Section 4.2) that were assigned to the **resource** r_i of the test post p_i, that is the set T_{r_i}, and that were assigned by the **user** u_i of the test post p_i, that is the set T_{u_i}.
    - Take the posts (contained in the training set described in Section 4.2) that contain the tags of $T_{r_i} \cap T_{u_i}$
    - Add such posts to the training set described in Section 4.2 ($UR^{d_i}_{u_i,r_i}$). Hence, the training set is formed by the posts of $UR^{d_i}_{u_i,r_i} \cup (T_{r_i} \cap T_{u_i})$.

5.2 Relevance measures

Once the set of candidates is obtained, they will be scored according to several measures in order to select the most relevant ones. The following scoring measures have been applied before to perform feature selection. Here, they will be adapted to select posts, that, in fact, they are examples instead of features. All of them depend on two parameters, which will be defined before presenting them. The parameter a will be the number of tags that certain post p_j shares with the post of test p_i (in its representation through the resource of the post as described in Section 4.3) and the parameter b will be the number of tags that certain post p_j has (again in its representation through the resource of the post as described in Section 4.3), but the post of test p_i does not.

- From Information Retrieval (IR), *document frequency df* [21] and F_1 [22].
- A family of measures coming from Information Theory (IT). These measures consider the distribution of the words over the categories. One of the most widely adopted [18] is the *information gain* (IG), which takes into account either the presence of the word in a category or its absence, whereas others are the *expected cross entropy for text* (CET) or χ^2 [17]. They are all defined in terms or probabilities which, in turn, are defined from the parameters mentioned above.
- Those which quantify the importance of a feature f in a category c by means of evaluating the quality of the rule $f \rightarrow c$, assuming that it has been induced by a Machine Learning (ML) algorithm [18] (in this paper changing feature by example/post). Some of these measures are based on the percentage of successes and failures of the applications of the rules as, for instance, the *Laplace* measure (L) which slightly modifies the percentage of success and the *difference* (D). Other measures that deal with the number of examples of the category in which the feature occurs and the distribution of the examples over the categories are, for example, the *impurity level* (IL) [20]. Some other variants of the foresaid measures studying the absence of the feature in the rest of the categories have also been adopted [18], leading respectively to the L_{ir}, D_{ir} and IL_{ir} measures.

5.3 Recent posts and most relevant posts

A TRS that provides the most on-fashion folksonomy tags would be desirable. This suggest emphasizing the most recent posts more than the oldest ones. For this purpose, a penalizing function is applied to the score granted by a measure. However, some measures reaches negative values, then an increasing function that guaranties a positive value should be applied before the penalizing function in order to keep the ranking the measure gives. The option adopted will be to use the arc tangent and to apply a translation of $\frac{\pi}{2}$. Then, the penalizing functions will be of the form $\frac{1}{(1+\frac{t}{d})^e}$, where t is the time the post was posted, d is the time unit and e is a parameter that controls the penalizing degree. Therefore, if m is the score granted by a measure, the final score granted to each post will be

$$\left(\arctan(m) + \frac{\pi}{2}\right) \cdot \frac{1}{(1+\frac{t}{d})^e}$$

Once the ranking of the posts is established, it is necessary to define a cutoff for selecting the most relevant ones. Some statistics in a folksonomy show that for a given test post its training set might contain either too few posts or too many ones. Both extreme situations are detrimental for the machine learning systems. Applying a percentage of posts to select the most relevant ones avoids neither having too few posts nor to many ones. The alternative used in this paper consists of applying an heuristic able to considerably reduce the posts selected when initially there are too many of them and also able to slightly reduce the posts selected when initially there are too few of them. If n is the original number of posts, such heuristic is defined as follows

$$floor(2 \cdot n)^{0.75}$$

6 Learning to Recommend

The key point of this paper is to provide a ranked set of tags adapted to a user and a resource. Therefore, it could be beneficial to have a learning system able to rank the tags and to indicate the user which tag is the best and which one is the worst for the resource. Taking into account this fact, a preference learning system can not be applied since that kind of methods yield a ranking of the examples (posts) rather than a ranking of categories (tags)[11].

As the input data are multi-category, a system of this kind is expected to be used. However, these systems do not provide a ranking. They can be adapted to produce a partial ranking in the following way: It is possible to take the labels they return and to place them first as a whole and to place the rest of the labels also as a whole afterwards. Obviously, this approach does not establish an order among the labels they recommend but it orders all those labels it returns as a whole with regard to the labels it does not provide.

The system we need must provide a global ranking of labels. Therefore a multi-label system could be used, but again they need an adaptation to deal with ranking problems. In fact, some multi-label classification systems perform a ranking and then they obtain the multi-label classification [26]. Hence, it is possible to obtain a ranking directly from them.

Elisseeff and Weston [6] propose a multi-label system based on Support Vector Machines (SVM), which generates a ranking of categories. The drawback is that the complexity is cubic and although they perform an optimization to reduce the order to be quadratic, they admit that such complexity is too high to apply to real data sets.

Platt [19] uses SVM to obtain a probabilistic output, but just for a binary classification and not for multi-category. A priori one might think about performing as many binary classification problems as the number of tags (categories) that appear in the training set. The problem would turn them into decide if a post is tagged with certain tag or not. But this becomes unfeasible since we are talking of about hundreds of thousands of tags.

With regard to the problem of tag recommendation, Godbole and Sarawagi in [8] present an evolution of SVM based on extending the original data set with extra features containing the predictions of each binary classifier and on modifying the margin of SVMs in multi-label classification problems. The main drawback is that they perform a classification rather than a ranking.

In this framework, LIBLINEAR ([5] and [7]) is an open source library [3] which is a recent alternative able to accomplish multi-category classification through logistic regression, providing a probabilistic distribution before the classification.

This paper proposes to use this probability distribution to rank the tags, taking as most suitable tag the one with the highest probability value. In the same sense the most discordant tag will be the one with the lowest probability.

This work uses the default LIBLINEAR configuration after a slight modification of the output. The evaluation in this case takes place when a resource is presented to the user. Then, a ranking of tags (the tags of the ranking will be all which appear in the categories of the training set) is provided by the learning model.

If such resource has not been previously tagged, the ranking is generated according to a priori probability distribution. It consists of ranking the tags of the user according to the frequency this user has used them before. Therefore, no learning process is performed in this particular case.

7 Performance Evaluation

So far, no consensus about an adequate metric to evaluate a recommender has been reached [10]. Some works do not include quantitative evaluation [28] or they include it partially [16]. However, the so called LeavePostOut or LeaveTagsOut proposed in [15] and [10] sheds light on this issue. They pick up a random post for each user and they provide a set of tags for this post based on the whole folksonomy except such post. Then, they compute the precision and recall [12]

[3] available at http://www.csie.ntu.edu.tw/∼cjlin/liblinear/

as follows

$$precision(T) = \frac{1}{|D|} \sum_{(u,r) \in D} \frac{|\mathcal{T}^+(u,r) \cap \mathcal{T}(u,r)|}{|\mathcal{T}(u,r)|} \tag{7}$$

$$recall(T) = \frac{1}{|D|} \sum_{(u,r) \in D} \frac{|\mathcal{T}^+(u,r) \cap \mathcal{T}(u,r)|}{|\mathcal{T}^+(u,r)|} \tag{8}$$

where D is the test set, $\mathcal{T}^+(u,r)$ are the set of tags user u has assigned to resource r (positive tags) and $\mathcal{T}(u,r)$ are the set of tags the system has recommended to user u to assign to resource r. The F_1 measure could be computed from them as

$$F_1 = \frac{1}{|D|} \sum_{(u,r) \in D} \frac{2|\mathcal{T}^+(u,r) \cap \mathcal{T}(u,r)|}{|\mathcal{T}(u,r)| + |\mathcal{T}^+(u,r)|} \tag{9}$$

The evaluation adopted in this paper consists of computing the F_1, but just considering at most the first five tags of the ranking. Notice that such kind of evaluation quantifies the quality of a classification rather than the quality of a ranking.

8 Experiments

8.1 Data Sets

The experiments were carried out over the ECML PKDD Dicovery Challenge 2009 datasets [4]. This work studies the Task 1: Content-Based Tag Recommendations and Task 2: Graph-Based Recommendations of the 2009 Challenge. The test dataset of the former contains posts, whose user, resource or tags are not contained in the post-core at level 2 of the training data whereas the latter assures that the user, resource, and tags of each post in the test data are all contained in the training data's post-core at level 2.

The post-core at level 2 is got through cleaning dump and removing all users, tags, and resources which appear in only one post. This process is repeated until convergence and got a core in which each user, tag, and resource occurs in at least two posts.

The tags were cleaned by removing all characters which are neither numbers nor letters from tags. Afterwards,those tags which were empty after cleaning or matched one of the tags imported, public, systemimported, nn, systemunfiled were removed.

The cleaned dump contains all public bookmarks and publication posts of BibSonomy [5] until (but not including) 2009-01-01. Posts from the user dblp (a mirror of the DBLP Computer Science Bibliography) as well as all posts from users which have been flagged as spammers have been excluded.

[4] http://www.kde.cs.uni-kassel.de/ws/dc09/dataset

[5] http://www.bibsonomy.org

To make the experiments, the datasets of the Tasks 1 and 2 were split into 2 different datasets. The former is made up by bookmark posts whereas the latter by bibtex posts. These sets will be respectively called bm09 no core and bt09 no core for Task 1 and bm09 core and bt09 core for Task 2.

8.2 Discussion of results

This section deals with the experiments carried out. A binary representation of the examples was empirically chosen. Hence, the value of a feature will be 1 if this feature appears in the example and 0 otherwise. For each one, several tag sets are provided depending on the parameters described before:

- The four ways of collaboration: resource, user, union and intersection.
- The twelve measures for selecting relevant posts.
- The penalizing degree of the oldest posts. Several values were checked. Those are 0, 0.0625, 0.25 and 1.

Data	Kind of Collaboration	Measure	Penalizing degree	F_1
'bm09 no core'	Intersection	IL_{ir}	0	28.54%
'bt09 no core'	Intersection	IL_{ir}	0.0625	28.56%
'bm09 core'	Intersection	IG	0	30.90%
'bt09 core'	Intersection	IG	0.0625	37.07%

Table 1. The parameters and performance of the best settings in training data for all post collections

Table 1 shows the best setting parameters together with their F_1 computed when at most 5 tags are returned, obtained using training sets. The parameters of Table 1 were established to classify the test datasets, obtaining the results shown in Table 2.

Data	F_1
'bm09 no core'	7.28%
'bt09 no core'	6.75%
'bm09 core'	24.21%
'bt09 core'	28.76%
Task 1 'bm09 and bt09 no core'	6.98%
Task 2 'bm09 and bt09 core'	26.25%

Table 2. The performance of test data for all post collections

The performance corresponding to Tasks 1 and 2 can be seen in the two last rows of Table 2.

Table 3 shows the effect of including collaboration, post selection and a penalization of the oldest posts.

Data	F_1 no Collaboration	F_1 no post selection	F_1 no penalization
'bm09 no core'	28.02%	27.25%	28.54%
'bt09 no core'	28.53%	27.30%	28.51%
'bm09 core'	29.32%	26.32%	30.90%
'bt09 core'	36.10%	34.74%	36.84%

Table 3. The effect of collaboration, post selection and a penalization of the oldest posts

All the experiments carried out allow to conclude that the collaboration slightly improves the performance of the recommender. Particularly, the collaboration using resources and using both resources and users by intersection offer the best results with regard to the collaboration using users and using both resources and users by union. Furthermore, collaboration by intersection grants the best results. Including measures to select promising posts improves the recommender. Although the behavior among them is quite similar, measures coming from the Information Theory field together with those based on the impurity level provide the best results. The former seem to be more adequate to the core data whereas the latter improve the results of the no core data. The effect of time differs from one collection to another. The best results are reached without taking into account the time (parameter $e = 0$) for the bookmark collections, either core or no core versions. However, it seems that penalizing oldest posts improves the performance for the bibtex collections, either core or no core versions.

9 Conclusions

This work proposes a TRS based on a novel approach which learns to rank tags from previous posts in a folksonomy using a logistic regression based system. The TRS includes several ways of collaboration among users and resources. It also includes a selection of promising posts using scoring measures and penalizing the oldest ones.

The collaboration using intersection of tags that both users assign and resources have improves the performance of the recommender with regard to other types of collaboration. Selecting posts using scoring measures makes the recommender provide best tags, although in general the behavior of all of them is quite similar. However, the Information Theory measures offers best results for the core data and the impurity level measures do it for the no core data. Finally, penalizing oldest posts improves the results for the bibtex collections, but it does not obtain satisfactory results for bookmarks collections.

References

1. B. Adrian, L. Sauermann, and T. Roth-Berghofer. Contag: A semantic tag recommendation system. In *Proceedings of I-Semantics' 07*, pages 297–304, 2007.
2. H.S. Al-Khalifa. Coolrank: A social solution for ranking bookmarked web resources. In *Innovations in Information Technology, 2007. Innovations '07. 4th International Conference on*, pages 208–212, 2007.
3. Pierpaolo Basile, Domenico Gendarmi, Filippo Lanubile, and Giovanni Semeraro. Recommending smart tags in a social bookmarking system. In *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, pages 22–29, 2007.
4. Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM.
5. S.S. Keerthi C. J. Lin, R. C. Weng. Trust region newton method for logistic regression. *Journal of Machine Learning Research*, (9):627–650, 2008.
6. Andre Elisseeff and Jason Weston. A kernel method for multi-labelled classification. In *In Advances in Neural Information Processing Systems 14*, pages 681–687. MIT Press, 2001.
7. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, August 2008.
8. Shantanu Godbole and Sunita Sarawagi. Discriminative methods for multi-labeled classification. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *PAKDD*, volume 3056 of *Lecture Notes in Computer Science*, pages 22–30. Springer, 2004.
9. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Trend detection in folksonomies. In Yannis S. Avrithis, Yiannis Kompatsiaris, Steffen Staab, and Noel E. O'Connor, editors, *Proceedings of the First International Conference on Semantics And Digital Media Technology (SAMT)*, volume 4306 of *LNCS*, pages 56–70, Heidelberg, 12 2006. Springer.
10. Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In Alexander Hinneburg, editor, *Workshop Proceedings of Lernen - Wissensentdeckung - Adaptivit (LWA 2007)*, pages 13–20, sep 2007.
11. Thorsten Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM.
12. Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 75–83, 2008.
13. Ralf Krestel and Ling Chen. The art of tagging: Measuring the quality of tags. pages 257–271. 2008.
14. Sigma On Kee Lee and Andy Hon Wai Chun. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ann semantic structures. In *ACOS'07: Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science*, pages 88–93, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).
15. Leandro Balby Marinho and Lars Schmidt-Thieme. Collaborative tag recommendations. In Springer Berlin Heidelberg, editor, *Studies in Classification, Data Analysis, and Knowledge Organization*, pages 533–540, 2008.

16. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM Press. paper presented at the poster track.
17. D. Mladenic and M. Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In *Proc. 16th International Conference on Machine Learning ICML-99*, pages 258–267, Bled, SL, 1999.
18. E. Montañés, I. Díaz, J. Ranilla, E.F. Combarro, and J. Fernández. Scoring and selecting terms for text categorization. *IEEE Intelligent Systems*, 20(3):40–47, May/June 2005.
19. John C. Platt and John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.
20. J. Ranilla and A. Bahamonde. Fan: Finding accurate inductions. *International Journal of Human Computer Studies*, 56(4):445–474, 2002.
21. G. Salton and M. J. McGill. *An introduction to modern information retrieval.* McGraw-Hill, 1983.
22. F. Sebastiani. Machine learning in automated text categorisation. *ACM Computing Survey*, 34(1), 2002.
23. Börkur Sigurbjörnsson and Roelof van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 327–336, New York, NY, USA, 2008. ACM.
24. Sanjay Sood, Sara Owsley, Kristian Hammond, and Larry Birnbaum. Tagassist: Automatic tag suggestion for blog posts. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.
25. M. Tatu, M. Srikanth, and T. D'Silva. Rsdc'08: Tag recommendations using bookmark content. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 96–107, 2008.
26. G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.
27. M. Vojnovic, J. Cruise, D. Gunawardena, and P. Marbach. Ranking and suggesting tags in collaborative tagging applications. Technical Report MSR-TR-2007-06, Microsoft Research, 2007.
28. Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *WWW2006: Proceedings of the Collaborative Web Tagging Workshop*, Edinburgh, Scotland, 2006.
29. Shiwan Zhao, Nan Du, Andreas Nauerz, Xiatian Zhang, Quan Yuan, and Rongyao Fu. Improved Recommendation based on Collaborative Tagging Behaviors. In *Proceedings of the International ACM Conference on Intelligent User Interfaces (IUI2008)*, Canary Islands, Spain, 2008.

Content- and Graph-based Tag Recommendation: Two Variations

Johannes Mrosek, Stefan Bussmann, Hendrik Albers, Kai Posdziech,
Benedikt Hengefeld, Nils Opperman, Stefan Robert and Gerrit Spira

FH Gelsenkirchen, Department of Computer Sciences, Neidenburger Strasse 43,
45877 Gelsenkirchen, Germany,
mrosek@internet-sicherheit.de, stefan.bussmann@gmx.net, joschgg@gmx.de,
kaip@gmx.net, hengefeld@web.de, nilsoppermann@web.de, FSMARINE@gmx.de,
gerrit.spira@gmx.de

Abstract. We describe two variants of our approach to tackle the task 1 & 2 of the ECML PKDD Discovery Challenge 2009 where each contenter had to identify up to 5 tags for each resource of a given set of either bibtex-like references to publications or bookmarks. The quality of the results was measured against the tags that users of the data source (www.bibsonomy.org) had originally assigned to the resources (F1 measure). In our approach, we either generate tags (from the content of the given resource data or after crawling additional resources) or we request tags from tagging services. We call each of this tag sources a *tag recommender*. We then combine the results of the tag recommenders based on weighting factors. The weighting factors are determined experimentally by comparing generated and expected tags based on the available training data. This general idea is also used for the graph-based approach required to solve task 2. Here again, the final tag recommendations are computed from the individual results of the different tag-recommending algorithms. In the preliminary result list, we ranked second for task 1 (Group 2) and nineth for task 2 (Group 1).

Key words: content-based graph-based tag recommendation bibsonomy

1 Preliminaries

Assigning tags to resources can be an effective instrument to organize an information space. Users interacting with this information space may utilize the tags to identify relevant resources or groups of resources. User-driven tag assignment is a popular way to ensure at least some sort of tag quality[1]. Often, the users assigning tags are supported by algorithmic tag recommendation. This may be as simple as offering auto-complete fields for entering tags that display tags already assigned by users, or it may be based on a full-fledged analysis of the resource

[1] in the sense that the tags indeed help users to find or organize resources, for recent results on tag quality compare [2, 3].

the user wants to tag. This analysis may present a set of automatically identified tags to the user. Within the later context, compare [1], we describe two variants of our approach to content-based tag recommendation which we applied to task 1 and 2 of the ECML PKDD Discovery Challenge 2009. We acted as two teams (with completely independent implementations), each team deploying its own variation of the overall approach – and each with different success. In the following we first describe the solution and results of group 1[2] for tasks 1. This will be followed by a brief presentation of the differences of the solution for task 1 of group 2[3] and their results. We conclude with details of the solution for task 2 of group 1.

2 Content-based Tag Recommendation

This attempt on content-based tag recommendation uses different sources to generate tag candidates. These candidates are combined on the basis of appropriate weighting factors which have been assigned to the particular sources ex ante. The first kind of source are web services, which offer information for known resources. This information contains tags which already have been assigned to those resources. In this case we query del.icio.us and citeulike.org. The second kind of source are the resources themselves. In case of bookmarks the content of the according websites is crawled and analyzed. For bibtex entries the information contained in the bibtex table is taken into account. So tag candidates are determined without using any external service. The last source is also a web service called tagthe.net. It already has an engine, which recommends tags for a given URL.

2.1 Harvesting Tag Candidates

The first step to a tag recommendation is the accumulation of candidates and additional information from the several sources. For every URL in the bookmark table, the del.icio.us service is queried. It can be accessed via a feed. The URL md5 hash of the resource is inserted into a URL-pattern. When the resulting URL is accessed, all available information is returned. It includes a count which specifies how often the URL was posted, and a list of top tags assigned to it. Each tag is supplemented with an information about the frequency with witch it has been assigned to the given resource.
In some cases it is possible to find tag candidates for bibtex entries at citeulike.org. The description or misc field of the bibtex table often contain a citeulike-id. With this id the according citeulike page can be called and crawled for the assigned tags. These tags are already classified as ”tag”, ”publisher” and ”author”. Numerical information like a count cannot be obtained.
A similar kind of data is provided by the service tagthe.net. It generates classified tags for a submitted URL automatically and independent of the document

[2] Group 1: Mrosek, Bussmann, Albers, Posdziech.

[3] Group 2: Hengefeld, Opperman, Robert, Spira.

format behind it. The available categories are "tag", "author", "person" and "location". Like citeulike.org, tagthe.net returns no numerical information. Anyway this service can be seen as a backup system, because it can provide tag candidates for every URL. Thus it is called for all URLs in the bookmark and the bibtex table. Unfortunately there is no information about the algorithmics of the service available.
In addition to the citeulike-ids the bibtex table contains further interesting information. The title, journal and description fields contain words that can be potentially used as concise tags. Hence after comparison with a stop word list all remaining words in these fields are interpreted as tag candidates.
The last source for tag candidates are the websites behind the URLs in the bookmark table. After crawling and parsing the site's source code the words are counted and checked against a stop word list. Furthermore they are classified by the location of their appearance like in "meta keywords", "meta description", "title" and "body".

2.2 Selection Tags from the Candidates

The recommendation system has a hierarchical structure. This means there is one meta recommender which relies on several separate source recommenders, one for each source described in section 1.1. These recommenders provide up to 20 tag candidates for a queried content id. Every tag candidate is complemented by a score the according recommender assigns to it. This score can vary between 0 and 1. How the recommender constitutes that score depends on the source and is described in section 1.3. After the source recommenders have made their suggestions, the meta recommender takes all of the intermediate results and determines the actual tags. Therefor it combines a candidate's frequency of appearance in all sources k with the score s_i $(1 \leq i \leq k)$ provided by the source recommenders. s_i is already influenced by a weighting factor for the individual sources. How this factor is determined will be described in section 1.4. The final score s for a tag candidate can be determined by the formula

$$s = k \cdot (s_1 + s_2 + ... + s_k). \tag{1}$$

Note that s can be a value greater than 1. When the meta recommender has calculated these aggregated scores for all candidates, they are ordered by this new information. The five tags with the highest scores are selected as recommended tags. In order to prohibit recommendation of tags with a very low score, an optional filter can be set. This helps to enhance the precision.

2.3 Calculating the Individual Scores

As already stated the calculation of the single scores differs depending on the source. The reason for this is the additional information, which is provided besides the tags. Every source offers a different kind of information. To calculate a del.icio.us score $s_{delicious}$ for a tag, the number of posts n which assigned it to the resource is devided by the total number p of posts for the resource at del.icio.us. After that the result is multiplied by the weight $w_{delicious}$ constituted for del.icio.us.

$$s_{delicious} = w_{delicious} \cdot \frac{n}{p} \tag{2}$$

The scores for tagthe.net and citeulike candidates are determined as follows: For each of the provided categories a weighting factor $c_{categorie}$ is defined: $c_{tag} = 1.0$, $c_{author} = 0.3$, $c_{publisher=0.2}$, $c_{location} = 0.2$, $c_{person} = 0.1$. The score $s_{tagthe/cite}$ of each tag candidate is the result of the product of the source's weight $w_{tagthe/cite}$ and the categorie's weight $c_{categorie}$. In one case there is an exception from this practice. As can be detected in the training data the tag "juergen" appears pretty often. So this special tag is always handled with a score of 1.0. Otherwise all scores are determined by

$$s_{tagthe/cite} = w_{tagthe/cite} \cdot c_{categorie}. \tag{3}$$

The tag candidates generated from the bibtex entries are treated in a similar way. For some fields a weighting factor c_{field} is set: $c_{title} = 0.55$, $c_{journal} = 0.25$, $c_{description} = 0.2$. Only tags from the title field are used as a suggestion for the meta recommender. These candidates get a higher score if they also appear in the journal or description field. The individual scores for these fields are added to the initial title-score of 0.55.

The scoring process for the crawled content of a website is a little more complex. It is based on the information about the location of appearance as described in section 1.1. Three steps are passed until the final score is determined. In the first step a tag's frequency of appearance $n_{location}$ in the individual locations is counted. This value is weighted by a factor $c_{location}$ which is related to the location's importance. E.g. a word from the title or the keyword-list is rather a good tag than one from the body. All counts are multiplied by their weighting factors get summed up to a kind of weighted frequency n_{wfreq} for the whole resource

$$\begin{aligned} n_{wfreq} = n_{title} \cdot c_{title} + n_{description} \cdot c_{description} + \\ n_{keywords} \cdot c_{keywords} + n_{body} \cdot c_{body}. \end{aligned} \tag{4}$$

If a tag appears at different locations, its score is raised in step two. This takes into account the fact that such a tag is a good tag in most cases. To raise the score, n_{wfreq} is multiplied by a factor f_{cat} which is related to the number of categories the tag appeared in.

$$f_{cat} = \begin{cases} 1, & 1 \text{ categorie} \\ 1.5, & 2 \text{ categories} \\ 3, & 3 \text{ categories} \\ 5, & 4 \text{ categories} \end{cases} \tag{5}$$

The modified weighted frequency n^*_{wfreq} is determined by

$$n^*_{wfreq} = n_{wfreq} \cdot f_{cat}. \tag{6}$$

In the last step, n^*_{wfreq} is normalized to the common score interval $[0, 1]$. Therefor every n_{wfreq} is divided by the highest n_{wfreq} which is reached for the crawled resource. The result is a score s_{cc} for a tag from the crawled content. The number of tags which are returned to the meta recommender is limited. Only the 20 tags with the highest scores are chosen.

2.4 Calculating the Source Weights

A key point in this two-stage recommendation approach is the calculation of suitable weighting factors for the different sources. Their tag-quality varies in a wide range. Thus handling the candidates of the individual recommenders as if they were of similar quality leads to bad results. As a foundation for the weighting factor calculation the postcore-2 of the training dataset was used. For every source the set of tag assignments it can supply was determined and an according tas file was generated. E.g. to calculate a factor for citeulike, a tas file with all content ids, which are associated with bibtex entries containing a citeulike-id, was generated. Corresponding to the tas files a result file was created for every source. This result was independent of the meta recommender and all the other sources. The result files were measured with f1-score against the tas files created before. The first attempt was to use the f1 score a result file achieved as the weighting factor for the according source recommender.

However when testing the complete recommendation process against postcore-2, experiments with various weighting factors pointed out a better choice. Using the precision value which has been computed for the various result files to get the f1-score, provides better overall results. Thus all the weighting factors for the particular sources assumed in the meta recommender match the precision the respective source recommender reaches for the subset of tag assignments it can supply.

2.5 Results

Table 1 displays the results each recommender achieved for his own subset of the postcore-2 training data. The intermediate results in the first five rows show

recall, precision and f1-score for every source recommender. The weighting factor later used for the meta recommender is the particular precision as described in section 1.4.
Row five and six present the results of the meta recommender for the training data. The first is generated without a filter whereas the second one uses a filter to avoid tag recommendations with very low scores as described in section 1.2.

Table 1. Comparison of the results for test and training data set

Dataset	Recommender	Supplied Posts	Recall	Precision	F1-Score
Training	citeulike.org	5285 / 6372	0.446	0.134	0.206
	del.icio.us	38383 / 40882	0.418	0.353	0.383
	tagthe.net	40468 / 51580	0.066	0.053	0.059
	bibtex	22341 / 22341	0.155	0.127	0.139
	web content	33193 / 40882	0.150	0.123	0.135
	meta	63107 / 64120	0.350	0.254	0.294
	meta with filter	62104 / 64120	0.344	0.269	0.302
Test	**Meta/Overall** with filter	30844 / 43002	0.132	0.103	**0.116**

The table shows that del.icio.us and citeulike produce good tag candidates. Recommendations made by other sources have a much lower quality. The meta recommenders result for the test dataset is far below the result of the training data set.

2.6 Conclusion

The comparison of the meta recommender's results for training and test data leads to the conclusion that the choice of sources was suboptimal. As the training results were computed on the postcore-2 the web services provided tags with a high quality. Every resource was posted in bibsonomy at least two times. So the probability that it is contained in the web services' databases as well, was pretty good. Thus the decrease in score might be explainable by the unpopularity of the resources in the test dataset.
In conclusion the two stage approach is a good attempt for popular resources. To raise the result quality for unpopular resources too, further sources with previously assigned tags have to be added.

2.7 Group 2: Variations for Task 1

Table 2. Comparison of the results for test and training data set

Dataset		Recommender	Recall	Precision	F1-Score
Training	Bookmarks	del.icio.us	0.391	0.280	0.326
		WebCrawler	0.139	0.099	0.116
		DataSet	0.113	0.092	0.102
	Bibtex	WebCrawler	0.250	0.128	0.169
		GoogleScholar	0.087	0.073	0.079
		DataSet	0.083	0.061	0.070
	Meta		0.334	0.213	0.260
Test	**Meta/Overall**		0.214	0.155	**0.180**
	Meta	w CiteULike	0.218	0.157	0.183

Harvesting Tags: In addition to citeulike and del.icio.us, group 2 implemented a tag recommender using Google Scholar for bibtex entries. Using the title data only, links to the resource itself or similar resources were harvested. From the first ten entries three were selected by counting identic words in the titles of the referenced documents (ignoring certain stop words). In the competition, the Google scholar tag recommender harvested roughly 60.000 links for 24.000 bibtex data entries. Subsequently, the Web content crawler of group 2 was used to obtain candidate tags from the three selected resources.

Furthermore, the web content crawler was able (to a certain extend) to parse PDF documents in addition to HTML documents. Also the implementation of the citeulike tag recommender differed: a recent database dump of the citeulike data was used which made it possible to search for DOI-ids or URLs directly and to determine the overall frequency of tags in the citeulike data set. TagTheNet was not used. A straighforward tag recommender (called Data set recommender below) that analysed the relevant fields of the data entries complemented the set of recommenders.

Selecting Tags: No additional filtering for small scores was used. If less than 5 tags were harvested, the remaining slots for tags were filled by randomly drawing tags from the 30 tags most often used in bibsonomy[4]. The weights for a recommender could be different for bibtex and bookmark context. The weights for the

[4] Not a very successful idea – first evaluations show that this was rather counterproductive.

recommenders were chosen manually, based on experiences while experimenting with the training data. Tags recommended by more than one recommender were rated significantly higher by multiplying their score with a factor which depends exponentially on the number of recommendations.[5]

Results: The results of group 2 for task 1 are presented in table 2. Additional quantitative data are shown in table 3.

Please note that due to a persistent problem with the CiteULike tag recommender, it is not listed in the training data and haven't been used in the actual competition. Therefore, the relevant F1-score for the ranking in the competition is **0.180** (second place). Some more details on the impact of the different tag recommenders will be presented at the workshop.

Table 3. Additional quantitative data

	Competition	Training
Tags harvested	1432413	8389379
Bookmarks tagged	16898	263004
Bibtex entries tagged	26104	158924

3 Graph-Based Tag Recommendation

The graph-based approach adapts the same hierarchical recommender structure the content-based approach is based on[6]. This means that there is also one meta recommender which combines the results of subordinated recommenders.
Instead of using external sources, these recommenders implement different algorithms. Every algorithm processes bibsonomy's graph structure and the contained user information in a different way to generate a set of tag recommendations.
Like for the content-based approach every tag is valuated with a suitable weighting factor. This weighting factor also depends on the quality of the subordinated recommender. It is determined in the same way as for the source recommenders in Task 1 (cmp. section 1.4). The used benchmark data set contains the whole postcore-2. For processing the final test data set, three different algorithms were used to supply the meta recommender. All of these algorithms have a different focus on evaluating the graph-structure. The specific operation methods are presented in sections 2.1 to 2.3.

[5] value = value * Math.pow((1.0 + 3*(count - 1)/10), count);

[6] The following section documents the work of group 1 only

3.1 Tag by Resource

The first algorithm selects tags based on the resource information. Therefor all tags which have already been assigned to a queried resource are determined. Additionally each tag's frequency of appearance n_{local} in combination with the resource is counted. To calculate the score s_{tr}, this value is taken and divided by the number n_{post} the resource was posted. The result is multiplied by the weighting factor w_{tr} for this recommender. Like in section 1, the recommenders precision for postcore-2 is used for this purpose.

$$s_{tr} = w_{tr} \cdot \frac{n_{local}}{n_{post}} \tag{7}$$

If two tags reach the same score, the one with the higher frequency of appearance in the entire postcore is preferred.

3.2 Tag by User

The second algorithm recommends and scores tags with a special relation to the user. Algorithmically this approach is very similar to the one described in section 2.1. At first the set of tags is identified which the user who makes a post has assigned to other posts previously. The frequency of usage for the tags n_{local} is also determined. From this set of tags the n_{local}-value for the most used one is taken as a reference value n_{max}. The score s_{tu} this recommender calculates for a tag is as follows

$$s_{tu} = w_{tu} \cdot \frac{n_{local}}{n_{max}} \tag{8}$$

The weighting factor w_{tu} is determined as usual (cmp. Section 2.1).

