Workshop proceedings

CBRecSys 2014

Workshop on New Trends in Content-based Recommender Systems

October 6, 2014
RecSys 2014, Sillicon Valley

Edited by
Toine Bogers, Marijn Koolen and Iván Cantador

Preface

While content-based recommendation has been applied successfully in many different domains, it has not seen the same level of attention as collaborative filtering techniques have. In recent years, competitions like the Netflix Prize, CAMRA, and the Yahoo! Music KDD Cup 2011 have spurred on advances in collaborative filtering and how to utilize ratings and usage data. However, there are many domains where content and metadata play a key role, either *in addition to* or *instead of* ratings and implicit usage data. For some domains, such as movies the relationship between content and usage data has seen thorough investigation already, but for many other domains, such as books, news, scientific articles, and Web pages we do not know *if* and *how* these data sources should be combined to provided the best recommendation performance.

The *CBRecSys 2014* workshop aims to address this by providing a dedicated venue for papers dedicated to all aspects of content-based recommendation. We issued a Call for Papers asking for submissions of novel research papers (both long and short) addressing recommendation in domains where textual content is abundant (e.g., books, news, scientific articles, jobs, educational resources, Web pages, etc.) as well as dedicated comparisons of content-based techniques with collaborative filtering in different domains. Other relevant topics included opinion mining for text/book recommendation, semantic recommendation, content-based recommendation to alleviate cold-start problems, as well as serendipity, diversity and cross-domain recommendation.

Each submission was received by three members of the program committee consisting of experts in the field of recommender systems and information retrieval. We selected 7 long papers and 3 short papers for presentation at the workshop. We are also happy to have professor Pasquale Lops of the University of Bari "Aldo Moro" to give a keynote presentation on semantics-aware content-based recommender systems.

We thank all PC members, our keynote speaker as well as authors of accepted papers for making CBRecSys 2014 possible. We hope you will enjoy the workshop!

Toine Bogers, Marijn Koolen, Iván Cantador

Table of Contents

Organization

Workshop organizers

- Toine Bogers, *Aalborg University Copenhagen, Denmark*
- Marijn Koolen, *University of Amsterdam, the Netherlands*
- Iván Cantador, *Universidad Autónoma de Madrid, Spain*

Program committee

- Linas Baltrunas, *Telefónica Research, Spain*
- Alejandro Bellogín, *Universidad Autónoma de Madrid, Spain*
- Shlomo Berkovsky, *NICTA, Australia*
- Robin Burke, *DePaul University, USA*
- Pablo Castells, *Universidad Autónoma de Madrid, Spain*
- Federica Cena, *Universita' degli Studi di Torino, Italy*
- Paolo Cremonesi, *Politecnico di Milano, Italy*
- Marco De Gemmis, *University of Bari "Aldo Moro", Italy*
- Ernesto W. De Luca, *Potsdam University of Applied Sciences, Germany*
- Tommaso Di Noia, *Politecnico di Bari, Italy*
- Peter Dolog, *Aalborg University, Denmark*
- Juan M. Fernández-Luna, *Universidad de Granada, Spain*
- Ignacio Fernández-Tobas, *Universidad Autónoma de Madrid, Spain*
- Frank Hopfgartner, *Technische Universität Berlin, Germany*
- Juan F. Huete, *Universidad de Granada, Spain*
- Marius Kaminskas, *University College Cork, Ireland*
- Alexandros Karatzoglou, *Telefónica Research, Spain*
- Birger Larsen, *Aalborg University, Denmark*
- Pasquale Lops, *University of Bari "Aldo Moro", Italy*
- Alan Said, *Centrum Wiskunde & Informatica, The Netherlands*
- Markus Schedl, *Johannes Kepler University, Austria*
- Giovanni Semeraro, *University of Bari "Aldo Moro", Italy*
- Nava Tintarev, *University of Aberdeen, UK*
- Marko Tkalčič, *Johannes Kepler University, Austria*
- David Vallet, *Google, Australia*
- Markus Zanker, *University of Klagenfurt, Austria*

Semantics-aware Content-based Recommender Systems

Pasquale Lops
Department of Computer Science
University of Bari “Aldo Moro”
Via E. Orabona, 4, I70126
Bari, Italy
lops@di.uniba.it

ABSTRACT

Content-based recommender systems (CBRS) filter very large repositories of items (books, news, music tracks, TV assets, web pages?) by analyzing items previously rated by a user and building a model of user interests, called user profile, based on the features of the items rated by that user. The user profile is then exploited to recommend new potentially relevant items.

CBRS usually use textual features to represent items and user profiles, hence they inherit the classical problems of natural language ambiguity. The ever increasing interest in semantic technologies and the availability of several open knowledge sources have fueled recent progress in the field of CBRS. Novel research works have introduced semantic techniques that shift a keyword-based representation of items and user profiles to a concept-based one.

In this talk I will focus on the main problems of CBRS, such as limited content analysis, and overspecialization, showing the current research directions for overcoming them, including

- top-down semantic approaches, based on the use of different open knowledge sources (ontologies, Wikipedia, DBpedia)
- bottom-up semantic approaches, based on the distributional hypothesis, which states that "words that occur in the same contexts tend to have similar meanings"
- cross-language recommender systems and algorithms for learning multilingual content-based profiles
- the generation of serendipitous recommendations

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

Discovering Contextual Information from User Reviews for Recommendation Purposes

Konstantin Bauman
Stern School of Business
New York University
kbauman@stern.nyu.edu

Alexander Tuzhilin
Stern School of Business
New York University
atuzhili@stern.nyu.edu

ABSTRACT

The paper presents a new method of discovering relevant contextual information from the user-generated reviews in order to provide better recommendations to the users when such reviews complement traditional ratings used in recommender systems. In particular, we classify all the user reviews into the "context rich" *specific* and "context poor" *generic* reviews and present a *word-based* and an *LDA-based* methods of extracting contextual information from the *specific* reviews. We also show empirically on the Yelp data that, collectively, these two methods extract almost all the relevant contextual information across three different applications and that they are complementary to each other: when one method misses certain contextual information, the other one extracts it from the reviews.

Keywords

Recommender systems; Contextual information; Online reviews; User-generated content

1. INTRODUCTION

The field of Context-Aware Recommender Systems (CARS) has experienced extensive growth since the first papers on this topic appeared in the mid-2000's [3] when it was shown that the knowledge of contextual information helps to provide better recommendations in various settings and applications, including Music [8, 9, 12, 13], Movies [5], E-commerce [17], Hotels [10], Restaurants [14].

One of the fundamental issues in the CARS field is the question of what context is and how it should be specified. According to [2, 7], context-aware approaches are divided into *representational* and *interactional*. In the representational approach, adopted in most of the CARS papers, context can be described using a set of observable contextual variables that are known a priori and the structure of which does not change over time. In the interactional approach [4, 11], the contextual information is not known a priori and either needs to be learned or modeled using latent approaches, such as the ones described in [11]. Although most of the CARS literature has focused on the representational approach, an argument has been made that the context is not known in advance in many CARS applications and, therefore, needs to be discovered [3].

In this paper, we focus on the interactional approach to CARS and assume that the contextual information is *not* known in advance and is latent. Furthermore, we focus on those applications where rating of items provided by the users are supplemented with user-generated reviews containing, the contextual information, among other things. For example, in case of Yelp, user reviews contain valuable contextual information about user experiences of interacting with Yelp businesses, such as restaurants, bars, hotels, and beauty & spas. By analyzing these reviews, we can discover various types of rich and important contextual information that can subsequently be used for providing better recommendations.

One way to discover this latent contextual information would be to provide a rigorous formal definition of context and discover it in the texts of the user-generated reviews using some formal text mining-based context identification methods. This direct approach is difficult, however, because of the complex multidimensional task of defining the unknown contextual information in a rigorous way, identifying what constitutes context and what does not in the user-generated reviews, and dealing with complexities of extracting it from the reviews using text mining methods.

Therefore, in this paper we propose the following *indirect* method for discovering relevant contextual information from the user-generated reviews. First, we observe that the contextual information is contained mainly in the *specific* reviews (those that describe specific visit of a user to an establishment, such as a restaurant) and hardly appears in the *generic* reviews (the reviews describing overall impressions about a particular establishment). Second, words or topic describing the contextual information should appear much more frequently in the specific than in the generic reviews because the latter should mostly miss such words or topics. Therefore, if we can separate the specific from the generic reviews, compare the frequencies of words or topics appearing in the specific vs. the generic reviews and select these words or topic having high frequency ratios, then they should contain most of the contextual information among them. This background work of applying the frequency-based method to identifying the important context-related words and topics paves the way to the final stage of inspecting these lists of words and topics.

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

In this paper, we followed this indirect approach and developed an algorithm for classifying the reviews into the "context rich" specific and "context poor" generic reviews. In additional, we present a *word-based* and an *LDA-based* methods of extracting contextual information from the specific reviews. We also show that, together, these two methods extract *almost all* the relevant contextual information across three different applications (restaurants, hotels, and beauty & spas) and that they are complementary to each other: when one method misses certain contextual information, the other one extracts it from the reviews and vice versa. Furthermore, in those few cases when these two methods fail to extract the relevant contextual information, these types of contexts turned out to be rare (appear infrequently in the reviews) and are more subtle (i.e., it is hard to describe such contexts in crisp linguistic terms).

[1, 10, 14] present some prior work on extracting contextual information from the user-generated reviews. Although presenting different approaches, these three references have one point in common: in all the three papers the types of contextual information are a priori known. Therefore, the key issue in these papers is determination of the specific values of the known contextual types based on the reviews.

Although significant progress has been made on learning context from user-generated reviews, nobody proposed any method of separating the reviews into specific and generic and presented the particular methods of extracting the contextual information from the reviews that are described in this paper.

This paper makes the following contributions. First, we proposed two novel methods, a *word-based* and an *LDA-based,* of extracting the contextual information from the user-generated reviews in those CARS applications where contexts are not known in advance. Second, we validated them on three real-life applications (Restaurants, Hotels, and Beauty & Spas) and experimentally showed that these two methods are (a) complementary to each other (whenever one misses certain contexts, the other one identifies them and vice versa) and (b) collectively, they discover almost all the contexts across the three different applications. Third, we show that most of this contextual information can be discovered quickly and effectively.

2. METHOD OF CONTEXT DISCOVERY

The key idea of the proposed method is to extract the contextual information from the user-generated reviews. However, not all the reviews contain rich contextual information. For example, *generic* reviews, describing overall impressions about a particular restaurant or a hotel, such as the one presented in Figure 1, contain only limited contextual information, if any. In contrast, the *specific* visits to a restaurant or staying in a hotel may contain rich contextual information. For example, the review presented in Figure 2 and describing the specific dining experience in a restaurant contains such contextual information as "lunch time," with whom the person went to the restaurant, and the date of the visit. Therefore, the first step in the proposed approach is to separate such *generic* from the *specific* reviews, and we present a particular separation method in Section 2.1.

After that, we use the specific/generic dichotomy to extract the contextual information using the two methods proposed in this paper, the first one based on the identification of the most important context-related words and the second one on the popular LDA method [6]. These two approaches are described in Section 2.2 and 2.3 respectively.

Figure 1: An example of a *generic* review

Figure 2: An example of a *specific* review

2.1 Separating Reviews into Specific and Generic

The main idea in separating specific from generic reviews lies in identification of certain characteristics that are prevalent in one type but not in the other type of review. For example, users who describe particular restaurant experiences tend to write long reviews and extensively use past tenses (e.g., "I came with some friends for lunch today"), while generic reviews tend to use present tense more frequently (e.g., "they make wonderful pastas").

In this work, we identified several such features for separating the generic from the specific reviews, including (a) the length of the review, (b) the total number of verbs used in the review and (c) the number of verbs used in past tenses. More specifically, we used the following measures in our study:

- *LogSentences*: logarithm of the number of sentences in the review plus one[1].
- *LogWords*: logarithm of the number of words used in the review plus one.
- *VBDsum*: logarithm of the number of verbs in the past tenses in the review plus one.
- *Vsum*: logarithm of the number of verbs in the review plus one.
- *VRatio* - the ratio of VBDsum and Vsum ($\frac{VBDsum}{Vsum}$).