3.3 Tag by User Similarity

The last algorithm determines tags, which have been used by similar users. The similiarity between two users is defined over the number of equal resources posted by them.

Therefor the first task is the determination of similarities between all users. These are calculated by the Tanimoto-Score under consideration of the posted resources. These resources are used as comparison criterion for different users.

In the recommending process the top five of the similar users are identified. The tags they assigned to the posted resource are accumulated and sorted by their frequency of appearance n_{count} in the posts of the top five users group. As a reference value the maximum frequency of appearance n_{max} of a tag in the same group is utilized. The final score s_{us} for a tag generated by this recommender is calculated with the following formula:

$$s_{us} = w_{us} \cdot \frac{n_{count}}{n_{max}} \tag{9}$$

3.4 Results

The results for Task two are arranged the same way as the results for the content-based approach in task one. The recall, precision and f1-score values for the training datatset are shown individually for every algorithm. Furthermore results are visualized for meta recommenders which use different combinations of the subordinated recommenders. A filter like the one described in 1.2 is tested too.
The recommendations for the test dataset have been made by a meta recommender which uses all of the three subordinated recommenders and a filter. They are a good deal worse than the results for the training dataset.

Table 4. Comparison of the results for test and training data set

Dataset	Recommender	Supplied Posts	Recall	Precision	F1-Score
Training	1. by Resource	64120	0.731	0.586	0.651
	2. by User	64120	0.349	0.205	0.258
	3. by User-Sim.	34738	0.265	0.271	0.268
	1. + 2. + 3	64120	0.774	0.517	0.620
	1. + 2.	64120	0.846	0.570	0.681
	1. + 2. with filter	64120	0.846	0.576	0.685
Test	1. + 2. + 3. with filter	778 / 778	0.389	0.262	0.313

3.5 Conclusion

Like for task 1 there is a dramatic decrease of the f1-score in the test. In this case the explanation can be found in the composition of the training data files. When evaluating the methods for task 2 with the training data, the post, a recommendation is made for, is not excluded from the dataset. Thus the tags of the post are definitely in the training dataset. This increases the probability that the expected tags are recommended. As a consequence, the scores are higher as for the test data which are not included in post-core 2.
In conclusion, the evaluation of the training data is not valid, but the algorithm works correct for the test data.

4 Final remarks

The competition offered an excellent test bed for our approach to tag recommendation. Though we are pretty happy with the results obtained, much work remains to be done:

- setting the weights for the different recommenders could be improved (iterative algorithmic optimisation; cleaning/choosing training data),
- relations to similar resources in the bibsonomy data set (identified for example with the help of the recommended tags or based on Web crawling / Google scholar results) could be explored: if, for a given resource x, a *similar* resource y in the bibsonomy data sets is identified, tag candidates can be drawn from resource y (either directly or via further graph-based or recursive similarity-based analysis),
- fine-tuning of the individual tag recommenders, for example by coupling the Web crawler to the tag database,

to name just a few open topics. We want to thank the organizers[7] and to express our hope that this stimulating and exciting competition will be continued in the future and that some of our results and suggestions may help others to develop further improved tag recommenders.

References

[1] Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag Recommendations in Folksonomies. In: PKDD 2007, Springer (2007)

[2] Sen, S., Vig, J., Rield, J.: Learning to Recognize Valuable Tags. In: IUT'09, ACM (2009)

[3] Zhang, S., Farooq, U., Carroll, J.M.: Enhancing Information Scent: Identifying and Recommending Quality Tags. In: GROUP'09, ACM (2009)

[7] and the anonymous reviewers and Wolfram Conen for their valuable comments.

A Two-Level Learning Hierarchy of Concept Based Keyword Extraction for Tag Recommendations

Hendri Murfi and Klaus Obermayer

Neural Information Processing Group, TU Berlin
Franklinstr. 28/29, 10587 Berlin, Germany
{henri,oby}@cs.tu-berlin.de
http://ni.cs.tu-berlin.de

Abstract. Textual contents associated to resources are considered as sources of candidate tags to improve the performance of tag recommenders in social tagging systems. In this paper, we propose a two-level learning hierarchy of a concept based keyword extraction method to filter the candidate tags and rank them based on their occurrences in concepts existing in the given resources. Incorporating user-created tags to extract the hidden concept-document relationships distinguishes the two-level from the one-level learning version, which extracts concepts directly using terms existing in textual contents. Our experiment shows that a multi-concept approach, which considers more than one concept for each resource, improves the performance of a single-concept approach, which takes into account just the most relevant concept. Moreover, the experiments also prove that the proposed two-level learning hierarchy gives better performances than one of the one-level version.

Key words: Recommender system, Social tagging, Machine learning, Keyword extraction, Concept extraction

1 Introduction

Social tagging is intended to make resources increasingly easy to discover and recover over time. Discovery enables users to find new content of their interest shared by other users. This social indexing gives a promising index quality because it is done by human beings, who understand the content of the resource, as opposed to software, which algorithmically attempts to determine its meaning. Moreover, it is done collectively among users, that is, it uses a collective human intelligence as an index extractor. Recovery enables a user to recall content that was discovered before. It should be easier because the tags are both originated by, and familiar to, its primary users. However, Golder et al. [9] identify three major problems with the current social tagging systems: polysemy, synonymy, and level variation. The first two inherit the problems of natural language, while the third one refers to the phenomenon of users tagging content at different levels of abstraction. Other problems are dealing with word forms, nouns in singular, nouns in plural, abbreviations, and misspelled words.

To direct users towards the consistency of the tags, the system usually has a service that assists users in the tagging process, by automatically recommending an appropriate set of tags. The service is a mediated suggestion system, that is, the service does not apply the recommended tags automatically, rather it suggests a set of appropriate tags and allows the user to select tags from the set they find appropriate. Moreover, the tag recommendation can serve many purposes such as consolidating the vocabulary across the users, giving a second opinion what a resource is about and, the important thing, increasing the success of searching because of the consistency [14].

In practice, the standard tag recommenders are services that recommend the most popular tags used for either a particular resource or a whole system. There are other methods proposed from a diversity of approaches to recommend tags from user-created tags (folksonomy) such as information retrieval [23, 28], graph-based approaches [11], collaborative filtering [14], machine learning [10, 15]. Recently, people consider textual contents associated to the resources as sources of candidate tags to improve the performance of tag recommenders. For example, Xu et al. [31] suggest content-based (and context-based) tags based on analysis and classification of the tagged content and context. This not only solves the cold start problem, but also increases the tag quality of those objects that are less popular. Tatu et al. [29] use natural language processing tools to extract important terms (nouns, adjectives and named entities) from the textual contents. They conclude that the understanding of the contents improves the quality of the tag recommendations.

In this paper, we also consider the textual contents associated to resources as sources of candidate tags to improve the performance of the tag recommender in the social tagging system. To achieve this goal, we propose a two-level learning hierarchy of concept based keyword extraction as a tag recommendation method. Firstly, the method extracts concepts, which can be considered as a set of related words, using nonnegative matrix factorization (NMF) from training document collections using a two-level learning hierarchy: at the lower level the method extracts concepts and concept-document relationships using user-created tags. Having these relationships, the method populates the concepts with terms existing in textual contents of resources at the higher level. Next, the tag recommender finds the relevant concepts to a given resource and then scales terms of the resource based on their occurrences in the concepts. The terms having the highest scores are set as keywords and recommended as tags. Incorporating the user-created tags to extract the hidden concept-document relationships distinguishes the two-level from the one-level learning version, which extracts concepts directly from terms existing in textual contents. The main advantage of this approach is that NMF algorithm decomposes more compact document representations. Also, the concept extraction from textual contents is handled by nonnegative least squares algorithm which is much more efficient than NMF algorithm. Therefore, the two-level learning hierarchy approach is not only more efficient but also more reliable because it uses tags created by users who understand the content of documents. Moreover, the approach may have richer vocabularies because it can combine vocabularies from tag space and content space. Our experiment shows that a multi-concept approach, which considers more than one concept for each resource, improves the f-measure values of a single-concept approach, which takes into account just the most relevant concept, about 10%. Moreover, the experiments also prove that the proposed two-level learning hierarchy has f-measure values 13% better than one of the one-level version.

The rest of the paper is organized as follows: Section 2 discusses a concept extraction method using nonnegative matrix factorization and our proposed two-level learning hierarcy method. Section 3 describes the existing keyword extraction methods and the proposed concept based keyword extraction methods. In Section 4, we describe a tag recommendation algorithm which combines keywords, which are extracted by the keyword extraction methods, with user-created tags in training data. In Section 5, we show our experiments and results. We conclude and give a summary in Section 6.

2 Concept Extraction

Many researchers are trying to address questions about concepts and, in this section, we consider one of them that defines the concepts as a set of related terms. These definitions are proposed and used by some researchers such as [21] or [27]. They use clustering methods to extract the concepts from training document collections. *Formal concept analysis* (FCA) [5,26] and *Latent semantic analysis* (LSA) [3, 7] are other methods to perform this task .

2.1 A One-Level Learning Hierarchy for Concept Extraction

There are some disadvantages of *singular value decomposition* (SVD) to extract concepts from a document collection as used by LSA. Its negative values make a semantic interpretation difficult. What we would really like to say is that a concept is mostly concerned with some subset of terms, but any semantic interpretation is difficult because of these negative values. To circumvent this problem, a new method which maintains the nonnegative structure of original documents has been proposed. The method uses *nonnegative matrix factorization* (NMF) [17] rather than SVD to extract the concepts from document collections.

Let V be a $m \times n$ term-by-document matrix whose columns are document vectors and a positive integer $k < min(m, n)$. In this paper, we use NMF to extract concepts from the term-by-document matrix V. NMF problem is how to find a nonnegative $m \times k$ matrix W and a nonnegative $k \times n$ matrix H to minimize the functional [2]:

$$\begin{aligned} \min_{W,H} \quad & f(W,H) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(V_{ij} - (WH)_{ij}\right)^2 \\ \text{subject to} \quad & W_{ia} \geq 0, H_{aj} \geq 0, \forall i, a, j \ . \end{aligned} \tag{1}$$

The constrained optimization problem above is convex on either W or H, but not on both, hence realistic possible solutions usually correspond to local minima. The product WH is called a nonnegative matrix factorization of V, although V is not necessarily equal to the product WH. Clearly the product WH is an rank-k approximation to V. An appropriate decision on the value of k is critical in practice, but the choice of k is very often problem dependent. In most cases, however, k is usually chosen such that $k << min(m,n)$.

The most popular approach for the NMF problem is the multiplicative update algorithm proposed by Lee and Seung [18]. To either overcome shortcomings related to convergence properties or to speed up this algorithm, researchers have proposed modifications of the algorithm or even created new ones [2]. In general, the algorithms can be divided into three general classes: multiplicative update algorithms [18,19], gradient descent algorithm [6,12], and alternating least squares algorithms [20, 24].

Because all elements of the matrix W and H are nonnegative, we can interpret them immediately as following: Each column of W corresponds to a set of related terms called concepts and each element w_{ia} of matrix W represents the degree to which term i belongs to concept a. Each element h_{aj} of matrix H represents the degree to which document j is associated to concept a. Next, we call this type of concept extraction as *an one-level learning hierarcy method.*

2.2 A Two-Level Learning Hierarchy for Concept Extraction

In case where the training documents are accompanied by user-created keywords, it is a good idea to incorporate the valuable information in learning process. For this reason, we propose a new learning scheme that uses the keywords for extracting concepts from a document collection. The learning scheme consists of two-level learning hierarchy. At the lower level, concepts and concept-document relationships are discovered using the user-created keywords. Having these relationships, the concepts are populated by terms existing in textual contents of documents at higher level. We expect this method to be successful because the hidden document structures are discovered using keywords collectively created by users. Another advantage of this approach is that NMF algorithm uses more compact document representations. On the other hand, the concept extraction from textual contents is handled by nonnegative least squares algorithm which is much more efficient than NMF algorithm. Therefore, this two-level learning hierarchy approach is not only more efficient but also more reliable because it uses tags created by users who understand the content of documents. Moreover, the approach may have richer vocabularies because it can combine vocabularies from tag space and content space. The detail algorithm of this method is described in Algorithm 1.

Algorithm 1 A two level learning hierarchy for concept extraction

```
Let $V$ be the tag by document matrix, and $X$ be the term by document matrix
Find the tag by concept matrix $W$ and the concept by document matrix $H$ from
  $V = WH$ using nonnegative matrix factorization (see Section 2.1) to minimize the
  functional:
      $f(W, H) = \frac{1}{2} \|V - WH\|^2$
Find the term by concept matrix $T$ from $X = TH$ using nonnegative least squares
  algorithm, e.g. [2]:
    – Solve for $T$ in matrix equation $HH^T T^T = HX^T$
    – Set all negative elements in $T$ to 0
```

3 Concept Based Keyword Extraction

Keyword extraction is the task of automatically selecting a small set of important, topical terms within the textual content of a document. The fact that the keywords are extracted means that the selected terms are present in the document [16]. In general, the task of automatically extracting keywords can be divided into two stages:

1. Selecting candidate terms in the document
2. Filtering out the most significant ones to serve as keywords and rejecting those that are inappropriate

There are various methods proposed for selecting candidate terms. The first one is n-gram extraction, that is, extracting uni-, bi-, or tri-grams, removing those that begin or end with a stop word [8]. Another one is more linguistically oriented using natural language processing (NLP) method such as NP-chunker or part-of-speech (PoS) [13]. Filtering uses either simple statistics, where a weighting schema is applied to rank words accoding to their score [1, 25], or machine learning, where the ranking function is defined by a statistical model derived from training set with manually assigned keywords [13, 22, 30].

In this section, we propose a machine learning based filtering method, that is, a method that uses concepts extracted from textual contents of documents. The method finds the relevant concepts to a given document and then scales terms of the document based on their occurrences in the concepts. The terms having the highest scores are set as keywords. The method can be considered as unsupervised learning when we use the one-level learning hierarchy. It means that the method does not need labeled data for the training process. Moreover, the method becomes a supervised method when the user-created tags are used in learning process for the two-level learning hierarchy approach. Two variants of the concept based keyword extraction method are described in detail in the following sections.

3.1 Single-Concept Based Keyword Extraction

The two-level learning hierarchy extracts concepts from a training document collection. Having these concepts, the single-concept based keyword extraction method finds the most relevant concept to a given document and then scales the candidate terms existing in the document based on their occurrence in the concept. The relevance of a concept c with a document d is calculated using the following cosine distance measure:

$$\text{rel}(\mathbf{d}, c) = \frac{\mathbf{d}^T V_c}{\|\mathbf{d}\| \, \|V_c\|} \tag{2}$$

The detail algorithm of this approach is described in Algorithm 2.

Algorithm 2 The single-concept based keyword extraction

```
1: Let column $c$ of $W$ $(W_c)$ be concept $c$ and $\mathbf{d}$ be a document
2: Let $rel(\mathbf{d}, c) = \frac{\mathbf{d}^T W_c}{\|\mathbf{d}\|\|W_c\|}$
3: Find the most relevant concept to the document $\mathbf{d}$, i.e., concept $c'$ where $c' = \text{argmax}_c(rel(\mathbf{d}, c))$
4: Scale terms existed in the document based on the most relevant concept, i.e.,
       $d1_i = w_{ic'} d_i$
       $d2_j = t_{jc'} d_j$
5: Combine the normalized terms:
       $\tilde{\mathbf{d}} = (1 - \alpha)\,(\mathbf{d1}/\,\|\mathbf{d1}\|) + \alpha(\mathbf{d2}/\,\|\mathbf{d2}\|)$
6: Select first $n$ non zero terms of the ranked $\tilde{\mathbf{d}}$ as keywords
```

3.2 Multi-Concept Based Keyword Extraction

The multi-concept based keyword extraction assumes that a document may contain more than one relevant concept. The detail algorithm of the multi-concept based keyword extraction method is described in Algorithm 3.

4 A Hybrid Tag Recommender

In our experiment, we use a hybrid recommender as described in detail in Algorithm 4. The recommender checks if a given resource exists in the training data.

Algorithm 3 The multi-concept based keyword extraction

```
Let column $c$ of $W$ ($W_c$) be concept $c$ and $\mathbf{d}$ be a document
Let $rel(\mathbf{d}, c) = \frac{\mathbf{d}^T W_c}{\|\mathbf{d}\| \|W_c\|}$
Scale terms existed in the document using the concepts, i.e.,
    $d1_i = \sum_c \mathrm{rel}(\mathbf{d}, c) w_{ic} d_i$
    $d2_j = \sum_c \mathrm{rel}(\mathbf{d}, c) t_{jc} d_j$
Combine the normalized terms:
    $\tilde{\mathbf{d}} = (1 - \alpha) \left(\mathbf{d1} / \|\mathbf{d1}\|\right) + \alpha(\mathbf{d2} / \|\mathbf{d2}\|)$
Select first $n$ non zero terms of the ranked $\tilde{\mathbf{d}}$ as keywords
```

If this is the case then the tag space based recommenders are suggested. The collaborative recommender is used if a given user has profiles in system. Otherwise, the most popular tag by resource method is used as tag recommender. If the resource appears for the first time then the recommender examines the content of the resource using the concept based keyword extraction algorithm. Boosting the extracted tags if they have been used by the user before. If neither any tags nor any keywords are suggested then the most popular tags in the training data are recommended. Using the user-created tags aims to direct the standardization and consistency of supplied tags, while using the tags extracted from textual contents intend especially to overcome the cold start problem.

Algorithm 4 A hybrid tag recommendation algorithm

```
input : a post $P < user, resource >$
if $P.resource$ exists in system then
    if $P.user$ exists in system then
        $P.tags \Leftarrow$ collaborative tags by $P.user$
    else
        $P.tags \Leftarrow$ most popular tags by $P.resource$
    end if
else
    $P.tags \Leftarrow$ the keywords by $P.resource$
    Boosting tags of $P.tags$ that exist in tag space of $P.user$
end if
if $P.tags = \phi$ then
    $P.tags \Leftarrow$ most popular tags
end if
Select top $n$ of the ranked $P.tags$ as recommended tags
```

5 Experiment

We apply our proposed recommender methods (Algorithm 4) for ECML PKDD Discovery Challenge 2009[1]. The task of the competition requires the development of a content-based tag recommendation method for BibSonomy[2], a web based social bookmarking system that enables users to tag both web pages (bookmark) and scientific publications (bibtex). The organizers of the competition made available a training set of examples consisting of the resources accompanied with their user-created tags. A testing data will be provided in order to evaluate proposed recommenders. Each bookmark is described by its URL, a description of the URL that usually is the title of the web page and an extended description of the bookmark supplied by the user. Each bibtex is associated with values of bibtex fields such as title, author, booktitle, journal, series, volume, number, etc. BibtexKey, bibtexAbstract, URL, and description of the publication can be specified. Some statistics of the data are shown in Table 1.

Table 1. Statistics of Experiment data

	Bookmark	Bibtex
Training	181,491	72,124
Testing	16,898	26,104

In our experiment, we use textual contents associated to each resource as content of the resources. For the bookmark, the contents are the description of the URL and the extended description. Title and abstract are textual contents associated to the bibtex. A bookmark is identified by its URL address (*url_hash*) attribute and a bibtex by its title (*simhash1*) attribute. Therefore, a document, bookmark or bibtex, is represented by the description given to the document by all users that bookmarked the document.

Let D be a testing data set, consisting of $|D|$ examples $(\mathbf{r}_i, T_i), i = 1...|D|$. Let T_i be the set of tags created by users for a resource $\mathbf{r}_i$ and P_i be the set of tags predicted by a recommender for a resource $\mathbf{r}_i$. The precision, recall, and F-measure for recommender f on testing data set D is calculated as follows:

$$\text{Precision} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|T_i \cap P_i|}{|P_i|}$$

$$\text{Recall} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|T_i \cap P_i|}{|T_i|}$$

$$\mathrm{F-Measure} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{2\,|T_i \cap P_i|}{|P_i| + |T_i|}$$

[1] http://www.kde.cs.uni-kassel.de/ws/dc09

[2] http://www.bibsonomy.org

We perform our experiment in java platform and use Lucene[3] for creating the tag-by-resource matrix and the term-by-resource matrix. The other processes are conducted on the Weka[4] framework, an open source machine learning software.

5.1 Experiment Settings

For each of the method of our experiment the settings we used to run them are described as following:

Concept-based keyword extraction . For creating the term-by-resource matrix, resources are parsed and a dictionary of terms is created using a standard word tokenization method. The terms are words, special characters are removed, and Snowball Porter stemming and standard stop words of English and German are applied. Finally, the term-by-resource matrix is created using a term frequency weighting scheme.

Extracting concepts from the term-by-resource matrix is an important step to find keywords from new resources. The optimal number of concepts (k), which captures most concepts in the training document collection, remains difficult to find. The method that is usually used for a practical purpose is a heuristic approach. However, because of the memory usage, simulations are usually conducted on the maximum number of concepts that can be extracted. In our experiment, we extract 200 concepts for the training document collection. For this task, we use the nonnegative double SVD initialization method [4] that conducts no randomization and the projected gradient method [20] that converges to a local minimum. We expect these combining methods leads to converge to a unique solution with a minimum error.

There is another parameter α that should be optimized in the two-level learning hierarchy approach. The parameter reflects the portion of the tag space and the content space as sources of tags for the recommender. In our experiment, we set the parameter $\alpha = 0.25$ for the single-concept method and $\alpha = 0.05$ for the multi-concept method, which are the optimal values we get using a heuristic method.

Collaborative recommendation [14] . For a given tag-by-user matrix X, a given user u, a given resource r, and integer k and n, the set $T(u, r)$ of n recommended tags is calculated by:

$$T(u,r) = \text{argmax}^n_{t \in T} \sum_{v \in N^k_u} sim(X_u, X_v)\delta(v,t,r) \tag{3}$$

where N^k_u is k nearest neighbors of u in X, $\delta(v,t,r) = 1$ if $(v,t,r) \in folksonomy$ and 0 else. Therefore, the only parameter to be tuned is the number of neighbors k. For that, multiple runs where performed where k incremented until a point where no more improvement in the results were observed.

[3] http://lucene.apache.org/

[4] http://www.cs.waikato.ac.nz/ml/weka/

Most popular tags by resource . For a given resource we count how many posts a tag occur together with that resource. We use tags that occur most often together with that resource as recommendation.

Most popular tags . For each tags we count in how many posts it occurs. We then use tags that occur most often as recommendation.

5.2 Single- vs. Multi-Concept Method

Fig. 1 shows the performances of the single- and multi-concept based keyword extraction on testing data. From Fig. 1, we can calculate that recall, precision, and f-measure of the multi-concept approach are, on average, 10%, 15% and 12%. The recall is likely to increase when the number of recommended tags gets bigger, while the precision is reduced for the bigger numbers of tags. Fig. 1 also shows the performance of the single-concept approach in the similar pattern and its f-measure is, on average, 11%. From both curves, we conclude that the multi-concept approach, which assumes that a resource may contain more than one concept, improves f-measure of the single-concept method, on average, 10%. The improvement occurs in all numbers of recommended tags. These results verify that associating of resources with more than one concept gives better performance than just considering the main concept of resources. In other words, some minor concepts of a resource should also be examined for getting the better performance of the keyword extraction.

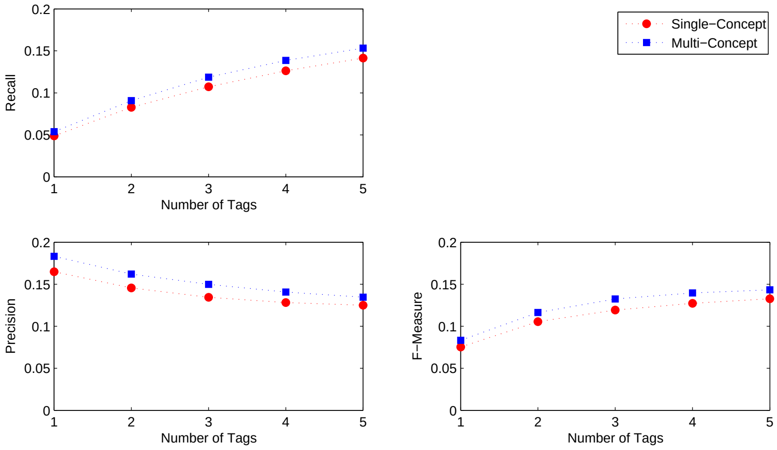


Fig. 1. Performance comparison of single- and multi-concept approach

5.3 One- vs. Two-Level Learning Hierarchy

In this section, we examine the performance of our proposed two-level learning hierarchy approach compared to the one-level version. Fig. 2 shows the performance of the one-level learning hierarchy multi-concept based keyword extraction and the two-level learning hierarchy multi-concept based keyword extraction. From the figure, we see that the two-level learning method has better recall, precision and f-measure. Its f-measure values, on average, are 13% better than one of the one-level learning approach. The detailed recall, precision, and f-measure values of the optimal performance of the two-level learning hierarchy are given in Table 2.

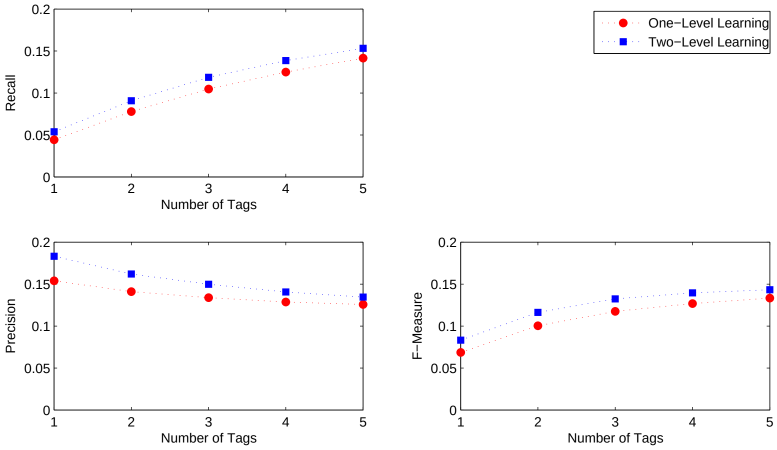


Fig. 2. Performance comparison of one- and two-level learning hierarchy approach

Table 2. Performance of two-level learning hierarchy of multi-concept based keyword extraction method for each number of recommended tags using the optimal parameter $\alpha = 0.05$

Num. of Tags	Recall	Precision	F-Measure
1	0.0538	0.1832	0.0832
2	0.0908	0.1621	0.1164
3	0.1187	0.1498	0.1324
4	0.1386	0.1406	0.1396
5	0.1533	0.1345	0.1433

6 Summary

In this paper, we propose a two-level learning hierarchy concept based keyword extraction method for task1 of ECML PKDD Discovery Challenge 2009, that is, a content-based tag recommendation. The tag recommendation method explores tags from textual contents of resources using concepts existing in the textual contents of the resources. A multi-concept approach, which considers more than one concept for each resource, improves the performance of a single-concept approach, which only considers the most relevant concept. Moreover, our experiment demonstrates that the proposed two-level learning hierarchy method outperforms the common one-level learning approach for all performance measures, e.g. recall, precision, and f-measure.

7 Acknowledgments

The authors would like to thank Nicolas Neubauer for useful discussions, comments and suggestions in writing this paper. The first author was funded partly by a scholarship by the DAAD.

References

1. K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Canadian Conference on Artificial Intelligence*, pages 417–426, Montreal, Canada, 2000.
2. M. W. Berry, M. Browne, A. N. Langville, P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 15(1):155–173, 2007.
3. W. M. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37:573–595, 1995.
4. C. Boutsidis and E. Gallopoulos. Svd-based initialization: A head start on nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.
5. C. Carpineto and G. Romano. Using concept lattices for text retrieval and mining. In B. G. et al., editor, *Formal Concept Analysis*, pages 161–179. Springer-Verlag, 2005.
6. M. Chu, F. Diele, R. Plemmons, and S. Ragni. Optimality, computation, and interpretations of nonnegative matrix factorization. Technical report, Department of Mathematics, North Carolina State University, 2005.
7. S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
8. E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 668–673, 1999.
9. S. Golder and B. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.
10. P. Heymann, D. Ramage, and H. G. Molina. Social tag prediction. In *Proceeding of SIGIR*, 2008.

11. A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *Proceeding of the 3rd European Semantic Web Conference*, pages 411–426, 2006.
12. P. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 1994.
13. A. Hulth. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction.* PhD thesis, Stockholm University, 2004.
14. R. Jaschke, L. Marinho, A. Hotho, L. S. Thieme, and G. Stumme. Tag recommendations in folksonomies. In J. N. Kok, J. Koronacki, R. L. de Ma'ntaras, S. Matwin, D. Mladenic, and A. Skowron, editors, *The 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 506–514, 2007.
15. I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceeding of the ECML/PKDD Discovery Challenge*, 2008.
16. F. W. Lancaster. *Indexing and Abstracting in Theory and Practice.* Library Association Publishing, 1998.
17. D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.
18. D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing System*, pages 556–562, 2001.
19. C. J. Lin. On the convergence of multiplicative update algorithms for non-negative matrix factorization. *IEEE Transactions on Neural Networks*, 18:1589–1596, 2007.
20. C. J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19:2756–2779, 2007.
21. D. Lin and P. Pantel. Concept discovery from text. In *Proceedings of the International Conference on Computational Linguistics*, pages 577–583, 2002.
22. O. Medelyan and I. H. Witten. Domain-independent automatic keyphrase indexing with small training set. *Journal of the American Society for Information Science and Technology*, 59(7):1026–1040, 2008.
23. G. Mishne. Autotag: A collaborative approach to automated tag assignment for weblog posts. In *Proceeding of the 15th International Conference on World Wide Web*, pages 953–954, 2006.
24. P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.
25. C. Paice and W. Black. A three-pronged approach to the extraction of key terms and semantic roles. In *Proceedings of the International Conference on Recent Advances in NLP*, pages 357–363, Borovets, Bulgaria, 2003.
26. U. Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology*, 40:521–543, 2005.
27. M. L. Reinberger and P. Spyns. Unsupervised text learning for the learning of dogma-inspired ontologies. In P. Buitelaar, P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Method, Application and Evaluation*, pages 29–43. IOS Press, 2005.
28. S. C. Sood, S. H. Owsley, K. J. Hammond, and L. Birnbaum. Tagassist: Automatic tag suggestion for blog posts. In *Proceeding of the International Conference on Weblogs and Social Media (ICWSM)*, 2007.
29. M. Tatu, M. Srikanth, and T. D'Silva. Rsdc'08: Tag recommendation using bookmark content. In *Proceeding of the ECML/PKDD Discovery Challenge*, 2008.

30. P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.
31. Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. In *Proceeding of Collaborative Web Tagging Workshop*, 2006.

STaR: a Social Tag Recommender System

Cataldo Musto, Fedelucio Narducci, Marco de Gemmis,
Pasquale Lops, and Giovanni Semeraro

Department of Computer Science, University of Bari "Aldo Moro", Italy
{musto,narducci,degemmis,lops,semeraro}@di.uniba.it

Abstract. The continuous growth of collaborative platforms we are recently witnessing made possible the passage from an 'elitary' Web, written by few and read by many, towards the so-called Web 2.0, a more 'user-centric' vision, where users become active contributors in Web dynamics. In this context, collaborative tagging systems are rapidly emerging: in these platforms users can annotate resources they like with freely chosen keyword (called tags) in order to make retrieval of information and serendipitous browsing more and more easier. However, as tags are handled in a simply syntactical way, collaborative tagging systems suffer of typical Information Retrieval (IR) problems like polysemy and synonymy: so, in order to reduce the impact of these drawbacks and to aid at the same time the so-called tag convergence, systems that assist the user in the task of tagging are required. The goal of these systems (called tag recommenders) is to suggest a set of relevant keywords for the resources to be annotated by exploiting different approaches. In this paper we present a tag recommender developed for the ECML-PKDD 2009 Discovery Challenge. Our approach is based on two assumptions: firstly, if two or more resources share some common patterns (e.g. the same features in the textual description), we can exploit this information supposing that they could be annotated with similar tags. Furthermore, since each user has a typical manner to label resources, a tag recommender might exploit this information to weigh more the tags she already used to annotate similar resources.

Key words: Recommender Systems, Web 2.0, Collaborative Tagging Systems, Folksonomies

1 Introduction

The coming of Web 2.0 has changed the role of Internet users and the shape of services offered by the World Wide Web. Since web sites tend to be more interactive and user-centric than in the past, users are shifting from passive consumers of information to active producers. By using Web 2.0 applications, users are able to easily publish content such as photos, videos, political opinions, reviews, so they are identified as *Web prosumers*: *pro*ducers + con*sumers* of knowledge. One of the forms of user-generated content (UGC) that has drawn more attention from the research community is *tagging*, which is the act of annotating resources of interests with free keywords, called *tags*, in order to help users in organizing, browsing and searching resources through the building of a socially-constructed classification schema, called *folksonomy* [18]. In contrast to systems where information about resources is only provided by a small set of experts, collaborative tagging systems take into account the way individuals conceive the information contained in a resource [19]. Well-known example of platforms that embed tagging activity are Flickr[1] to share photos, YouTube[2] to share videos, Del.icio.us[3] to share bookmarks, Last.fm[4] to share music listening habits and Bibsonomy[5] to share bookmarks and lists of literature. Although these systems provide heterogeneous contents, they have a common core: once a user is logged in, she can post a new resource and choose some significant keywords to identify it. Besides, users can label resources previously posted from other users. This phenomenon represents a very important opportunity to categorize the resources on the web, otherwise hardly feasible. The act of tagging resources from different users is the social aspect of this activity; in this way tags create a connection among users and items. Users that label the same resource by using the same tags could have similar tastes and items labeled with the same tags could have common characteristics.

Many would argue that the power of tagging lies in the ability for people to freely determine the appropriate tags for a resource without having to rely on a predefined lexicon or hierarchy [11]. Indeed, folksonomies are fully free and reflect the user mind, but they suffer of the same problems of unchecked vocabulary. Golder et. al. [5] identified three major problems with current tagging systems: *polysemy*, *synonymy*, and *level variation*. Polysemy refers to situations where tags can have multiple meanings: for example a resource tagged with the term *turkey* could indicate a news taken from an online newspaper about politics or a recipe for Thanksgiving' Day. When multiple tags share a single meaning we refer to it as synonymy. In collaborative tagging systems we can have simple morphological variations (for example we can find 'blog', 'blogs', 'web log', to identify a common blog) but also semantic similarity (like resources tagged with 'arts' versus 'cultural heritage'). The third problem, called level variations, refers to the phenomenon of tagging at different level of abstraction. Some people can annotate a web page containing a recipe for roast turkey with the tag 'roast-turkey' but also with a simple 'recipe'.

In order to avoid these problems, in the last years many tools have been developed to facilitate the user in the task of tagging and to aid the tag convergence [4]: these systems are know as tag recommenders. When a user posts a resource in a Web 2.0 platform, a tag recommender suggests some significant keywords to label the item following some criteria to filter out the noise from the complete tag space.

[1] http://www.flickr.com
[2] http://www.youtube.com
[3] http://delicious.com/
[4] http://www.last.fm/
[5] http://www.bibsonomy.org/

This paper presents STaR (Social Tag Recommender system), a tag recommender system developed for the ECML-PKDD 2009 Discovery Challenge. The idea behind our work is that folksonomies create connections among users and items, so we tried to point out two concepts:

– Resources with similar content could be annotated with similar tags;
– A tag recommender needs to take into account the previous tagging activity of users, by weighting more tags already used to annotate similar resources.

In this work we identify two main aspects in the tag recommendation task: firstly, each user has a typical manner to label resources (for example using personal tags such as 'beautiful', 'ugly', 'pleasant', etc. which are not connected to the content of the item, or simply tagging using general tags like 'politics', 'sport', etc.); next, similar resources usually share common tags: when a user posts a resource r on the platform, our system takes into account how she (if she is already stored in the system) and the entire community previously tagged resources similar to r in order to suggest relevant tags. Next, we develop this model and we tested it on a dataset extracted from BibSonomy.

The paper is organized as follows. Section 2 analyzes related work. The general problem of tag recommendation is introduced in Section 3. Section 4 explains the architecture of the system and how the recommendation approach is implemented. The experimental section carried out is described in Section 5.1, while conclusions and future works are drawn in last section.

2 Related Work

Previous work in the tag recommendation area can be broadly divided into three classes: *content-based*, *collaborative* and *graph-based* approaches.

In the content-based approach, a system exploits some textual source with Information Retrieval-related techniques [1] in order to extract relevant unigrams or bigrams from the text. Brooks et. al [3], for example, develop a tag recommender system that automatically suggests tags for a blog post extracting the top three terms exploiting TF/IDF scoring [14]. The system presented by Lee and Chun [8] recommends tags retrieved from the content of a blog using artificial neural networks. The network is trained based on statistical information about word frequencies and lexical information about word semantics extracted from WordNet. The collaborative approach for tag recommendation, instead, presents some analogies with collaborative filtering methods [2]. In the model proposed by Mishne and implemented in AutoTag [12], the system suggests tags based on the other tags associated with similar posts in a given collection. The recommendation process is performed in three steps: first, the tool finds similar posts and extracts their tags. All the tags are then merged, building a general folksonomy that is filtered and reranked. The top-ranked tags are suggested to the user, who selects the most appropriate ones to attach to the post. TagAssist [16] improves the AutoTags' approach performing a lossless compression over existing tag data. It finds similar blog posts and suggests a subset of the associated tag through a

Tag Suggestion Engine (TSE) which leverages previously tagged posts providing appropriate suggestions for new content. In [10] the tag recommendations task is performed through a user-based collaborative filtering approach. The method seems to produce good results when applied on the user-tag matrix, so they show that users with a similar tag vocabulary tend to tag alike. The problem of tag recommendation through graph-based approaches has been firstly addressed by Jäschke et al. in [7]. They compared some recommendation techniques including collaborative filtering, PageRank and FolkRank. The key idea behind FolkRank algorithm is that a resource which is tagged by important tags from important users becomes important itself. The same concept holds for tags and users, thus the approach uses a graph whose vertices mutually reinforce themselves by spreading their weights. The evaluation showed that FolkRank outperforms other approaches. Schmitz et al. [15] proposed association rule mining as a technique that might be useful in the tag recommendation process. In literature we can find also some hybrid methods integrating two or more approaches (mainly, content and collaborative ones) in order to reduce their typical drawbacks and point out their qualities. Heymann et. al [6] present a tag recommender that exploits at the same time social knowledge and textual sources. They suggest tags based on page text, anchor text, surrounding hosts, adding tags used by others users to label the URL. The effectiveness of this approach is also confirmed by the use of a large dataset crawled from del.icio.us for the experimental evaluation. A hybrid approach is also proposed by Lipczak in [9]. Firstly, the system extracts tags from the title of the resource. Afterwards, based on an analysis of co-occurrences, the set of candidate tags is expanded adding also tags that usually co-occur with terms in the title. Finally, tags are filtered and reranked exploiting the information stored in a so-called ”personomy”, the set of the tags previously used by the user.

Finally, in [17] the authors proposed a model based on both textual content and tags associated with the resource. They introduce the concept of *conflated tags* to indicate a set of related tag (like blog, blogs, ecc.) used to annotate a resource. Modeling in this way the existing tag space they are able to suggest various tags for a given bookmark exploiting both *user* and *document* models. They win the previous edition of the Tag Recommendation Challenge.

3 Description of the Task

STaR has been designed to participate at the ECML-PKDD 2009 Discovery Challenge[6]. In this section we will firstly introduce a formal model for recommendation in folksonomies, then we will analyze the specific requirements of the task proposed for the Challenge.

[6] http://www.kde.cs.uni-kassel.de/ws/dc09

3.1 Recommendation in Folksonomies

A collaborative tagging system is a platform composed of users, resources and tags that allows users to freely assign tags to resources. Following the definition introduced in [7], a folksonomy can be described as a triple (U, R, T) where:

- U is a set of *users*;
- R is a set of *resources*;
- T is a set of *tags*.

We can also define a tag assignment function TAS: $U \times R \rightarrow T$. The tag recommendation task for a given user $u \in U$ and a resource $r \in R$ can be finally described as the generation of a set of tags $\text{TAS}(u, r) \subseteq T$ according to some relevance model. In our approach these tags are generated from a ranked set of *candidate tags* from which the top n elements are suggested to the user.

3.2 Description of the ECML-PKDD 2009 Discovery Challenge

The 2009 edition of the Discovery Challenge consists of three recommendation tasks in the area of social bookmarking. We compete for the first task, content-based tag recommendation, whose goal is to exploit content-based recommendation approaches in order to provide a relevant set of tags to the user when she submits a new item (Bookmark or BibTeX entry) into Bibsonomy.

The organizers make available a training set with some examples of tag assignment: the dataset contains *263,004 bookmark* posts and *158,924 BibTeX* entries submitted by 3,617 different users. For each of the 235,328 different URLs and the 143,050 different BibTeX entries were also provided some textual metadata (such as the title of the resource, the description, the abstract and so on).

Each candidate recommender is evaluated by comparing the real tags (namely, the tags a user adopts to annotate an unseen resource) with the suggested ones. The accuracy is finally computed using classical IR metrics, such as Precision, Recall and F1-Measure (Section 5.1).

By analyzing the aforementioned requirements, we designed STaR thinking at a *prediction task* rather than a *recommendation* one. Consequently, we will try to emphasize the previous tagging activity of the user, also looking for connections and patterns among resources. All these decisions will be thoroughly analyzed in the next section describing the architecture of STaR.

4 STaR: a Social Tag Recommender System

STaR (Social Tag Recommender) is a content-based tag recommender system, developed at the University of Bari. The inceptive idea behind STaR is to improve the model implemented in systems like TagAssist [16] or AutoTag [12]. Although we agree with the idea that resources with similar content could be annotated with similar tags, in our opinion Mishne's approach presents two important drawbacks:

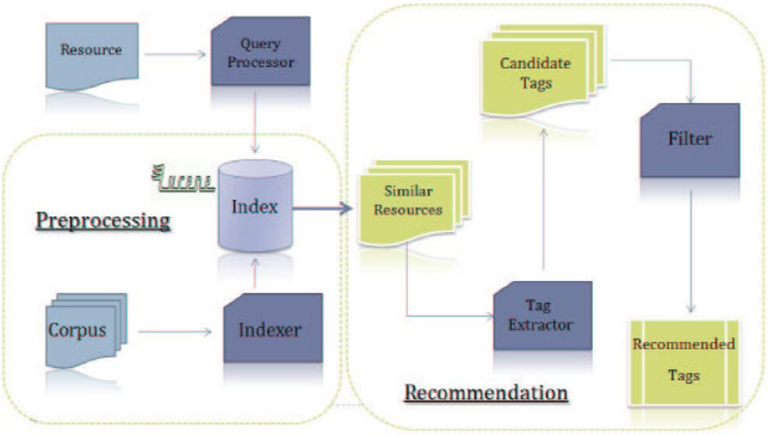


Fig. 1. Architecture of STaR

1. The tag reranking formula simply performs a sum of the occurrences of each tag among all the folksonomies, without considering the similarity with the resource to be tagged. In this way tags often used to annotate resources with a low similarity level could be ranked first.
2. The proposed model does not take into account the previous tagging activity performed by users. If two users bookmarked the same resource, they will receive the same suggestions since the folksonomies built from similar resources are the same.

We will try to overcome these drawbacks, by proposing an approach based on the analysis of similar resources capable also of weighting more the tags already selected by the user during her previous tagging activity. Figure 1 shows the general architecture of STaR. The recommendation process is performed in four steps, each of which is handled by a separate component.

4.1 Indexing of Resources

Given a collection of resources (*corpus*), a preprocessing step is performed by the *Indexer* module, which exploits Apache Lucene[7] to perform the indexing step. As regards *bookmarks* we indexed the title of the web page and the extended description provided by users. For the *BibteX* entries we indexed the title of the publication and the abstract. Let U be the set of users and N the cardinality of this set, the indexing procedure is repeated $N + 1$ times: we build an index for each user (*Personal Index*) storing the information on her previously tagged resources and an index for the whole community (*Social Index*) storing the information about all the resources previously tagged by the community.

[7] http://lucene.apache.org

Following the definitions presented in Section 3.1, given a user $u \in U$ we define $PersonalIndex(u)$ as:

$$PersonalIndex(u) = \{r \in R | \exists t \in T : \text{TAS}(u, r) = t\} \tag{1}$$

where TAS is the tag assignment function TAS: $U \times R \rightarrow T$ which assigns tags to a resource annotated by a given user. *SocialIndex* represents the union of all the user personal indexes:

$$SocialIndex = \bigcup_{i=1}^{N} PersonalIndex(u_i) \tag{2}$$

4.2 Retrieving of Similar Resources

At the end of the preprocessing step STaR is able to take into account users requests. Every user interacts with STaR by providing information about a resource to be tagged. In the Query Processing step the system acquires data about the user (her language, the tags she uses more, the number of tags she usually uses to annotate resources, etc.) before processing (through the elimination of not useful characters and punctuation) and submitting the query against the *SocialIndex* stored in Lucene. If the user is recognized by the system since it has previously tagged some other resources, the same query is submitted against her own *PersonalIndex*, as well. We used as query the title of the web page (for bookmarks) or the title of the publication (for BibTeX entries). In order to improve the performances of the Lucene Querying Engine we replaced the original Lucene Scoring function with an Okapi BM25 implementation[8]. BM25 is nowadays considered as one of the state-of-the art retrieval models by the IR community [13].

Let D be a corpus of documents, $d \in D$, BM25 returns the top-k resources with the highest similarity value given a resource r (tokenized as a set of terms $t_1 \ldots t_m$), and is defined as follows:

$$sim(r, d) = \sum_{i=1}^{m} \frac{n_{t_i}^r}{k_1((1-b) + b * \frac{length_r}{avgLength_r}) + n_{t_i}^r} * idf(t_i) \tag{3}$$

where $n_{t_i}^r$ represents the occurrences of the term t_i in the document d, $length_r$ is the length of the resource r and $avgLength_r$ is the average length of resources in the corpus. Finally, k_1 and b are two parameters typically set to 2.0 and 0.75 respectively, and $idf(t_i)$ represents the inverse document frequency of the term t_i defined as follows:

$$idf(t_i) = log \frac{N + df(t_i) + 0.5}{df(t_i) + 0.5} \tag{4}$$

[8] http://nlp.uned.es/ jperezi/Lucene-BM25/

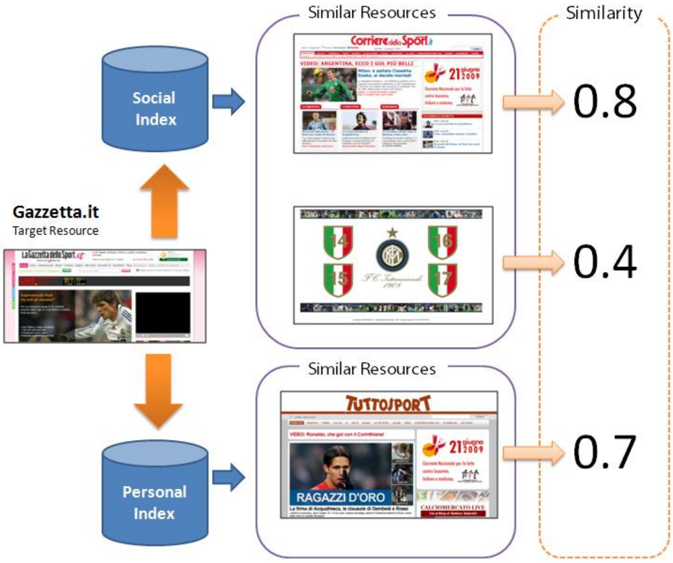


Fig. 2. Retrieving of Similar Resources

where N is the number of resources in the collection and $df(t_i)$ is the number of resources in which the term t_i occurs.

Given user $u \in U$ and a resource r, Lucene returns the resources whose similarity with r is greater or equal than a threshold β. To perform this task Lucene uses both the *PersonalIndex* of the user u and the *SocialIndex*. More formally:

$$PersonalRes(u,q) = \{r \in PersonalIndex(u) | sim(q,r) \geq \beta\} \tag{5}$$

$$SocialRes(q) = \{r \in SocialIndex | sim(q,r) \geq \beta\} \tag{6}$$

Figure 2 depicts an example of the retrieving step. In this case the target resource is represented by Gazzetta.it, one of the most famous Italian sport newspaper. Lucene queries the *SocialIndex* and returns as the most similar resources an online newspaper (Corrieredellosport.it) and the official web site of an Italian Football Club (Inter.it). The *PersonalIndex*, instead, returns another online newspaper (Tuttosport.com). The similarity score returned by Lucene has been normalized.

4.3 Extraction of Candidate Tags

In the next step the *Tag Extractor* gets the most similar resources returned by the Apache Lucene engine and produces the set of candidate tags to be suggested, by computing for each tag a score obtained by weighting the similarity score returned by Lucene with the normalized occurrence of the tag. If the *Tag Extractor* also gets the list of the most similar resources from the user *PersonalIndex*, it will produce two partial folksonomies that are merged, assigning a weight to each folksonomy in order to boost users' previously used tags.

Formally, for each query q (namely, the resource to be tagged), we can define a set of tags to recommend by building two sets: $candTags_p$ and $candTags_s$. These sets are defined as follows:

$$candTags_p(u,q) = \{t \in T | t = TAS(u,r) \wedge r \in PersonalRes(u,q)\} \tag{7}$$

$$candTags_s(q) = \{t \in T | t = TAS(u,r) \wedge r \in SocialRes(q) \wedge u \in U\} \tag{8}$$

In the same way we can compute the relevance of each tag with respect to the query q as:

$$rel_p(t,u,q) = \frac{\sum_{r \in PersonalRes(u,q)} n_r^t * sim(r,q)}{n^t} \tag{9}$$

$$rel_s(t,q) = \frac{\sum_{r \in SocialRes(q)} n_r^t * sim(r,q)}{n^t} \tag{10}$$

where n_r^t is the number of occurrences of the tag t in the annotation for resource r and n^t is the sum of the occurrences of tag t among all similar resources.

Finally, the set of *Candidate Tags* can be defined as:

$$candTags(u,q) = candTags_p(u,q) \cup candTags_s(q) \tag{11}$$

where for each tag t the global relevance can be defined as:

$$rel(t,q) = \alpha * rel_p(t,q) + (1-\alpha) * rel_s(t,q) \tag{12}$$

where α (PersonalTagWeight) and $(1-\alpha)$ (SocialTagWeight) are the weights of the personal and social tags respectively.

Figure 3 depicts the procedure performed by the *Tag Extractor*: in this case we have a set of 4 Social Tags (Newspaper, Online, Football and Inter) and 3 Personal Tags (Sport, Newspaper and Tuttosport). These sets are then merged, building the set of *Candidate Tags*. This set contains 6 tags since the tag *newspaper* appears both in social and personal tags. The system associates a score to each tag that indicates its effectiveness for the target resource. Besides, the scores for the Candidate Tags are weighted again according to SocialTagWeight (α) and PersonalTagWeight ($1-\alpha$) values (in the example, 0.3 and 0.7 respectively), in order to boost the tags already used by the user in the final tag rank. Indeed, we can point out that the social tag 'football' gets the same score of the personal tag 'tuttosport', although its original weight was twice.

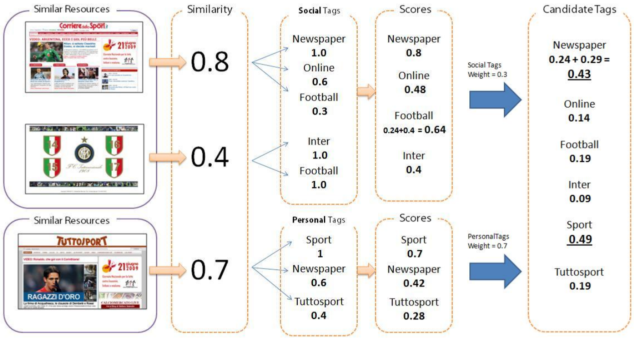


Fig. 3. Description of the process performed by the Tag Extractor

4.4 Tag Recommendation

The *Tag Extractor* produces the set of the *Candidate Tags*, a ranked set of tags with their relevance scores. This set is exploited by the *Filter*, a component which performs the last step of the recommendation task, that is removing those tags not matching specific conditions: we fix a threshold for the relevance score between 0.20 to 0.25 and we return at most 5 tags. These parameters are strictly dependent from the training data.

Formally, given a user $u \in U$, a query q and a threshold value γ, the goal of the filtering component is to build $recommendation(u, q)$ defined as follows:

$$recommendation(u, q) = \{t \in candTags(u, q) | rel(t, q) > \gamma\} \tag{13}$$

In the example in Figure 3, setting a threshold $\gamma = 0.20$, the system would suggest the tags *sport* and *newspaper*.

5 Experimental Evaluations

5.1 Experimental Session

In this experiment we measure the performance of STaR in the Task 1 of the ECML-PKDD 2009 Discovery Challenge. This experimental evaluation was carried out according to the instructions provided from the organizers of the Challenge 2009. The test set was released 48 hours before the end of the competition. Every participant uploaded a file containing the tag predictions, and for each post only five tags were considered. F1-Measure was used to evaluate the accuracy of recommendations, thus for each post Precision and Recall were computed by comparing the recommended tags with the true tags assigned by the users. The case of tags was ignored and all characters which are neither numbers nor letters were removed. Results are presented in Table 1.

Table 1. Results of the ECML-PKDD 2009 Discovery Challenge

#Tag	Precision	Recall	F1
1	19.51	6.89	10.19
2	16.34	10.10	12.53
3	14.55	12.16	13.25
4	13.56	13.53	13.55
5	**13.56**	**13.53**	**13.55**

STaR finished the ECML-PKDD Discovery Challenge 2009 with an overall F-measure of 13.55. As showed in the table above, exploiting only the first recommended tag the system reaches almost 20% in precision. The value of the recall increases with the number of recommended tags reaching the 13.5% in the fourth and fifth tag. In the future we will perform a more in-depth study in order to compare the predictive accuracy of STaR with different configurations of parameters.

6 Conclusions and Future Work

In this paper we presented STaR, a tag recommender designed and implemented to participate to the ECML-PKDD 2009 Discovery Challenge. The idea behind our work was to discover similarity among resources in order to exploit communities and user tagging behavior. In this way our recommender system was able to suggest tags for users and items still not stored in the training set. The experimental sessions showed that users tend to reuse their own tags to annotate similar resources, so this kind of recommendation model could benefit from the use of the user personal tags before extracting the social tags of the community (we called this approach user-based).

In the future we will implement a methodology to suggest tags when the set of similar items returned by Lucene is empty. The system should be able to extract significant keywords from the textual content associated to a resource (title, description, etc.) that has not similar items, maybe exploiting structured data or domain ontologies. Another issue to investigate is the application of our methodology in different domains such as multimedia environment. In this field discovering similarity among items just on the ground of textual content could be not sufficient. Finally, textual content suffers from syntactic problems like polysemy (a keyword with two or more meanings) and synonymy (two or more keywords with the same meaning). These problems hurt the performance of the recommender. We will try to establish if a semantic document indexing could improve the performance of the recommender.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison-Wesley, 1999.

2. D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceeding of the 15th International Conference on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.
3. C. H. Brooks and N. Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM Press.
4. C. Cattuto, C. Schmitz, A. Baldassarri, V. D. P. Servedio, V. Loreto, A. Hotho, M. Grahl, and G. Stumme. Network properties of folksonomies. *AI Communications*, 20(4):245–262, December 2007.
5. S. Golder and B. A. Huberman. The Structure of Collaborative Tagging Systems. *Journal of Information Science*, 32(2):198–208, 2006.
6. P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, New York, NY, USA, 2008. ACM.
7. R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In Alexander Hinneburg, editor, *Workshop Proceedings of Lernen - Wissensentdeckung - Adaptivit?t (LWA 2007)*, pages 13–20, September 2007.
8. Sigma On Kee Lee and Andy Hon Wai Chun. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ann semantic structures. In *ACOS'07: Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science*, pages 88–93, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).
9. M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 84–95, 2008.
10. Leandro B. Marinho and Lars Schmidt-Thieme. Collaborative tag recommendations. pages 533–540. 2008.
11. Adam Mathes. Folksonomies - cooperative classification and communication through shared metadata. http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html, December 2004.
12. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 953–954, New York, NY, USA, 2006. ACM.
13. Stephen E. Robertson, Steve Walker, Micheline H. Beaulieu, Aarron Gull, and Marianna Lau. Okapi at trec. In *Text REtrieval Conference*, pages 21–30, 1992.
14. G. Salton. *Automatic Text Processing.* Addison-Wesley, 1989.
15. Christoph Schmitz, Andreas Hotho, Robert Jschke, and Gerd Stumme. Mining association rules in folksonomies. In V. Batagelj, H.-H. Bock, A. Ferligoj, and A. ?iberna, editors, *Data Science and Classification (Proc. IFCS 2006 Conference)*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270, Berlin/Heidelberg, July 2006. Springer. Ljubljana.
16. Sanjay Sood, Sara Owsley, Kristian Hammond, and Larry Birnbaum. TagAssist: Automatic Tag Suggestion for Blog Posts. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.
17. M. Tatu, M. Srikanth, and T. D'Silva. Rsdc'08: Tag recommendations using bookmark content. In *Proceedings of ECML PKDD Discovery Challenge (RSDC08)*, pages 96–107, 2008.

18. Thomas Vander Wal. Folksonomy coinage and definition. Website, Februar 2007. `http://vanderwal.net/folksonomy.html`.
19. Harris Wu, Mohammad Zubair, and Kurt Maly. Harvesting social knowledge from folksonomies. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 111–114, New York, NY, USA, 2006. ACM Press.

Combining Tag Recommendations Based on User History

Ilari T. Nieminen

Helsinki University of Technology
ilari.nieminen@tkk.fi

Abstract. This paper describes our attempt at Task 2 of ECML PKDD Discovery Challenge 2009. The task was to predict which tags a given user would use on a given resource using methods that only utilize the graph structure of the training dataset, which was a snapshot of BibSonomy. The approach combines simple recommendation methods by weighting recommendations based on the tagging history of the user.

1 Introduction

Collaborative tagging systems or folksonomies have steadily gained popularity in the recent years. Users are free to choose the tags they want to use, and while this may be a main reason behind the popularity of these systems, it is also one of the biggest problems these systems face. As users come up with new tags they forget the tags they used to use, making it difficult to find the previously tagged content. Tag recommendation can help both in search and in keeping the users' tagging practices consistent. Tag recommendation can be defined as the problem of finding suitable tags or labels to a given resource for a given user.

Tag recommendation can be an important element in a folksonomy as it can help users employ the tags consistently as well as help users to use same tags for similar resources. This can improve searching within the users' own resources as well as the folksonomy.

We present a method for tag recommendation that combines several baseline methods and collaborative filtering. Combining the results makes use of the past performance of the recommenders.

2 Tag Recommendation

2.1 Collaborative Filtering for Folksonomies

Collaborative filtering (CF), a popular method used in recommender systems can be adapted for tag recommendation. The description here is based on [1].

Folksonomy can be understood as a tuple $\mathcal{F} = (U, R, T, Y)$, where U is the set of users, T is the set of tags and R is the set of resources (bookmarks and BibTeX entries in the case of BibSonomy [2]) and $Y \subseteq U \times R \times T$ is the tag assignment relation.

Projections $\pi_{UR}Y \in 0, 1^{|U|\times|R|}$, $(\pi_{UR}Y)_{u,r} := 1$ iff $\exists t \in T$ s.t. $(u, r, t) \in Y$ and $\pi_{UT}Y \in 0, 1^{|U|\times|T|}$, $(\pi_{UT}Y)_{u,t} := 1$ iff $\exists r \in R$ s.t. $(u, r, t) \in Y$ let us define the "tag neighbourhood" and "resource neighbourhood" of the users. The set of k nearest neighbours for a user u using the neighbourhood matrix X is

$$N_u^k := \operatorname*{argmax}^k_{u \in U} \operatorname{sim}(x_u, x_v) \tag{1}$$

where sim is the cosine similarity

$$\operatorname{sim}(x, y) := \frac{x \cdot y}{||x|| \, ||y||} \tag{2}$$

The set of recommendations for a given user-resource pair (u, r) is

$$T'(u, r) := \operatorname*{argmax}^n_{t \in T} \sum_{v \in N_u^k} \operatorname{sim}(x_u, x_v)\delta(v, r, t) \tag{3}$$

where $\delta(v, r, t) := \text{iff}(v, r, t) \in Y$.

2.2 Baseline Methods

The following are a collection of simple recommendation methods, which do not produce very good recommendations and have few redeeming qualities except that they are computationally inexpensive.

Popular tags for a resource. If the users of the folksonomy are homogenous, this method can be expected to perform almost as well as CF methods. However, if the users have very different tagging habits or if people use different tags from different languages, performance for the minorities can be expected to suffer.

Popular tags for a user. Some users use relatively few but obscure tags, which means that the popular tags for resource -recommender will not work. Collaborative recommendations also will not work well, as the user will probably have very few applicable "tag neighbours" and the "resource neighbours" will most likely not use the same tags. For example, user 483 used the tag "allgemein" a total of 2237 times in the 9003 posts. In other words, given a post by this user at random, there is almost a 25% chance it is tagged "allgemein".

Globally popular tags. Recommending the most used tags is perhaps the simplest possible method.

We used several variants of the aforementioned recommenders. These and the method used to combine the recommendations are described in chapter 4.1.

3 Data Description and Preprocessing

The provided training data contains three files: *bibtex,* bookmark and *tas. The* bibtex and *bookmark files describe the content of the links and BibTeX entries, respectively. The* tas file contains the tag assignments. Also provided was the post-core at level 2 [3], which contained a reduced set, which contained only those users, resources and tags that appear at least in two posts. The test set for this task was known to have the users, resources and tags from this set.

We processed bookmarks and BibTeX entries identically. The only information extracted from the "bookmark" and "bibtex" tables were the hash values which identified the resources. We used the `url_hash` and `simhash1` columns and did not attempt to combine duplicate resources. The `url_hash` considers two resources different if there are any differences in the url, such as a trailing slash.

To retain a slightly better neighbourhoods for the collaborative filtering approach we used full training set to calculate the neighbourhoods, but removed the tags that could not appear in the results. The difference between this and the post-core at level 2 was that this left several partial posts to the training data.

No effort was made to separate functional tags (such as "myown" and "toread") from descriptive tags, which are considerably more interesting in tag recommendation.

Some of the most used tags in BibSonomy are used by a small minority, such as "juergen" (3101 posts, 2 users). In total, in the subset of tags that are contained in the post-core 2 there are 273 tags that have been used at least 100 times by at most 5 people. A measure for the popularity of the tag, which takes into account the number of users of a tag can be defined as

$$\text{popularity}(t) = \log(N_t) * \log(N_t^*), \tag{4}$$

where N_t is the number of times the tag t has been used and N_t^* is number of users for the tag t.

This measure can be used to improve tag recommendation methods which would not otherwise give weights to different tags.

Table 1. Tags ordered by number of uses and "popularity"

Number of uses	Popularity
bookmarks	software
zzztosort	web
video	web20
software	video
programming	blog
web20	bookmarks
books	programming
media	internet
tools	tools
web	social

As can be seen from Table 1, sorting the tags by their "popularity" removes the unlikely tag "zzztosort" while preserving a sensible selection of popular tags.

4 Results

4.1 Combining Recommendations

The baseline methods can yield good results on certain users, but they are generally worse than the alternatives. However, combining the baseline results with results from collaborative filtering or other methods can be used to improve the general results. The problem of combining results is in evaluating the trustworthiness of the recommender results.

In tag recommendation, there are multiple "items" that are recommended, and besides the similarity between the user and the neighbours of the user there are few evident factors that could be used to weight the tags when combining different methods. In our method, we used the training data to predict the recent posts of the users (1-100 posts, but at most 20% of the user's all posts)

In our approach, we took the arbitrary set of methods shown in Table 2 and assigned weights to different tags by calculating the weighted sum over all recommenders using the per-user per-post weighted sum

$$w_t := \sum_p [t \in T'] * 0.9^k f_p \tag{5}$$

where f_p is the F-measure of the method $p \in 1, .., 7$ on the validation set, and k is the position of the tag in the recommendation. This reduces the weight of the tag slightly so that the methods with smaller F-measure have a better possibility of getting a likely tag in the final results. The final recommendation are the five $t \in T$ with the highest w_t.

Table 2. Recommendation methods

Method
Collaborative filtering (UR neighbourhood)
Collaborative filtering (UT neighbourhood)
Most frequent tags by resource
Most frequent tags by resource (popularity > 3)
Most frequent user tags
Most frequent user tags (popularity > 3)
Most popular global tags

Prior to the competition, we performed a test with the training data. The posts were divided into three sets based on the post date. The first 80% was selected to work as a training set, the following 10% as the validation set and the last 10% were used for testing. The method weights were computed from the validation set. The resulting weights were tested on the test set, showing a modest 5% improvement in the F-measure over the best baseline method in the test.

4.2 Experiment on the Competition Set

The weights for the methods were assigned to the users in the competition set by generating recommendations for recent posts with all the methods listed in the previous section. The amount of posts was chosen was up to 100 posts, but at most 20% of the user's all posts. After this, the F-measure for each method was used to generate a mixing profile for each user. Then the recommendations were made for the competition set and these were combined using the equation 5. The results are summarized in Table 3.

Table 3. Results on the competition set

Method	F-measure with 5 tags
CF-UR	0.2084
CF-UT	0.2317
resource tags	0.3067
resource tags (popularity > 3)	0.2940
user tags	0.0935
user tags (popularity > 3)	0.0050
popular tags	0.0354
combined	0.2952

One of the baselines (resource tags) outperforms the combined result slightly on the competition set. Some of the recommendations, such as “resource tags”, can contain very unlikely tags when the resource itself is tagged only a few times and contains unpopular tags; this was not taken into account when combining the recommendations. A possible solution for this problem is to not recommend unlikely (unpopular) tags if the user hasn't used them in the past.