Given these characteristics, we used the classical K-means clustering method to separate all the reviews into the "specific" vs. "generic" clusters. We describe the specifics of this separation method, as applied to our data, in Section 3.2.

Once the two types of reviews are separated into two different classes, we next apply the word-based and LDA-based methods described in the next two sections.

[1]We added one avoid the problem of having empty reviews when logarithm becomes $-\infty$.

2.2 Discovering Context Using Word-based Method

The key idea of this method is to identify those words (more specifically, nouns) that occur with a significantly higher frequency in the specific than in the generic reviews. As explained earlier, many contextual words describing the contextual information fit into this category. We can capture them by analyzing the dichotomy between the patterns of words in the two categories of reviews, as explained below, and identify them as follows:

1. For each review R_i, identify the set of nouns N_i appearing in it.
2. For each noun n_k, determine its weighted frequencies $w^s(n_k)$ and $w^g(n_k)$ corresponding to the specific (s) and generic (g) reviews, as follows
$$w^s(n_k) = \frac{|R_i : R_i \in specific \text{ and } n_k \in N_i|}{|R_i : R_i \in specific|}$$
and
$$w^g(n_j) = \frac{|R_i : R_i \in generic \text{ and } n_k \in N_i|}{|R_i : R_i \in generic|}.$$
3. Filter out the words n_k that have low overall frequency, i.e.,
$$w(n_k) = \frac{|R_i : n_k \in N_i|}{|R_i : R_i \in generic \text{ or } R_i \in specific|} < \alpha,$$
where α is a threshold value for the application (e.g., $\alpha = 0.005$).
4. For each noun n_k determine ratio of its specific and generic weighted frequencies: $ratio(n_k) = \frac{w^s(n_k)}{w^g(n_k)}$.
5. Filter out nouns with $ratio(n_k) < \beta$ (e.g $\beta = 1.0$).
6. For each remaining noun n_k left after filtering step 5, find the set of senses $synset(n_k)$ using WordNet[2][16].
7. Combine senses into groups g_t having close meanings using WordNet taxonomy distance. Words with several distinct meanings can be represented in several distinct groups.
8. For each group g_t determine its weighted frequencies $w^s(g_t)$ and $w^g(g_t)$ through frequencies of its members as:
$$w^s(g_t) = \frac{|R_i : R_i \in specific \text{ and } g_k \cap N_i \neq \emptyset|}{|R_i : R_i \in specific|}.$$
9. For each group g_t determine ratio of its specific and generic weighted frequencies as $ratio(g_t) = \frac{w^s(g_t)}{w^g(t_t)}$.
10. Sort groups by $ratio(g_t)$ in its descending order.

As a result of running this procedure, we obtain a list of groups of words that are sorted based on the metric ratio defined in Step 9 above. Furthermore, the contextual words are expected to be located high in the list (and we empirically show it in Section 4).

[2]WordNet is a large lexical database of English. Words are grouped into sets of cognitive synonyms, each expressing a distinct concept. Function $synset(word)$ returns a list of lemmas of this word that represent distinct concepts.

2.3 Discovering Context Using LDA-based Method

The key idea of this method is to generate a list of topics about an application using the well-known LDA approach [6] and identify among them those topics corresponding to the contextual information for that application. In particular, we proceed as follows:

1. Build an LDA model on the set of the specific reviews.
2. Apply this LDA model to all the user-generated reviews in order to obtain the set of topics T_i for each review R_i with probability higher than certain threshold level.
3. For each topic t_k from the generated LDA model, determine its weighted frequencies $w^s(t_k)$ and $w^g(t_k)$ corresponding to the specific (s) and generic (g) reviews, as follows
$$w^s(t_k) = \frac{|R_i : R_i \in specific \text{ and } t_k \in T_i|}{|R_i : R_i \in specific|}$$
and
$$w^g(t_k) = \frac{|R_i : R_i \in generic \text{ and } t_k \in T_i|}{|R_i : R_i \in generic|}.$$
4. Filter out the topics t_k that have low overall frequency, i.e.,
$$w(t_k) = \frac{|R_i : t_k \in T_i|}{|R_i : R_i \in generic \text{ or } R_i \in specific|} < \alpha,$$
where α is a threshold value for the application (e.g., $\alpha = 0.005$).
5. For each topic t_k determine the ratio of its specific and generic weighted frequencies: $ratio(t_k) = \frac{w^s(t_k)}{w^g(t_k)}$.
6. Filter out topics with $ratio(t_k) < \beta$ (e.g. $\beta = 1.0$).
7. Sort the topics by $ratio(t_k)$ in the descending order.

As a result of running this procedure, we obtain a list of LDA topics that is sorted using the ratio metric defined in Step 5 above. Since the contextual information is usually related to the specific user experiences, we expect that these contextual LDA topics will appear high in the generated list, as in the case of the word-based method described in Section 2.2.

We next go through the lists of words and topics generated in Sections 2.2 and 2.3 and select the contextual information out of them. As is shown in Section 4, this contextual information is usually located high on these two lists and therefore can be easily identified and extracted from them. The specifics are further presented in Section 4. As we can see, the list generation methods described in Sections 2.2 and 2.3 lie at the core of our context extraction methodology and make the final context selection process easy.

In summary, we proposed a method of separating the reviews pertaining to the specific user experiences from the generic reviews. We also proposed two methods of generating contextual information, one is based on the LDA topics and another on generating list of words relevant to the contextual information.

In Section 3, we empirically validate our methods and will show their usefulness and complementarity in Section 4.

Category	*Restaurants*		*Hotels*		*Beauty & Spas*	
Cluster	specific	generic	specific	generic	specific	generic
Number of reviews	168	132	195	105	173	127
Number of reviews with context	146	25	127	13	103	9
% of reviews with context	87%	19%	65%	12%	59%	7%

Table 1: Specific vs. Generic Statistics

3. EXPERIMENTAL SETTINGS

To demonstrate how well our methods work in practice, we tested them on the Yelp data (www.yelp.com) that was provided for the RecSys 2013 competition. In particular, we extracted the contextual information from the reviews pertaining to restaurants, hotels and beauty & spas applications using the word-based and the LDA-based approaches. We describe the Yelp data in Section 3.1 and the specifics of our experiments in Section 3.2.

3.1 Dataset Descriptions

The Yelp dataset contains reviews of various businesses, such as restaurants, bars, hotels, shopping, real estate, beauty & spas, etc., provided by various users of Yelp describing their experiences visiting these businesses, in addition to the user-specified ratings of these businesses. These reviews were collected in the Phoenix metropolitan area (including towns of Scottsdale, Tempe and Chandler) in Arizona over the period of 6 years. For the purposes of this study, we used all the reviews in the dataset for all the 4503 restaurants (158430 reviews by 36473 users), 284 hotels (5034 reviews by 4148 users) and 764 beauty & spas (5579 reviews by 4272 users). We selected these three categories of businesses (out of 22 in total) because they contained some of the largest numbers of reviews and also differed significantly from each other.

The data about these businesses is specified with the following attributes: business ID, name, address, category of business, geolocation (longitude/latitude), number of reviews, the average rating of the reviews, and whether the business is open or not. The data about the users is specified with the following attributes: user ID, first name, number of reviews, and the average rating given by the user. Finally, the reviews are specified with the following attributes: review ID, business ID, user ID, the rating of the review, the review (textual description), and the date of the review. For instance, Figures 1 and 2 provide examples of restaurant reviews.

3.2 Applying the proposed methods

We applied our context discovery method to the three Yelp applications from Section 3.1 (Restaurants, Hotels and Beauty & Spas). As a first step, we have separated all the user-generated reviews into the specific and generic classes, as explained in Section 2.1. In order to determine how well this method works on the Yelp data, we manually labeled 300 reviews into specific vs. generic for each of the three applications used in this study (i.e., restaurants, hotels and beauty & spas - 900 reviews in total). This labeled data was used for computing performance metrics of our separation algorithm. The results of this performance evaluation are reported in Section 4.

We have also counted the number of occurrences of contextual information in generic and specific reviews. The results presented in Table 1 support our claim that specific reviews contain richer contextual information than generic reviews across all the three applications.

Second, we have applied the word-based method described in Section 2.2 to the Yelp data. Initially, we generated sets of nouns for restaurants, hotels and beauty & spas applications respectively. After we computed the weighted frequencies of nouns and filtered out infrequent and low-ratio words (having the thresholds values of $\alpha = 0.005$, $\beta = 1.0$), only 1495, 1292 and 1150 nouns were left in the word lists for restaurants, hotels and beauty & spas cases respectively. Finally, we combined the remaining words into groups, as described in Step 7, using the Wu&Palmer Similarity measure [19] with the threshold level of 0.9. As a result, we obtained 835, 755, 512 groups of similar nouns for the restaurants, hotels and beauty & spas categories.

Third, we have applied the LDA-based method described in Section 2.3 to the Yelp data. Initially, we pre-processed the reviews using the standard text analysis techniques by removing punctuation marks, stop words, high-frequency words, etc. [15]. Then we ran LDA on the three preprocessed sets of reviews with $m = 150$ topics for each of the three applications using the standard Python module *gensim*[18]. After generating these topics, we removed the most infrequent ones, as described in Step 4 of the LDA-based approach (setting the parameter $\alpha = 0.005$) and low-ratio topics (Step 6) having the parameter $\beta = 1.0$. As a result, we were left with 135, 121 and 110 topics for each of the three applications.

We describe the obtained results in the next section.

4. RESULTS

First, the results of separation of the user-generated reviews into the specific and generic classes are presented in Table 2 that has the following entries:

- *AvgSentences*: the average number of sentences in reviews from the generic or specific cluster.
- *AvgWords*: the average number of words in reviews from the cluster.
- *AvgVBDsum*: the average number of verbs in past tense in reviews from the claster.
- *AvgVsum*: the average number of verbs in reviews from the cluster.
- *AvgVRatio*: the average ratio of VBDsum and Vsum for reviews from the cluster.

Category	*Restaurants*		*Hotels*		*Beauty & Spas*	
Cluster	specific	generic	specific	generic	specific	generic
AvgSentences	9.59	5.04	10.38	5.58	9.36	4.54
AvgWords	129.42	55.97	147.81	65.48	134.5	50.88
AvgVBDsum	27.07	1.09	28.87	1.58	25.8	1.03
AvgVsum	91.54	23.93	107.43	28.88	107.22	25.65
AvgVRatio	0.43	0.02	0.40	0.06	0.38	0.03
Size	59.3%	40.7%	67.8%	32.2%	59.2%	40.8%
AvgRating	3.53	4.03	3.57	3.81	3.76	4.35
Silhouette	0.446		0.424		0.461	
Precision	0.87	0.89	0.83	0.92	0.83	0.94
Recall	0.83	0.91	0.83	0.92	0.88	0.90
Accuracy	0.89		0.88		0.90	

Table 2: Clusterization quality

- *Size*: size of the cluster in percents from the number of all reviews in the category (restaurants, hotels and beauty & spas).
- *AvgRating*: the average rating for reviews from the cluster.
- *Silhouette*: the silhouette measure of the clusterization quality (showing how separable the clusters are).
- *Precision*: the precision measure for the cluster.
- *Recall*: the recall measure for the cluster.
- *Accuracy*: the overall accuracy of clusterization with respect to the manual labeling.

As we can see from Table 2, the separation process gives us two groups of reviews that are significantly different in *all* the presented parameters. Further, this difference is observed not only in terms of the five parameters used in the k-means clustering method used to separate the generic from the specific reviews (first five rows in Table 2), but also in terms of the average rating (AvgRating) measure (that is significantly higher for the generic than for the specific reviews across all the three categories). Also, the silhouette measure is more than 0.4 for all the three categories and is as high as 0.46 for one of them, demonstrating significant separation of the two clusters. Finally, note that the Accuracy measure is around 0.9 across the three categories of reviews (with respect to the labeled reviews - see Section 3.2), which is a good performance result for separating the reviews.