5 Conclusion

In these experiments, the weights of the recommenders are based on their past performance, but it is likely that there are several features that can be used to estimate these weights from statistical features of the user, such as the average “popularity” of the user's tags and the number of distinct tags. We would like to study these numbers for correlations. Recommendations by other methods, such as FolkRank [1] could be added to improve the performance on the dense parts of the data.

The obtained results were less than stellar; in retrospect, more attention should have been paid to the combining of the results and especially the fact that the results of the recommendations were far from independent. Some method for filtering the results should have been applied, perhaps by modifying the weights for the individual tags by using the information whether the target user has used a certain tag before and how popular the tag is. Simple methods should not be completely neglected, as they can provide useful results for users who do not conform to the tagging practices of the mainline users of the folksonomy.

6 Discussion

F-measure works as a performance measure for tag recommendation to a certain extent, but the utility of tag recommendation methods for usability and search within a folksonomy should be confirmed with user tests. Combining different tag recommendation results with different weights at different times may cause the recommendation to feel inconsistent.

Searching within a folksonomy is sometimes unnecessarily difficult. A part of the problem is that users tend to use only a few tags per post. One improvement for these tagging systems would be to ask for applicability of a set of tags that are similar to the ones user has already chosen. It might make sense to distinguish between the problems of tag prediction, that is, predicting the tags user will choose, and tag recommendation, the problem of finding descriptive tags for a resource.

7 Acknowledgements

The author acknowledges Heikki Kallasjoki's technical assistance and Mari-Sanna Paukkeri's comments. This work was supported by the Academy of Finland through the Adaptive Informatics Research Centre that is a part of the Finnish Centre of Excellence Programme.

References

1. Jäschke, R., Marinho, L.B., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In Kok, J.N., Koronacki, J., de Mántaras, R.L., Matwin, S., Mladenic, D., Skowron, A., eds.: PKDD. Volume 4702 of Lecture Notes in Computer Science., Springer (2007) 506–514
2. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: BibSonomy: A social bookmark and publication sharing system. In: Proc. Workshop on Conceptual Structure Tool Interoperability at the Int. Conf. on Conceptual Structures. (2006) 87–102
3. Batagelj, V., Zaversnik, M.: Generalized cores (2002) cs.DS/0202039, http://arxiv.org/abs/cs/0202039.

Factor Models for Tag Recommendation in BibSonomy

Steffen Rendle and Lars Schmidt-Thieme

Machine Learning Lab, University of Hildesheim, Germany
{srendle,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. This paper describes our approach to the ECML/PKDD Discovery Challenge 2009. Our approach is a pure statistical model taking no content information into account. It tries to find latent interactions between users, items and tags by factorizing the observed tagging data. The factorization model is learned by the Bayesian Personal Ranking method (BPR) which is inspired by a Bayesian analysis of personalized ranking with missing data. To prevent overfitting, we ensemble the models over several iterations and hyperparameters. Finally, we enhance the top-n lists by estimating how many tags to recommend.

1 Introduction

In this paper, we describe our approach to task 2 of the ECML/PKDD Discovery Challenge 2009. The setting of the challenge is personalized tag recommendation [1]. An example is a social bookmark site where a user wants to tag one of his bookmark and the tag recommender suggest the user a personalized list of tags he might want to use for this item.

Our approach to this problem is a pure statistical model using no content information. It relies on a factor model related to [2] where the model parameters are optimized for the maximum likelihood estimator for personalized pairwise ranking [3]. Furthermore, we use a smoothing method for reducing the variance in the factor models. Finally, we provide a method for estimating how many tags should be recommended for a given post. This method is model independent and can be applied to any tag recommender.

2 Terminology and Formalization

We follow the terminology of [2]: U is the set of all users, I the set of all items/ resources and T the set of all tags. The tagging information of the past is represented as the ternary relation $S \subseteq U \times I \times T$. A tagging triple $(u, i, t) \in S$ means that user u has tagged an item i with the tag t. The *posts* P_S denotes the set of all distinct user/ item combinations in S:

$$P_S := \{(u, i) | \exists t \in T : (u, i, t) \in S\}$$

Our models calculate an estimator $\hat{Y}$ for S. Given such a predictor $\hat{Y}$ the list Top of the N highest scoring items for a given user u and an item i can be calculated by:

$$\mathrm{Top}(u, i, N) := \underset{t \in T}{\overset{N}{\operatorname{argmax}}}\, \hat{y}_{u,i,t} \tag{1}$$

where the superscript N denotes the number of tags to return. Besides $\hat{y}_{u,i,t}$ we also use the notation of a rank $\hat{r}_{u,i,t}$ which is the position of t in a post (u, i) after sorting all tags by $\hat{y}_{u,i,t}$:

$$\hat{r}_{u,i,t} := |\{t' : \hat{y}_{u,i,t'} > \hat{y}_{u,i,t}\}|$$

3 Factor Model

Our factorization model (FM) captures the interactions between users and tags as well as between items and tags. The model equation is given by:

$$\hat{y}_{u,i,t} = \sum_f \hat{u}_{u,f} \cdot \hat{t}^U_{t,f} + \sum_f \hat{i}_{i,f} \cdot \hat{t}^I_{t,f} \tag{2}$$

Where $\hat{U}$, $\hat{I}$, $\hat{T}^U$ and $\hat{T}^I$ are feature matrices capturing the latent interactions. They have the following types:

$$\hat{U} \in \mathbb{R}^{|U| \times k}, \quad \hat{I} \in \mathbb{R}^{|I| \times k},$$
$$\hat{T}^U \in \mathbb{R}^{|T| \times k}, \quad \hat{T}^I \in \mathbb{R}^{|T| \times k}$$

Note that this model differs from the factorization model in [2] where the model equation is the Tucker Decomposition.

3.1 Optimization Criterion

Our optimization criterion is an adaption of the BPR criterion (Bayesian Personalized Ranking) [3]. The criterion presented in [3] is derived for the task of item recommendation. Adapted to tag recommendation, the optimization function for our factor model is:

$$\begin{aligned}\text{BPR-O}\textsc{pt} := \sum_{(u,i) \in P_S} \sum_{t^+ \in T^+_{u,i}} \sum_{t^- \in T^-_{u,i}} &\ln \sigma(\hat{y}_{u,i,t^+} - \hat{y}_{u,i,t^-}) \\ &- \lambda(||\hat{U}||^2 + ||\hat{I}||^2 + ||\hat{T}^U||^2 + ||\hat{T}^I||^2)\end{aligned} \tag{3}$$

That means BPR-Opt tries to optimize the pairwise classification accuracy within observed posts. Note that it differs from [2] by optimizing for pairwise classification (log-sigmoid) instead of AUC (sigmoid).

3.2 Learning

The model is learned by the LearnBPR algorithm [3] which is a stochastic gradient descent algorithm where cases are sampled by bootstrapping. In the following, we show how we apply this generic algorithm to the task of optimzing our model paramaters for the task of tag recommendation. The gradients of our model equation (2) with respect to the model parameters $\Theta = \{\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I\}$ are:

$$\frac{\partial \text{BPR-Opt}}{\partial \Theta}$$

$$= \sum_{(u,i)\in P_S} \sum_{t^+ \in T^+_{u,i}} \sum_{t^- \in T^-_{u,i}} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{y}_{u,i,t^+} - \hat{y}_{u,i,t^-}) - \lambda \frac{\partial}{\partial \Theta} ||\Theta||^2$$

$$\propto \sum_{(u,i)\in P_S} \sum_{t^+ \in T^+_{u,i}} \sum_{t^- \in T^-_{u,i}} \frac{-e^{-(\hat{y}_{u,i,t^+} - \hat{y}_{u,i,t^-})}}{1 + e^{-(\hat{y}_{u,i,t^+} - \hat{y}_{u,i,t^-})}} \cdot \frac{\partial}{\partial \Theta}(\hat{y}_{u,i,t^+} - \hat{y}_{u,i,t^-}) - \lambda \Theta$$

That means, we only have to compute the derivations of our model equation $\hat{y}_{u,i,t}$ with respect to each model parameter from $\Theta = \{\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I\}$:

$$\frac{\partial}{\partial \hat{u}_{u,f}} \hat{y}_{u,i,t} = \hat{t}^U_{t,f}$$

$$\frac{\partial}{\partial \hat{i}_{u,f}} \hat{y}_{u,i,t} = \hat{t}^I_{t,f}$$

$$\frac{\partial}{\partial \hat{t}^U_{t,f}} \hat{y}_{u,i,t} = \hat{u}_{u,f}$$

$$\frac{\partial}{\partial \hat{t}^I_{t,f}} \hat{y}_{u,i,t} = \hat{i}_{i,f}$$

These derivations are used in the stochastic gradient descent algorithm shown in figure 1.

The method presented so far has the following hyperparameters:

- $\alpha \in \mathbb{R}^+$ learning rate
- $\lambda \in \mathbb{R}^+_0$ regularization parameter
- $\mu \in \mathbb{R}$ mean value for initialization of model parameters
- $\sigma^2 \in \mathbb{R}^+_0$ standard deviation for initialization of model parameters
- $k \in \mathbb{N}^+$ feature dimensionality of factorization

Reasonable values for all parameters can be searched on a holdout set. The learning rate and the initialization parameters are only important for the learning algorithm but are not part of the optimization criterion or model equation. Usually, the values found for α, μ, σ^2 on the holdout generalize well.

In contrast to this, the regularization and dimensionality are more important for the prediction quality. In general, when the regularization is chosen properly, the higher the dimensionality the better. In our submitted result, we use an ensemble of models with different regularization and dimensionality.

```
procedure LearnBPR($P_S, \hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$)
    draw $\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$ from $N(\mu, \sigma^2)$
    repeat
        draw $(u, i, t^+, t^-)$ uniformly from $P_S \times T^+_{u,i} \times T^-_{u,i}$
        $d \leftarrow \hat{y}_{u,i,t^+} - \hat{y}_{u,i,t^-}$
        for $f \in 1, \ldots, k$ do
            $\hat{u}_{u,f} \leftarrow \hat{u}_{u,f} + \alpha\left(\frac{e^{-d}}{1+e^{-d}} \cdot (\hat{t}^U_{t^+,f} - \hat{t}^U_{t^-,f}) + \lambda \cdot \hat{u}_{u,f}\right)$
            $\hat{i}_{i,f} \leftarrow \hat{i}_{i,f} + \alpha\left(\frac{e^{-d}}{1+e^{-d}} \cdot (\hat{t}^I_{t^+,f} - \hat{t}^I_{t^-,f}) + \lambda \cdot \hat{i}_{i,f}\right)$
            $\hat{t}^U_{t^+,f} \leftarrow \hat{t}^U_{t^+,f} + \alpha\left(\frac{e^{-d}}{1+e^{-d}} \cdot \hat{u}_{u,f} + \lambda \cdot \hat{t}^U_{t^+,f}\right)$
            $\hat{t}^U_{t^-,f} \leftarrow \hat{t}^U_{t^-,f} + \alpha\left(\frac{e^{-d}}{1+e^{-d}} \cdot -\hat{u}_{u,f} + \lambda \cdot \hat{t}^U_{t^-,f}\right)$
            $\hat{t}^I_{t^+,f} \leftarrow \hat{t}^I_{t^+,f} + \alpha\left(\frac{e^{-d}}{1+e^{-d}} \cdot \hat{i}_{i,f} + \lambda \cdot \hat{t}^I_{t^+,f}\right)$
            $\hat{t}^I_{t^-,f} \leftarrow \hat{t}^I_{t^-,f} + \alpha\left(\frac{e^{-d}}{1+e^{-d}} \cdot -\hat{i}_{i,f} + \lambda \cdot \hat{t}^I_{t^-,f}\right)$
        end for
    until convergence
    return $\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$
end procedure
```

Fig. 1. Optimizing our factor model for equation (3) with bootstrapping based stochastic gradient descent. With learning rate α and regularization λ.

3.3 Ensembling Factor Models

Ensembling factor models with different regularization and dimensionality is supposed to remove variance from the ranking estimates. There are basically two simple approaches of ensembling predictions $\hat{y}^l_{u,i,t}$ of l models:

1. Ensemble of the **value estimates** $\hat{y}^l_{u,i,t}$:

$$\hat{y}^{\text{ev}}_{u,i,t} := \sum_l w_l \cdot \hat{y}^l_{u,i,t} \tag{4}$$

2. Ensemble of the **rank estimates** $\hat{r}^l_{u,i,t}$:

$$\hat{y}^{\text{er}}_{u,i,t} := \sum_l w_l \cdot (|T| - \hat{r}^l_{u,i,t}) \tag{5}$$

   That means tags with a high rank (low $\hat{r}$) will get a high score $\hat{y}$.

Where w_l is the weighting parameter for each model.

Whereas ensembling value estimates is effective for models with predictions on the same scale, rank estimates are favorable in cases where the $\hat{y}$ values of the different models have no direct relationship.

Ensembling Different Factor Models For our factor models the scales of $\hat{y}$ depend both on the dimensionality and the regularization parameter. Thus we use the rank estimates for ensembling factor models with different dimensionality and regularization. In our approach we use a dimensionality of $k \in \{64, 128, 256\}$ and regularization of $\lambda \in \{10^{-4}, 5 \cdot 10^{-5}\}$. As the prediction quality of all of our factor models are comparable, we have chosen identical weights $w_l = 1$.

Ensembling Iterations Within each factor model we use a second ensembling strategy to remove variance. Besides the hyperparameters, another problem is the stopping criterion of the learning algorithm (see figure 1). We stop after a predefined number of iterations (2000) – we have chosen an iteration size of $10 \cdot |S|$ single draws. In our experiments the models usually converged already after about 500 iterations but in the following iterations the ranking alternates still a little bit. To remove the variance, we create many value estimates from different iterations and ensemble them. I.e. after the first 500 iterations we create each 50 iterations a value estimate for each tag in all test posts and ensemble these estimates with (4). Again there is no reason to favor an iteration over another, so we use identical weights $w_l = 1$. This gives the final estimates for each model. The models with different dimensionality and regularization are ensembled as described above.

4 Baseline Models

Besides our factorization model we also consider several baseline models and ensembles of these models. The models we pick as baselines are *most-popular by item* (mpi), *most-popular by user* (mpu), *item-based knn* (knni) and *user-based knn* (knnu).

The most-popular models are defined as follows:

$$\hat{y}^{\text{mpi}}_{u,i,t} = |\{u' \in U : (u', i, t)\}|$$
$$\hat{y}^{\text{mpu}}_{u,i,t} = |\{i' \in I : (u, i', t)\}|$$

The k-nearest-neighbour models (knn) are defined as follows:

$$\hat{y}^{\text{knni}}_{u,i,t} = \sum_{(u,i',t) \in S} \text{sim}_{i,i'}$$
$$\hat{y}^{\text{knnu}}_{u,i,t} = \sum_{(u',i,t) \in S} \text{sim}_{u,u'}$$

To measure $\text{sim}_{i,i'}$ and $\text{sim}_{u,u'}$ respectively, we first fold/ project the observed data tensor in a two dimensional matrix F^U and F^I:

$$f^I_{i,t} = |\{u : (u, i, t) \in S\}|$$
$$f^U_{u,t} = |\{i : (u, i, t) \in S\}|$$

After the folding we apply cosine similarity to compare two tag vectors:

$$\mathrm{sim}_{i,i'} = \frac{\sum_t f^I_{i,t} \cdot f^I_{i',t}}{\sqrt{\sum_t (f^I_{i,t})^2} \cdot \sqrt{\sum_t (f^I_{i,t})^2}}$$

$$\mathrm{sim}_{u,u'} = \frac{\sum_t f^U_{u,t} \cdot f^U_{u',t}}{\sqrt{\sum_t (f^U_{u,t})^2} \cdot \sqrt{\sum_t (f^U_{u,t})^2}}$$

We tried different weighted ensembles of the baseline models using the value estimate ensembling method. Even though these ensembles produce quite good results, in our experiments they did not outperform the factor models and furthermore adding baselines to the factor models did not result in a significant improvement of the factor models. Thus our final submission only consists of the factor models.

5 Adaptive List Length

In contrast to the usual evaluation scheme of tag recommendation, in this challenge the recommender was free to choose the length of the list of the recommendations in a range from a length of 0 to 5. The evaluation functions are:

$$\mathrm{Prec}(S_{test}) := \underset{(u,i) \in P_{S_{test}}}{\mathrm{avg}} \frac{|\,\mathrm{Top}(u,i,\min(5,\#_{u,i})) \cap \{t|(u,i,t) \in S_{test}\}|}{\min(5,\#_{u,i})}$$

$$\mathrm{Recall}(S_{test}) := \underset{(u,i) \in P_{S_{test}}}{\mathrm{avg}} \frac{|\,\mathrm{Top}(u,i,\min(5,\#_{u,i})) \cap \{t|(u,i,t) \in S_{test}\}|}{|\{t|(u,i,t) \in S_{test}\}|}$$

$$\mathrm{F1}(S_{test}) := \frac{2 \cdot \mathrm{Prec}(S_{test}) \cdot \mathrm{Recall}(S_{test})}{\mathrm{Prec}(S_{test}) + \mathrm{Recall}(S_{test})}$$

Where $\#_{u,i}$ is the number of tags the recommender estimates for a post.

There are three simple ways to estimate $\#_{u,i}$:

- **Global estimate**:

$$\#^G_{u,i} := \frac{|S|}{|P_S|}$$

- **User estimate**:

$$\#^U_{u,i} := \frac{|\{(i',t) : (u,i',t) \in S\}|}{|\{i' : (u,i',t) \in S\}|}$$

- **Item estimate**:

$$\#^I_{u,i} := \frac{|\{(u',t) : (u',i,t) \in S\}|}{|\{u' : (u',i,t) \in S\}|}$$

Based on these simple estimators, a combined post size can be produced by a linear combination:

$$\#^E_{u,i} := \beta_0 + \beta_G \#^G_{u,i} + \beta_U \#^U_{u,i} + \beta_I \#^I_{u,i}$$

In our approach we use $\#^E_{u,i}$ and optimize β on the holdout set for maximal F1. We found that choosing an adaptive length of the recommender list significantly improved the results over a fixed number.

6 Experimental Results

6.1 Sampling of Holdout Set

As the test of the challenge was released two days before the submission deadline, we tried to generate representative holdout-sets. We created two test sets, one following the leave-one-post-per-user-out protocol [1] and a second one by uniformly sampling posts with the constraint that the dataset should remain a 2-core after moving a post into the test set. These two sets were used as holdout sets for algorithm evaluation and hyperparameter selection. In the following, we report results for the second holdout set, because its characteristics (in terms of number of users, items and posts) are closer to the real test set.

6.2 Results

The results of the method presented so far can be found in table 2 and 3. As you can see, the single baseline models result in low quality but ensembles can achieve a good quality. In contrast to this, our proposed factor models generate better recommendations. The best possible ensemble (optimized on test!) of the baselines achieves a score of 0.330 on the challenge set whereas our factor ensemble (not optimized on test) results in 0.345.

	mpu	mpi	mp-ens	knni	knnu	knn-ens	knn+mp-ens
holdout	0.249	0.351	-/0.423	0.401	0.371	-/0.445	-/0.473
challenge	0.098	0.288	0.290/0.317	0.209	0.295	0.293/0.320	0.299/0.330

Fig. 2. F-Measure quality for the baselines methods. For the ensembles, we report two results: one for an ensemble with identical weights and one with optimal weights that have been optimized on the test! set. For sure this is an optimistic value that might not be found using the holdout split.

An interesting finding is that the results on the challenge test set largely differs from both of our holdout sets. But as all methods suffer, we assume that the tagging behavior in the challenge test set is indeed different from the one in the training set. Especially, the baseline most-popular-by-user dropped largely from 24.9% to 9.8% – this might indicate that personalization is difficult to achieve on

	single FM	FM-ens	FM-ens adaptive list length
holdout	0.495 ± 0.002	0.498	0.522
challenge	-	0.345	**0.356**

Fig. 3. F-Measure quality for the factorization methods. Single FM reports the average quality of each factorization model. FM-ens is the unweighted ensemble and finally we report the ensemble with the adaptive list length, i.e. predicting sometimes less than 5 tags.

the challenge test set using the provided training set. Non-personalized methods or content-based methods could benefit from the difference in both sets. Also methods that can handle temporal changes in the tagging behaviour might improve the scores.

7 Conclusion

In this paper, we have presented a factor model for the task of tag recommendation. The model tries to describe the individual tagging behavior by four low-dimensional matrices. The model parameters are optimized for the personalized ranking criterion BPR-Opt [3]. The length of the recommended lists is adapted both to the user and item. Our evaluation indicates that our approach outperforms ensembles of baseline models which are known to give high quality recommendations [1].

8 Acknowledgements

The authors gratefully acknowledge the partial co-funding of their work through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006. For your inquiries please contact info@mymediaproject.org.

References

1. Jaeschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in social bookmarking systems. AICOM (2008)
2. Rendle, S., Marinho, L.B., Nanopoulos, A., Thieme, L.S.: Learning optimal ranking with tensor factorization for tag recommendation. In: KDD '09: Proceeding of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2009)
3. Rendle, S., Freudenthaler, C., Gantner, Z., Thieme, L.S.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009). (2009)

Content-based and Graph-based Tag Suggestion

Xiance Si, Zhiyuan Liu, Peng Li, Qixia Jiang, Maosong Sun

State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Dept. of Computer Science&Technology, Tsinghua University, Beijing 100084, China
{adam.si, lzy.thu, pengli09, qixia.jiang}@gmail.com, sms@tsinghua.edu.cn
http://nlp.csai.tsinghua.edu.cn

Abstract. Social tagging is a popular and convenient way to organize information. Automatic tag suggestion can ease the user's tagging activity. In this paper, we exam both content-based and graph-based methods for tag suggestion using the BibSonomy dataset, and describe our methods for ECML/PKDD Discovery Challenge 2009 submissions . In content-based tag suggestion, we propose a fast yet accurate method named Feature-Driven Tagging. In graph-based tag suggestion, we apply DiffusionRank to solve the problem, and get a better result than current state-of-the-art methods in cross-validation.

1 Introduction

Social tagging, aka, folksonomy, is a popular way to organize resources like documents, bookmarks and photos. Resource, tag and user are three essential parts in a social tagging system, a user uses tags to describe resources. Tag suggestion system eases the process of social tagging. It can suggest tags to new resources based on previous tagged resources.

To promote related research, ECML/PKDD organizes a open contest of tag suggestion systems, named Discovery Challenge 2009 (DC09 in short). In this contest a snapshot of users, documents and tags in the online bookmarking system BibSonomy is provided. Each team trains their suggestion system on the snapshot, and test the performance on the same test dataset. There are 3 tasks in the contest. Task 1 focuses on suggesting tags by the content of the resources, i.e, content-based tag suggestion. Task 2 focuses on suggesting tags by the tripartite links between resources, tags and users, i.e., graph-based tag suggestion. Task 3 puts the suggestion system into real-life situation by integrating it with BibSonomy website, and see which system predicts the user's intention best.

In this paper, we describe our methods for the three tasks. For Task 1 and 3, we propose a fast tag suggestion method called Feature-Driven Tagging (FDT). FDT indexes tags by features, where feature can be word, resource ID, user ID or others. For each feature, FDT keeps a list of weighted tags, the higher the weight, the more likely the tag is suggested by the feature. For a new resource, each feature in it suggests a list of weighted tags, the suggestions are combined according to the importance of features to get the final suggestion. Compared to other methods, FDT provides suggestions faster, and the speed is only related with the number of features in the resource(number of words in the content).

For Task 2, we apply two existing methods, most popular tags and FolkRank, for graph-based suggestion. Furthermore, we propose to use a new graph-based ranking model, DiffusionRank, for tag suggestion. The method of "most popular tags" is the simplest collaborative-filtering based methods. It recommends the most popular tags of the resources used by other users. FolkRank is based on PageRank [1] on user-resource-tag tripartite graph, which was first proposed as a tag suggestion method in [2]. DiffusionRank was originally proposed for combating web spam [3], which has also been successfully used in social network analysis [4] and search query suggestion [5]. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow. Compared to PageRank, DiffusionRank provides more flexible mechanism to make the ranking scores related to initial values of the vertices, which is important for graph-based tag suggestion.

The paper is organized as follows. Section 2 formulates the problem of tag suggestion. Section 3 introduces our method for content-based tag suggestion. Section 4 introduces our method for graph-based tag suggestion. Section 5 describes the dataset, experiment settings and the result. Section 6 introduces related work on tag suggestion. Section 7 concludes the paper.

2 Problem Formulation

We adopt the model of social tagging proposed by Jaschke et al [2]. A social tagging data set is defined as a tuple $\mathbb{F} := (U, T, R, Y)$, where U is the set of users, T is the set of tags and R is the set of resources. Y is a ternary relation between U, T and R, $Y \subseteq U \times T \times R$. $(u, r, t) \in Y$ is called a tag assignment, which means user u assigned the tag t to resource r. A resource $r \in R$ can be described with a piece of text, like titles of a paper or user-edited description of a website. We denote the words in the text as $\{w_i\}$.

Resources, users and tags form a graph $G = (V, E)$, where $V = U \bigcup R \bigcup T$, and $E = \{\{u, t\}, \{u, r\}, \{r, t\} | (u, t, r) \in Y\}$. The goal of tag suggestion is to predict the set of tags $\{t\}$ for a given pair of user and resource (u, r).

In related literature, social tags are also called folksonomy, the pair of a resource and a user is also called a post.

3 Content-based Tag Suggestion

In this section, we propose a content-based tag suggestion method named Feature-Driven Tagging(FDT). Briefly speaking, FDT is a voting model, where each feature in the resource votes for their favorite tags, and the final scores of tags are averaged by the importance of the features. Figure 1 illustrates the tagging procedure of FDT, it consists of 3 steps: feature extraction, feature weighting and tag voting. For a resource with content, FDT first extracts features from the content. Features include but are not limited to words, resource ID and user ID. Then, FDT weights each feature by their importance in the resource, we explain different ways to compute the importance of features later in this section. In the voting step, each feature contributes a weighted list of tags, the higher the weight, the more likely we should suggest the tag. Weight of a tag from different features are combined by the importance of each feature, thus creates the final weighted list of tags. In the tagging process, all parameters are indexed by feature, we do not need to iterate over all tags (as in text categorization approaches) or resources (as in neighborhood-based approaches), so it is called Feature-Driven Tagging.

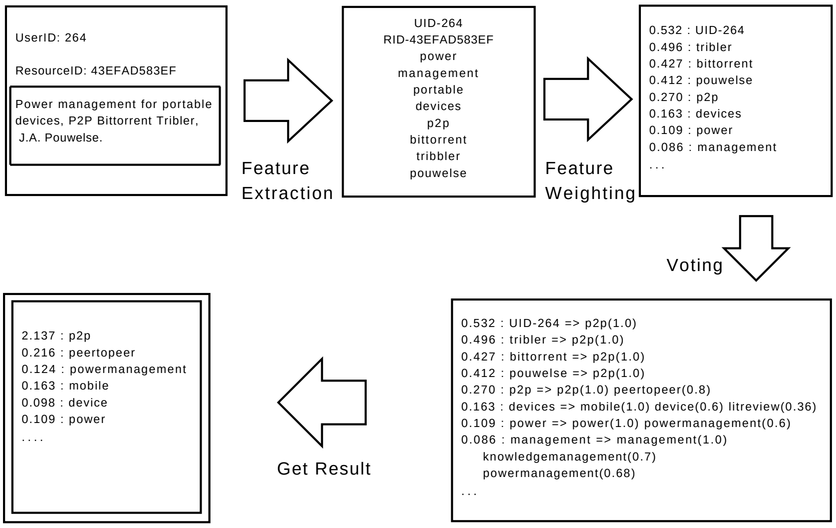

UserID: 264
ResourceID: 43EFAD583EF
Power management for portable devices, P2P Bittorrent Tribler, J.A. Pouwelse.
Feature Extraction
UID-264
RID-43EFAD583EF
power
management
portable
devices
p2p
bittorrent
tribbler
pouwelse
Feature Weighting
0.532 : UID-264
0.496 : tribler
0.427 : bittorrent
0.412 : pouwelse
0.270 : p2p
0.163 : devices
0.109 : power
0.086 : management
...
Voting
0.532 : UID-264 => p2p(1.0)
0.496 : tribler => p2p(1.0)
0.427 : bittorrent => p2p(1.0)
0.412 : pouwelse => p2p(1.0)
0.270 : p2p => p2p(1.0) peertopeer(0.8)
0.163 : devices => mobile(1.0) device(0.6) litreview(0.36)
0.109 : power => power(1.0) powermanagement(0.6)
0.086 : management => management(1.0) knowledgemanagement(0.7) powermanagement(0.68)
...
Get Result
2.137 : p2p
0.216 : peertopeer
0.124 : powermanagement
0.163 : mobile
0.098 : device
0.109 : power
....

Fig. 1. The procedure of Feature-Driven Tagging.

3.1 Feature Extraction

We extract features from different sources. Word features are extracted from textual content of resources, we use them to capture the relationship between words and tags. For bibtex, the textual content is *title* + *bibtexAbstract* + *journal* + *booktitle* + *annote* + *note* + *description*; For bookmark, it is *description* + *extended*. We also include *simhash*1 and the user ID of a resource as a feature. The same publication or website share the same *simhash*1, we use it to capture the tags assigned by other users. We use user ID as a feature so as to model a user's preferences of tagging.

3.2 Compute the Importance of Features

We use two methods to assess the importance of features in a resource. The first and most intuitive one is $TF \times IDF$. $TF \times IDF$ is widely used in information retrieval, text categorization and keyword extraction [6]. We use *log*-version of $TF \times IDF$, which computes as

$$TF \times IDF(f) = log(\frac{n(f)}{N} + 1) * log(\frac{|R|}{df(f)} + 1) \quad (1)$$

where $n(f)$ is the number of occurrences of f in this resource, and N is the total number of features occurred in this resource. $|R|$ is the total number of resources, $df(f)$ is the number of resources f has occurred in. The +1 is to avoid zero or negative weights.

The other method we used is $TF \times ITF$, ITF stands for Inverse Tag Frequency, it computes as follows,

$$TF \times ITF(f) = log(\frac{n(f)}{N} + 1) * log(\frac{|T|}{ntag(f)} + 1) \quad (2)$$

where $|T|$ is the total number of tags, and $ntag(f)$ is the number of tags f has co-occurred with. ITF implies that the more tags a feature co-occurs with, the less specific and important the feature is.

3.3 Feature-Tag Correlation

In FDT, each feature is associated with a weighted list of tags. We denote this as a matrix Θ, where $\theta_{i,j}$ is the weight of tag t_j to feature f_i, the size of Θ is $|F| \times |T|$, F is the set of all features. Although Θ is large, it is extremely sparse, so each feature only associates with a small number of tags.

We use three different methods to compute Θ offline, they are co-occurrence count(CC), Mutual Information (MI) and χ^2 statistics (χ^2). Co-occurrence count is computed by

$$CC(f,t) = n(f,t)/n(t) \quad (3)$$

where $n(f,t)$ is the number of co-occurrences of feature f and tag t, and $n(t)$ is the total number of occurrences of tag t. CC is a naive way to find the most important tags for a feature.

In MI, we model each feature or tag as a binary-valued probabilistic variable, the value of which means occur in a document(1) or not(0). Then, we can compute the Mutual Information between a feature and a tag by

$$MI(f,t) = \sum_{f' \in f, \bar{f}} \sum_{t' \in t, \bar{t}} p(f',t') log(\frac{p(f',t')}{p(f')p(t')}) \quad (4)$$

where $f' = f$ means feature f occurs in the resource, and $f' = \bar{f}$ means it doesn't occur, the same is for t'. MI computes the shared information between f and t, the higher it is, the more correlated f and t are.

χ^2 has been used for feature selection in text categorization [7], it also find the correlation between a feature and a category, here we use the tag as category. χ^2 is computed as follows,

$$\chi^2(f,t) = \frac{N(AD-BC)^2}{(A+C)(B+D)(A+B)(C+D)} \tag{5}$$

where $A = n(f,t)$, $B = n(f,\bar{t})$, $C = n(\bar{f},t$, $D = n(\bar{f},\bar{t})$.

After we get Θ by one of the above methods, we make Θ sparse by picking the largest K values in Θ and set other values to 0. We test $K = 30000, 50000$ and 100000, as K increases, the F1 measure increases. When $K > 50000$, the F1-measure doesn't change a lot, so we use $K = 50000$ in all experiments. For each row in Θ, we first find the largest value $\theta_{i,max}$, then set all values in this row to $\theta_{i,j} = \theta_{i,j}/\theta_{i,max}$. We compare the performance of these 3 methods in the experiment section.