We next extracted the contextual information from the specific reviews (produced in the previous step) using the word- and the LDA-based methods. As explained in Section 3.2, we obtained the sorted lists of 835, 755, 512 groups of words for restaurants, hotels and beauty & spas categories respectively using the word-based approach. We went through these three lists and identified the contextual variables among them - they are marked with the check marks in Column 4 (Word) in Tables 3, 4 and 5 (the numbers in parentheses next to them identify the first occurrences of the group of words in the sorted lists of the groups of words produced by the word-based method).

Similarly, as explained in Section 3.2, we obtained the sorted lists of 135, 121 and 110 topics for restaurants, hotels and beauty & spas categories respectively using the LDA-based approach. We also went through these three lists and identified the contextual variables among them - they are marked with the check marks in Column 5 (LDA) in Tables 3, 4 and 5 (the numbers in parentheses next to them also identify the first occurrences of the topics in the sorted lists of the topics produced by the LDA-based method).

	Context variable	Frequency	Word	LDA
1	Company	56.3%	✓(1)	✓(6)
2	Time of the day	34.8%	✓(77)	✓(21)
3	Day of the week	22.5%	✓(2)	✓(15)
4	Advice	10.7%	✓(13)	✓(16)
5	Prior Visits	10.2%	X	✓(26)
6	Came by car	7.8%	✓(267)	✓(78)
7	Compliments	4.9%	✓(500)	✓(74)
8	Occasion	3.9%	✓(39)	✓(19)
9	Reservation	3.0%	✓(29)	X
10	Discount	2.9%	✓(4)	X
11	Sitting outside	2.4%	X	✓(64)
12	Traveling	2.4%	X	X
13	Takeout	1.9%	✓(690)	X

Table 3: Restaurants

As Table 3 demonstrates, we identified the following types of contexts for the Restaurants category:

- *Company:* specifying with whom the user went to the restaurant (e.g., with a spouse, children, friends, coworkers, etc.).
- *Time of the day:* this context variable contains information about the time of the day, such as morning, evening and mid-day.
- *Day of the week:* specifying the day of the week (Monday, Tuesday, etc.).
- *Advice:* specifying the type of an advice given to the user, such as a recommendation from a friend or a review on Yelp. This context indicates that the user knows the opinions of other parties about the restaurant before going there.
- *Prior Visits:* specifying if the user is the first time visitor or a regular in the restaurant.

- *Came by car:* specifying if the user came to the restaurant by car or not.
- *Compliments:* specifying any types of discounts or special offers that user recieved during his visit, such as happy hour, free appetizer, special offer etc.
- *Occasion:* specifying the special occasion for going to the restaurant, such as birthday, date, wedding, anniversary, business meeting, etc.
- *Reservation:* specifying if the user made a prior reservation in the restaurant or not.
- *Discount:* specifying if the user used any types of discount deals that he or she obtained before coming to the restaurant, such as groupon/coupon, a voucher and a gift certificate.
- *Sitting outside:* specifying if the user was sitting outside (vs. inside) the restaurant during his visit.
- *Takeout:* specifying if the user did not stay in the restaurant but ordered a takeout.

Note that some of this contextual information was found using either the word-based (Company, TimeOfTheDay, DayOfTheWeek, Advice, CameByCar, Compliments, Occasion, Reservation, Discount and Takeout) or the LDA-based method (Company, TimeOfTheDay, DayOfTheWeek, Advice, PriorVisits, CameByCar, Compliments, Occasion and SitOutside).

To validate the context extraction process, we went through the 400 restaurant reviews (produced as described in Section 3.2) and identified by inspection the contextual information in these reviews. This allowed us to identify the contextual information that served as the "ground truth". Table 3 contains all the contextual information that we have found in these 400 reviews (13 different types). Note that the word- and the LDA-based methods collectively found *all* this contextual information, except for the Traveling context (that determines if the user visited the restaurant while on a travel trip in the city or that he/she lives in that city) - 12 different types of context (out of 13).

Furthermore, column 3 in Table 3 presents the frequencies with which particular types of contextual variables appear in the specific reviews of restaurants. Note that the most frequently occurring popular contexts are discovered by both the word- and the LDA-based methods. The differences between the two methods come in discoveries of less frequent contexts. It is interesting to observe that the PriorVisits context was discovered by the LDA but not by the word-based method. This is the case because this context is usually represented by such expressions as "first time," "second time," "twice" and so on, which are hard to capture by the word-based method because none of these expressions contain a clearly defined "strong" noun capturing this context. In contrast, the LDA-based approach captured this context because LDA managed to combine the aforementioned expressions into one topic.

On the other hand, such contexts as Reservation, Discount and Takeout were captured well by the word-based method since all the three contexts have clearly defined nouns characterizing these contexts (e.g., "reservation," "groupon" and "takeout" respectively). In contrast, the LDA-based method did not capture them because these words ("reservation," "groupon" and "takeout") got lost among some other irrelevant topics.

Finally, nether method has discovered the Traveling context because it (a) is very infrequent and (b) is described in more subtle ways, making it difficult to capture it.

In addition to Restaurants, we have also examined the Hotels and the Beauty & Spas categories. The results are presented in Tables 4 and 5 with 10 types of contexts being discovered for the Hotels case and 10 types for the Beauty & Spas categories. Also, both methods missed the CityEvent context (an event happening in the city which is the cause of traveling to that city and staying in the hotel) for the Hotels and captured all the contextual information for the Beaty & Spas application.

As these tables demonstrate, the word- and the LDA-based methods are complementary to each other: some contexts were discovered by one but not by the other method. Further, collectively, these two methods discover most of the contextual information across the three applications examined in this paper.

Figure 3 presents the performance of the word-based discovery method across the three applications (restaurants, hotels and beauty& spas). On X-axis are the ordinal numbers of the groups of words in the word-based list produced as described in Section 3.2. On the Y-axis are the cumulative number of contexts $y(x)$ discovered by examining the first x groups of words on the list. Each line in Figure 3 corresponds to the appropriate application. The jumps on the curves correspond to the number of the first occurrence of the next contextual variable in the list of groups of words. As we can see from Figure 3, word-based method identified eight contextual variables for each application within the

	Context variable	Frequency	Word	LDA
1	Company	37.3%	✓(4)	✓(11)
2	Occasion	24.3%	✓(1)	✓(6)
3	Reservation	12.9%	✓(18)	X
4	Time of the year	12.4%	✓(94)	✓(30)
5	Came by car	9.4%	✓(381)	✓(65)
6	Day of the week	7.4%	✓(207)	✓(41)
7	Airplane	4.9%	✓(57)	✓(40)
8	Discount	4.4%	✓(23)	X
9	Prior Visits	3.7%	X	✓(57)
10	City Event	3.4%	X	X
11	Advice	1.9%	✓(134)	✓(31)

Table 4: Hotels

	Context variable	Frequency	Word	LDA
1	Company	30.1%	✓(47)	✓(22)
2	Day of the week	18.9%	✓(8)	X
3	Prior Visits	15.2%	X	✓(25)
4	Time of the day	13.2%	✓(3)	✓(4)
5	Occasion	9.6%	✓(15)	✓(29)
6	Reservation	9.4%	✓(167)	✓(1)
7	Discount	9.2%	✓(46)	✓(39)
8	Advice	4.1%	✓(2)	✓(8)
9	Stay vs Visit	3.1%	X	✓(19)
10	Came by car	1.8%	✓(113)	✓(75)

Table 5: Beauty & Spas

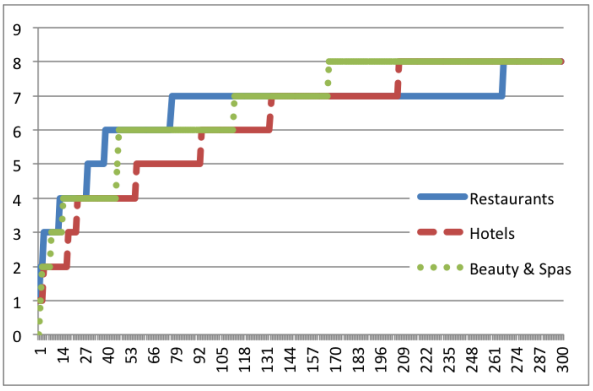


Figure 3: Word-based method

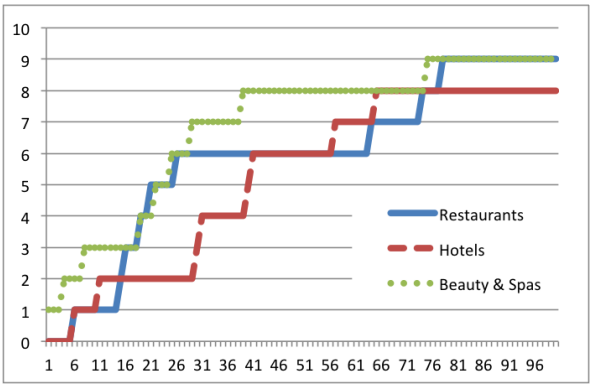


Figure 4: LDA-based method

first 300 groups of words on the list. Moreover, the first four contextual variables were identified from only first 30 groups of words on the list. This supports our earlier observation that many contextual variables appear relatively high on the list of words groups and therefore could be easily identified.

Figure 4 presents similar curves for the LDA-based method. This method managed to identify 9 contextual variables for restaurants and hotels applications, and 8 contextual variables for the beauty & spas application from the first 78 topics on the list of all the topics. Moreover, the first 6 topics were identified within just the first 41 topics. This further supports the earlier observation that many contextual variables appear high on the topics list and therefore could be easily identified.

As discussed before, the word- and the LDA-based methods are complementary to each other. In our three applications all the identified contextual variables could be identified within the first 78 LDA-topics and 29 groups of words in case of restaurants, 65 topics and 23 groups of words in case of hotels, and 75 topics and 8 groups of words in case of beauty & spas. Therefore, combination of the word- and the LDA-based methods idetifies almost all the frequent contextual variables by examining only the top several items on the two lists.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented two novel methods for systematically discovering contextual information from user-generated reviews. The first *word-based* method identifies the most important nouns that appear more frequently in the specific than in the generic reviews, and many important contextual variables appear high in this sorted list of nouns. The second *LDA-based* approach constructs a sorted list of topics generated by the popular LDA method [6]. We also show in the paper that many important types of context appear high in the list of the constructed topics. Therefore, these contexts can easily be identified by examining these two lists, as Figures 3 and 4 demonstrate.

We validated these two methods on three real-life applications (Yelp reviews of Restaurants, Hotels, and Beauty& Spas) and empirically showed that the word- and the LDA-based methods (a) are complementary to each other (whenever one misses certain contexts, the other one identifies them and vice versa) and (b) collectively, they discover almost all the contexts across the three different applications. Furthermore, in those few cases when these two methods fail to extract the relevant contextual information, the missed contexts turned out to be rare (appear infrequently in the reviews) and are more subtle (i.e., it is hard to describe these contexts in crisp terms). Finally, we showed that most of the contextual information was discovered quickly and effectively across the three applications.

As a future research, we plan to use other text mining methods in addition to the word-based and the LDA-based approaches and compare their effectiveness with the two methods presented in the paper. Hopefully, these improvements will help us to discover even more subtle and low-frequency contexts. Since the proposed word-based and LDA-based methods constitute general-purpose approaches, they can be applied to a wide range of applications, and we plan to test them on various other (non-Yelp based) cases to demonstrate broad usefulness of these methods.

6. REFERENCES

[1] S. Aciar. Mining context information from consumers reviews. In *Proceedings of Workshop on Context-Aware Recommender System*. ACM, 2010.

[2] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[3] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 217–253. Springer US, 2011.

[4] S. Anand and B. Mobasher. Contextual recommendation. In B. Berendt, A. Hotho, D. Mladenic, and G. Semeraro, editors, *From Web to Social Web: Discovering and Deploying User and Content Profiles*, volume 4737 of *Lecture Notes in Computer Science*, pages 142–160. Springer Berlin Heidelberg, 2007.

[5] J. F. T. Ante Odic, Marko Tkalcic and A. Kosir. Predicting and detecting the relevant contextual information in a movie-recommender system. In *Interacting with Computers*, 25(1), pages 74–90. Oxford University Press, 2013.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[7] P. Dourish. What we talk about when we talk about

context. *Personal Ubiquitous Comput.*, 8(1):19–30, Feb. 2004.

[8] N. Hariri, B. Mobasher, and R. Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–138, New York, NY, USA, 2012. ACM.

[9] N. Hariri, B. Mobasher, and R. Burke. Query-driven context aware recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 9–16, New York, NY, USA, 2013. ACM.

[10] N. Hariri, B. Mobasher, R. Burke, and Y. Zheng. Context-aware recommendation based on review mining. In *IJCAI' 11, Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2011)*, pages 30–36, 2011.

[11] X. Jin, Y. Zhou, and B. Mobasher. Task-oriented web user modeling for recommendation. In *Proceedings of the 10th international conference on User Modeling*, UM'05, pages 109–118, Berlin, Heidelberg, 2005. Springer-Verlag.

[12] M. Kaminskas and F. Ricci. Location-adapted music recommendation using tags. In J. Konstan, R. Conejo, J. Marzo, and N. Oliver, editors, *User Modeling, Adaption and Personalization*, volume 6787 of *Lecture Notes in Computer Science*, pages 183–194. Springer Berlin Heidelberg, 2011.

[13] J. Lee and J. Lee. Context awareness by case-based reasoning in a music recommendation system. In H. Ichikawa, W.-D. Cho, I. Satoh, and H. Youn, editors, *Ubiquitous Computing Systems*, volume 4836 of *Lecture Notes in Computer Science*, pages 45–58. Springer Berlin Heidelberg, 2007.

[14] Y. Li, J. Nie, Y. Zhang, B. Wang, B. Yan, and F. Weng. Contextual recommendation based on text mining. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 692–700, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[15] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[16] G. A. Miller. Wordnet: A lexical database for english. *COMMUNICATIONS OF THE ACM*, 38:39–41, 1995.

[17] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Trans. on Knowl. and Data Eng.*, 20(11):1535–1549, Nov. 2008.

[18] R. Rehurek and P. Sojka. Software framework for topic modelling with large corpora. In *Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks*, pages 46–50, Valletta, Malta, 2010. University of Malta.

[19] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

HybridRank : A Hybrid Content-Based Approach To Mobile Game Recommendations

Anthony Chow
Group Digital Life
Singapore Telecommunications Ltd
awjchow@singtel.com

Min-Hui Nicole, Foo, Ph.D.
Group Digital Life
Singapore Telecommunications Ltd
nicolefoo@singtel.com

Giuseppe Manai, Ph.D.
Group Digital Life
Singapore Telecommunications Ltd
giuseppe@singtel.com

ABSTRACT

The massive number of mobile games available necessitates a technique to help the consumer find the right game at the right time. This paper introduces HybridRank, a novel hybrid algorithm to deliver recommendations for mobile games. This technique is based on a personalised random walk approach, with the incorporation of both content-based and user-based information in the formulation of the recommendations. This technique is evaluated against traditional neighbourhood based collaborative filtering and content-based recommendation algorithms. This paper also explores the fact that this algorithm can also be used to help alleviate the cold start problem that is associated with little user data.[1] Online evaluations were conducted and results yield that the approach presented performed the best in both a controlled testing environment as well as in live production. This algorithm is currently implemented in a live mobile game platform developed by Singapore Telecommunications Ltd called WePlay.

Categories and Subject Descriptors

H.4 [**Recommender Systems**]: personalised pagerank,random walk,online evaluation,cold-start,hybrid recommender

General Terms

Experimentation, Algorithms, Measurement

1. INTRODUCTION

Singapore Telecommunications Ltd (SingTel) launched WePlay, a mobile game app store in early 2014. With an increasing number of mobile games available to the consumer, there is a need for the development of a mobile game recommendation system. Key work on this domain has been done by Xbox [2] and others [3]. This paper presents a novel approach, HybridRank, which is a hybrid content-based recommender system using a biased random walk model. This is an adaptation of the ItemRank algorithm [10] for the incorporation of both content-based and user-based signals to generate recommendations. Online evaluations were conducted and results yield that the approach presented performed best when compared against user-based collaborative filtering[16] and content-based filtering approaches.[15]

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

2. RELATED WORK

Recommendation algorithms can be broadly classified into two well known techniques: collaborative filtering methods and content-based methods. User-user collaborative filtering starts by placing the user in a vector space of their explicit and implicit activities. A nearest neighbour algorithm with a defined distance metric is then applied to identify what items the users might like based on behaviours of users that are most similar to them. Such an algorithm typically faces issues of data sparsity [9] and algorithm complexity[5].

Unlike collaborative filtering, content based recommendation systems takes the approach of item-to-item correlation. With this technique, the system learns to recommend items that are similar to the ones that the user liked in the past. The similarity is calculated based on features associated with the items being compared.

There is also an increasing focus on the ability to combine both user and content metadata together in a hybrid way to generate recommendations such as using Naive Bayes [1] or clustering techniques [7]. Graph based approaches of using concept graphs [12] and Markov Chains [6] have also been presented. Such approaches usually model the users as a bipartite graph where the nodes are users and items, and a link is drawn between the nodes if a user has done an activity on the item, i.e. watched a movie, liked a restaurant or listened to a song.

This paper explores new approaches towards the graph-based hybrid recommender system problem. It draws heavily on the Pagerank algorithm [8]. This algorithm has been adapted by ItemRank [10], as well as LBSNRank[4].The Pagerank algorithm computes an importance score for each node, and one can use this important score as a measure of importance within the network to be used to provide recommendations.

3. PROBLEM DESCRIPTION

A recommender system deals with a set of users u_i where $i = 1, ..., n$ and a set of items p_j where $j = 1, ..., m$. For each user, item pair, (u_i, p_j) the system generates a score that

will describe the relationship between u_i and p_j captured in a relationship score r_{ij}. To generate this score, this paper proposes HybridRank, an algorithm that combines both items features and user behaviours in a novel way for recommendations. The key steps in setting up the algorithm is to generate two matrices based on user and feature correlation respectively.

3.1 User and Feature Correlation Matrices

The user correlation graph G_u draws the correlation between games via user co-occurrence, i.e, a link appears between 2 games g_i and g_j if one or more users have downloaded both games. It is noted that multiple user co-occurrence matrices can be developed, where links between users can be drawn not only when they have downloaded both games, but also the frequency of which they have viewed and played the games on the platform. For these co-occurrence matrices, define matrix $\mathbb{M} \in \mathbb{R}^{n*m}$ where n is number of games and m is number of users. $\mathbb{M}_{x_{ij}}$ represents the number of times a user u_i has conducted an action x on the game g_j. Examples of actions would be viewed, downloaded or played. Their respective correlation matrix can then be generated via the inner product of the matrices as follows:

$$\mathbb{U}_x = \mathbb{M}_x \cdot \mathbb{M}_x^\intercal \tag{1}$$

The feature correlation graph G_f on the other hand draws the correlation between games via feature co-occurrence, i.e. a link appears between two games g_i and g_j if both games share one or more metatags. For example, two games that share the same developer, or price points will share a link. The feature set F_{ij} is defined as the set of features which belong to both g_i and g_j where $i \neq j$ and $i, j \in S_{availablegames}$. These features can typically be generated from two sources. The first source will be structured information provided by the developer, the second being user generated content like reviews. This paper focuses on utilising structured metatags provided by the former source.

It is noted there are some features that are more important in determining game similarity as compared to others, for example two games sharing the same game mechanics is considered more similar than two games sharing the same price point. [14]. As such, a set of weights β_k associated with each feature can be defined. This set of weights can be learned, defined via empirical experiments or assigned via a TF-IDF on the content vector of the items[13]. Specifically for the experiments conducted a hierarchy of metadata was defined, and weights were assigned from qualitative user feedback. With that, the feature correlation matrix can be defined as follows:

$$\mathbb{F}_{ij} = \sum_{k=1}^{n} \beta_k \mathbf{1}_k \tag{2}$$

$\mathbf{1}_k$ will be 1 if both games g_i and g_j share feature k, and 0 otherwise. We normalise both matrices, $\mathbb{U}$ and $\mathbb{F}$ column-wise to generate stochastic matrices $\tilde{\mathbb{U}}$ and $\tilde{\mathbb{F}}$, such that each column sums up to 1. While the former is a symmetrical matrix, the normalised matrices are not symmetric. The diagonals are also 0 by definition. These two correlation matrices become valuable graphic model to indicate correlations between games. The weights associated with the links provide approximate measures of games relation.

4. HYBRIDRANK: THE ALGORITHM

The idea underlying HybridRank is that a hybrid combination of both the user and feature correlation graph can be used to forecast the user preferences in a content-based approach. The personalised page rank algorithm has been shown to be a good algorithm to be used for such a use case as in [10]. This algorithm offers key properties of propagation and attenuation. Utilising the relationships between games, captured by both the feature and user correlation matrices, $\tilde{\mathbb{U}}$ and $\tilde{\mathbb{F}}$, the personalised page rank algorithm is able to propagate preferences through the graph from a given starting point. As the preferences move further away from the seed nodes, the influence of the user preferences diminishes, and such an attenuation property is aptly captured by the said algorithm. The personalised page rank algorithm is defined as below:

$$PR_{u_i} = \alpha \cdot M \cdot PR_{u_i} + (1 - \alpha) \cdot d_{u_i} \tag{3}$$

PR_{u_i} refers to the personalised page rank vector for a particular user u_i, which gives an indication of the importance the different nodes in the system to the user u_i. M refers to the stochastic matrix which captures the connectivity between all the nodes in the system. This paper uses the feature correlation matrix $\tilde{\mathbb{F}}$ to represent the connections between the games. The vector d_{u_i} is often referred to as the teleport vector, which allows the introduction of bias into the system to a given user u_i. This generates a static score distribution vector of all the items that user has consumed or has an opinion for. For example, the j^{th} element of the vector d_{u_i} will be 1 if the user u_i has downloaded the game, and 0 otherwise. The vector will then be normalised to sum to 1.

The HybridRank algorithm builds on this idea by introducing the user correlation matrix $\mathbb{U}$ to build this vector d_{u_i}. Let the set $\mathcal{D}^{dl}_{u_i}$, $\mathcal{D}^{v}_{u_i}$ and $\mathcal{D}^{p}_{u_i}$ be the set of games that the user u_i has downloaded, viewed and played respectively. The vector d_{u_i} can be defined as follows:

$$d^j_{u_i} = \gamma \cdot \sum_{k \in \mathcal{D}^{dl}_{u_i}} \tilde{\mathbb{U}_{dl}}_{jk} + \eta \cdot \sum_{k \in \mathcal{D}^{v}_{u_i}} \tilde{\mathbb{U}_{v}}_{jk} + \theta \cdot \sum_{k \in \mathcal{D}^{p}_{u_i}} \tilde{\mathbb{U}_{p}}_{jk} \tag{4}$$

Next normalise the vector to sum to one, to obtain $\tilde{d}_{u_i}$. For this paper, equal weights have been assigned to the weights γ, η and θ and further optimisation is underway. In the simple case, where the user has only downloaded one game, the vector $\tilde{d}_{u_i}$ will simply be the j^{th} column of the matrix $\mathbb{U}$ corresponding to the game that the user has downloaded. This draws upon not only the user preferences, but also assigns a bias towards games that are close in relationship to the games selected via a simple collaborative approach or captured via the user co-occurrence matrix.

Linear algebra approaches via power iteration can be used to solve equation 3. There has been research to improve computation efficiency, one of them being in [17]. Also, in terms of complexity, [10] has shown that such a computation is efficient from both computation and memory resources.