FDT has low computation complexity when tagging. For a resource with n features, the complexity of tagging is $O(nm)$, where m is the average tags for each feature in Θ. m is usually a small number, in our model it is 4.63 for bibtex and 5.81 for bookmark. Note that the complexity of FDT is not related to the total number of training documents, tags or users. Nearest neighbor methods have to search in the entire training data set, so the complexity is at least $O(|R|)$. Multi-label classifier methods have to train a classifier for each one of tags, so the complexity is at least $O(|T|)$. Furthermore, the model of FDT is related with K, which is around 10^5, it is small enough to load in the main memory.

4 Graph-based Tag Suggestion

4.1 Method Preliminaries

The basic idea of graph-based tag suggestion is to construct a graph with users, resources and tags as vertices and build edges according to user tagging behaviors. After building the graph, we can adopt some graph-based ranking algorithms to rank tags for a specific user and resource. Then the top-ranked tags are recommended to users.

To describe the graph-based methods more clearly, we first give some mathematical notations. For the folksonomy $\mathbb{F} := (U, T, R, Y)$, we firstly convert it into an undirected tripartite graph $G_{\mathbb{F}} = (V, E)$. In $G_{\mathbb{F}}$, the vertices consists of users, resources and tags, i.e., $V = U \bigcup R \bigcup T$. For each tagging behavior of user u assigning tag t to resource r, we will add edges between u , r and t, i.e., $E = \{\{u,r\},\{u,t\},\{r,t\}|(u,t,r) \in Y\}$.

In $G_{\mathbb{F}}$, we have the set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and the set of edges $E = \{(v_i, v_j) \mid \text{There is an edge between } v_i \text{ and } v_j\}$. For a given vertex v_i, let $N(v_i)$ be the set of vertices that are neighbors of v_i. We have $w(v_i, v_j)$ as the weight of the edge (v_i, v_j). For an undirected graph, $w(v_i, v_j) = w(v_j, v_i)$. Let $w(v_i)$ be the degree of v_i, and we have

$$w(v_i) = \sum_{v_j \in N(v_i)} w(v_j, v_i) = \sum_{v_j \in N(v_i)} w(v_i, v_j) \tag{6}$$

Based on the graph, we can employ various graph-based ranking methods to recommend tags. In this paper, we first introduce two existing methods, including "most popular tags" and "FolkRank". Furthermore, we propose to use a new ranking model, DiffusionRank, for graph-based tag suggestion.

4.2 Most Popular Tags

We first introduce a simple but effective method for tag suggestion. Some notations are given as below, which is identical with [2]. For a user $u \in U$, we denote all his/her tag assignments as $Y_u := Y \bigcap(\{u\} \times T \times R)$. Accordingly, we have Y_r and Y_t. Based on the same principle, we can define $Y_{u,t} := Y \bigcap(\{u\} \times \{t\} \times R)$ for $u \in U$ and $t \in T$. We also have $Y_{t,r}$ accordingly. Furthermore, we denote all tags that user $u \in U$ have assigned as $T_u := \{t \in T | \exists r \in R : (u, t, r) \in Y\}$.

There are variants of "most popular tags" as shown in [8], which are usually restricted in different statistical range. For example, most popular tags of *folksonomy* recommends the most popular tags of the whole set of folksonomy. Therefore, it recommends the same set of tags for any user and resource, which suffers from cold-start problems and has no consideration on personalization.

A reasonable variant of "most popular tags" is recommending the tags that globally are most specific to the resource. The method is named as most popular tags by *resource*:

$$T(u, r) = \underset{t \in T}{\overset{n}{\operatorname{argmax}}}(|Y_{t,r}|) \tag{7}$$

Since users might have specific preferences for some tags, which should have been used by him/her, thus we can use the most popular tags by *user*. As shown in [8], the performance is poor if we use most popular tags by *user* in isolation. If we mix the most popular tags of *user* and *resource*, the performance will be much better than each of them. The simplest way to mix the effect of users and resources on tags is to add the counts and then sort:

$$T(u, r) = \underset{t \in T}{\overset{n}{\operatorname{argmax}}}(|Y_{t,r}| \times |Y_{u,t}|) \tag{8}$$

4.3 FolkRank

FolkRank is originally proposed in [2] which is based on user-resource-tag tripartite graph. In FolkRank, two random surfer model is employed on the tripartite graph. The ranking values of vertices are computed using the following formula:

$$PR(v_i) = \lambda \sum_{v_j \in N(v_i)} \frac{w(v_j, v_i)}{w(v_j)} PR(v_j) + (1 - \lambda) p(v_i) \tag{9}$$

where $PR(v_i)$ is the PageRank value and p_{v_i} is the preference to v_i. Suppose we have an adjacent matrix $\mathbf{A}$ to represent the graph $G_{\mathbb{F}}$:

$$A(i, j) = \begin{cases} 0 & \text{if } (v_i, v_j) \notin E \\ \frac{w(v_i, v_j)}{w(v_j)} & \text{if } (v_i, v_j) \in E \end{cases}$$

With the matrix, we can rewrite the Equation 9 as:

$$\mathbf{s} = \lambda \mathbf{A}\mathbf{s} + (1 - \lambda)\mathbf{p} \tag{10}$$

where $\mathbf{s}$ is the vector of PageRank scores of vertices, and $\mathbf{p}$ is the vector of preferences of vertices.

A straightforward idea of graph-based tag suggestion is to set preference to the user and resource to be suggested for, and then compute ranking values using PageRank in Eq. (10). However, as pointed out in [8], this will make it is difficult for other vertices than those with high edge degrees to become highly ranked, no matter what the preference values are.

Based on above analysis, we described FolkRank as follows. To generate tags for user u and resource r, we have to:

1. Let $\mathbf{s}^{(0)}$ be the stable results of Eq. (10) with $\mathbf{p} = \mathbf{1}$, i.e., the vector composed by 1's.
2. Let $\mathbf{s}^{(1)}$ be the stable results of Eq. (10) with $\mathbf{p} = \mathbf{0}$, but p(u) = 1 and p(r) = 1.
3. Compute $\mathbf{s} := \mathbf{s}^{(1)} - \mathbf{s}^{(0)}$.

Therefore, we can rank tags according to their final values in $\mathbf{s}$, where the top-ranked tags are suggested to user u for resource r.

4.4 DiffusionRank

DiffusionRank was originally proposed for combating web spam [3], which has also been successfully used in social network analysis [4] and search query suggestion [5]. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow.

For a graph $G = \{V, E\}$, denote $f_i(t)$ is the heat on vertex v_i at time t, we construct DiffusionRank as follows. Suppose at time t, each vertex v_i receives an amount of heat, $M(v_i, v_j, t, \Delta t)$, from its neighbor v_j during a period Δt. The received heat is proportional to the time period Δt and the heat difference between v_i and v_j, namely $f_j(t) - f_i(t)$. Based on this, we denote $M(v_i, v_j, t, \Delta t)$ as

$$M(v_i, v_j, t, \Delta t) = \gamma(f_j(t) - f_i(t))\Delta t$$

where γ is heat diffusion factor, i.e. the thermal conductivity. Therefore, the heat difference at node v_i between time $t + \Delta t$ and time t is equal to the sum of the heat that it receives from all its neighbors. This is formulated as:

$$f_i(t + \Delta t) - f_i(t) = \sum_{v_j \in N(v_i)} \gamma(f_j(t) - f_i(t))\Delta t \tag{11}$$

The process can also be expressed in a matrix form:

$$\frac{\mathbf{f}(t + \Delta t) - \mathbf{f}(t)}{\Delta t} = \gamma \mathbf{H}\mathbf{f}(t) \tag{12}$$

where $\mathbf{f}$ is a vector of heat at vertices at time t, and $\mathbf{H}$ is

$$H(i,j) = \begin{cases} -1 & \text{if } i = j \\ 0 & \text{if } (v_i, v_j) \notin E \\ \frac{w(v_i,v_j)}{w(v_j)} & \text{if } (v_i, v_j) \in E \end{cases} \tag{13}$$

If the limit $\Delta t \to 0$, the process will become into

$$\frac{d}{dt}\mathbf{f}(t) = \gamma \mathbf{H}\mathbf{f}(t) \tag{14}$$

Solving this differential equation, we have $\mathbf{f}(t) = e^{\gamma t \mathbf{H}}\mathbf{f}(0)$. Here we could extend the $e^{\gamma t \mathbf{H}}$ as

$$e^{\gamma t \mathbf{H}} = \mathbf{I} + \gamma t \mathbf{H} + \frac{\gamma^2 t^2}{2!}\mathbf{H}^2 + \frac{\gamma^3 t^3}{3!}\mathbf{H}^3 + \cdots \tag{15}$$

The matrix $e^{\gamma t \mathbf{H}}$ is named as the diffusion kernel in the sense that the heat diffusion process continues infinitely from the initial heat diffusion.

γ is an important factor in the diffusion process. If γ is large, the heat will diffuse quickly. If γ is small, the heat will diffuse slowly. When $\gamma \to +\infty$, heat will diffuse immediately, and DiffusionRank becomes into PageRank.

As in PageRank, there are random relations among vertices. To capture these relations, we use a uniform random relation among different vertices as in PageRank. Let $1 - \lambda$ denote the probability that random surfer happens and λ is the probability of following the edges. Based on the above discussion, we can modify DiffusionRank into

$$\mathbf{f}(t) = e^{\gamma t \mathbf{R}}\mathbf{f}(0),\ \mathbf{R} = \lambda \mathbf{H} + (1-\lambda)\frac{1}{N}\mathbf{1} \tag{16}$$

In application, a computation of $e^{\gamma t \mathbf{R}}$ is time consuming. We usually to approximate it to a discrete form

$$\mathbf{f}(t) = (\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^{Mt}\mathbf{f}(0) \tag{17}$$

Without loss of generality, we use one unit time for heat diffusion between vertices and their neighbors, we have

$$\mathbf{f}(1) = (\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^{M}\mathbf{f}(0) \tag{18}$$

We could iteratively calculate $(\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^{M}\mathbf{f}(0)$ by applying the operator $(\mathbf{I} + \frac{\gamma}{M}\mathbf{R})$ to $\mathbf{f}(0)$. Therefore, for each iteration, we could diffuse the heat values at each vertices using the following formulation:

$$\mathbf{s} = (1 - \frac{\gamma}{M})\mathbf{s} + \frac{\gamma}{M}(\lambda \mathbf{A}\mathbf{s} + (1-\lambda)\frac{1}{N}\mathbf{1}) \tag{19}$$

where M is the number of iterations. As analyzed in [3], for a given threshold ϵ, we can compute to get M such that $\|((\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^{M} - e^{\gamma \mathbf{R}})\mathbf{f}(0)\| < \epsilon$ for any $\mathbf{f}(0)$ whose sum is one. Similar to [3], in this paper we set $M = 100$ for DiffusionRank.

Different from FolkRank, in DiffusionRank we set the initial values $\mathbf{f}(0)$ for vertices to indicate the preferences. To suggest tags to user u for resource r, we set $\mathbf{f}(0) = \mathbf{0}$, but for $\mathbf{f}_u(0) = 1$ and $\mathbf{f}_r(0) = 1$. After running DiffusionRank on the tripartite graph, we rank tags according to their ranking scores and the top-ranked tags are suggested to user u for resource r.

5 Experiments

5.1 Data Set

We use the given BibSonomy data set to validate our methods, it is a snapshot of the BibSonomy system until Jan 1, 2009. The data set contains two parts, *bibtex* and *bookmark*. In *bibtex*, the resources are citation of research papers or books, with title, author and other information. In *bookmark*, the resources are website URLs with a user-provided short description. Additionally, the contest organizer provide two *postcore-2* data sets. In the *postcore-2* data sets, the organizer removed all users, tags, and resources which appear in only one post. The process was iterated until convergence and got a core in which each user, tag, and resource occurs in at least two posts. Batagelj et al [9] provided a detailed explanation of postcore building . The basic statistics of these data sets are lists in Table 1

Table 1. Basic statistics of the full bibtex and bookmark data sets. Mean Len. is the mean number of words in the corresponding text content.

Name	#posts	#tags	#users	#words	Mean Length	Mean #tags/user
bibtex	158,912	50,855	1,790	278,106	47.67	60.75
bookmark	263,004	56,424	2,679	293,026	11.83	57.78
bibtex(pcore2)	22,852	5,816	788	48,401	59.21	31.75
bookmark(pcore2)	41,268	10,702	861	47,689	12.23	60.26

To validate and tune our methods, we split each of the four dataset into 5 equal-sized subset randomly, and perform 5-fold cross validation on them.

5.2 Evaluation Metrics

We use precision, recall and F1 measure as the evaluation metrics. Precision is the number of correct suggested tags multiplied by the total number of tags suggested. Recall is the number of correct suggested tags multiplied by the total number of tags of original post. F1 measure is a geometry mean of precision and recall, $F1 = 2Precsion \times Recall/(Precision + Recall)$. For each post, we only consider the first 5 tags suggested.

5.3 Content-based Tag Suggestion

To test the performance of our content-based method, we run 5-fold cross validation using the given training data. Additionally, for each fold, we remove all posts in the postcore set from the test data, since posts in postcore will not appear in the final test data. We remove stopwords, punctuation marks and all words shorter than 2 letters from the data set, and convert all text to lowercase. We remove words, resource IDs and user IDs appear in less than 5 post. We treat bibtex and bookmark separately.

We use search-based kNN as our baseline method, this is proposed by Mishne [10] for suggesting tags to blog posts. In our experiment, we index the training data by Lucene[1] indexing package. For a test post, we use $TF \times IDF$ to select 10 top words. Then, we use these words to construct a weighted query, and search the training data with it. We take all tags from Lucene returned top-k documents, weight each tag using the corresponding document's relevance score, and sum the weights of duplicated tags. We take the first 5 tags as the suggested tags. In search-based kNN, k is a parameter to tune. After using $k = 1, 2, 3, 4, 5$, we use $k = 1$ as the final k, since it has the best F1 measure.

We list the mean precision, recall and F1 value for bibtex and bookmark data in Table 2 and 3 respectively. We experimented with the different combination of methods for weighting features and estimating Θ matrix.

In the bibtex dataset, FDT(TFITF+MI) has the similar performance as the search-based kNN methods. In the bookmark dataset, FDT(TFITF+MI) has the best performance, which is 3 percentage better than search-based kNN.

Table 2. P, R and F1 of search-based kNN and different learning methods for FDT on the bibtex dataset. All averaged over 5 folds.

Method	Precision	Recall	F1
search-based kNN	**0.2792**	0.2324	**0.2537**
FDT(TFIDF+CC)	0.2517	0.2152	0.2320
FDT(TFIDF+MI)	0.1822	0.1652	0.1733
FDT(TFIDF+χ^2)	0.2261	0.2235	0.2248
FDT(TFITF+CC)	0.2513	0.2173	0.2330
FDT(TFITF+MI)	0.2432	**0.2526**	0.2478
FDT(TFITF+χ^2)	0.2216	0.2246	0.2231

In the training data, the number of post from each user roughly follows the power law distribution, where most users have less than 100 posts, and the top 4 users have 50% of all posts. If we treat all posts as equal, then the model may bias to the preference of several super users. To know the performance of the methods on super users and common users, we run other two experiments. In the first experiment, we train the model using posts from all users, then check its performance on each of the top n users and all the rest users separately. In

[1] http://lucene.apache.org

Table 3. P, R and F1 of search-based kNN and different learning methods for FDT on the bookmark dataset. All averaged over 5 folds.

Method	Precision	Recall	F1
kNN	0.2935	0.2409	0.2646
FDT(TFIDF+CC)	**0.4036**	0.2411	**0.3018**
FDT(TFIDF+MI)	0.3580	0.1892	0.2475
FDT(TFIDF+χ^2)	0.2675	0.2486	0.2577
FDT(TFITF+CC)	0.3633	0.2404	0.2893
FDT(TFITF+MI)	0.3143	**0.2610**	0.2852
FDT(TFITF+χ^2)	0.2284	0.1831	0.2033

the second experiment, we train and test models using only posts from each of the top n users and all the rest users separately. For bibtex dataset, we choose $n = 4$, for bookmark dataset, we choose $n = 5$. The results for bibtex and bookmark are listed in Table 4 and Table 5 respectively. In these experiments, we use FDT(TF*ITF+MI) for bibtex data and FDT(TF*IDF+CC) for bookmark data. In the result table, the column *Trained(ALL)* means all methods are trained on full training data. The column *Trained(USER)* means each method is trained using only posts from corresponding group of users. In the method name, *kNN(2463)* means the method used is search-based kNN, and test data set are all post of user 2463, rest means all other users. The same naming rule applies to *FDT(xxxx)*.

For each group of test data, we have 4 different models, they are kNN trained by all users, kNN trained by this group, FDT trained by all users and FDT trained by this user. As the result shows, for groups of super users, kNN-based models have best performance. For common users (the *rest* group), FDT-based models performs better. This result follows our intuition. In this data set, super users have different tag preference than common users. kNN suggest tags using most similar resources, it is less affected by the overall distribution of resources, so it fits to the . FDT relies on the global statistics of feature-tag relationship, it is less effective to fit a special user's preference. In practical situation, we can get the best performance by choosing different model for different group of users.

One interesting observation is about the user #2732. When trained with all posts, FDT performs much better(0.6308 vs 0.2300) on #2732 than trained with #2732's own posts. We examined the posts of #2732, found that many posts contains only three tags: *genetic*, *programming* and *algorithm*, and the number of posts by #2732 is large. When we use all posts to train FDT(TFITF+MI), these three tags have a large Mutual Information value with many features, especially the user id feature "UID-2732", so FDT can predict tags for posts of #2732 with high accuracy. When trained only with #2732's posts, the Mutual Information between features and these three tags is much smaller, since these three tags appears everywhere and can be seen as stopwords in tags. Small Mutual Information of these three tags means FDT will make wrong prediction about most posts of #2732, which leads to a decreasing in F1-measure.

Table 4. P, R and F1 of search-based kNN and FDT on different set of users on the bibtex dataset. Trained(ALL) means that we train the model using posts from all users, and test the performance on given set of users. Train(USER) means that both training and testing use posts from the given set of users. The % column indicates the size of corresponding user group, as the percentage in all posts. All averaged over 5 folds.

Trained(ALL)	P	R	F1	Trained(USER)	P	R	F1	%
kNN(2463)	0.1323	0.1352	0.1337	kNN(2463)	0.1376	0.1377	0.1376	24.27
kNN(2651)	0.1561	0.1573	0.1567	kNN(2651)	0.1953	0.1910	**0.1932**	12.40
kNN(3180)	0.4278	0.2771	0.3364	kNN(3180)	0.4440	0.2807	0.3440	9.20
kNN(2732)	0.6267	0.3915	0.4819	kNN(2732)	0.6517	0.4422	0.5269	3.78
kNN(rest)	0.3207	0.2530	0.2829	kNN(rest)	0.3202	0.2579	0.2857	50.35
FDT(2463)	0.1066	0.1869	0.1358	FDT(2463)	0.1100	0.2055	**0.1429**	24.27
FDT(2651)	0.1022	0.1285	0.1138	FDT(2651)	0.1126	0.1818	0.1391	12.40
FDT(3180)	0.3656	0.3334	**0.3488**	FDT(3180)	0.3688	0.3274	0.3469	9.20
FDT(2732)	0.8763	0.4927	**0.6308**	FDT(2732)	0.3142	0.1814	0.2300	3.78
FDT(rest)	0.3101	0.2516	0.2778	FDT(rest)	0.3260	0.2559	**0.2867**	50.35

Table 5. P, R and F1 of search-based kNN and FDT on different set of users on the bookmark dataset. Trained(ALL) means that we train the model using posts from all users, and test the performance on given set of users. Train(USER) means that both training and testing use posts from the given set of users. The % column indicates the size of corresponding user group, as the percentage in all posts. All averaged over 5 folds.

Trained(ALL)	P	R	F1	Trained(USER)	P	R	F1	%
kNN(1747)	0.5523	0.4877	0.5180	kNN(1747)	0.6513	0.5566	**0.6003**	19.90
kNN(2977)	0.4554	0.4072	0.4299	kNN(2977)	0.5567	0.5154	**0.5353**	9.48
kNN(483)	0.1002	0.1365	0.1156	kNN(483)	0.2375	0.2227	**0.2299**	3.56
kNN(275)	0.2102	0.1947	0.2022	kNN(275)	0.3413	0.3059	**0.3226**	3.40
kNN(421)	0.2749	0.0867	0.1318	kNN(421)	0.2787	0.1080	0.1557	2.26
kNN(rest)	0.1921	0.1627	0.1762	kNN(rest)	0.2007	0.1643	0.1807	61.41
FDT(1747)	0.5306	0.4169	0.4670	FDT(1747)	0.3592	0.2325	0.2823	19.90
FDT(2977)	0.4437	0.3622	0.3988	FDT(2977)	0.4162	0.3367	0.3722	9.48
FDT(483)	0.1684	0.2653	0.2060	FDT(483)	0.1642	0.2637	0.2024	3.56
FDT(275)	0.1887	0.1610	0.1738	FDT(275)	0.2531	0.1760	0.2076	3.40
FDT(421)	0.4044	0.1258	0.1920	FDT(421)	0.4328	0.1339	**0.2045**	2.26
FDT(rest)	0.2462	0.2133	0.2286	FDT(rest)	0.2502	0.2204	**0.2344**	61.41

For final test, we use FDT(ITF+MI) for bibtex and FDT(IDF+CC) for bookmark. The test data of DC09 has a different distribution with the training data. Most top ranked users don't appear in the test data. So we removed the top ranked users from the training data, use the *rest* group of users to train the model for final suggestion. The p/r/f1 on final test data are 0.1388/0.1049/0.1189 respectively. Compared to the cross validation results, the performance dropped a lot on final test data. One reason is that FDT does not suggest tags that are not in the training data. There are 93756 tags in the training data and 34051 tags in the test data, the overlapped tags are only 15194. To achieve better performance, suggesting new tags should be considered in the future.

5.4 Graph-based Tag Suggestion

In experiments, we compare the results of three graph-based methods, most popular tags, FolkRank and DiffusionRank.

Here we first demonstrate the results using 5-fold cross validation on training dataset. In Table 6, we show the best performance of various methods on bibtex dataset. In this table, we also demonstrate the performance of the content-based method kNN , which achieves the best result when $k = 2$. For the method of most popular tags, we use "mpt+resource" to indicate most popular tags by *resource*, and "mpt+mix" to indicate most popular tags by mixing *resource* and *user*. For FolkRank, the best result is achieved when damping factor $\lambda = 0.01$ with 100 iterations. DiffusionRank obtains the best result when damping factor $\lambda = 0.85$, maximum number of iterations $max_it = 10$ and diffusion factor $\gamma = 0.1$. From the table, we can see that most popular tags by mix achieves the best F_1-measure, which has the largest precision. While for DiffusionRank, it achieves the best recall.

Table 6. Best performance of various methods on bibtex training dataset. All values are averaged over 5 folds.

Method	Precision	Recall	F_1-measure
kNN	0.3664	0.4307	0.3959
mpt+resource	0.3949	0.3765	0.3855
mpt+mix	**0.4211**	0.4014	**0.4110**
FolkRank	0.3222	0.4459	0.3741
DiffusionRank	0.3347	**0.4630**	0.3885

In Table 7, we show the best performance of various methods on bookmark dataset. kNN achieves the best performance when $k = 2$. For FolkRank, the best result is achieved when damping factor $\lambda = 0.0001$ with 10 iterations. DiffusionRank obtains the best result when damping factor $\lambda = 0.85$, maximum number of iterations $max_it = 10$ and diffusion factor $\gamma = 0.01$. Furthermore, we also restrict the scores of suggested tags should be no less than 1/5 of score of first-ranked tags. From the table, we can see that DiffusionRank achieves the best F_1-measure, which has the largest precision.

Table 7. Best performance of various methods on bookmark training dataset. All values are averaged over 5 folds.

Method	Precision	Recall	F_1-measure
kNN	0.2855	0.2892	0.2873
mpt+resource	0.3345	0.2798	0.3047
mpt+mix	0.3606	0.3017	0.3285
FolkRank	0.3288	**0.3309**	0.3298
DiffusionRank	**0.3772**	0.3266	**0.3501**

From the above two tables, we find that on the bibtex dataset the method of most popular tags by mix is the best, and on bookmark dataset DiffusionRank achieves the best result. Therefore, for task 2 of rsdc'09, we use the two methods to train ranking models separately on bibtex and bookmark. Using the original result and evaluation program provided by the challenge organizer, we obtain the evaluation results on test dataset, as shown in Table 8. From the table, we find that the absolute values are much smaller than what are shown in Table 6 and 7.

Table 8. Evaluation result on test dataset of rsdc'09.

Tag Number	Precision	Recall	F_1-measure
1	0.1483	0.4229	0.2196
2	0.2301	0.3477	0.2769
3	0.2960	0.3113	0.3034
4	0.3418	0.2840	0.3102
5	0.3760	0.2601	0.3075

Besides the above analysis, we want to investigate the performance of FolkRank and DiffusionRank as their parameters change.

In Table 9 and 10, we demonstrate the performance of FolkRank on bibtex training dataset and bookmark training dataset as its parameters, the damping factor λ and maximum number of iterations (denoted as "max-it" in tables) change. From the both tables, we find the performance of FolkRank improves as damping factor shrinks, which indicates the effect of preference values are growing larger. That is to say the generalization of FolkRank by passing values iteratively on graphs may harm the performance. Moreover, it seems that the maximum number of iterations of FolkRank does not effect the results significantly.

In Table 11 and 12, we demonstrate the performance of DiffusionRank on bibtex training dataset and bookmark training dataset as its parameters, the

Table 9. Performance of FolkRank on bibtex training dataset. All values are averaged over 5 folds.

λ	max-it	Precision	Recall	F_1-measure
0.85	10	0.2943	0.4072	0.3417
0.5	10	0.3053	0.4225	0.3545
0.1	10	0.3198	0.4425	0.3713
0.01	10	**0.3222**	**0.4459**	**0.3741**
0.01	100	**0.3222**	**0.4459**	**0.3741**
0.001	10	0.3219	0.4455	0.3738
0.0001	10	0.3219	0.4455	0.3738

Table 10. Performance of FolkRank on bookmark training dataset. All values are averaged over 5 folds.

λ	max-it	Precision	Recall	F_1-measure
0.85	10	0.2989	0.3008	0.2998
0.5	10	0.3038	0.3058	0.3048
0.1	10	0.3198	0.3218	0.3208
0.01	10	0.3275	0.3297	0.3286
0.01	100	0.3275	0.3297	0.3286
0.001	10	**0.3288**	**0.3309**	**0.3298**
0.0001	10	**0.3288**	**0.3309**	**0.3298**

diffusion factor γ and maximum number of iterations (denoted as "max-it" in tables) change. Here the damping factor λ is set to 0.85. We also find that the performance of DiffusionRank improves as diffusion factor shrinks, which indicates the effect of initial values is growing larger. Similar to FolkRank, the generalization of DiffusionRank by passing values iteratively on graphs may also harm the performance. It is also the same as FolkRank that the maximum number of iterations of DiffusionRank does not effect the results significantly.

From the experiments on both bibtex and bookmark training datasets, we can see that DiffusionRank always outperforms FolkRank with some specific parameters, which is more significant on bookmark dataset. Although in this dataset, FolkRank does not outperform the method of most popular tags, in [8] we know that in some datasets, FolkRank outperforms most simple methods including the method of most popular tags. Therefore, more experiments still need to be done to investigate the efficiency of DiffusionRank compared to FolkRank and other graph-based methods for tag suggestion.

Furthermore, the number of suggested tags should be specified in advance in FolkRank and DiffusionRank. However in some conditions, we do not have to recommend as many tags as specified. For DiffusionRank, we set the maximum number of suggested tags is 5. If we further require the suggested tags should have the ranking values no less than 1/5 of the ranking value of the first-ranked tag, the performance of precision, recall and F_1-measure will be improved to 0.3772, 0.3266 and 0.3501 on bookmark training dataset. Therefore, we use the altered DiffusionRank for the bookmark test set of task 2 in rsdc'09.

Table 11. Performance of DiffusionRank on bibtex training dataset. In all experiments, damping factor λ is set to $\lambda = 0.85$. All values are averaged over 5 folds.

γ	max-it	Precision	Recall	F_1-measure
2.0	10	0.3279	0.4537	0.3807
1.0	10	0.3331	0.4609	0.3867
0.1	10	**0.3347**	**0.4630**	**0.3885**
0.1	100	**0.3347**	**0.4630**	**0.3885**
0.01	10	**0.3347**	**0.4630**	**0.3885**

Table 12. Performance of DiffusionRank on bookmark training dataset. In all experiments, damping factor λ is set to $\lambda = 0.85$. All values are averaged over 5 folds.

γ	max-it	Precision	Recall	F_1-measure
2.0	10	0.3336	0.3357	0.3346
1.0	10	0.3370	0.3392	0.3381
0.1	10	0.3403	0.3425	0.3414
0.1	100	0.3403	0.3425	0.3414
0.01	10	**0.3406**	**0.3428**	**0.3417**

6 Related Work

Ohkura et al [11] proposed a Support Vector Machine-based tag suggestion system. They train a binary classifier for each tag to decide if this tag should be suggested. Katakis et al [12] use a hierarchical multi-label text classifier to find the proper tags for a document. They cluster all tags using modified k-means, use one classifier to decide which clusters a document belongs to, then use another cluster-specific classifier to decide which tags in the cluster belongs to the document. Mishne [10] use a search-based nearest neighbor method to suggest tags, where the tags of a new document is collected from the most relevant documents in the training set. Lipczak et al [13] extract keywords from the title of a document, then filter them with a user's used tags to get the final suggestion. These methods all use the content of a document, we call them content-based methods. Tatu et al [14] combine tags from similar documents and extracted keywords to provide tag suggestions. They have the best performance in the first ECML/PKDD Discovery Challenge task.

Another class of tag suggestion system is based on the links between users, tags and resources, which does not take the content of resources into consideration. Since the method of "most popular tags" also does not consider the content of resources, in this paper we regard it as a member of graph-based tag suggestion approach. Xu et al [15] use collaborative filtering to suggest tags for URL bookmarks. Jaschke et al [2] proposed FolkRank, a PageRank-like iterative algorithm to find the most related tags for a resource in its neighbor users and tags. PageRank is originally used for ranking web pages only according to the topology of web graph. However, in PageRank we can set preference values to a subset of pages to make the PageRank values biased to these pages and their neighbors. In fact, FolkRank is used to compute the relatedness between tags and the specific user and resource by setting the given user and resource to high preference values in PageRank.

Recently, a new graph-based ranking method, DiffusionRank [3], is proposed for anti-spam of web pages. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow. Based on the property of heat always flow from high to low, the ranking values of DiffusionRank are related to initial values of vertices. Therefore, DiffusionRank provides a more flexible method to rank tags by setting high initial values to the given user and resource. In this paper, we for the first time propose to use DiffusionRank for graph-based tag suggestions.

7 Conclusion

In this paper, we study the problem of tag suggestion and describe our methods for content-based and graph-based suggestion. For content-based tag suggestion, we propose a new method named Feature-Driven Tagging for fast content-based tag suggestion. Cross validation on the training data shows that FDT outperforms wildly-used search-based kNN, especially when suggesting tags for long-tail users. For graph-based tag suggestion, we study most popular tags, FolkRank, and propose a DiffusionRank-based method. Experiments show that on bibtex dataset the method of most popular tags by mixing of user and resource performs best, and on bookmark dataset, DiffusionRank outperforms other methods.

Work remains to be done. First, currently we use empirical methods to estimate the parameters for FDT, like CC, MI and ITF. We will consider learn a Θ matrix directly by optimizing a tag-related loss function. Second, evaluation using final test data of DC09 shows that the F1 value drops a lot than cross validation on the training data, especially for content-based methods. This suggests we should pay attention to out-of-vocabulary tags. Third, more information should be considered, such like time-stamp, to suggest better tags in real-world situation.

Acknowledgments

This work is supported by the National Science Foundation of China under Grant No. 60621062, 60873174 and the National 863 High-Tech Project under Grant No. 2007AA01Z148.