5. EXPERIMENT RESULTS

The HybridRank algorithm was developed and deployed in two separate live experiments in relation to mobile game recommendations. The first experiment was done in a controlled fashion with a preloaded web prototype with a group of 526 users. The second experiment was done on the production app WePlay with over 100,000 users in Indonesia.

5.1 Online Evaluation 1

The seed dataset has 78099 users and 199 games. Each game also comes with its set of 149 set of metadata, including tags that are temporal in nature, i.e. whether it is trending, top grossing or curated by marketing team for that week. The following shows the distribution of the tags available. As from equation 2, weights β were assigned to several top-level categories. This was purely done via qualitative assumptions and reasoning.

Tag Type	Tag Count	β
Developers	79	0.2
Categories	25	0.3
Price Ranges	11	0.05
ESBN Ratings	6	0.1
In-App Goods	2	0.05
Others (Motivations, Goals etc)	26	0.3

Table 1: Distribution of metatags available and weights assigned

A testing portal was developed and sent to 526 digitally savvy members of SingTel Digital Advisor Panel[1]. These users were asked to select up to five mobile games that they like, and four separate lists of recommendations were provided generated by HybridRank, kNN Collaborative Filtering, Top Grossing and Baseline. The baseline algorithm randomly selects games from the entire catalog. Each set of recommendations exposed seven games. The users were asked to then choose the mobile games they like across all the lists provided.

To evaluate the results, users were segmented[2] across the dimensions of *externalised gratifications* and *internalised fulfilment*. The former comprises of factors associated with basic progression in the game, scoring, beating the competition. The latter involves the altruistic sharing of knowledge and experience, helping others in game progression and gaining respect and trusted recognition. Table 2 gives the definition and distribution of users across the segments.

[1]This is a panel of 15,000 users across South East Asia maintained by SingTel Group Digital Life to help in testing of new digital products. The users in this study have been screened to have played at least a mobile game in the past one month.
[2]This framework is an ongoing research by the team in Group Digital Life SingTel in an effort to deeper understand the gamer's psyche, fundamentally based on Maslow′s Hierarchy of Needs [11]

Type	Definitions	No of testers
Trend Seeker and Contributor (Grp1)	Testers are at the forefront and actively contribute online reviews to share knowledge	34
Trend Seeker (Grp2)	Testers are at the forefront and less actively/seldom contribute online reviews to share knowledge	77
Contributor (Grp3)	Testers who are not at the forefront but are actively involved in information exchanges with like-minded gamers through online reviews	41
Social (Grp4)	Testers who take heed from what their friends/social circle play	128
Indifferent (Grp5)	Testers who just want to stay in the game and are indifferent to what others say	246

Table 2: Distribution and definitions of user testing groups

Table 3 shows the performance results of the four algorithms across the five different segments. Success of the algorithms were measured by comparing the lift in average number of games selected against baseline. It can be seen that HybridRank provided maximal lift in segments of users who are indifferent and social gamers. For the small segment of users who are trend seekers, it appears that the top grossing algorithm performed the best. This could be because the HybridRank did not take into consideration global market features in the development of the item metadata.

Grp1	HyRank	CF	TopG	Baseline
Average	2.12	1.56	1.88	1.15
Lift	+0.846	+0.359	+0.641	0
Grp2	HyRank	CF	TopG	Baseline
Average	1.756	1.536	1.878	0.927
Lift	+0.895	+0.658	+1.03	0
Grp3	HyRank	CF	TopG	Baseline
Average	1.922	1.481	1.805	0.974
Lift	+0.973	+0.52	+0.853	0
Grp4	HyRank	CF	TopG	Baseline
Average	2.023	1.585	1.781	0.914
Lift	+1.214	+0.735	+0.9487	0
Grp5	HyRank	CF	TopG	Baseline
Average	1.671	1.276	1.459	0.825
Lift	+1.02	+0.546	+0.768	0

Table 3: Results of recommendations lift across the different groups

5.2 Online Evaluation 2

In this second evaluation, the algorithm was exposed to over 100,000 users in Indonesia in the live WePlay app with over 900 games to recommend from. The section evaluated recommends games that are similar to the selected game. This particular use-case can be likened to the cold start problem, where there are no previous preferences of the user and

the only preference being the current selected game. The HybridRank algorithm was compared with two other algorithms. The first being a commonly used content-based algorithm via an euclidean distance metric on the feature vector of the games as in [15] and the second being a baseline that chooses the more popular items within the same category as the selected game. The experiment was conducted live on the platform in Indonesia for a period of one month in an out of time validation fashion. The entire base was exposed to the algorithms in an alternating day fashion. The click through rates of the suggested game were measured - the higher the click through rate, the more effective the algorithm was considered to be.

Country	Baseline	Content-based	HybridRank
Indonesia	6.3%	7.1%	13.3%
Lift	0	+0.127	+1.11

Table 4: Evaluation of algorithms on live production environment

From the results HybridRank was shown to serve as a better algorithm in recommending games in a user cold-start scenario. The hybrid approach of using both user and feature correlation proved superior to the typical content-based approach.

6. CONCLUSION

This paper presents HybridRank, a personalised pagerank approach that incorporates both content metadata and user collaborative features in a novel approach. The algorithm was compared against state of the art collaborative filtering algorithms as well as content based approaches in live environment, with the conclusion that the hybrid approach performs better against the algorithms that were compared against. Also the algorithm proved to be able to help alleviate the cold start problem. Future work will include the incorporation of context such as user location, global trends in mobile gaming as well as custom curated metadata to the approach.

7. ACKNOWLEDGEMENTS

The authors would like to thank the WePlay team for the data and allowing us to conduct evaluation of our algorithms on the live app.

8. REFERENCES

[1] Andrew I. Schein et al. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '02)*, pages 253–260, 2002.

[2] Noam Koenigstein et al. The xbox recommender system. *Proceedings of the sixth ACM conference on Recommender systems (RecSys '12)*, pages 281–284, 2012.

[3] Pavle Skocir et al. The mars - a multi-agent recommendation system for games on mobile phones. *KES-AMSTA'12 Proceedings of the 6th KES international conference on Agent and Multi-Agent Systems: technologies and applications*, pages 104–113, 2012.

[4] Zhaoyan Jin et al. Lbsnrank: personalized pagerank on location-based social networks. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 980–987, 2012.

[5] Diego Fernandez Fidel Cacheda, Victor Carneiro and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):2:1–2:33, 2011.

[6] Francois Fouss, Alain Pirotte, Jean michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19:2007, 2006.

[7] Byeong Man Kim, Qing Li, Chang Seok Park, Si Gwan Kim, and Kim Ju Yeon. A new approach for combining content-based and collaborative filters. *J. Intell. Inf. Syst.*, 27(1):79–91, 2006.

[8] R. Motwani L. Page, S. Brin and T. Winograd. The pagerank citation ranking: Bringing order to the web. *Technical report, Stanford University*, 1998.

[9] Dimitris Plexousakis Manos Papagelis and Themistoklis Kutsuras. Alleviating the sparsity problem of collaborative filtering using trust inferences. *iTrust'05 Proceedings of the Third international conference on Trust Management*, pages 224–239, 2005.

[10] Augusto Pucci Marco Gori. Itemrank: A random-walk based scoring algorithm for recommender engines. *International Joint Conferences on Artificial Intelligence*, pages 2766–2771, 2007.

[11] A. H. Maslow. A theory of human motivation. *Psychological Review*, 50:370–396, 1943.

[12] Le Quang Thang Nguyen Duy Phuong and Tu Minh Phuong. A graph-based method for combining collaborative and content-based filtering. *PRICAI '08 Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, pages 859 – 869, 2008.

[13] Incheon Paik and Hiroshi Mizugai. Recommendation system using weighted tf-idf and naive bayes classifiers on rss contents. *JACIII*, 14:631–637, 2010.

[14] Rapeepisarn K. Paireekreng, W. and K.W. Wong. Personalised mobile game recommendation system. *6th International Game Design and Technology Workshop and Conference*, 2008.

[15] F. et al. Ricci. *Recommender systems handbook*, chapter Content-based Recommender Systems: State of the Art and Trends. Springer, Berlin, 2011.

[16] Bell R.M. and Koren Y. Improved neighborhood based collaborative filtering. *KDD 2007 Netflix Competition Workshop*, 2007.

[17] Christopher D Manning Sepandar D Kamvar, Taher H Haveliwala and Gene H Golub. Extrapolation methods for accelerating pagerank computations. *WWW '03 Proceedings of the 12th international conference on World Wide Web*, pages 261–270, 2003.

Exploiting FrameNet for Content-Based Book Recommendation

Orphée De Clercq
LT3, Language and Translation Technology Team
Ghent University
orphee.declercq@ugent.be

Michael Schuhmacher
Research Group Data and Web Science
University of Mannheim
michael@informatik.uni-mannheim.de

Simone Paolo Ponzetto
Research Group Data and Web Science
University of Mannheim
simone@informatik.uni-mannheim.de

Véronique Hoste
LT3, Language and Translation Technology Team
Ghent University
veronique.hoste@ugent.be

ABSTRACT

Adding semantic knowledge to a content-based recommender helps to better understand the items and user representations. Most recent research has focused on examining the added value of adding semantic features based on structured web data, in particular Linked Open Data (LOD). In this paper, we focus in contrast on semantic feature construction from text, by incorporating features based on semantic frames into a book recommendation classifier. To this purpose we leverage the semantic frames based on parsing the plots of the items under consideration with a state-of-the-art semantic parser. By investigating this type of semantic information, we show that these frames are also able to represent information about a particular book, but without the need of having explicitly structured data describing the books available. We reveal that exploiting frame information outperforms a basic bag-of-words approach and that especially the words relating to those frames are beneficial for classification. In a final step we compare and combine our system with the LOD features from a system leveraging DBpedia as knowledge resource. We show that both approaches yield similar results and reveal that combining semantic information from these two different sources might even be beneficial.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.4 [**Information Systems Applications**]: Miscellaneous

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

Keywords

Content-Based Recommender Systems, Semantic Frame, Linked Data

1. INTRODUCTION

Recommender systems are omnipresent online and constitute a significant part of the marketing strategy of various companies. In recent years, a lot of advances have been made in constructing collaborative filtering systems, whereas the research on content-based recommenders had lagged somewhat behind. Similar to evolutions in information retrieval research, the focus has been more on optimizing tools and finding more sophisticated techniques leveraging for example big data than on the actual understanding or processing of the items or text at hand.

In Natural Language Processing (NLP), on the other hand, huge advances have been made in processing text both from a lexical and semantic perspective. In this respect, we believe it is important to test whether a content-based recommender system might actually benefit from plugging in more semantically enriched text features, which is the purpose of the current research. In this paper we wish to investigate to what extent leveraging semantic frame information can help in recommending books to users. We chose to work with books, since these typically contain a chronological description of certain actions or events which might be indicative for the interests of a particular reader. Someone might enjoy reading historical novels, for example, but is more prone to those novels where a love history is explained in closer detail than those where a typical revenge story is portrayed. We hypothesize that the semantic frames and or events in these two types of historical novels will be different. In other words, we wish to investigate to what extent deep semantic parsing of the plots describing a book following the FrameNet paradigm can help for recommendation.

In order to validate these claims we performed an extensive analysis on a book recommendation dataset which was provided in the framework of the 2014 ESWC challenge. What is particularly interesting about this dataset is that all the books have been mapped to their corresponding DBpedia URIs which allows us to directly compare externally

gained semantic information as available in the Linked Open Data cloud (LOD) with internal semantic information based on the plots themselves. Our analysis reveals that although some frames and events are good indicators of genres derived from external DBpedia information, they do represent some additional information which might help the recommendation process.

To actually verify this finding we test the added value of incorporating frame information as semantic features in a basic recommender system. We see that exploiting this kind of semantic information outperforms a standard bag-of-words unigram baseline and that incorporating frame elements and lexical units evoking the frames allows for the best overall performance. If we compare our best system to a system levering semantic LOD information, we observe that our frames approach is not able to outperform this system. We do find, however, that if we combine these two semantic information sources into one system we get the best overall performance. This might indicate that combining semantic information from different sources, i.e. from the linguistically grounded implicit frame features and the explicit, ontology grounded DBpedia features, is beneficial.

The remainder of this paper is structured as follows. In Section 2 we describe some related work with an explicit focus on the added value of semantic information for recommender systems. In Section 3 we then explain in closer detail the construction and reasoning behind the semantic frame-enhancement. We then continue by describing the actual experimental setup (Section 4) and have a closer analysis of the results (Section 5). We finish with some concluding remarks and ideas for future work (Section 6).

2. RELATED WORK

In content-based recommender systems, the items to be recommended are represented by a set of features based on their content, whereas a user is represented by his profile. To build a recommender both information sources are compared. Most content-based recommenders use quite simple retrieval models, such as keyword matching or the vector space model with basic TF-IDF weighting [15]. A problem with these models is that they tend to ignore semantic information. To overcome this one can use Explicit Semantic Analysis (ESA) [10] instead of TF-IDF weighting which allows to represent a document as a weighted vector of concepts. Another way to add more linguistic knowledge is to use for example information from Wordnet as done by [6, 3]. An alternative is to use language models to represent documents. This was done for example by [16] when exploring content-based filtering of calls for papers. Besides retrieval models, machine learning techniques where a system learns the user profile and classifies items as interesting or not are also used for content-based recommenders. One of the first to do this was [2] using a Naïve Bayes classifier.

When it comes to adding semantic information to recommender systems we see that currently leveraging *Linked Open Data* (LOD) is a popular research strand. [11] and [18] were among the first to use LOD for recommendation. The former use this information to build open recommender systems whereas the latter built a music recommender using collaborative filtering techniques. [4] was the first to really leverage LOD to build a content-based recommender and the first to exploit the semantics of the relations in the link hierarchy. They use LOD information from `DBpedia`, `Freebase` and `LinkedMDB` as the only background knowledge for a movie recommender system and show that thanks to this ontological information the quality of a standard content-based system can be improved. In more recent work, the semantic item descriptions based on LOD have been merged with positive implicit feedback in a graph-based representation to produce a hybrid top-N item recommendation algorithm, SPrank [17], which further underlines the added value of this kind of data. Moreover, in 2014 in order to spark research on LOD and content-based recommender systems, a shared task was organized by the same authors, i.e. the ESWC-14 Challenge[1].

Table 1: Example of a frame

Frame: KILLING		
The KILLER or CAUSE causes the death of the VICTIM.		
FEs	KILLER	**John** drawned Martha.
	VICTIM	I saw **heretics** beheaded.
	CAUSE	**The rockslide** killed nearly half of the climbers.
	INSTRUMENT	It's difficult to suicide **with only a pocketknife**.
LUs	..., kill.v, killer.n, killing.n, lethal.a, liquidate.v, liquidation.n, liquidator.n, lynch.v, massacre.n,massacre.v, matricide.n, murder.n, murder.v, murderer.n,...	

In content-based recommendation, the advances that have been made were made possible thanks to the availability of designated datasets. These include data for predicting music, Last.FM[2], and or movies, MovieLens[3]. Up till now little research has been performed on other genres, such as books. The ESWC challenge, however, made a book recommendation dataset available which is mapped to DBpedia. DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and makes them available as linked RDF data [14]. This dataset will be used as our main data source. In this paper, we focus on the feature construction for a classifier in that we also incorporate semantic features based on the semantic frames present within the items to be recommended. This is, to our knowledge, the first approach that tries to leverage this kind of data and is one way of tackling the issue of Limited Content Analysis within recommender systems [4]. In order to validate these claims we will compare and combine our best system with a system exploiting LOD.

3. FRAME-ENHANCEMENT

In this section we give some more information about why we believe exploiting frame information might help with recommendations. First, we introduce some basic concepts and theory after which we explain how we apply a state-of-the-art semantic frame parser to our dataset and provide a first analysis. We hypothesize that a plot description tells more about a book than using more global semantic classification based on external semantic information as provided by the LOD cloud. This reasoning can be transferred to other data sources having a large number of textual information.

[1] `http://challenges.2014.eswc-conferences.org/index.php/RecSys`
[2] `http://labrosa.ee.columbia.edu/millionsong/`
[3] `http://grouplens.org/datasets/movielens/`

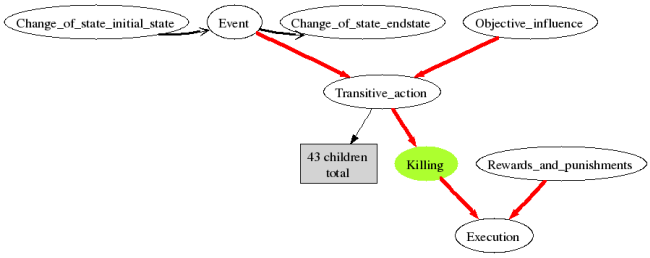


Figure 1: Example of *Inheritance* relations related to the KILLING frame.

3.1 Frame semantics and FrameNet

Following the basic assumption that the meanings of most words can best be understood on the basis of a semantic frame, FrameNet [9] was developed as a linguistic resource storing considerable information about lexical and predicate-arguments semantics in English.

FrameNet is grounded in the theory of frame semantics [7, 8]. This theory tries to describe the meaning of a sentence by characterizing the background knowledge required to understand this sentence. This knowledge is presented in an idealized, i.e. prototypical, form. A frame is thus a structured representation of a concept. It can be a description of a type of event, relation or entity, and the participants in it. In Table 1 we present an example of such a frame, *KILLING*. We see it is a semantic class containing various predicates, also known as lexical units (LUs), evoking the described situation, e.g. *killer, murder, lethal*. Moreover, it illustrates that within FrameNet each frame comes with a set of semantic roles, i.e. frame elements (FEs), which can be perceived as the participants and/or properties of a frame which are of course also lexicalized in the text itself, e.g. *Killer: John, Instrument: with only a pocketknife*.

FrameNet's latest release (1.5) contains 877 frames and about 155K exemplar sentences.[4] An interesting aspect of the FrameNet lexicon is that asymmetric frame relations can relate two frames, thus forming a complex hierarchy containing both *is-a* like and non-hierarchical relations [22]. In this work, we are particularly interested in the former type, also known as *Inheritance* relations. This type of relation entails that the child frame is a subtype of the parent frame. If we look for instance at our Killing example, of which the taxonomy is visualized in Figure 1[5], we are able to find out that this frame is a child of the frame Transitive_action, which is in turn a child of both the frame Objective_Influence and, more interestingly, the frame Event. This taxonomy thus enables us to find even more semantic properties about specific frames.

3.2 Exploiting FrameNet

3.2.1 Book dataset

For the research described in this paper, we worked with the dataset of the ESWC challenge which is in fact a re-

[4]This release is available at `http://framenet.icsi.berkeley.edu`

[5]This graph was produced using the FrameGrapher tool, `https://framenet.icsi.berkeley.edu/fndrupal/FrameGrapher`

Table 2: Example of two sentences of a plot description and its resulting frames.

PLOT	The [Prince], the protagonist, is [named] Alexander. His [father], [Prince] Baudouin, is [**murdered**] by the [King] of Cornwall, [King] [March]. [When] Alexander [comes] of [age], he [sets out] to Camelot to [seek] justice from [King] Arthur and to [**avenge**] the [**death**] of his [father]....
FRAMES	Leadership, Appointing, Kinship, Leadership, **Killing**, Leadership, Leadership, Calendric_unit, Temporal_collocation,Arriving, Calendric_unit, Departing,Seeking_to_achieve, Leadership, **Revenge**, **Death**, Kinship.

elaborated version of the LibraryThing dataset[6]. This dataset contains books that are part of a particular user's online catalog containing the books he/she has read or owns. For the challenge, the books available in the dataset have been mapped to their corresponding DBpedia URIs [17]. Based on the available information we were able to download the plot description of each book from its corresponding Wikipedia page (this plot information is lacking in DBpedia). In this way we envisaged to investigate whether knowing more about what is actually happening in a book can enhance the recommendation. We worked with a subset by only including books of which a uniform and unambiguous DBpedia link was available and that actually contained plot information on Wikipedia. In total our final dataset contains 5,063 books with an average plot length of 312 words[7].

In order to annotate the semantic frames, each plot was parsed using the state-of-the-art frame-semantic parser SEMAFOR [5]. This parser extracts semantic predicate-argument structures from text using a statistical model and is trained on the FrameNet 1.5 release. It takes as input the text as such, performs some preprocessing steps and outputs on a sentence-per-sentence basis all frames that are present within a text. These frames are represented by one of the 877 possible frame names and also the lexical units and frame elements (both generic and lexicalized form) are output. An example is presented in Table 2. This is the plot description of the book *The Prince and the Pilgrim*. In the text itself, the lexical units evoking the frames are indicated in square brackets. The frames and LUs which are represented in bold are those frames which actually constitute an Event. Finding out which books are events can be done by exploiting the taxonomy (cfr. supra) which enables us in a way to find out more semantic properties of specific frames. Intuitively, we can state that especially those Event frames give most information about what is happening within a book: the above-mentioned book is clearly a revenge story. However, the other frames might also pinpoint important aspects, e.g. the repetition of the Leadership and Kinship frames could inform us that this novel is about royalty and family.

What this example also illustrates is that the SEMAFOR parser is not 100% accurate. For example, the name of a particular king – King *March* – is interpreted by the parser as evoking the frame Calendric_unit. We should thus keep

[6]`http://www.macle.nl/tud/LT/`

[7]This dataset will also be made available to the research community in due time

in mind that a certain amount of noise is also introduced into our dataset. Moreover, some frames such as Arriving or Temporal_collocation, are correctly labeled but do not really contribute interesting semantic information.

For all books in our dataset we parsed the plots using SEMAFOR, after which we also filtered out those frames which can have the Event frame as a parent. Some data statistics regarding these annotations are presented in Table 3, which reveal that the information we have available is rather skewed.

Table 3: Plot annotation statistics representing the average number of real and unique frames and events per book and their standard deviations

	# Avg	Stdev	# Avg unique	Stdev
Frames	197	205	96	61
Events	42	45	22	15

3.2.2 Semantic frames versus Linked Open Data

As previously mentioned, we hypothesize that using frames might represent different information than using semantic information represented in the LOD-cloud. The books dataset we have at hand is particularly useful to verify this claim since all books have been mapped to their DBpedia URIs.

In order to do so, we relied on a manual subdivision of all books in genres based on LOD. This classification was made by [23] by parsing the abstract (`dbo: abstract`), the genre (`dbo:literaryGenre`, `dbp:genre`) and the subject (`dcterms: subject`) of each book against a regular expression pattern of thirty distinct genres. The authors performed this step to allow for more data coverage. However, by doing so they also made a combination of various LOD information categories which enables us to directly compare these with our semantic frames. If we have a look at our running example, *The Prince and the Pilgrim*,[8], we notice that this book is classified under the *Fantasy* genre.

Based on this genre mapping, we calculated the gain ratio [19] of our semantic frames representation with relation to the genres, thus considering the frames as features allowing to do genre classification. These gain ratios can then be observed as feature weights, and ranked according to the amount of information they add to discriminating between the thirty possible genres. We start our analysis by first only considering the semantic frame annotations. It became apparent, however, that it might be more interesting to also closer inspect those frames which are Events since these intuitively better represent what is actually happening.

The result of these analyses in presented in Table 4. Because of space constraints, we only represent the five genres representing most books of our dataset. This table each time contains the ten top features (frames and events), i.e. those with the highest gain ratio. The cell colour represents the manual analysis, indicated in light grey are those frames and events occurring only within one particular genre. In darker grey the frames and events which are representative for a specific genre are indicated. Regarding the frames, we see that it is more difficult to find distinctive features correlating with the genre (light grey). In the upper part, only the *Science Fiction* and *Crime* genre contain truly representative frames based on our manual analysis (dark grey). If we go to the level of the Events, we see that this already allows for finding more unique events per genre. Again, the *Science Fiction* and *Crime* genre are best represented. When we had a closer look at other discriminating features we found the same tendency. In the *Crime* genre, for example, other Events such as Verdict, Revenge, Execution, Robbery all appeared within the top twenty features.