References

1. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)

2. Jaschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag Berlin, Heidelberg (2007) 506–514
3. Yang, H., King, I., Lyu, M.R.: Diffusionrank: a possible penicillin for web spamming. In: Proceedings of SIGIR. (2007) 431–438
4. Ma, H., Yang, H., Lyu, M.R., King, I.: Mining social networks using heat diffusion processes for marketing candidates selection. In: Proceeding of CIKM. (2008) 233–242
5. Ma, H., Yang, H., King, I., Lyu, M.R.: Learning latent semantic relations from clickthrough data for query suggestion. In: Proceeding of CIKM. (2008) 709–718
6. Manning, C., Raghavan, P., Schtze, H.: Introduction to information retrieval. Cambridge University Press New York, NY, USA (2008)
7. Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, MORGAN KAUFMANN PUBLISHERS, INC. (1997) 412–420
8. Jaschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in social bookmarking systems. AI Communications **21**(4) (2008) 231–247
9. Batagelj, V., Zaversnik, M.: Generalized cores (2002)
10. Mishne, G.: Autotag: a collaborative approach to automated tag assignment for weblog posts. In: Proceedings of the 15th international conference on World Wide Web, ACM New York, NY, USA (2006) 953–954
11. Ohkura, T., Kiyota, Y., Nakagawa, H.: Browsing system for weblog articles based on automated folksonomy. In: Proceedings of the WWW 2006 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, at WWW. Volume 2006. (2006)
12. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. ECML PKDD Discovery Challenge 2008 75
13. Lipczak, M.: Tag Recommendation for Folksonomies Oriented towards Individual Users. ECML PKDD Discovery Challenge 2008 84
14. Tatu, M., Srikanth, M., D'Silva, T.: Rsdc'08: Tag recommendations using bookmark content. ECML PKDD Discovery Challenge 2008
15. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. In: Collaborative Web Tagging Workshop at WWW2006. (2006)

RSDC'09: Tag Recommendation Using Keywords and Association Rules

Jian Wang, Liangjie Hong and Brian D. Davison

Department of Computer Science and Engineering
Lehigh University, Bethlehem, PA 18015 USA
{jiw307, lih307, davison}@cse.lehigh.edu

Abstract. While a webpage usually contains hundreds of words, there are only two to three tags that would typically be assigned to this page. Most tags could be found in related aspects of the page, such as the page own content, the anchor texts around the page, and the user's own opinion about the page. Thus it is not an easy job to extract the most appropriate two to three tags to recommend for a target user. In addition, the recommendations should be unique for every user, since everyone's perspective for the page is different. In this paper, we treat the task of recommending tags as to find the most likely tags that would be chosen by the user. We first applied the TF-IDF algorithm on the limited description of the page content, in order to extract the keywords for the page. Based on these top keywords, association rules from history records are utilized to find the most probable tags to recommend. In addition, if the page has been tagged before by other users or the user has tagged other resources before, that history information is also exploited to find the most appropriate recommendations.

1 Introduction

Social bookmarking services allow users to share and store references to various types of World Wide Web (WWW) resources. Users can assign tags to these resources, several words best describing the resource content and his or her opinion. To assist the process of assigning tags, some services would provide recommendations to users as references. In Tatu et al. [5] work, they mentioned that the average number of tags in RSDC'08 bookmarking data is two to three. Thus, it is not an easy task to provide reasonable tag recommendations for the resource with only two to three related tags on average. Tag recommendation is a challenge task in ECML PKDD 2009 where participants should provide either content-based or graph-based methods to help users to assign tags. This work shows some results that aim to this challenge.

The challenge provides description of the resources and posts of the tag. Description contains some basic information about the resources and post is the tuple of user, tag and resource. In the challenge, there are two types of resources, normal web pages, named as *bookmark*, and research publications, named as *bibtex*, with different schemas of descriptions. A post records the resource and the tags assigned to it by a particular user. The task is to provide ***new*** tags to ***new*** resources with high F-Measure performance on the top five recommendations. The difficulties of this challenge fall in:

- How to take advantage of the record content itself, while the description is very limited? For example, *bookmark* is only described with the title of the web page and a short summary while *bibtex* is usually described with title, publication name, and authors of the paper.
- How to utilize history information to recommend tags which do not appear in the page content? Though we can use keywords to help find possible tags, tags are not just keywords. Tags could be user's opinion about the page, the category of the page, so on and so forth. This kind of tag might be tracked by using history information.
- How to choose the most appropriate two to three tags among the potential pool? By analyzing the page content and history information, we might have a pool which contains the reasonable tag recommendations. Yet we cannot recommend all those to the user. Instead of that, only two to three tags need to be extracted from that pool.

In order to solve the above problems, we propose tag recommendation using both keywords in the content and association rules from history records. After we end with a pool which contains potential appropriate tags, we introduce a method, named *common and combine*, to extract the most probable ones to recommend. Our evaluation showed that integrating association rules can give better F-Measure performance than simply using keywords.

Besides using association rules, some history information will be used more directly, if the resource has been tagged before or the target user tagged other documents before. These history records would greatly improve recommendation performance.

In this paper, we tuned some parameters in our recommendation system to generate the best F-Measure performance while recommending at most five tags.

2 Related Work

Lipczak [3] proposed a recommendation system mainly based on individual posts and the title of the resource. The key conclusion of their experiments is that, they should not only rely on tags previously attached when making recommendations. Sparsity of data and individuality of users greatly reduce the usefulness of previous tuple data. Looking for potential tags they should focus on the direct surrounding of the post, suggesting a graph-based method. Tatu et al. [5] proposed a recommendation system that takes advantage of textual content and semantic features to generate tag recommendations. Their system outperformed other systems in last year's challenge. Katakis et al. [2] proposed a multilabel text classification recommendation system that used titles, abstracts and existing users to train a tag classifier.

In addition, Heymann et al. [1] demonstrated that “Page text was strictly more informative than anchor text which was strictly more informative than surrounding hosts”, which suggests that we do not have to crawl other information besides page content. They also showed that the use of association rules can help to find recommendations with high precision.

3 Dataset Analysis and Processing

3.1 Dataset from the Contest

Three table files were provided by the contest, including *bookmark*, *bibtex* and *tas*. The *bookmark* file contains information for bookmark data such as contentID, url, url-hash, description and creation date. The *bibtex* file contains information for bibtex data such as contentID, and all other related publication information. The *tas* file contains information for (user, tag, resource) tuple, as well as the creation date. The detailed characteristics for these files could be found in Table 1. In this work, all contents were transformed into lower case since the evaluation process of this contest ignores case. In the mean time, we filtered the latex format when we exported bibtex data from the database.

Table 1. Detail information about training dataset, provided by the contest

file	# of lines	information
bookmark	263,004	content_id (matches tas.content_id), url_hash (the URL as md5 hash), url, description, extended description, date
bibtex	158,924	content_id (matches tas.content_id), journal, volume chapter, edition, month, day, booktitle, editor, year howPublished, institution, organization, publisher address, school, series, bibtexKey, url, type, description annote, note, pages, bKey, number, crossref, bibtexAbstract simhash0, simhash1, simhash2, entrytype, title, author, misc
tas	1,401,104	userID, tag, content_id (matches bookmark.content_id or bibtex.content_id) content_type (1 = bookmark, 2 = bibtex), date

3.2 Building Experiment Collection

We considered and tried merging duplicate records together in training process yet found it did not help much. Thus we kept the duplicate records when building our experiment collections. Since our proposed tag recommendation approach does not involve a training process, we did not separate the dataset into training one and testing one at first. We evaluated our recommendation system on all documents in the given dataset. Based on the type of documents, there are three different collections in our dataset:

bookmark collection from dataset provided We created a collection *bookmark_more* to contain all bookmark information which were provided by the contest training dataset. Every document in the collection corresponds to a unique contentID in *bookmark* file. It contains all information for that record, including *description* and *extended description*. There are 263,004 documents in this collection.

During the experiment, we crawled the external webpage for every contentID. Yet the performance showed that the external webpage are not as useful as the simple description provided by the contest. Regardless of performance, it also cost too much time, which is not realistic for online tag recommending. In addition, an external webpage usually contains too many terms, which makes it even harder to extract two to three appropriate terms to recommend as tags.

bibtex collection from dataset provided We created a collection *bibtex_original* to contain all bibtex information which were provided by the original dataset. Every document in the collection corresponds to a unique contentID in *bibtex* file. It contains all information for that record, including all attributes in Table 1 except simhash0, simhash1 and simhash2. There are 158,924 documents docs in this collection.

bibtex collection from external resources If the url of a bibtex record points to some external websites such as portal.acm.org and citeseer, we crawled that webpage and extracted useful information for this record. All these documents are stored in another collection. Similarly, every document in the collection corresponds to a unique contentID in *bibtex* file. There are 3,011 documents in this collection *bibtex_parsed*.

4 Keyword-AssocRule Recommendation

We consider the tag recommendation problem as to find the most probable terms that would be chosen by users. In this paper, $P(X)$ indicates the probability of term X to be assigned to the document as tag. For every document, the term with high $P(X)$ has the priority to be recommended.

4.1 Keyword Extraction

In this step, our assumption is that the more important this term in the document, the more probable for this term to be chosen as tag.

We used two term weighting functions, TF-IDF and Okapi BM25 [4] to extract "keywords" from resources. In a single collection, we calculated TF-IDF and BM25 value for every term in every document.

For TF-IDF, the weighting function is defined as follows:

$$TF - IDF = TF_{t,d} \times IDF_t \tag{1}$$

where $TF_{t,d}$ is the term frequency that equal to the number of occurrences of term t in document d. IDF_t is *inverse document frequency* that is defined as:

$$IDF_t = log \frac{N}{df_t} \tag{2}$$

where df_t is the number of documents in the collection that contain a term t and N is the total number of documents in the corpus.

For Okapi BM25, the weighting function is defined as follows:

$$BM25 = \sum_{i=1}^{n} IDF(q_i) \frac{TF_{t,d}(1 + k_1)}{TF_{t,d} + k_1(1 - b + b \times \frac{L_d}{L_{ave}})} \tag{3}$$

where $TF_{t,d}$ is the frequency of term t in document d and L_d and L_{ave} are the length of document d and the average document length for the whole collection. $IDF(q_i)$ here is defined as

$$IDF(q_i) = log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{4}$$

The terms in the single document are ranked according to its TF-IDF or BM25 value in decreasing order. A term with high value or high rank is considered to be more important in the document. Thus $P_k(X)$ can be calculated by Algorithm 1.

Algorithm 1 To calculate $P_k(X)$, by using results from keyword extraction method

```
for all documents in the collection do
  rank all terms according to TF-IDF or BM25 value in decreasing order
  for all term $X$ in the document do
    $P_k(X) = 100 - rank(X)$;
    {//$rank(X)$ = 1 indicated the top position, 2 indicated the second position, etc. }
  end for
end for
```

As shown in Table 2, TF-IDF performed better than BM25 in tag recommendation process. The following processes in this work were all performed based on results of TF-IDF method.

4.2 Using Association Rules

Recent work by Heymann et al. [1] showed that using association rules could help to find tag recommendation with high precision. They expanded their recommendation pool in decreasing order of confidence. In this paper, we used

Table 2. Performance of key word extraction method in every collection, while recommending at most 5 tags.

collection	BM25			TF-IDF		
	recall	precision	f-measure	recall	precision	f-measure
bibtex_original	0.0951	0.0561	0.0706	0.0989	0.0592	0.0741
bibtex_parsed	0.1663	0.1059	0.1294	0.1800	0.1158	0.1409
bookmark_more	0.1186	0.0940	0.1049	0.1189	0.0943	0.1052

alternative approaches to deeply analyze association rules, which are found in history information. It does help to extract tags which are more likely to be used by users.

Finding association rules in history records We used three key factors in association rules, including *support*, *confidence* and *interest*. Every unique record is treated as the basket and the tags (X, Y, etc.) associated with every record are treated as the items in the basket. For every rule $X \rightarrow Y$, *support* is the number of records that contain both X and Y. *Confidence* indicates the probability of Y in this record if X already associates with the record, i.e., $P(Y|X)$. *Interest* is $P(Y|X) - P(Y)$, showing how much more possible that X and Y associating with the record together.

Table 3. Sample Association rules found in training dataset

bookmark				bibtex			
$X \rightarrow Y$	confidence	support	interest	$X \rightarrow Y$	confidence	support	interest
$blog \rightarrow software$	0.0541	291	0.0454	$systems \rightarrow algorithms$	0.2886	295	0.2757
$blogs \rightarrow blogging$	0.1345	291	0.1333	$algorithms \rightarrow systems$	0.0492	295	0.0470
$blogging \rightarrow blogs$	0.2910	291	0.2885	$systems \rightarrow genetic$	0.2847	291	0.2721
$artery \rightarrow cardiology$	0.9510	291	0.9506	$genetic \rightarrow systems$	0.0497	291	0.0475
$photos \rightarrow photography$	0.3149	290	0.3138	$tagging \rightarrow folksonomy$	0.5097	288	0.5085
$photography \rightarrow photos$	0.3142	290	0.3131	$folksonomy \rightarrow tagging$	0.5115	288	0.5103
$learning \rightarrow foodcooking$	0.1004	290	0.1000	$genetic \rightarrow and$	0.0466	273	0.0441

The rules $X \rightarrow Y$ we constructed all have support > 10, thus at least 10 resources in our training dataset contain both X and Y as tags. As we mentioned before, we did not separate the dataset into training and testing sets. During evaluation, some records might benefit from the rule it contributed at first, yet at least 9 more resources also contributed to the rule. The support limit here is chosen arbitrarily. Two sets of rules are constructed independently, one for bookmark dataset and another one for bibtex dataset. Some sample rules are showed in Table 3.

Choosing appropriate recommendations by using association rules Here the problem becomes to be:

If $X \rightarrow Y$ exists in the association rules, how possible that term Y should be recommended when X is likely to be recommended?

Given $P(X)$ and the confidence value $P(Y|X)$, $P(Y)$ could be calculated according to law of total probability, which is sometimes called as law of alternatives:

$$P(Y) = \sum_{n} P(Y \cap X_n) \tag{5}$$

or

$$P(Y) = \sum_{n} P(Y \mid X_n)P(X_n) \tag{6}$$

since $P(Y \cap X_n) = P(Y \mid X_n)P(X_n)$. According to the above equations, the algorithm to calculate $P_a(Y)$, which is called $Assoc(Y)$ in this paper, is shown in Algorithm 2.

Algorithm 2 To calculate $P_a(X)$, by using association rules

```
for all documents in the collection do
  for all term $X$ in the document do
    for all association rule $X \rightarrow Y$ do
      $P_a(Y) += (confidence\ of\ X \rightarrow Y) * P_k(X)$;
      {//$P_k(X)$ is calculated by Algorithm 1}
    end for
  end for
end for
```

4.3 Combining Keyword Extraction with Association Rules Results

After $P_k(X)$ and $P_a(X)$ are calculated for every term in the document, one method, in Algorithm 3, is to linearly combine the two values to calculate the final probability $P_c(X)$ for recommending a term X.

Similarly, term with higher $P_c(X)$, i.e., higher rank in Combined results has the priority to be recommended.

The experiments showed that *weight* could affect the F-Measure performance and the optimal *weight* to combine is different for every collection. Figure 1 shows the effect of *weight* in *bibtex_parsed* collection, where F-Measure reaches the peak during increase of *weight* from 0.1 to 0.9. This trend is similar in other two collections. Our experiments indicated that the optimal *weight* to achieve best F-Measure for *bibtex_parsed*, *bibtex_original*, *bookmark_more* is 0.7, 0.5 and 0.5, respectively. The evaluation results with optimal *weight* for every collection, in this step, is shown in the second column of Table 4. Compared to the TF-IDF results in the first column, it is obvious that the association rules can greatly help to improve the F-Measure performance.

Another method we found that worked well is *common and combine*. In *common* step, if the term in top rank of keyword extraction results do have $Assoc(X) > 0$, then recommend this term. In *combine* step, extract terms with

Algorithm 3 To calculate $P_c(X)$, by linearly combining results from TF-IDF & association rules

```
for all documents in the collection do
    $TF-IDFmax$ = maximum $TF-IDF(X)$ for all terms in this document
    $Assocmax$ = maximum $Assoc(X)$ for all terms in this document
    for all term $X$ in the document do
        $TF-IDF(X) = TF-IDF(X)/TF-IDFmax$;
        {// normalize $TF-IDF(X)$ value}
        $Assoc(X) = Assoc(X)/Assocmax$;
        {// normalize $Assoc(X)$ value, $Assoc(X) = P_a(X)$ in Algorithm 2.}
        $Combined(X) = TF-IDF(X) * weight + Assoc(X) * (1 - weight)$;
        {//linearly combine the two values}
        rank terms according to decreasing order of Combined(X);
        $P_c(X) = 100 - rank\ of\ Combined(X)$;
        {//rank = 1 indicated the top position, 2 indicated the second position, etc.}
    end for
end for
```

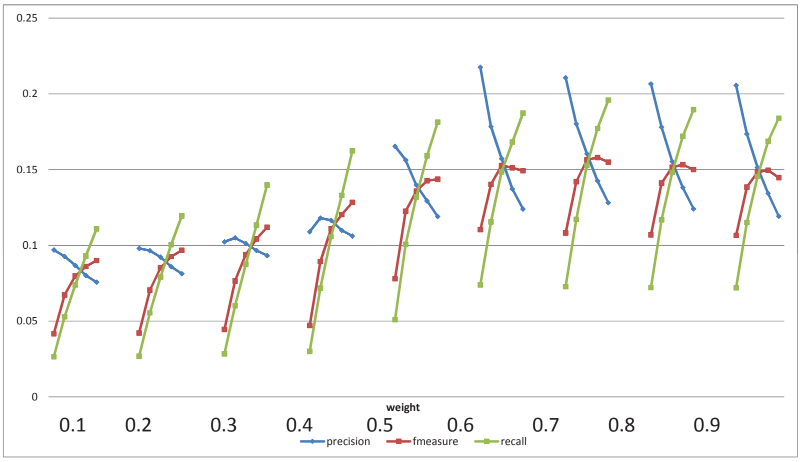


Fig. 1. Performance for different weight in bibtex_parsed. For every weight, no. of tags changes from 1 to 5.

Table 4. Performance for only using TF-IDF results, linearly combining results of TF-IDF & association rules, and common & combine the two results, for the top N tag recommendations

TF-IDF				linearly combining results				common & combine			
bibtex_original											
Top N	recall	precision	f-measure	Top N	recall	precision	f-measure	Top N	recall	precision	f-measure
1	0.0199	0.0636	0.0304	1	0.0339	0.1105	0.0519	1	0.0344	0.1123	0.0527
2	0.0378	0.0610	0.0467	2	0.0593	0.0979	0.0739	2	0.0619	0.1018	0.0770
3	0.0579	0.0603	0.0591	3	0.0824	0.0900	0.0860	3	0.0848	0.0927	0.0886
4	0.0787	0.0598	0.0680	4	0.1046	0.0849	0.0937	4	0.1065	0.0867	0.0956
5	0.0989	0.0592	0.0741	5	0.1244	0.0802	0.0975	5	0.1264	0.0816	0.0992
bibtex_parsed											
Top N	recall	precision	f-measure	Top N	recall	precision	f-measure	Top N	recall	precision	f-measure
1	0.0708	0.2033	0.1050	1	0.0728	0.2106	0.1081	1	0.0723	0.2155	0.1083
2	0.1138	0.1710	0.1367	2	0.1171	0.1802	0.1419	2	0.1212	0.1871	0.1471
3	0.1438	0.1487	0.1462	3	0.1527	0.1605	0.1565	3	0.1549	0.1635	0.1591
4	0.1665	0.1316	0.1470	4	0.1771	0.1425	0.1580	4	0.1778	0.1432	0.1586
5	0.1800	0.1158	0.1409	5	0.1959	0.1281	0.1549	5	0.1968	0.1291	0.1559
bookmark_more											
Top N	recall	precision	f-measure	Top N	recall	precision	f-measure	Top N	recall	precision	f-measure
1	0.0388	0.1285	0.0596	1	0.0449	0.1547	0.0696	1	0.0460	0.1599	0.0715
2	0.0693	0.1172	0.0871	2	0.0846	0.1400	0.1055	2	0.0872	0.1487	0.1099
3	0.0919	0.1080	0.0993	3	0.1133	0.1286	0.1205	3	0.1165	0.1358	0.1254
4	0.1077	0.1001	0.1038	4	0.1375	0.1202	0.1283	4	0.1415	0.1255	0.1330
5	0.1189	0.0943	0.1052	5	0.1581	0.1132	0.1319	5	0.1623	0.1172	0.1361

high $P_c(X)$ for recommendation. The total number of tags to recommend is controlled by k, the number of tags to check in the *common* step is *common-no, and the number of tags to extract in the combine step is* combine-no. Detailed steps are shown in Algorithm 4.

Since the evaluation of this contest only cares for the first 5 tags to recommend, we set $k = 5$. If *common-no = 10 and* combine-no = 5, the results for all three collections are shown in third column of Table 4.

Generally speaking, F-Measure increases with the increase of *common-no and reaches the peak near* common-no = 20. At the same time, it reaches its highest point as *combine-no increases, and remains the same level with the further increase of* combine-no. Since the total number of tags to recommend is fixed to be 5, the *combine* step will stop before it reaches the limit of how many tags to check, i.e., *combine-no. Thus if* combine-no is greater than a certain number, it won't affect the f-measure performance anymore.

Since the recommendations would be further modified by history results, we set $k = 80$, *common-no=10, and* combine-no=80 here.

If only recommending at most 5 tags, the F-Measure performance of all above methods, including only using TF-IDF, linearly combining results of TF-IDF & association rules, and *common & combine* the two, are shown in Figure 2. It is obvious that using association rules can greatly enhance the TF-IDF performance, either by linear combination or *common & combine. Common & combine* method is slightly better than linearly combining the two.

4.4 Checking Resource or User Match with History Records

In this section, historical information is used more directly. We performed 10-fold cross validation to report the performance in this section.

Algorithm 4 Common and Combine, to integrate results from TF-IDF & association rules

```
for all documents in the collection do
  $count = 0$;
  {// common step}
  for $i = 1$ to common-no do
    {// common-no is the parameter to tune. It controls how many terms to check in TF-IDF results.}
    extract term $X$ with TF-IDF rank = $i$;
    if $Assoc(X) > 0$ then
      recommend this term $X$;
      $count++$;
      {//count is the number of tags that have been recommended in common step.}
    end if
    $i++$;
  end for
  {// combine step}
  rank all terms by $P_c(X)$ in decreasing order.
  {// $P_c(X)$ is the combination value of keyword extraction and association rules results, calculated by Algorithm 3.}
  for $j = 1$ to $(k - count)$ do
    {// $k$ is the total number of tags to recommend, $k - count$ is the number of tags to recommend in combine step}
    extract the term $Y$ with (rank in $P_c(X)$ results) = $j$;
    if $Y$ is not in the recommendation list then
      recommend this term $Y$;
    end if
    $j++$;
    if $j >$ combine-no then
      {// combine-no is the parameter to tune. It controls how many terms to check in combined results of TF-IDF & association rules.}
      exit the combine step;
    end if
  end for
end for
```

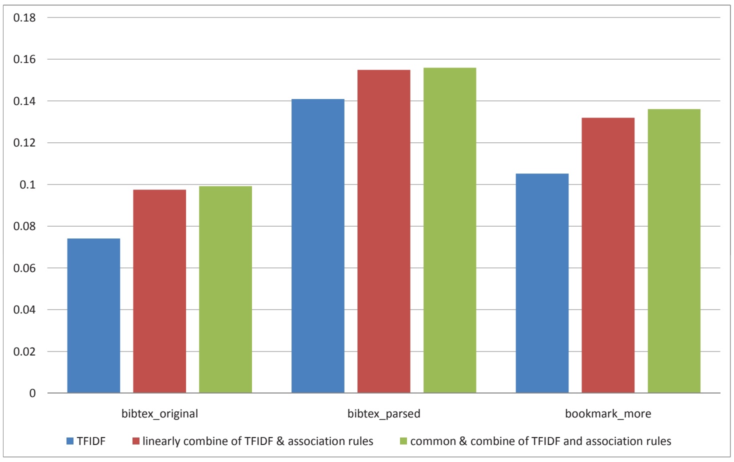


Fig. 2. F-Measure performance for different methods in all collections. For every method, at most 5 tags are recommended. The three methods to compare are only using TF-IDF, linearly combining results of TF-IDF & association rules, and common & combine

Resource match If the bookmark or bibtex in the testing dataset already appeared before in training dataset, regardless of which user assigned the tags, the tags that were assigned before would be directly inserted into our recommendation list for this document. These tags from historical information have higher priority than the tags that were recommended in previous steps.

User match Suppose the tags that are assigned by users previously in the training dataset, regardless of to which documents, make up the user's *tagging vocabulary*. Our assumption here is that every user prefers to use tags in his/her own tagging vocabulary, as long as the tags are relevant to the document. Thus the tags in the user's tagging vocabulary would be given higher priority. The *common and combine* algorithm is again applied here. In *common* step, if the terms with high rank in previous steps do appear in user's tagging vocabulary, then recommend this term. In *combine* step, extract terms with high ranks in previous steps to recommend. The number of tags to check in the *common* step is *common-no, and the number of tags to extract in the combine step is* combine-no.

The two parameters, *common-no and* combine-no, are tuned to achieve the best F-Measure performance when recommending at most 5 tags. *common-no is fixed to be 53 in Figure 3, while* combine-no increases from 1 to 5. In that figure, it shows that F-Measure increases and reaches the peak point at *combine-no* = *1. In Figure 4,* combine-no is fixed to be 1 and *common-no increases from 1 to 80. F-Measure increases with the initial increase of* common-no and reaches the peak point in the middle. In this work, we set *common-no* = *53, and* combine-no = 1.

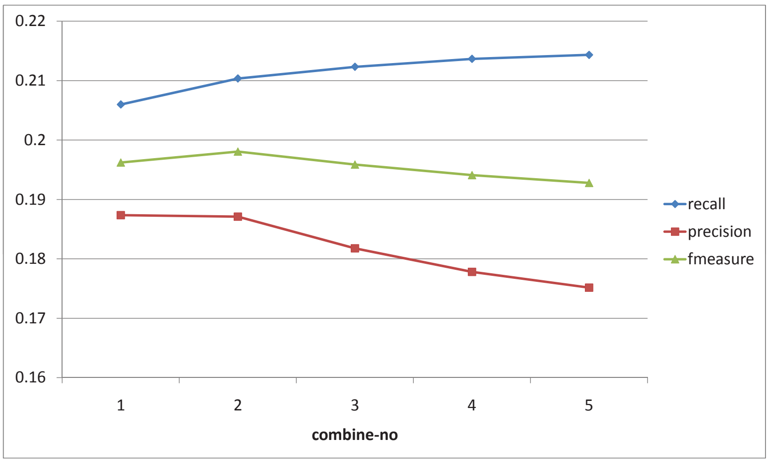


Fig. 3. In *common and combine* method for checking user match, performance for different *combine-no and fixed* common-no = 53. At most 5 tags are recommended.

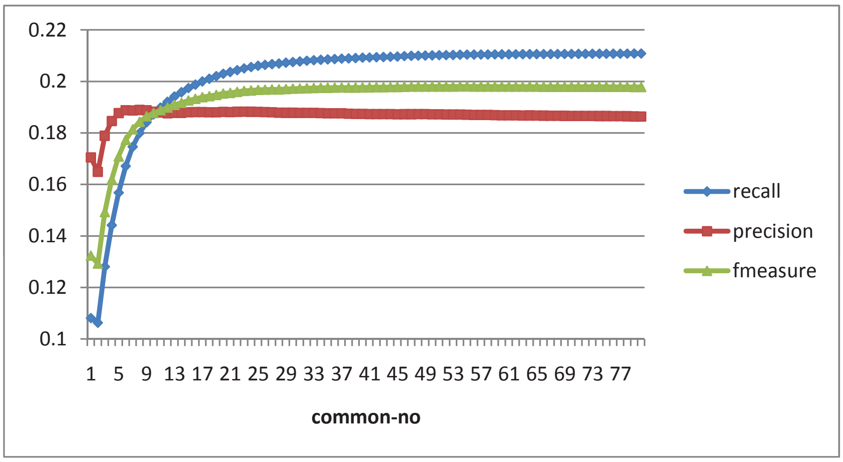


Fig. 4. In *common and combine* method for checking user match, performance for different *common-no and fixed* combine-no = 1. At most 5 tags are recommended.

Exact match with same user and same resource In this step, if user has tagged the same document in the training dataset, then the tags he used before for this document would be directly recommended again.

4.5 Combining Results in all Collections

According to the performance of each collection, our priority to combine the results is shown in Table 5.

Table 5. Priority to combine results

Priority	Method
Higher to lower	Tags from records that has exact match with same user and same bookmark/bibtex
	Tags from records that has match with same user
	Tags from records that has match with same resource (bookmark url or bibtex publication)
	common & combine results of *bibtex_parsed*
	common & combine results of *bibtex_original*
	common & combine results of *bookmark_more*

For example, if a record both exists in *bibtex_parsed* and *bibtex_original*, the results for this record are chosen from *bibtex_parsed* instead of *bibtex_original*, since the former one has higher priority.

If we only consider to combine the *common & combine* results for all three collections, the best performance is shown in column *without checking the history records* of Table 6.

Table 6. Performance for without checking history records, resource match with higher priority and user match with higher priority, for the top N tag recommendations

without checking history records				resource match higher				user match higher			
Top N	recall	precision	f-measure	Top N	recall	precision	f-measure	Top N	recall	precision	f-measure
1	0.0415	0.1395	0.0639	1	0.0835	0.2312	0.1226	1	0.0867	0.2396	0.1273
2	0.0783	0.1305	0.0979	2	0.1344	0.2143	0.1652	2	0.1374	0.2220	0.1698
3	0.1059	0.1204	0.1126	3	0.1667	0.1980	0.1810	3	0.1684	0.2064	0.1855
4	0.1292	0.1115	0.1197	4	0.1915	0.1866	0.1890	4	0.1916	0.1954	0.1935
5	0.1510	0.1046	0.1235	5	0.2118	0.1778	0.1933	5	0.2104	0.1871	0.1981

If step *Tags from records that match with same user* has lower priority than *tags from records that match with same resource*, the best result is shown in column *resource match higher* of Table 6. Otherwise, the best result is shown in column *user match higher* of Table 6. The results indicate that even for those bookmarks that were tagged by other users before, it is still beneficial to consider the target user's own tagging vocabulary.

To sum up, the best performance on training dataset is shown in Table 7, including the detailed results only for bookmark and bibtex.

5 Conclusions and Future Work

In this paper, we proposed a tag recommendation system using keywords in the page content and association rules from history records. If the record resource

Table 7. Best performance on training dataset, only for bookmarks and only for publications, for the top N tag recommendations

all resources				only for bookmark				only for bibtex			
Top N	recall	precision	f-measure	Top N	recall	precision	f-measure	Top N	recall	precision	f-measure
1	0.0867	0.2396	0.1273	1	0.0771	0.2364	0.1163	1	0.1025	0.2448	0.1445
2	0.1374	0.2220	0.1698	2	0.1296	0.2215	0.1635	2	0.1504	0.2228	0.1796
3	0.1684	0.2064	0.1855	3	0.1613	0.2056	0.1808	3	0.1803	0.2076	0.1930
4	0.1916	0.1954	0.1935	4	0.1843	0.1932	0.1887	4	0.2038	0.1990	0.2014
5	0.2104	0.1871	0.1981	5	0.2035	0.1841	0.1933	5	0.2218	0.1921	0.2059

or target user appeared before, the history tags would be used as references to recommend, in a more direct way. Our experiments showed that association rules could greatly improve the performance with only keyword extraction method, while history information could further enhance the F-Measure performance of our recommendation system.

In the future, other keyword extraction method can be implemented to compare with TF-IDF performance. In addition, graph-based methods could be combined with our recommendation approach to generate more appropriate tag recommendations.

Acknowledgments

This work was supported in part by a grant from the National Science Foundation under award IIS-0545875.

References

1. P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, New York, NY, USA, 2008. ACM.
2. I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.
3. M. Lipczak. Tag recommendation for folksonomies oriented towards individual users. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.
4. S. E. Robertson. Overview of the OKAPI projects. *Journal of Documentation*, 53(1):3–7, 1997.
5. M. Tatu, M. Srikanth, and T. D'Silva. Rsdc'08: Tag recommendations using bookmark content. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 96–107, 2008.

Understanding the user: Personomy translation for tag recommendation

Robert Wetzker[1], Alan Said[1], and Carsten Zimmermann[2]

[1] Technische Universität Berlin, Germany
[2] University of San Diego, USA

Abstract. This paper describes our approach to the challenge of graph-based tag recommendation in social bookmarking services. Along the ECML PKDD 2009 Discovery Challenge, we design a tag recommender that accurately predicts the tagging behavior of users within the Bibsonomy bookmarking service. We find that the tagging vocabularies among folksonomy users differ radically due to multilingual aspects as well as heterogeneous tagging habits. Our model overcomes the prediction problem resulting from these heterogeneities by translating user vocabularies, so called *personomies*, to the global folksonomy vocabulary and vice versa. Furthermore we combine our user-centric translation approach with item-centric methods to achieve more accurate solutions. Since our method is purely graph-based, it can also readily be applied to other folksonomies.

1 Introduction

Over the last years, social bookmarking services, such as Delicious[3], Bibsonomy[4] and CiteULike[5], have grown rapidly in terms of usage and perceived value. One distinguishing feature provided by these services is the concept of tagging - the labeling of content with freely chosen keywords (tags). Tagging enables users to describe and categorize resources in order to organize their bookmark collections and ease later retrieval. Social bookmarking services are therefore the classic example of collaborative tagging communities, so called *folksonomies* [1][2]. The consumer-centric (collaborative) tagging aspect differentiates social bookmarking from other content sharing community services, such as Flickr[6] or YouTube[7], where tags are generally assigned by the content creator [3].