From this analysis we could deduce that both the frames and events might deliver the same type of information as the LOD, with the events being more representative. However, what becomes clear is that the frames also contribute more information. They can represent what is happening within a book. If we again consider our running example (cfr. Table 2), which is classified as *Fantasy*, we feel that enriching a recommender with semantic frame, and especially with event information, might account for a better recommendation. This brings us to the actual experiments.

[8]http://dbpedia.org/resource/The_Prince_and_the_Pilgrim

4. EXPERIMENTS

For our experiments we focus on the generation of new, semantic features. In our experimental setting we aim to evaluate the contribution of those features and thus do not explicitly focus on engineering towards a top recommendation performance.

4.1 Experimental Set-Up and Evaluation

We opt to add our semantic features to an existing recommender system [23], which participated, and performed well, in the ESWC'14 Challenge. Though we do apply feature weighting and feature selection as described below, the overall item classification and collaborative-filtering elements of the base system remain unchanged. This allows us to directly compare the predictive power of the frame-based features with the DBpedia-based features used by the original system, in particular as both approaches are different utilizations of the same information source, i.e. Wikipedia, and dataset, i.e. the ESWC RecSys Challenge data.

We use a reduced version of the dataset, based on a filtering of the 5,063 books that were retained as having sufficient plot information available (Section 3.2). This dataset has binary ratings and consists of 53,665 user-item-rating triples (6,162 users, 4,251 items) in the training data and 50,654 triples (6,180 users, 4,311 items) in the evaluation dataset.

Even though this is a binary classification task, we opt to output the positive class likelihood and not the final binary classification in order to avoid making a decision about the cut-off for the likelihood values. Consequently, we evaluate with root-mean-squared error (RMSE) to capture also the degree of confidence between the classification and the gold-standard test dataset[9]. RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - x_i)^2}$$

in which X_i is the prediction and x_i the response value, i.e. the correct value for the task at hand, and m is the number of items for which a prediction is made. Speaking in practical terms, the lower the RMSE value the better,

[9]Obtained from the ESWC'14 Challenge Chairs upon request.

Table 4: Top ten features with the highest gain ratios in the five most popular LOD genres. Light grey cells represent genre-unique and dark grey ones genre-representative features.

	Fantasy	Science Fiction	History	Children	Crime
FRAMES	Jury_deliberation	Beyond_compare	Representing	Memorization	Extradition
	Bond_maturation	Becoming_dry	Intentional_traversing	Measure_area	Go_into_shape
	Intentional_traversing	Containment_relation	Dominate_competitor	Estimated_value	Exporting
	Cause_to_rot	Dunking	Getting_vehicle_underway	Rope_manipulation	Becoming_dry
	Get_a_job	Exclude_member	Cause_to_rot	Degree_of_processing	Arson
	Beyond_compare	Representing	Beyond_compare	Jury_deliberation	Measure_area
	Representing	Jury_deliberation	Probability	Bond_maturation	Dominate_competitor
	Locale_by_ownership	Cause_to_rot	Jury_deliberation	Intentional_traversing	Containment_relation
	Ratification	Medium	Color_qualities	Cause_to_be_dry	Reading_aloud
	Commutation	Cause_change_of_phase	Get_a_job	Drop_in_on	Extreme_point
EVENTS	Surrendering_possession	Change_of_consistency	Eventive_affecting	Intentionally_affect	Endangering
	Dodging	Immobilization	Historic_event	Examination	Posing_as
	Immobilization	Execute_plan	Extradition	Absorb_heat	Experience_bodily_harm
	Renting	Cause_impact	Surrendering_possession	Cause_to_experience	Enforcing
	Reparation	Reparation	Corroding_caused	Fighting_activity	Cause_to_be_wet
	Heralding	Eventive_affecting	Dodging	Dodging	Intentionally_affect
	Soaking	Get_a_job	Clemency	Rope_manipulation	Intercepting
	Intentional_traversing	Cause_to_be_sharp	Intentional_traversing	Intentional_traversing	Change_resistance
	Cause_to_rot	Cause_to_rot	Cause_to_rot	Drop_in_on	Go_into_shape
	Get_a_job	Cause_change_of_phase	Get_a_job	Cause_to_be_dry	Extradition

because the closer the prediction confidence to the actual gold standard.

In addition, again motivated by wanting to avoid to choose a cut-off point for the class assignment, we follow [12] and evaluate with a receiver operating characteristic (ROC) curve and also compute the area under the curve (AUC) for it. While in contrast to RMSE, the ROC curve is computed only on the relative ordering of the predictions sorted by confidence values, it offers the advantage of understanding how a classifier would perform given different cut-off values. In addition, with ROC we can compare against recommender systems that output only an (implicit) ranking and no class confidence values.

The base system by [23] we extend is a simple content-based recommender which trains two Naïve Bayes classifiers[10] on book features acquired from DBpedia, one global classifier as background model and one per-user classifier to capture individual preferences, trained on a user-neighborhood of variable size. In our experiments, we leave this setting unchanged but only vary the different features for item representation. We experimented with five different feature representations, which is explained in closer detail in the next section.

4.2 Feature Representation

1. Baselines

First, we established two baselines: the first baseline was constructed by including the majority class based on the training data, in our case the majority class is '0'. As a second baseline we decided to include a bag-of-words approach containing token unigrams from all the different plots.

The next three groups of features all relate to the frame representation of the plots based on the SEMAFOR output (cfr. Section 3.2)

[10]Even though being a simple approach, in a preliminary experiment Naïve Bayes outperformed an SVM, motivating us to not compare different classifiers but focus on feature selection. In addition, Naïve Bayes was – as expected – significantly faster compared to other classifiers.

2. Frames

For the frames as such, we decided to include the resulting frame names (e.g. Killing, Kinship, Leadership) as a separate setting. In total this can lead to a maximum of 877 discriminating features, which is a large feature space shrinkage compared to the bag-of-words representation. This is why we decided to also take into consideration those particular words evoking the frames, the Lexical Units (e.g. murdered, father, Prince) on the one hand, and the lexical representations of the Frame Elements – the semantic roles – evoked by this frame on the other hand (e.g. Prince Baudouin, by the King of Cronwall, King March). In a final setting, we incrementally combine these various elements of data, thus giving more information to our classifier.

3. Events

As was illustrated in Section 3.2 the Events occurring within a book seem to intuitively represent important information of what is actually happening. This is why we also decided to perform the same experiments as with the frames but, this time only incorporating those frames which have a possible Event parent somewhere in the FrameNet hierarchy. Looking only at the Events further reduced our feature space to a maximum of 234 features. We therefore also made the same combinations as mentioned above with all possible LUs and FEs relating only to Events.

4. Taxonomy

In order to exploit the hierarchical structure of FrameNet even further, we decided to also investigate three other settings. First we explored whether including besides a frame also its direct parent, thus going one level up in the graph, might help. We did the same in the other direction, by only including the children which are at the bottom of our taxonomy (the leafs). Another way of incorporating this graph information was to calculate for each possible frame pair that was found in a plot its least common subsumer [20] (LCS), i.e. the parent both frames have in common resulting

in the shortest path. Since the FrameNet taxonomy as such is not hypercomplex, i.e. the maximum distance between two frames is twelve, we decided to filter out those parents which are too generic by manually inspecting the LCS.[11]

For the four above-mentioned setups, the same feature selection methods were employed. Of course in order to allow for a good representation, all word-based features (bow, LUs and FEs) were first tokenized, stemmed and filtered on stop words. For the automatic feature selection, we first use unsupervised feature attribute weighting by computing the standard TF-IDF weights since all our features are in the end derived from text (book plots).

$$TF - IDF_i = \ln(1 + tf_i) \ln(N/df_i)$$

Next, we use attribute selection by computing the gain ratio with relation to the binary class label in the training data:

$$R_G(Attr, Class) = (\mathbb{H}(Class) - \mathbb{H}(Class|Attr))/\mathbb{H}(Attr)$$

This should allow us to filter out noise or unimportant features. We keep only those features with a gain ratio larger than zero ($R_G > 0$).[12]

5. Linked Open Data (LOD)

In a final setup we compare our best setting with the LOD features used by the base system, i.e. properties and values from DBpedia, and apply the same feature weighting and selection process. The features in the base system were manually selected and contain explicit book attributes, as e.g. `dbo:author` (`db:Umberto_Eco`), but also categorical information as `dbo:literaryGenre` (`db:Historical_novel`), `dcterms:subject` (`category:Novels_set_in_Italy`) or `rdf:type` (`yago:PhilosophicalNovels`) and untyped Wikipedia links in general.

We use the same set of features as reported by [23], but, to remain consistent across all experimental settings, apply our feature selection and weighting approach and use our reduced training and test dataset. In addition, we tested the combination of the DBpedia features with our best-performing frame approach.

5. RESULTS

We report experimental results for the different feature settings in Table 5. Overall, the two best performing frame features are the *Frame elements* and the *Frames+LUs+FEs*, both achieve an RMSE of 0.6036. We see that the best result is obtained when making the combination between the *Frame elements* and the LOD system, RMSE of 0.5982. Looking at the AUC for the ROC curve, both features still perform very well, but not as good as the DBpedia features alone, which achieve the best overall AUC of 0.5588.

Considering the RMSE values, we observe that the majority baseline is easily outperformed by all different settings. Looking at the bag-of-words baseline, however, illustrates that having the words of the plot available for recommendation is already a quite difficult to beat baseline.

[11]We looked at the most frequent LCS nodes and excluded the first 10 generic nodes such as Artifact, Relation, Intentionally_affect, Gradable_attributes, Transitive_action.

[12]Preliminary experiments revealed that keeping all features as well as doing classifier-based features selection with OneR [13] with 5-fold-cross-validation on the training data constantly underperformed against this setting.

Table 5: Experimental results on test dataset (N = 50,654) with classifier trained on different feature types (best results per category in bold).

	Features	RMSE	AUC
Baselines	Majority voting (0)	0.7705	n/a
	Words as such	**0.6145**	0.5431
Frames	Frames as such	0.6272	0.5377
	Lexical units (LUs)	0.6266	0.5398
	Frame elements (FEs)	**0.6036**	**0.5468**
	Frames + LUs	0.6259	0.5389
	Frames + LUs + FEs	**0.6036**	0.5453
Events	Events as such	**0.6132**	0.5148
	Events + LUs	0.6259	**0.5310**
	Events + LUs + FEs	0.6237	0.5296
Taxonomy	Frames One up	**0.6244**	0.5297
	Frames Bottom	0.6253	0.5370
	Frames + LCS	0.6285	**0.5376**
LOD	DBpedia features	0.6022	**0.5588**
	DBpedia + FEs	**0.5982**	0.5498
	(DBpedia + FEs hybrid)	(0.5664)	(0.5571)

Contrary to our expectations, our settings with only frames or events do not outperform this baseline. We do see that the events as such, which constitute a much smaller feature space, perform slightly better than the frames. The bag-of-words baseline is only outperformed when using features actually presenting some sort of word filtering mechanism: the Frame Elements are the lexical representation of words which are evoked by certain frames in the form of semantic roles. Even though these features are extracted from the text, it performs better than the bag-of-words *(Words as such)* baseline approach (0.6145) which does not make use of any semantic information. Analyzing the R_G-ranked feature attributes revealed that also for the other best frame approach *Frames+LUs+FEs*, the dominant attributes are the *Frame elements*, these were ranked highest. What is strange is that we do not find a similar trend when performing the same combination with our Event frames. This is probably because the feature space is too small to make a well-informed decision.

Figure 2 presents the ROC curves for our features, for the sake of readability only the most interesting curves are plotted. As to be expected from the AUC values, all curves are very close together. Besides not being far away from the diagonal, for no curve a clear cut-off value is recognizable. We observe that the DBpedia features are slightly better for the left and partially the middle part of the curve, leading to the interpretation that those features are superior for recommender systems which focus on quality. Comparing the best frames-based approach (FEs) with the bag-of-words baseline (Words), we see that FEs are mostly better than just words, with some exception around a false positive rate of around 0.23.