Most folksonomy solutions assist users during the bookmarking process by recommending tags. Thus, a user can select recommended tags from various sets in addition to entering tags manually. Despite their positive effect on usability, these recommenders are effective tools to limit tag divergence within

[3] http://delicious.com
[4] http://bibsonomy.org
[5] http://citeulike.org
[6] http://flickr.com
[7] http://www.youtube.com

folksonomies as they are generally considered to lower the ratio of misspellings and to increase the likelihood of tag reassignments. The design of a folksonomy tag recommender was one of the tasks of the ECML PKDD 2009 Discovery Challenge[8]. In the following sections, we describe our solution to this task as submitted.

Our approach is based on the observation that the tag vocabularies of users, their personomies, differ within a folksonomy. This heterogeneity is mainly caused by differences in the tags users constantly assign to categorize content and the multilingualism of the user base, as apparent for Bibsonomy. To overcome the problems caused by this heterogeneity, we propose a tagging model that translates the personomy of each user to the folksonomy vocabulary and vice versa. We find that our model is highly accurate as it characterizes an item by its tag spectrum before translating this spectrum to a user's personomy. We then combine the translational model with item-centric tag models to improve performance.

This paper is structured as follows: The introductory section presents a graph model for the underlying data structure and explains the actual goals of the challenge. This is followed by an analysis of different properties of the Bibsonomy dataset with respect to their impact on tag recommendation. Section 3 introduces and discusses the tag vocabularies found within folksonomies, before we present our recommendation algorithm and evaluation results.

1.1 Modeling folksonomies

According to [2], a folksonomy can be described as a tuple $F := (I, T, U, Y)$, where $I = \{i_1, \ldots, i_k\}$, $T = \{t_1, \ldots, t_l\}$ and $U = \{u_1, \ldots, u_m\}$ are finite sets of items, tags and users, and Y is a ternary relation whose elements are called *tag assignments*. A *tag assignment* (TAS) is defined by the authors as relation $Y \subseteq U \times T \times I$, so that the tripartite folksonomy hypergraph is given by $G = (V, E)$, where $V = I \cup T \cup U$ and $E = \{(i, t, u) | (i, t, u) \in Y\}$. The set of all bookmarks is then given as $BM = \{(i, u) | \exists t : (i, t, u) \in Y\}$. This graph structure is characteristic for all folksonomies.

1.2 The challenge: Graph-based tag recommendations (Task 2)

The ECML PKDD 2009 Discovery Challenge consists of different tasks related to the problem of tag recommendation. Testbeds for all tasks are different snapshots of the Bibsonomy bookmarking service. Our solution contributes to the task of "Graph-Based Recommendations (Task 2)“ as it does not consider the content of the given resources. The recommendation task in this setting resembles the problem of link prediction within G given a user and an item node. The recommender thus needs to estimate $P(t|i, u)$, the probability of observing a given tag when a combination of user and item has been observed.

[8] http://www.kde.cs.uni-kassel.de/ws/dc09

1.3 Related work

One of the first works on tag recommendation in folksonomies is [4], where the authors compare the performance of co-occurrence based tag recommenders and more complex recommenders based on the FolkRank algorithm [2]. Furthermore they report only minor improvements of the FolkRank method over the co-occurrence approaches. Parts of their analysis is performed on a snapshot of the Bibsonomy dataset.

Further related work was presented by the participants of last years challenge on tag recommendation. The authors of [5] enrich tag vocabularies by terms extracted from all bookmarks' meta data, such as user given descriptions or the abstract, author, year etc. information in case of publications. More similar to our work is the approach in [6]. The team combines the keywords found within a resource's title and the actual tags previously assigned to a resource with the tags from a user's personomy. Each vocabulary is mapped to the global tag vocabulary using co-occurrence tables. This mapping is similar to our translation process. However, the fusion of different sources differs from our method and no user-centric optimization of parameters was reported.

2 The dataset

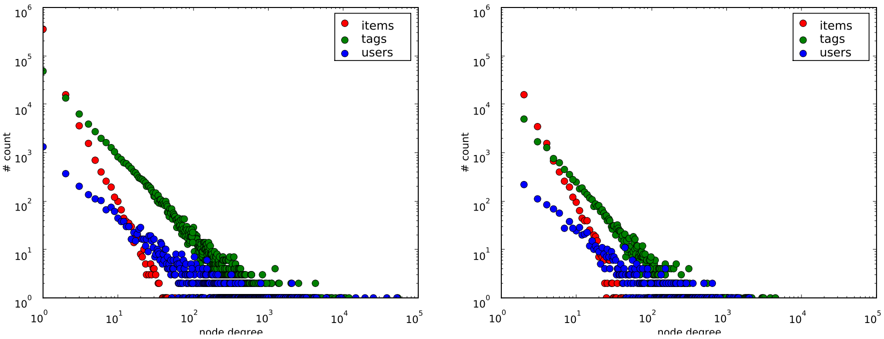
The Bibsonomy bookmarking service allows its users to bookmark URLs and publications in parallel. This hybrid approach makes Bibsonomy different from other bookmarking communities such as Delicious or CiteULike. For each web bookmark, participants are given the URL, the title and an optional description of the resource as provided by the user during the bookmarking process. Bookmarked publications generally come with information about the title, the authors, the abstract or other common bibliographic attributes. The completeness of this information is not guaranteed, and many attribute fields are left empty. Table 1 gives an overview of the node statistics found within the dataset for p-core levels one and two[9].

Table 1. Different node set sizes of the Bibsonomy dataset for p-core levels 1 and 2.

p-core	$\|E\|$	$\|BM\|$	$\|BM_{BIB}\|$	$\|BM_{URL}\|$	$\|I\|$	$\|T\|$	$\|U\|$
1	1,401,104	421,928	263,004	158,924	378,378	93,756	3,617
2	253,615	64,120	41,268	22,852	22,389	13,276	1,185

For the construction of our recommender we ignore the meta-data attached to the bookmarked content and only consider the graph G. Furthermore, we do not distinguish between URLs and publications, but merge both node sets to the item set I. Both decisions result in a loss of information which potentially

[9] The p-core of a folksonomy graph has the characteristic that all contained nodes appear in at least p bookmarks. See [4] for details.



(a) 1-core

(b) 2-core

Fig. 1. Node degree distributions within the full Bibsonomy dataset (a) and the dataset at p-core level 2. Node degree distributions found within folksonomies generally exhibit power law characteristics with few highly connected nodes and many nodes with low occurrence. This characteristic is obtained when cutting the graph to its level 2-core.

reduces the overall accuracy of our recommender. However, we believe this loss to be compensated by the general applicability of our approach and the improved validity of the presented results.

Figure 1 shows the node degree distributions of the full dataset and its 2-core. As previously reported for other folksonomies (e.g. [2][7]), we find the node degree distributions of Bibsonomy to exhibit power law characteristics, with very few and very frequent nodes on one end and many infrequent nodes on the other. This characteristic is basically obtained when reducing G to its 2-core. However, we find that this reduction drastically reduces the impact of some previously very influential users. These users tend to have a rather ”organizational“ background and bookmark large collections of similar resources which are of no interest to other users. Most of these resources therefore fall victim to the p-core reduction, which also explains the drastic cut on items in Table 1.

3 Tag vocabularies

As users bookmark items only once[10], we cannot estimate $P(t|i,u)$ directly from previous observations. Instead, we base our estimates on other distributions. The most basic sources are the overall tag distribution and the tags previously assigned by the user, or to the item.

Global tag distributions. This is the distribution of all observed tag assignments within the training data. By definition, each tag of the test data has

[10] Note that a small percentage of user item combinations found in the given dataset occur in more than one bookmark.

to occur within this distribution. A recommender that only considers the global tag distribution would assume $P(t|u,i) \approx P(t)$. We will refer to such a recommender as *MostPopular* recommender.

Item tag distributions. This is the distribution of previously assigned tags for a given item. It was shown that the tag distributions of items converge to a characteristic tag spectrum over time. Furthermore, as reported by [8] and [9], the resulting tag distributions often follow a power law with few tags being assigned very frequently and most tags occurring in the long tail. If we neglect the personalization aspect of tagging, we can recommend tags by assuming $P(t|i,u) \approx P(t|i)$. However, our observation of $P(t|i)$ within the training data may be limited, i.e. information about most tags will be missing.

User tag distributions (Personomies). Each user develops his own vocabulary of tags over time called his *personomy.* Users will generally be interested in reassigning previously used tags as this will simplify content search later on. The interest in tag convergence often results in the frequent assignment of a limited number of category tags, and it was shown that user vocabularies develop power law characteristics over time [10]. A personomy based recommender would assume $P(t|i,u) \approx P(t|u)$. Once again, the distribution $P(t|u)$ estimated over the training data is likely to miss a variety of tags especially for users with few bookmarks.

The authors of [4] report that a tag model which combines user and item tag distributions into a unified distribution achieves sufficient recommendation accuracy. We will consider a hybrid recommder with $P(t|i,u) \approx \alpha P(t|i) + (1-\alpha)P(t|u)$ as an additional baseline approach during our evaluations (*MostPopular2d*).

4 Our approach

The design of our tag recommender is based on two intuitive assumptions:

1. **Tags are personalized.** Different users will assign different tags to the same item. This effect cannot only be explained by statistical variance. Instead, we find users developing their own category tags over time. One of the implications for the tag recommendation task is the problem of recommending the right version of a tag, especially in cases where synonymous tags exist. This includes different spellings, such as "web20“ versus "web2.0“. Even though semantically equal, these will be different for a user who assigns tags for content categorization. Furthermore, especially in multilingual folksonomies, we find that users often assign keywords from their mother tongue. This is of particular importance for Bibsonomy, where many users seem to come from Germany, with the effect that the tag distribution is a mixture of German and English words. Whereas some users tagged a site as "searchengine“ related we also find the German translation ("suchmaschine“) among the frequent tags.

2. **Tags describe items.** The authors of [11] report that the vast majority of assigned tags on Delicious identify the topic, the type or the owner of a URL. We can therefore assume that users will assign personomy tags depending on the item. This assumption is a simplification as it excludes tags, such as "toread“ or "self“, which actually refer to the user item relationship instead of the item itself. However, we believe that these tags cannot be easily predicted based on the given training data but require deeper knowledge about the user. Luckily, as reported by [12] for Delicious, *usage context* and *self reference* tags are relatively scarce compared to descriptive tags.

These assumptions directly influenced the basic design decisions for our recommender which suggests tags from a user's personomy with respect to the community opinion about the underlying item.

4.1 Translating personomies

We assume that each user has a distinctive vocabulary of tags. These tags can be translated to the community tag vocabulary by looking at co-occurrences within the shared item space. We are thus interested in the probability $P(t^u|t,u)$ that a user will assign a tag t^u from his personomy as next tag given that the item was tagged as t by another user. Based on previous knowledge about the users tagging behavior we can estimate $P(t^u|t,u)$ as

$$P(t^u|t,u) = \sum_{i \in I_u} P(t^u|i,u)P(i|t), \tag{1}$$

where I_u is the set of items previously bookmarked by the user. Based on $P(t^u|t,u)$ we can now translate the global folksonomy language to the personomy of a user.

For the recommendation task we are interested in the probability $P(t^u|i,u)$ for previously unseen user item combinations. For a new item with an observed tag vocabulary of $P(t|i)$, the probability that a user will assign one of his tags next is given as

$$P(t^u|i,u) = \sum_{t \in T_i} P(t^u|t,u)P(t|i). \tag{2}$$

Note that $\sum_{t^u \in T_u} P(t^u|i,u) = 1$ is only true if all tags in T_i have a mapping in T_u which is rather unlikely. Instead we expect $\sum_{t^u \in T_u} P(t^u|i,u)$ to decrease the more the given item deviates from the items previously bookmarked by the user.

4.2 The tag recommender

The tag recommender we propose, selects tags coming from three sources: the personomy of a user where tags are weighted by $P(t^u|i,u)$, the item vocabulary ($P(t|i)$) and the global vocabulary ($P(t)$). Including the item vocabulary is important, as many tags may be item specific and are thus unlikely contained within the personomy. We also include the global tag distribution $P(t)$ for cases where little is known about user and item alike. We assume a weighted linear combination of sources and estimate $P(t|i, u)$ as

$$P(t|i, u) \approx \alpha_u P(t^u|i, u) + \alpha_i P(t|i) + (1 - \alpha_u - \alpha_i)P(t), \tag{3}$$

with $0 \leq \alpha_u, \alpha_i, \alpha_u + \alpha_i \leq 1$. We then recommend the N tags with highest probability $P(t|i, u)$, where N is a parameter that needs to be optimized together with α_u and α_i. Optimization can take place on a global or a user-centric scale.

Global optimization. For the globally optimized model, we assume equal parameter settings for all users. As the Bibsonomy dataset is rather small, we can use a brute-force approach to find the combination of N, α_u and α_i that maximizes the F-measure. We do so by performing a 10-fold cross-validation on the training data. We call this user-centric tag recommender with global optimization UC_G.

User-centric optimization. As users are heterogeneous, it is not intuitive to assume shared parameter preferences. Instead, it seems straightforward to optimize parameters for each user separately. Once again, we do so by performing a cross-validation on the training data. We then use a brute-force approach to find the combination of N, α_u and α_i that maximizes the F-measure of each user averaged over all folds. We will refer to the user-centric tag recommender with local optimization as UC_L.

5 Evaluation

We trained the models of all recommender types on the 2-core version of the dataset. The parameters of the *MostPopular2d*, UC_G and the UC_L models were fine-tuned in a 10-fold cross-validation as described above. For the *MostPopular2d* recommender we found an α value of 0.5 to perform best. For the user-centric tag recommender with global optimization the maximal F_1 measure was achieved when setting $\alpha_u = 0.6$ and $\alpha_i = 0.4$. The weight of the global tag distribution thus resulted 0 which means that including the global vocabulary did not yield performance gains. For the *MostPopular2d* as well as the UC_G recommender the best number of tags to recommend was 5. Evaluating the accuracy of all recommender types during the cross validation, we found the user-centric tag recommender with local optimization (UC_L) to constantly outperform all other versions. We therefore submitted the predicted tags of the UC_L approach as our solution to the challenge.

The released test dataset consists of 778 bookmarks from 136 users linking to 667 items. Table 2 presents the achieved F_1 measures on the first five ranks for the various recommender types. The UC_L recommender we submitted achieved a

Table 2. Performance of various recommender types on the test data. Underlined values represent the configuration that performed best during training. The submitted recommender (UC_L) achieved an F_1 measure of 0.314. The best F_1 measure could have been achieved with a UC_G recommender always suggesting 3 tags (bold).

Recommender	F_1@1	F_1@2	F_1@3	F_1@4	F_1@5
$MostPopular$	0.021	0.038	0.051	0.051	0.059
$MostPopular2d_{\alpha=0.5}$	0.229	0.286	0.306	0.313	0.310
$UC_{G,\alpha_u=0.6,\alpha_i=0.4}$	0.246	0.326	**0.335**	0.334	0.330
UC_L	0.230	0.294	0.306	0.311	0.314

performance of 0.314. This result is somewhat disappointing as it is only slightly above the result of the simpler $MostPopular2d$ recommender. However, we find that the approach of vocabulary translation is generally superior as the results of the UC_G recommender are significantly better. We observe similar performance patterns when looking at the precision/recall curves plotted in Figure 2.

Investigating the reasons for the weak performance of the UC_L recommender, we find that the user distribution of the test set deviates from the trained one as shown in Figure 3. This deviation is likely to have a negative impact on the prediction quality as parameters have been tuned in expectation of a user distribution similar to the one of the training set. However, this problem is not

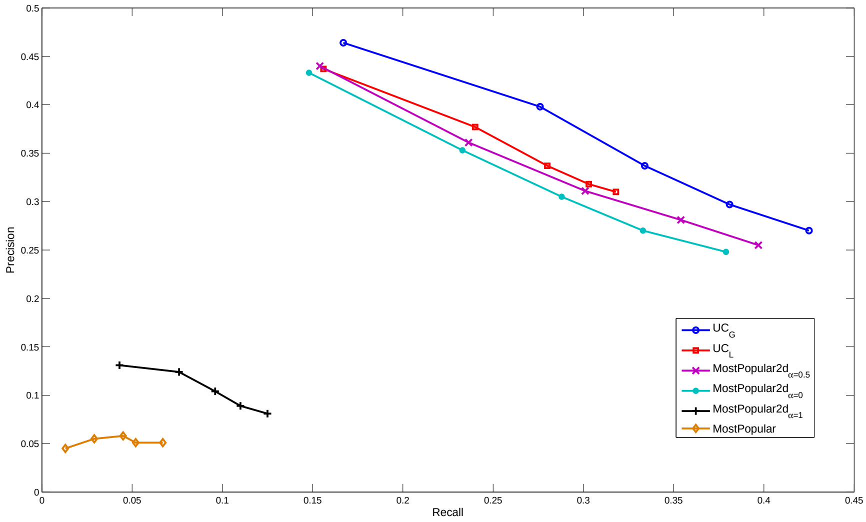


Fig. 2. Precision/Recall curves for various recommenders on the provided test data. The curve of the UC_L recommender appears "shorter" as this recommender suggests a variable number of tags.

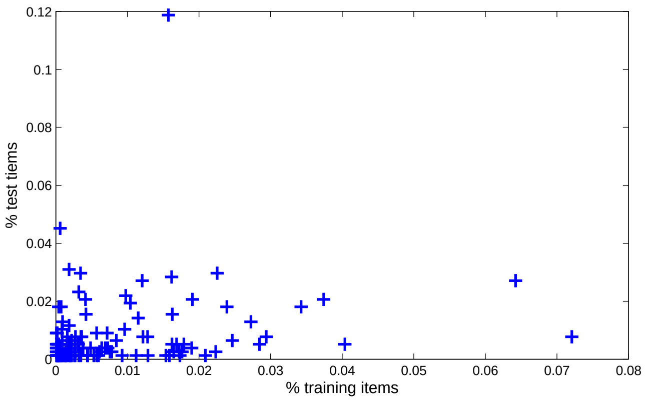


Fig. 3. Relative number of bookmarks per user in test and training data. The correlation between the user participation in the test set and the trained distribution is rather low ($\rho = 0.24$).

unique to the UC_L recommender but is expected to have a negative impact on the performance of all recommender types. Instead, we believe that the weak performance of the UC_L recommender is caused by an inadequate parameter tuning for users less present in the training data but frequent in the test set. Tuning α_u, α_i and N for these users often results in bad estimates due to missing data. Whereas the implications of these shortcomings are rather minor when users are distributed as in the training data, they seem to become major for the test distribution.

The fact that the test set is dominated by users with rather small training vocabularies is also reflected by the performance of the $MostPopular2d$ recommenders with α set to 0 and 1 as shown in Figure 2. Here, we find that a recommender which only suggest tags from a user's personomy ($\alpha = 1$) performs very bad, whereas an item based recommender ($\alpha = 0$) achieves nearly as good results as the mixture model $\alpha = 0.5$. This implies, that most tags of the test data are not present within a user's personomy at training time or, less likely, that the tagging behavior of users drastically changed in the test phase.

The inadequate modeling of infrequent users (and items) is an expected shortcoming of a purely graph-based recommendation approach. This is especially true for our personomy translation approach which requires the tags of the given item to have a mapping within the conditional distribution $P(t^u|t, u)$ (see equation 2). Incorporating the provided item meta-data may be a promising alternative to improve accuracy in scenarios where little is known about users and items from a graph perspective.

6 Conclusions

In this paper, we presented a novel approach to the challenge of graph-based tag recommendation in folksonomies. Building on the assumption that all users of a folksonomy have their own tag vocabulary, our approach translates the personomies of users to the global folksonomy vocabulary. Evaluation results show that this translation helps to significantly improve tag prediction performance. Furthermore, we fine-tuned our model by estimating parameters on a per user basis. Even though this user-centric approach performed rather disappointing during the challenge, we believe that user-level optimization will be essential for the success of future (tag) recommenders.

References

1. Wal, T.V.: Folksonomy coinage and definition, February 2, 2007, http://www.vanderwal.net/folksonomy.html.
2. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information retrieval in folksonomies: Search and ranking. In: Proc. of the 3rd European Semantic Web Conference (ESWC), Springer (2006)
3. Marlow, C., Naaman, M., Boyd, D., Davis, M.: Ht06, tagging paper, taxonomy, flickr, academic article, to read. In: HYPERTEXT '06, New York, NY, USA, ACM (2006)
4. Jäschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Lernen - Wissensentdeckung - Adaptivität (LWA) Workshop Proc. (2007)
5. Tatu, M., Srikanth, M., D'Silva, T.: Rsdc'08: Tag recommendations using bookmark content. In: Proc. of the ECML PKDD Discovery Challenge (RSDC08). (2008)
6. Lipczak, M.: Tag recommendation for folksonomies oriented towards individual users. In: Proc. of the ECML PKDD Discovery Challenge (RSDC08). (2008)
7. Heymann, Paul Garcia-Molina, H.: Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical report, Computer Science Department, Stanford University (2006)
8. Cattuto, C., Loreto, V., Pietronero, L.: Collaborative tagging and semiotic dynamics. PNAS **104** (2007)
9. Halpin, H., Robu, V., Shepherd, H.: The complex dynamics of collaborative tagging. In: WWW '07: Proc. of the 16th int. conf. on World Wide Web, New York, NY, USA, ACM (2007)
10. Zhang, D., Mao, R., Li, W.: The recurrence dynamics of social tagging. In: WWW '09: Proc. of the 18th int. conf. on World wide web, New York, NY, USA, ACM (2009)
11. Golder, S.A., Huberman, B.A.: Usage patterns of collaborative tagging systems. J. of Information Science **32**(2) (2006)
12. Bischoff, K., Firan, C.S., Nejdl, W., Paiu, R.: Can all tags be used for search? In: CIKM '08: Proc. of the 17th ACM conf. on Information and knowledge management, New York, NY, USA, ACM (2008)

A Tag Recommendation System based on contents

Ning Zhang, Yuan Zhang, and Jie Tang

Knowledge Engineering Group
Department of Computer Science and Technology, Tsinghua University, Beijing, China
zntsinghua1117@gmail.com, fancyzy0526@gmail.com and
jietang@tsinghua.edu.cn

Abstract. Social bookmarking tools become more and more popular nowadays and tagging is used to organize information and allow users to recall or search the resources. Users need to type the tags whenever they post a resource, so that a good tag recommendation system can ease the process of finding some useful and relevant keywords for users. Researchers have made lots of relevant work for recommendation system, but those traditional collaborative systems do not fit to our tag recommendation. In this paper, we present two different methods: a simple language model and an adaption of topic model. We evaluate and compare these two approaches and show that a combination of these two methods will perform better results for the task one of PKDD Challenge 2009.

1 Introduction

With the event of social resource sharing tools, such as BibSonomy[1], Flickr[2], del.icio.us[3], tagging has become a popular way to organize the information and help users to find other users with similar interests and useful information within a given category. Tags posted by a user are not only relevant to the content of the bookmark but also to the certain user. According to [3], the collection of a user’s tag assignments is his/her *personomy*, and *folksonomy* consists of collections of personomies. From the available training data, we can find that some tags might just be words extracted from the title, some tags might be the concept or main topic of the resource, and other might be very specific to a user.(see Table 1). The last three lines of the table show that the user 293 post tags like swss0603, swss0609, swss0602, which are very specific to the user. Since the test data for task one contains posts whose user, resource or tags are not in the training data, some traditional collaborative recommendation systems might not perform well. It is because most of the collaborative recommendation systems cannot recommend tags which are not in the tag set of the training data. This paper presents our tag recommendation system,

[1] http://www.bibsonomy.org
[2] http://www.flick.com
[3] http://del.icio.us

which is a combination of two methods: simple Language model and an adaption of topic model according to [7].

USER	TITLE	TAGS	SOURCE OF TAGS
37	SVG: Adobe	adobe, svg	title
787	SourceForge.net: delicious-java	api java, delicious	title
173	Reassessing Working Memory: Comment on Just and Carpenter (1992) and Waters and Caplan (1996)	psycholinguistics, review, workingmemory	concept or topic
293	A Semantic Web Primer	swss0603, ontolex2006, semwebss06, swss0602	specific to the user
293	The ABCDE Format Enabling Semantic Conference Proceedings	semwiki2006,swikig,wiki, eswc2006, semantic	specific to the user
293	Learning of Ontologies for the Web: the Analysis of Existent Approaches	ontologylearning, semanticweb,semwebss06, sw0809, sw080912, swss0609	specific to the user

Table 1: Examples of tag sources

The first method can extract some keywords from the description and other information of the post and they constitute a candidate set. Then we use the relevance of a word and a document to score the words in the candidate set and recommend the words with highest scores. The second method uses an ACT model [7], and it can get some conceptual or topic knowledge of the post. Given a test post, the model can score all the tags which have been posted previously and recommend the tags with highest scores.

These two methods focus on two different aspects. The first method will probably recommend some keywords extracted from the title while the second method uses a probabilistic latent semantic method to recommend tags which are similar to the post in terms of conceptual knowledge. Comparing these two methods, we can find that the tags recommended are always different. Consequently, the combination is a intuitive better way.

This paper is organized as follows: Section 2 reviews recent development in the area of social bookmark tag recommendation systems. Section 3 describes our proposed system and the combination method in details. In section 4, we present and evaluate our experimental results on the test data of ECML PKDD challenge and conclude the results in section 5.

2 Related work

The recent rise of Web 2.0 technologies has aroused the interest of many researchers to the tag recommendation system. Some approaches are based on collaborative information. For example, AutoTag[9] and TagAssist[8] use some information retrieval skills to recommend tags for weblog posts. They recommend tags based on the tags posted to the similar weblogs and they cannot recommend new tags which are not in the training file. FolkRank[4,5], which is an adaption of the famous PageRank algorithm, is a graph-based recommendation system. Also, it cannot recommend new tags not in the training file. The experimental results of FolkRank in [5] reveal that the FolkRank can outperform other collaborative methods. But to some extents FolkRank relies on a dense core of training file and it might not be fit to our task.

All the methods mentioned above are based on collaborative information and similarity between users and resources. However, in the cases when there are many new users and resources in the test data (our task one), those methods cannot perform well. In the RSDC '08 challenge, the participants[1,2] who use methods based on words extracted from the title or semantic knowledge and user's personomy can outperform other methods. Consequently, we propose our tag recommendation system mainly based on the contents.

3 Our Tag Recommendation System

3.1 Notations

First, we define notations used in this paper. We group the data in bookmark by its url_hash and data in bibtex by its simhash1. If some posts in bookmark or bibtex file have the same url_hash or simhash1, they are mapped to one resource r. In bookmark, we extract description, extended description while in bibtex, we extract journal, booktitle, description, bibtexAbstract, title and author. We define these information as the description of resource r. For each resource r and each user u who has posted tags to resource r, assuming that its description contains a vector $\mathbf{w}_d$ of N_d words; a vector $\mathbf{t}_d$ of T_d tags posted to this resource r by the user u_d.Then the training dataset can be represented as

$$D = \{(w_1, t_1, u_1), ..., (w_D, t_D, u_D)\}$$

Table 2 summarizes the notations.

3.2 Language Model

Language model is widely used in natural language processing applications such as speech recognition, machine translation and information retrieval. In our model, first we pick some words to form a candidate set of recommended tags and then score all the words in the candidate set and recommend words with highest scores for our tag

Symbol	Description
T	the collection of tags posted in the training data
R	the collection of resources posted in the training data (grouped by the url_hash or simhash1)
U	the collection of users who posted tags in the training data
D	training data set containing tagged resources. D={($\mathbf{w}_i$, $\mathbf{t}_i$,u_i,)}, which represents a set of pairs of resources and users, with the assigned tags by the corresponding users.
D’	The test data set containing resources and users. D’={(r_j, u_i)}$_{\{i,j\}}$. Note that: 1) either the user u_i or the resource r_j may not appear in the training data set.
N_d	number of word tokens in the $d \in D$
T_d	number of tags posted by user u to resource r in $d \in D$
$\mathbf{w}_d$	vector form of word tokens in $d \in D$
$\mathbf{t}_d$	vector form of tags in $d \in D$
u_d	the user in $d \in D$
$\tilde{T}(u,r)$	the set of tags that will be recommended for a given user u and a given resource r
z	hidden topic layer in ACT model

Table 2: Notations.

recommendation system. The candidate set C is composed with two subset, C_1 and C_2, i.e. $C = C_1 \cup C_2$.

We extract useful words from the description of the active resource r^* in the test data. Then we remove all the characters which are neither numbers nor letters and get rid of the stop words in the English dictionary. The rest words form the part of the candidate set C_1. For each $t_1 \in C_1$, we have the following generative probability:

$$P_1(t_1 \mid r^*) = \frac{N_d}{N_d + \lambda} \cdot \frac{tf(t_1, d)}{N_d} + (1 - \frac{N_d}{N_d + \lambda}) \cdot \frac{tf(t_1, D')}{N_{D'}} \qquad (1)$$

where N_d is the number of word tokens in the description d of r^*, tf(t_1,d) is the word frequency(i.e., occurring number) of word t_1 in the description of d of r^*, $N_{D'}$ is the number of word tokens in the entire test dataset, and tf(t_1,D’) is the word frequency of word t_1 in the collection D’. λ is the Dirichlet smoothing factor and is commonly set according to the average document length, i.e. $N_{D'}/|D'|$ in our cases.

As for C_2, in order to get more information about the new resource, we take the similarity between resources into consideration and add tags previously posted to the similar resource into C_2. The similarity of resource is determined by the url of the resource. Each url can be split into several sections, for example, ‘http://www.kde.cs.uni-kassel.de/ws/dc09’ will be split into three sections: ‘www.kde.cs.uni-kassel.de’, ‘ws’ and ‘dc09’. The similarity between r_1 and r_2 is defined as follows, sim(r_1,r_2) = 2^(number of the identical sections of url_1 and url_2 – maximum number of sections of url_1 and url_2). For each resource r, we will choose three most similar urls to the url of r and their corresponding resources form the neighbor of the resource r, noted as neigh(r). C_2 is compose of the tags previous posted to the neigh(r^*). For each $t_2 \in C_2$, we have the following generative probability:

$$P_2(t_2 \mid r^*) = \sum_{r \in neigh(r^*)} \left[(\frac{T_r}{T_r + \lambda} \cdot \frac{n(t_2, r)}{T_r} + (1 - \frac{T_r}{T_r + \lambda}) \cdot \frac{n(t_2, R)}{|T|}) \cdot sim(r, r^*) \right] \quad (2)$$

where $n(t_2,r)$ is the number of times that tag t_2 has been posted to the resource r and $n(t_2,R)$ is the number of times that tag t_2 has been posted in the training data. T_r is the number of tags posted to the resource r and λ is the Dirichlet smoothing factor and is commonly set according to the average document length, i.e. |T|/|R|

Now, we have the definition of the candidate set $C = C_1 \cup C_2$ and for an active resource r^*, we have $P_1(t_1|r^*)$ for $t_1 \in C_1$ and $P_2(t_2|r^*)$ for $t_2 \in C_2$. Then the set of recommended tags will be: $\widetilde{T}(u, r) \coloneqq \text{argmax}^n_{t \in C} (P_1(t|r^*) + P_2(t|r^*))$ where n is the number of recommended tags.

3.3 ACT model

The model we use is called Author-Conference-Topic (ACT) model[7], which is an adaptation of topic model. The initial model was used to simultaneously modeling papers, authors, and publication venues within a unified probabilistic topic model. The model utilizes the topic distribution to represent the inter-dependencies among authors, papers and publication venues. In our task, for each $(\mathbf{w}_d, \mathbf{t}_d, u_d) \in D$, we map the $\mathbf{w}_d$ to conference information, map $\mathbf{t}_d$ to author of the paper, map u_d to the publication venue. Consequently, after modeling, we can get probabilistic relations among description, tags and user given a post. In details, we can get these inter-dependencies: P(z|t), P(w|z) and P(u|z), where z is a hidden topic layer in the topic model. From this model, we can utilize the hidden topic layer to learn some conceptual knowledge of the post. The number of topic z can be set manually.

After obtaining the probability of P(z|t), P(w|z) and P(u|z) from the model, we can derive that for each word $w \in \mathbf{w}_d$, each tag $t \in \mathbf{t}_d$, we have:

$$P(w \mid t) = \sum_z P(w \mid z) P(z \mid t)$$
$$P(u \mid t) = \sum_z P(u \mid z) P(z \mid t)$$

To recommend tags for a given user u’ and a given resource r’, we will score all the tags $t \in T$ using probabilistic methods as follows:

$$P(t \mid u', r') \propto P(t, u' r') = P(t) P(u' \mid t) P(r' \mid t) \approx P(t) P(u' \mid t) \prod_{w \in r'} P(w \mid t) \quad (3)$$

There are two things that needed to be mentioned here. First, if the given user $u' \notin U$, which means the given user is a new user. Then we cannot obtain P(u’|t) in the equation (3), in that case we will set the value to 1 and the equation (3) will become:

$$P(t) \prod_{w \in r'} P(w \mid t) \approx P(t) P(r' \mid t) \propto P(t \mid r') \quad (4)$$

Because the user is a new one, so that we have nothing about his/her history of posting tags in our training data, in that case equation (4) is reasonable. Secondly, not all the words in the given resource are in our training data, so when calculating equation (3), we will ignore the word which has not appeared in the training data.

The set of recommended tags for a given user u' and a given resource r' will be: $\widetilde{T}(u', r') := \mathrm{argmax}_{t\in T}^{n} P(t|u', r')$ where n is the number of recommended tags, T is the collection of tags posted in the training file.

3.4 Combination

We have proposed two different methods to recommend tags, model one focuses on the useful words extracted from the description or title of the resource while the second model focuses on the conceptual knowledge and probabilistic relations among tags, resource and users. We are interested in the following problem: **Can we combine these two models to perform a better result for tag recommendation?**

```
Input: a given resource r and a given user u and the result of ATC model P(t),P(w|t)
       and P(u|t) for all tags, users, words in the training file.
Output: $\widetilde{T}(u, r)$the set of recommended tags
begin
       //Model one
       $\mathbf{w_1}$ ← words in the candidate set C
       foreach $w \in \mathbf{w_1}$ do
           $score1[w] = P_1(w|r) + P_2(w|r)$
       end
       max_score1 ← $\max_{w\in w_1}$score1[w]
       //Model two
       foreach $t \in T$ do
           $score2[t] = P(t)P(u|t)\prod_{w\in r} P(w|t)$
       end
       max_score2 ← $\max_{t\in T}$score2[t]
       //Combination
       foreach $t \in \mathbf{w_1} \cup T$ do
           score[t] = score1[t]+score2[t]*max_score1/max_score2
       end
       $\widetilde{T}(u, r) := \mathrm{argmax}_{t\in T}^{n}$score[t]
end
```

Algorithm 1: The combined tag recommendation system

We have tried some different approaches to combine these two models. A simple method is to combine the scores of these two models and recommend tags with highest scores after combination (Algorithm 1). We can make use of the two scores calculated in the two approaches and there are two things worthy to be noted here: 1) model one only calculates the scores of tags in the candidate set but model two calculates all the tags $t \in T$. 2) due to the different distribution of scores, we need to normalize the two scores before combination. In order to solve the first problem, we consider all the tags $t \in C \cup T$ where C is the candidate used in the model one. In terms of normalization, we make the $\|score1\|_\infty = \|score2\|_\infty$ and then add these two score, if a tag t is in the candidate set C but not in the T, the score2[t] = 0 and if a tag t is in T but not in C, then the score1[t] =0.

4 Experimental Results

4.1 Dataset

We evaluate our experimental results using the evaluation methods provided by the organizers of ECML PKDD discovery challenge 2009. The training set and the test set are strictly divided and we use the cleaned dump as our training set for our tag recommendation system.

Here are some statistical information about training data and test data: There are 1,401,104 tag assignments. 263,004 bookmarks are posted and among which there are 235,328 different url resources while 158,924 bibtex are posted and among which there are 143,050 different publications. From this, we can see that many resources appear just once in the training file. There are 3,617 users and 93,756 tags in all in the training file. The average number of tags posted to bookmark is 3.48 and the average number of tags posted to bibtex is 3.05.

In the test data, there are 43,002 tag assignments, 16,898 posted bookmarks and 26,104 posted bibtex. Among all the posts, there are only 1,693 bookmark resources and 2,239 bibtex resources which are in the training file. The average number of tags posted to bookmark is 3.81 and the average number of tags posted to bibtex is 3.82.

4.2 Data Preprocessing

The training data is provided by the organizers of the ECML PKDD, we establish three tables, bookmark, bibtex and tas in our MySQL database. In order to get the similarity between resources, we need to preprocess the url field. For each url in the bookmark, we eliminate the prefix such as ‘http://’, ‘https://’ and ‘ftp://’. Then we split the url by the character ‘/’. For example, a url ‘http://www.kde.cs.uni-kassel.de/ws/dc09’ will be split into ‘www.kde.cs.uni-kassel.de’, ‘ws’ and ‘dc09’. As we mentioned above, we define some information extracted from the table as the description of a resource r. We eliminate the stop words in the English dictionary and stem the words, for example, ‘knowledge’ will be stemmed to ‘knowledg’ and both ‘biology’ and ‘biologist’ will be stemmed to ‘biologi’.

4.3 Results and analysis

As performance measures we use precision, recall and f-measure. For a given user u and a given resource r, the true tags are defined as TAG(u,r), then the precision, recall and f-measure of the recommended tags $\widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})$ are defined as follows:

$$\mathrm{recall}\left(\widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})\right) = \frac{1}{|\mathrm{U}|}\sum\nolimits_{\mathrm{u}\in\mathrm{U}} \frac{|\mathrm{TAG}(\mathrm{u},\mathrm{r}) \cap \widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})|}{|\mathrm{TAG}(\mathrm{u},\mathrm{r})|}$$

$$\mathrm{precision}\left(\widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})\right) = \frac{1}{|\mathrm{U}|}\sum\nolimits_{\mathrm{u}\in\mathrm{U}} \frac{|\mathrm{TAG}(\mathrm{u},\mathrm{r}) \cap \widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})|}{|\widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})|}$$

$$\mathrm{f-measure}\left(\widetilde{\mathrm{T}}(\mathrm{u},\mathrm{r})\right) = \frac{2 \times \mathrm{recall} \times \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}}$$

4.3.1 Performance of Language model

In table 3, we show the performance of Language model on the test data provided by the organizers of ECML PKDD challenge 2009. We show the performance of bookmark, bibtex and the whole data respectively. From the table, we can find that the result of bookmark is better than that of bibtex and we can have a highest f-measure of 13.949% when the number of recommended tags is 10.

	bookmark(%)	bibtex(%)	overall(%)
1	5.256/18.287/8.165	3.535/13.714/5.620	4.207/15.509/6.619
2	9.298/17.417/12.124	6.399/12.571/8.481	7.534/14.474/9.910
3	12.467/16.789/14.308	9.030/11.941/10.284	10.377/13.845/11.862
4	14.755/16.114/15.405	11.242/11.399/11.320	12.619/13.251/12.927
5	16.314/15.634/15.967	12.889/10.879/11.799	14.231/12.747/13.448
6	17.288/15.195/16.174	14.361/10.460/12.104	15.507/12.320/13.731
7	17.964/14.874/16.273	15.564/10.103/12.253	16.503/11.977/13.880
8	18.459/14.610/16.311	16.531/9.779/12.289	17.285/11.677/13.938
9	18.827/14.430/16.338	17.280/9.496/12.256	17.884/11.434/13.949
10	19.127/14.270/16.346	17.894/9.243/12.190	18.374/11.218/13.931

Table 3: performance of language model on the test data, the numbers are shown in the following format: recall/precision/f-measure

4.3.2 Performance of ACT model

In table 4, we show the performance of Language model on the test data provided by the organizers of ECML PKDD challenge 2009. We show the performance of bookmark, bibtex and the whole data respectively. From the table, we can see that the performance of ACT model is worse than the language model. We have the highest f-measure of 3.077% when the number of recommended tags is set to five. Also, the performance of bookmark is a little bit better than the bibtex and the reason might be that description in bibtex has some information which is irrelevant to the main topic of the publication.

	bookmark(%)	bibtex(%)	overall(%)
1	2.142/6.800/3.258	0.944/2.758/1.406	1.415/4.346/2.135
2	3.523/5.628/4.333	1.431/2.320/1.770	2.253/3.620/2.778
3	4.519/4.825/4.667	1.885/2.076/1.976	2.920/3.156/3.034
4	5.179/4.196/4.636	2.167/1.868/2.007	3.351/2.783/3.041
5	5.829/3.815/4.612	2.466/1.721/2.027	3.788/2.544/3.044
6	6.418/3.536/4.560	2.724/1.582/2.002	4.175/2.350/3.077
7	6.870/3.257/4.419	2.977/1.483/1.980	4.507/2.180/2.939
8	7.377/3.059/4.324	3.205/1.393/1.942	4.844/2.048/2.878
9	7.849/2.891/4.225	3.557/1.346/1.953	5.244/1.953/2.846
10	8.289/2.746/4.126	3.721/1.276/1.900	5.516/1.854/2.775

Table 4: performance of ACT model on the test data, the numbers are shown in the following format: recall/precision/f-measure

4.3.3 Performance after combination

In table 5, we show the performance after the combination of these two models. From the table, we can see that after combination, our recommendation system works a little better than the Language model and has a highest f-measure of 14.398% when recommending five tags.

	final result(%)
1	4.624/15.271/7.099
2	7.753/14.550/10.116
3	10.626/14.900/12.405
4	12.738/14.944/13.753
5	13.916/14.915/14.398

Table 4: performance of ACT model on the test data, the numbers are shown in the following format: recall/precision/f-measure

The result after combination is shown in Fig. 1, together with the results of the previous two methods.

5 Conclusions

In this paper, we describe our tag recommendation system for the first task in the ECML PKDD Challenge 2009. We exploit two different models to recommend tags. The experimental results show that the Language model works much better than the ACT model and the combination of these two methods can improve the results.

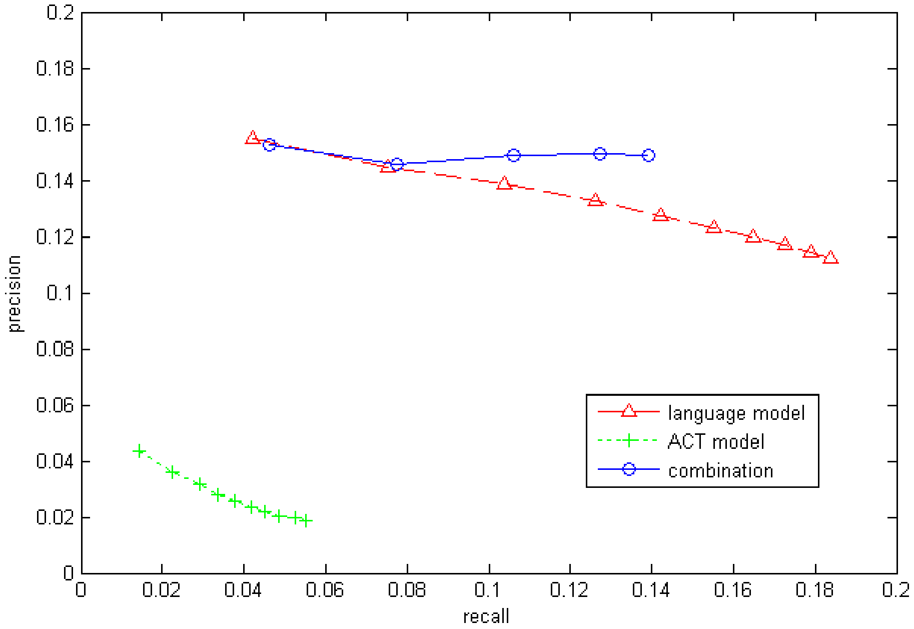


Fig.1 Recall and precision of tag recommendation system

However, we have an unexpected result of the poor ACT model, from the previous test using the separated test data from the training file, the ACT model can have a f-measure over 20%.

We need to further analyze the results to see why ACT has a poor result for the test data. Also, we can try to change the scoring scheme or expand the candidate set in the language model. Future work also includes some new methods of combination.

References

1. Marta Tatu, Munirathnam Srikanth, and Thomas D'Silva. RSDC'08: Tag Recommendations using Bookmark Content. In Proc. of ECML PKDD Discovery Challenge (RSDC 08), pp. 96-107
2. Marek Lipczak. Tag Recommendation for Folksonomies Oriented towards Individual Users. In Proc. of ECML PKDD Discovery Challenge(RSDC 08), pp. 84-95
3. Andreas Hotho, Robert Jaschke, Chiristoph Schmitz, and Gerd Stumme. BibSonomy: A Social Bookmark and Publication Sharing System. In Proc. the First Conceptual Structures Tool Interoperability Workshop at the 14th Int. Conf. on Conceptual Structures, pages 87-102, Aalborg, 2006. Aalborg Universitetsforlag.
4. Andreas Hotho, Robert Jaschke, Christoph Schmitz, and Gerd Stumme. FolkRank: A Ranking Algorithm for Folksonomies. In Proc. of FGIR 2006
5. Robert Jaschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gred Stumme. Tag Recommendations in Folksonomies. In Knowledge Discovery in

Database: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Database, Warswa, Poland, September 17-21, 2007, Proceedings, volume 4702 of LNCS, pages 506-514. Springer, 2007.

6. Mark Steyvers, Padhraic Smyth, and Thomas Griffiths. Probabilistic Author-Topic Models for Information Discovery. In Proc. of KDD'04.
7. Jie Tang, Ruoming Jin, and Jing Zhang. A Topic Modeling Approach and its Integration into the Random Walk Framework for Academic Search. In Proceeding of 2008 IEEE international Conference on Data Mining(ICDM'2008). pp. 1055-1060.
8. Sanjay C. Sood, Sara H.Owsley, Kristian J.Hammond, and Larry Birnbaum TagAssist: Automatic tag suggestion for blog posts. In Proc. the International Conference on Weblogs and Social Media(ICWSM 2007),2007.
9. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In WWW'06: Proceedings of the 15th international conference on World Wide Web, pages 953-954, New York, NY, USA, 2006. ACM Press.

A Collaborative Filtering Tag Recommendation System based on Graph

Yuan Zhang, Ning Zhang, and Jie Tang

Knowledge Engineering Group
Department of Computer Science and Technology, Tsinghua University, Beijing, China
fancyzy0526@gmail.com, zntsinghua1117@gmail.com and
jietang@tsinghua.edu.cn

Abstract. With the rapid development of web2.0 technologies, tagging become much more important today to organize information and help users search the information they need with social bookmarking tools. In order to finish the second task of ECML PKDD challenge 2009, we propose a graph-based collaborative filtering tag recommendation system. We also refer to an algorithm called FolkRank, which is an adaptation of the famous Page Rank. We evaluate and compare these two approaches and show that a combination of these two methods will perform better results for our task.

1 Introduction

Tagging is very useful for users to figure out other users with similar interests within a given category. Users with similar interests might post similar tags and similar resources might have similar tags posted to them. Collaborative filtering is widely used in automatic prediction system. The idea behind it is very simple: those who agreed in the past tend to agree again in the future. Traditional collaborative filtering systems have two steps. The first step is to look for users who share the same rating patterns with the active user whom the prediction is for. Then, the systems will use the ratings from those like-minded users found in the first step to calculate a prediction for the active user. Since all the tags, users and resources in the test data are also in the training file, we can make use of the history of users' tag, also called *personomy*[3] and tags previously posted to the resource to recommend tags for a active post. This paper presents our proposed tag recommendation system, which is a combination of two methods: one is an adaption of item-based collaborative filtering, the other is FolkRank according to [4,5].

As we mentioned above, collaborative filtering performs well for automatic prediction. However, current widely used collaborative filtering systems are for predicting the ratings of some products or recommend some products to users. For example, the famous websites, Amazon.com[1], Last.fm[2], eBay[3] apply this method to

[1] http://www.amazon.com

their recommendation systems. Our first method considers the tags previously posted to the resource and users' similarities to recommend tags. The second method is an application of the FolkRank algorithm in [4, 5].

These two methods have some common features. They both use the history of the user and tags previously posted to resource for recommendation. They are both suitable to the case that test data are in the training data. Both of them do not need to establish models in advance. But they are different to some extents. The first method just considers tags in the candidate set while the FolkRank will consider all the tags in the training data. Moreover, the first method focuses more on collaborative information while the second focuses on the graph information.

This paper is organized as follows: Section 2 introduces recent trends in the area of social bookmark tag recommendation systems. Section 3 describes our proposed system and the combination method in details. In Section 4, we present and evaluate our experimental results on the test data of ECML PKDD challenge 2009 and make some conclusions in Section 5.

2 Related work

Some researchers have already used some approaches based on collaborative information for tag recommendation systems. For example, AutoTag[7] and TagAssist[6] make use of information retrieval skills to recommend tags for weblog posts. They recommend tags based on the tags posted to the similar weblogs. Our first method is similar to these two approaches.

FolkRank in[4, 5] is a topic-specific ranking in folksonomies. The key idea of FolkRank algorithm is that a resource which is tagged with important tags by important users becomes important itself. In [5], the author compared the performance of some baseline methods and his FolkRank algorithm, and found that FolkRank outperformed other methods. His experimental results relied on a dense core of the training file and considering that our training data is a post-core two dataset, we decide to refer to this algorithm in our proposed tag recommendation system.

In the RSDC '08 challenge, the participants [1, 2] who make use of resource's similarities and users' personomy outperformed other approaches. Consequently, we consider using the collaborative information of resource's similarities and users' personomy in our tag recommendation system.

[2] http://www.last.fm

[3] http://www.eBay.com

3 Our Tag Recommendation

3.1 Notations

First, we define notations used in this paper. We group the data in bookmark by its url_hash and data in bibtex by its simhash1. If some posts in bookmark of bibtex file have the same url_hash or simhash1, they are mapped to one resource r. For each resource r_d, assuming a vector $\mathbf{t}_d$ of T_d tags posted to this resource r_d by the user u_d. Then the training dataset can be represented as

$$D = \{(r_1, t_1, u_1), ..., (r_D, t_D, u_D)\}$$

Table1 summarizes the notation.

Symbol	Description
T	the collection of tags posted in the training data
R	the collection of resources posted in the training data (grouped by the url_hash or simhash1)
U	the collection of users who posted tags in the training data
D	training data set containing tagged resources. D={(r_j, $\mathbf{t}_i$, u_i)}, which represents a set of pairs of resources and users, with the assigned tags by the corresponding users.
D’	The test data set containing resources and users. D’={(r_j, u_i)}. Note that: the user u_i, the resource r_j and the original tags posted by u_i to r_i appear in the training dataset.
N_d	number of word tokens in the d ∈ D
T_r	number of tags posted to resource r
$\mathbf{t}_d$	vector form of tags in d ∈ D
u_d	the user in d ∈ D
C(u, r)	the candidate set of tags to be recommended for a given user u and a given resource r
$\widetilde{T}$(u,r)	the set of tags that will be recommended for a given user u and a given resource r
n(t, r)	the number of times that the tag t has been posted to the resource r in the training dataset

Table1: Notations

3.2 Collaborative Filtering method

Our proposed collaborative filtering method for tag recommendation has two steps. First of all, for a given resource r and a given user u in the test dataset, we make use of the tags previously posted to the resource r in the training dataset and define them as the candidate set:

$$C(u,r)=\{t_i \mid (r,t_i,u')\in D, u'\in U\}$$

The second step is to score all the tags in the candidate set and recommend the tags with the highest scores. In our proposed tag recommendation system, we score the tags in the candidate set using the following equation for all tags $t\in C(u,r)$:

$$P(t\mid r)=\frac{T_r}{T_r+\lambda}\cdot\frac{n(t,r)}{T_r}+(1-\frac{T_r}{T_r+\lambda})\cdot\frac{n(t,R)}{|T|} \qquad (1)$$

where n(t,r) is the number of times that tag t has been posted to the resource r and n(t,R) is the number of times that tag t has been posted in the training data. T_r is the number of tags posted to the resource r and λ is the Dirichlet smoothing factor and is commonly set according to the average document length, i.e. $|T|/|R|$

In order to take the users' similarities into consideration, we change the equation (1) to the following equation:

$$P(t\mid r,u)=\frac{T_r}{T_r+\lambda}\cdot\frac{\sum_{u'\in U'}(sim(u,u')\cdot m(t,u,r))}{T_r}$$

$$+(1-\frac{T_r}{T_r+\lambda})\cdot\frac{\sum_{u'\in U'}(sim(u,u')\cdot m(t,u,R))}{|T|} \qquad (2)$$

where $U' = \{u|(r,\mathbf{t},u)\in D\}$ for a given resource r, m(t,u,r) is the number of times tag t has been posted to the resource r by the user u. The similarity of users sim(u,u') is define as follows,

$$sim(u,u')=\frac{\sum_{t\in T}n(t,u)\cdot n(t,u')}{\sqrt{\sum_{t\in T}n(t,u)^2}\cdot\sqrt{\sum_{t\in T}n(t,u')^2}} \qquad (3)$$

For a given user u and a given resource r, the set of recommended tags will be: $\widetilde{T}(u,r) := \arg^{n}_{t\in C(u,r)} P(t|r,u)$ where n is the number of recommended tags.

3.3 FolkRank algorithm

FolkRank is a graph-based algorithm whose basic idea is to rank all the tags and pick out tags which are relatively important given a user u and a resource r. This algorithm is derived from the PageRank algorithm, which is used by the Google Internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents. The purpose of PageRank is to measure the hyperlink's relative importance within the set. However, due to the structural differences between hyperlinks and our tag recommendation system, we cannot apply the PageRank to our tag recommendation system and a new FolkRank algorithm was introduced in [4, 5].

In order to apply a weight-spreading ranking scheme to recommend tags, we need to change the directed graph in PageRank to an undirected graph and change the corresponding ranking approach.

First, we convert the training dataset D into an undirected graph $G=(V,E)$. V is the set of the nodes in the graph, which is composed of all the tags, resources and users in the training file, i.e. $V = T\cup R\cup U$. E is the set of the edges in the graph, which is defined as the co-occurrences of tags and users, users and resources, tags and resources. $\mathrm{E} = \{\{\mathrm{u.t}\}, \{\mathrm{t,r}\}, \{\mathrm{u,r}\} \mid \{\mathrm{r,t,u}\} \in \mathrm{D}\}$ and each edge $\{\mathrm{u,t}\} \in \mathrm{E}$ has a weight $|\{\mathrm{r} \in \mathrm{R} | \{\mathrm{r,t,u}\} \in \mathrm{D}\}|$, each edge $\{\mathrm{t,r}\} \in \mathrm{E}$ has a weight $|\{\mathrm{u} \in \mathrm{U} | \{\mathrm{r,t,u}\} \in \mathrm{D}\}|$ and each edge $\{\mathrm{u,r}\} \in \mathrm{E}$ has a weight $|\{\mathrm{t} \in \mathrm{T} | \{\mathrm{r,t,u}\} \in \mathrm{D}\}|$. After having the graph format of the posts, we can spread the weight like PageRank as follows:

$$\overline{w} \leftarrow dA\overline{w} + (1-d)\overline{p} \tag{3}$$

where A is the adjacency matrix of G, $\vec{\mathrm{p}}$ is the random surfer component, and $\mathrm{d} \in [0,1]$ is a constant which controls the influence of the random surfer.

Usually, $\vec{\mathrm{p}}$ is set to the vector where all values equal to 1. But in order to recommend tags relevant to certain user and certain resource, we can change the $\vec{\mathrm{p}}$ to express user preferences. In our tag recommendation system, each user, tag, and resource get a preference weight of 1 but the active user and resource for recommendation get a preference of 1+|U| and 1+|R| respectively.

The FolkRank algorithm has a differential approach to see the ranking around the topics defined in the preference vector. This approach is to compare the rankings with and without the preference vector $\vec{\mathrm{p}}$. Assuming that $\mathbf{w}_0$ is the ranking after iteration with d = 1 while $\mathbf{w}_1$ is the ranking after iteration with d =0.625, then the final weight will be $\mathbf{w} = \mathbf{w_1} - \mathbf{w}_0$. Details can be found in Algorithm 1.

```
Input: the graph information of the training file, i.e. G = (V, E) where V =T ∪ R ∪ U and
$\mathrm{E} = \{\{\mathrm{u.t}\}, \{\mathrm{t,r}\}, \{\mathrm{u,r}\} \mid \{\mathrm{r,t,u}\} \in \mathrm{D}\}$, the adjacency matrix A, the given resource r and the
given user u.
Output: the ranking w of all tags ∈ T
begin
    //Initialize
    foreach t ∈ T, r ∈ R and u ∈ U do
        $w_0[t] = w_1[t]=1, w_0[r]= w_1[r]=2$ and $w_0[u] = w_1[u] =2$
    end
    foreach t ∈ T, r ∈ R and u ∈ U do
        p[t]=p[r]=p[u]=1
    end
    p[r] = 1+|R|
    p[u]= 1+|U|
    d = 0.625
    //iteration for $w_1$
    repeat
        $\vec{\mathrm{w}}_1 = \mathrm{d}\mathrm{A}\vec{\mathrm{w}}_1 + (1-\mathrm{d})\vec{\mathrm{p}}$
    until convergence
    //iteration for $w_0$
    repeat
        $\vec{\mathrm{w}}_0 = \mathrm{A}\vec{\mathrm{w}}_0$
    until convergence
    $\mathrm{w} = \vec{\mathrm{w}}_1 - \vec{\mathrm{w}}_0$
end
```

Algorithm 1: The FolkRank algorithm used in our tag recommendation system

3.4 Combination

We have proposed two different but similar methods for our tag recommendation system. Both are suitable to our case that the test data have already appeared in the training file, both make use of the similarity of users and resources, but the first method focuses more on the collaborative information while the second one focus more on the graph nodes and can spread the weight according to the co-occurrences. We hope to combine these two methods and get a better result.

We have tried some different approaches to combine these two methods. A simple method of combination is to multiply the scores of these two models and recommend tags with highest scores after combination. Details can be found in Algorithm 2.

```
Input: a given resource r and a given user u and the result of the two methods
Output: the set of recommended tags $\widetilde{T}(u,r)$
begin
    //collaborative method
    the candidate set $C(u,r) \leftarrow \{t|(r,t,u') \in D, u' \in U\}$
    foreach $t \in C$ do
            score1[t] = P(t|r,u) in equation(2)
    end
    //FolkRank algorithm
    foreach $t \in T$ do
            score2[t] = w, the output of the algorithm 1
    end
    //combination
    foreach $t \in T$ do
            score[t] = score1[t]×score2[t]
    end
  $\widetilde{T}(u,r) \coloneqq \mathrm{argmax}^{n}_{t \in T} \mathrm{score}[t]$
end
```

Algorithm 2: the combination method used in our tag recommendation system

4 Experimental Results

4.1 Dataset

We evaluate our experimental results using the evaluation methods provided by the organizers of ECML PKDD discovery challenge 2009. The training set and the test set are strictly divided and we use the post-core level 2 training file as our training dataset for our tag recommendation system.

The general statistical information of training data and test data can be found in the table 2 and table 3.

	tag assignments	|D|	|R|	|U|	|T|	average no. of tags
bookmark	916,469	263,004	235,328	2679	50,855	3.48
bibtex	484,635	158,924	143,050	1790	56,424	3.05
total	1,401,104	421,928	378,378	3617	93,756	3.32

Table2: the general statistical information about the training dataset

	tag assignments	|D|	|R|	|U|	|T|	average no. of tags
bookmark	1,465	431	387	91	587	3.40
bibtex	1,139	347	280	81	397	3.28
total	2,604	778	667	136	862	3.35

Table 3: the general statistical information about the test dataset

4.2 Experimental Result

As performance measures we use precision, recall and f-measure. For a given user u and a given resource r, the true tags are defined as TAG(u,r), then the precision, recall and f-measure of the recommended tags $\tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})$ are defined as follows:

$$\mathrm{recall}\left(\tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})\right) = \frac{1}{|\mathrm{U}|}\sum_{\mathrm{u}\in\mathrm{U}}\frac{|\mathrm{TAG}(\mathrm{u},\mathrm{r}) \cap \tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})|}{|\mathrm{TAG}(\mathrm{u},\mathrm{r})|}$$

$$\mathrm{precision}\left(\tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})\right) = \frac{1}{|\mathrm{U}|}\sum_{\mathrm{u}\in\mathrm{U}}\frac{|\mathrm{TAG}(\mathrm{u},\mathrm{r}) \cap \tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})|}{|\tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})|}$$

$$\mathrm{f}-\mathrm{measure}\left(\tilde{\mathrm{T}}(\mathrm{u},\mathrm{r})\right) = \frac{2 \times \mathrm{recall} \times \mathrm{precision}}{\mathrm{recall} + \mathrm{precision}}$$

4.2.1 Performance of Collaborative Filtering method

In table 4, we show the performance of collaborative filtering method on the test data provided by the organizers of ECML PKDD challenge 2009. From the table, we can see that this method has a highest f-measure of 30.002% when the number of recommended tags is 5.

4.2.2 Performance of FolkRank method

In table 4, we show the performance of FolkRank algorithm on the test data. From the table, we can find that the first method performs a little bit better than FolkRank and FolkRank has a highest f-measure of 28.837% when the number of recommended tags is 4.

	collaborative method	FolkRank algorithm	Combined result
1	13.000/37.147/19.262	14.132/40.231/20.917	14.400/41.512/21.381
2	20.220/31.362/24.588	22.827/33.419/27.126	21.309/38.368/27.400
3	26.760/28.813/27.749	28.326/29.092/28.704	25.117/37.125/29.962
4	32.571/27.035/29.546	32.783/25.739/28.837	27.744/36.739/31.614
5	36.569/25.435/30.002	36.229/23.342/28.392	28.670/36.225/32.008
6	39.079/23.811/29.592	38.826/21.208/27.423	29.409/35.981/32.364
7	41.205/22.670/29.248	40.733/19.262/26.155	29.763/35.935/32.560
8	42.860/21.896/28.985	42.096/17.625/24.847	29.901/35.880/32.619
9	43.863/21.089/28.483	43.227/16.195/23/570	29.933/35.803/32.606
10	45.367/20.591/28.325	44.620/15.077/22.539	29.984/35.769/32.622

Table 4: performance of two methods and combination on the test data, the numbers are shown in the following format: recall/precision/f-measure

4.2.3 Performance of combination

In table 4, we show the performance after the combination of the previous two methods. We are glad to see that the results after combination outperform these two methods. We have a 2% increase compared to the first method and a 4% increase compared to the second method. We have a highest f-measure of 32.622% when recommending 10 tags. The precision-recall plot in Fig.1 reveals the quality of our recommendation system.

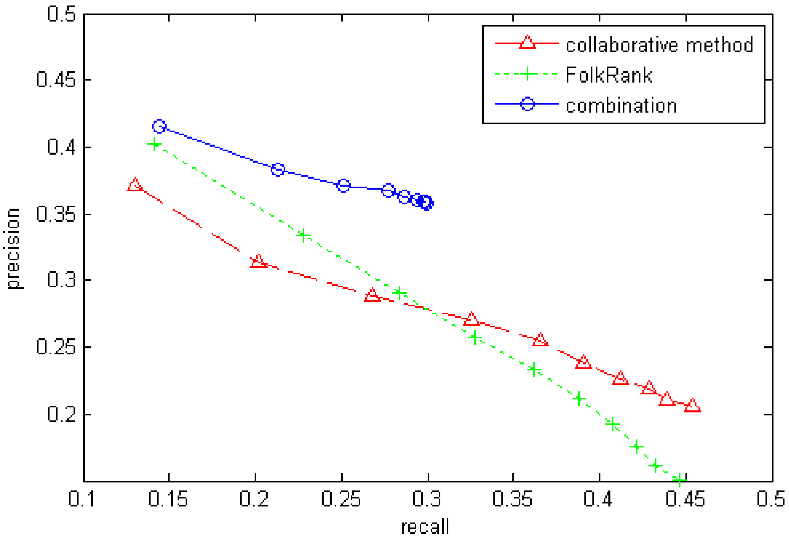


Fig.1 Recall and precision of tag recommendation system

5 Conclusions

In this paper, we describe our tag recommendation system for the second task in the ECML PKDD Challenge 2009. We exploit two different methods to recommend tags when tags, resources, users in the test data are also in the training file. The experimental results show that the combination of these two methods will gain a better result.

We need to further analyze the results to see which kind of information in the graph contributes more to the final ranking. Also, we can try to change the scoring scheme or expand the candidate set in our collaborative filtering method. Future work also includes some adaptations of PageRank for the tag recommendation system.

References

1. Marta Tatu, Munirathnam Srikanth, and Thomas D'Silva. RSDC'08: Tag Recommendations using Bookmark Content. In Proc. of ECML PKDD Discovery Challenge (RSDC 08), pp. 96-107
2. Marek Lipczak. Tag Recommendation for Folksonomies Oriented towards Individual Users. In Proc. of ECML PKDD Discovery Challenge(RSDC 08), pp. 84-95
3. Andreas Hotho, Robert Jaschke, Chiristoph Schmitz, and Gerd Stumme. BibSonomy: A Social Bookmark and Publication Sharing System. In Proc. the First Conceptual Structures Tool Interoperability Workshop at the 14th Int. Conf. on Conceptual Structures, pages 87-102, Aalborg, 2006. Aalborg Universitetsforlag.
4. Andreas Hotho, Robert Jaschke, Christoph Schmitz, and Gerd Stumme. FolkRank: A Ranking Algorithm for Folksonomies. In Proc. of FGIR 2006
5. Robert Jaschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gred Stumme. Tag Recommendations in Folksonomies. In Knowledge Discovery in Database: PKDD 2007, 11th European Conference on Principles and Practice of Knowledge Discovery in Database, Warswa, Poland, September 17-21, 2007, Proceedings, volume 4702 of LNCS, pages 506-514. Springer, 2007.
6. Sanjay C. Sood, Sara H.Owsley, Kristian J.Hammond, and Larry Birnbaum TagAssist: Automatic tag suggestion for blog posts. In Proc. the International Conference on Weblogs and Social Media(ICWSM 2007),2007.
7. Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In WWW'06: Proceedings of the 15th international conference on World Wide Web, pages 953-954, New York, NY, USA, 2006. ACM Press.Web, pages 953-954, New York, NY, USA, 2006. ACM Press.