We also compare our system with the hybrid recommender system from Ristoski at. al. [21] (AUC 0.5848), which was the second best system of the ESWC challenge and performed essentially equally well as the winning system. That system combined many different features, not only LOD, but also user ratings and explicit collaborative filtering approaches.[13]

[13]As that system only outputs scores for the purpose of ranking, we transformed those into confidences by dividing each

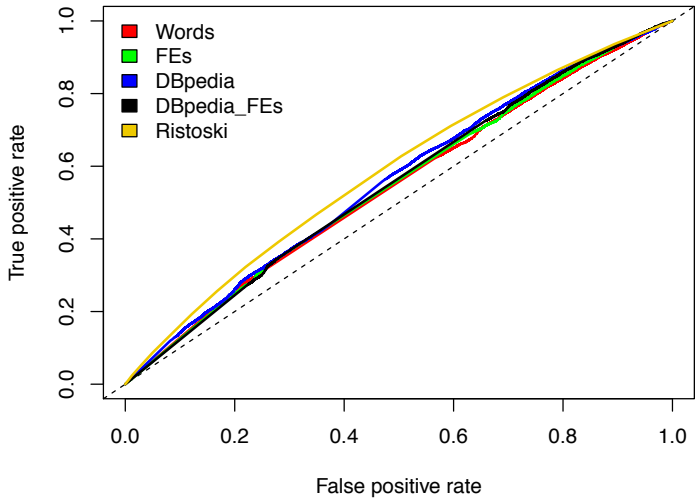


Figure 2: ROC curve for selected features and the top-performing ESWC system [21]

Looking at the ROC curve, it becomes clear that incorporating more and diverse features is beneficial in this setting. However, we have to note that this system combines different recommenders using the Borda rank aggregation method, which was not learned on the training data but manually selected while having knowledge about the test dataset (see also the comment below on our own combination model).

If we compare our best semantic frame results with the systems leveraging Linked Data, we see that we achieve a better performance (RMSE of 0.6022) when using the DBpedia features alone and that we get the best overall results when combining both our best system with the Linked Data (RMSE of 0.5982). In this way it appears that combining semantic information from different sources, i.e. from the linguistically grounded frames features and the explicit, ontology grounded DBpedia features, is beneficial in this setting. The AUC results, however, do not corroborate this finding.

Last, when not learning LOD+FEs together in one model, but separately and combine results with a simple linear combination (these results are presented in brackets in Table 5), as also done by [1], with $\lambda = 0.5$, we achieve better results (RMSE of 0.5664 and AUC of 0.5571). However, this notable improvement depends in the end on our knowledge of the test dataset, as it influenced our choice of a linear combination, instead of learning the combination of the different classifiers on the training data. Strictly speaking, this is thus not a valid experimental result, nevertheless it indicates there is most likely a better hybrid design with feature combinations that will better utilize the semantic frame features and should yield better results.

6. CONCLUSION

In this paper we have presented an alternative approach to add semantic information to a content-based book recommender system. We directly compared the addition of text internal semantic frame information with text external ontological information based on *Linked Open Data* (LOD), a popular research strand.

We have shown that parsing the book plots with a state-of-the-art semantic frame parser, SEMAFOR, delivers valuable additional semantic information. This information could enable a system to fully grasp what is happening within a book. One of the added values of FrameNet is that all frames are related in a taxonomy which allows you to pinpoint those Events forming the key components of a book. Based on a direct comparison between the frames and events and a list of genres derived from DBpedia attributes, we have shown that although these data sources show some similarities, the semantic frames should be able to represent more specific information about what is happening in a particular book.

In order to test this claim in closer detail, we have performed experiments where the focus was on generating new semantic features and find out what these can contribute to a book recommendation system using one global classifier as background model and one per-user classifier. We see that exploiting semantic frame information outperforms a standard bag-of-words unigram baseline and that especially incorporating frame elements and lexical units evoking the frames allows for the best overall performance. If we compare our best system to a system levering semantic LOD information, we observe that our frames approach is not able to outperform this system. We do find, however, that if we combine these two semantic information sources into one system we get the best overall performance. This might indicate that combining semantic information from different sources, i.e. from the linguistically grounded implicit frame features and the explicit, ontology grounded DBpedia features, is beneficial.

This work has inspired many ideas for future work. Con-

score by a constant.

sidering the current setup, we are aware that we completely relied on the output of one semantic frame parser, i.e. SEMAFOR. We believe that using a filtering mechanism beforehand, e.g. to filter out those frames and or events which are less meaningful or noisy, or that by applying a different parser or event extraction techniques new lights can be shed on the added value of this type of information. Also, since we now only relied on Wikipedia to extract book information, we had to reduce an original larger book data. We realize a lot of additional information about books can be found online, for example on Google Books, Amazon, GoodReads, etcetera. Also the same techniques can be used to extract other types of information from both the items and users under consideration for the recommendation task.

As mentioned at the end of Section 5 we would like to further investigate whether another hybrid design might yield better results. In this respect, it would be interesting to plug our semantic knowledge in a collaborative-filtering approach to see whether this can actually help the overall performance. Using our semantic frames we could also inspect in closer detail typical problems recommender systems face such as cold-start and data sparsity.

Acknowledgments

The work presented in this paper has been partly funded by the PARIS project (IWT-SBO-Nr. 110067). Furthermore, Orphée De Clercq was supported by an exchange grant from the German Academic Exchange Service (DAAD STIBET scholarship program). We would like to thank Christian Meilicke for his help in providing the manually derived genres and his help with building the original ESWC recommender system.

7. REFERENCES

[1] C. Basu, H. Hirsh, W. Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI'98*, pages 714–720, 1998.

[2] D. Billsus and M. J. Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180, 2000.

[3] M. Degemmis, P. Lops, and G. Semeraro. A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. *User Modeling and User-Adapted Interaction*, 17(3):217–255, 2007.

[4] T. Di Noia, R. Mirizzi, V. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *I-SEMANTICS '12 Proceedings of the 8th International Conference on Semantic Systems*, 2012.

[5] D. Dipanjan, A. F. T. Martins, N. Schneider, and N. A. Smith. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56, 2014.

[6] M. Eirinaki, M. Vazirgiannis, and I. Varlamis. Sewep: using site semantics and a taxonomy to enhance the web personalization process. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[7] C. J. Filmore. Frame semantics. *Linguistics in the Morning Calm*, pages 111–137, 1982.

[8] C. J. Filmore. Frames and the semantics of understanding. *Quaderni di Semantica*, IV(2), 1985.

[9] C. J. Filmore, C. R. Johnson, and M. R. L. Petruck. Background to framenet. *International Journal of Lexicography*, 16(3):235–250, 2003.

[10] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical intelligence*, 2007.

[11] B. Heitmann and C. Hayes. Using linked data to build open, collaborative recommender systems. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010.

[12] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[13] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.

[14] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2013.

[15] P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[16] G. Martín, S. Schockaert, C. Cornelis, and H. Naessens. An exploratory study on content-based filtering of call for papers. *Multidisciplinary Information Retrieval, Lecture Notes in Computer Science*, 8201:58–69, 2013.

[17] V. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of RecSys 2013*, 2013.

[18] A. Passant. dbrec: music recommendations using dbpedia. In *Proceedings of the 9th International Semantic Web Conference (ISWC'10)*, 2010.

[19] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[20] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence (IJCAI'95)*, 1995.

[21] P. Ristoski, E. L. Mencia, and H. Paulheim. A Hybrid Multi-Strategy Recommender System Using Linked Open Data. In *LOD-enabled Recommender Systems Challenge (ESWC 2014)*, 2014.

[22] J. Ruppenhofer, M. Ellsworth, M. R. L. Petruck, and C. R. Johnson. Framenet ii: Extended theory and practice. Technical report, 2005.

[23] M. Schuhmacher and C. Meilicke. Popular Books and Linked Data: Some Results for the ESWC-14 RecSys Challenge. In *LOD-enabled Recommender Systems Challenge (ESWC 2014)*, 2014.

A Hybrid Strategy for Privacy-Preserving Recommendations for Mobile Shopping

Toon De Pessemier
iMinds-WiCa-Ghent University
G. Crommenlaan 8 box 201
B-9050 Ghent, Belgium
toon.depessemier@ugent.be

Kris Vanhecke
iMinds-WiCa-Ghent University
G. Crommenlaan 8 box 201
B-9050 Ghent, Belgium
kris.vanhecke@ugent.be

Luc Martens
iMinds-WiCa-Ghent University
G. Crommenlaan 8 box 201
B-9050 Ghent, Belgium
luc1.martens@ugent.be

ABSTRACT

To calculate recommendations, recommender systems collect and store huge amounts of users' personal data such as preferences, interaction behavior, or demographic information. If these data are used for other purposes or get into the wrong hands, the privacy of the users can be compromised. Thus, service providers are confronted with the challenge of offering accurate recommendations without the risk of dissemination of sensitive information. This paper presents a hybrid strategy combining collaborative filtering and content-based techniques for mobile shopping with the primary aim of preserving the customer's privacy. Detailed information about the customer, such as the shopping history, is securely stored on the customer's smartphone and locally processed by a content-based recommender. Data of individual shopping sessions, which are sent to the store backend for product association and comparison with similar customers, are unlinkable and anonymous. No uniquely identifying information of the customer is revealed, making it impossible to associate successive shopping sessions at the store backend. Optionally, the customer can disclose demographic data and a rudimentary explicit profile for further personalization.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Filtering; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

Keywords

Recommender System, Shopping Assistant, Privacy, Mobile

1. INTRODUCTION

Data gathering and analysis, i.e. one of the fundamentals of traditional recommender systems, is a serious concern for many, increasingly privacy-aware users. A data collector may disclose personal information to untrusted parties. This could occur either on purpose, i.e., by selling the personal information to a third party, or involuntarily through a security breach. As a result, users are becoming apprehensive about using applications or services that collect personal data. For shopping applications for example, many customers are already concerned with the data collection practices related to loyalty programs [8, 11]. This can be exacerbated when the loyalty program is an application running on the customer's own smartphone. These devices contain a large amount of personal data such as the customer's phone number, e-mail address, and social networking account details.

Despite these privacy concerns, Mobile Shopping Assistants (MSAs) are becoming increasingly popular due to the benefits they offer to both customers and retailers. An MSA can enhance the shopping experience by incorporating features such as loyalty programs, discount vouchers, easy checkout, and various personalized services. The applications are easy and inexpensive to roll out because they can run on the customer's own smartphone. The retailer does not have to invest in specialized hardware and many customers are already familiar with smartphones and the concept of mobile apps. To address these privacy concerns, Put et al. [9] have created inShopnito, a transparent, privacy-preserving MSA that still offers all the features that customers and retailers have come to expect, including a rudimentary recommender system. In this paper, we have extended the MSA with an advanced, hybrid recommendation strategy. Section 3 describes this contribution in detail. As the security and privacy-enhancing technologies used for (anonymous) authentication and transactions have already been described [9], Section 2 of this paper provides only a brief overview of the functionality and the implications of privacy-preserving measures on recommendations.

2. PRIVACY-PRESERVING MOBILE SHOPPING

Preserving the customer's privacy during the usage of inShopnito is of primary importance, which has significant implications for the recommender. At registration time, the customer is issued an Idemix [5] anonymous credential containing attributes with personal information such as name, zip code, or gender. When the customer enters the store, the inShopnito MSA uses the credential to initiate a new shopping session on the store backend system. The customer chooses which attributes (name, zip code, gender, explicit profile) to disclose during this authentication phase. These different levels of privacy provide the customer the necessary

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

flexibility in the trade-off between privacy and personalization. The backend system knows that the customer has a valid credential, and it knows the content of the attributes that the customer opted to disclose. However, it does not know which particular customer it is dealing with, because no uniquely identifying information is disclosed during authentication. This also means that a customer's successive shopping sessions can not be tied together.

The customer can now proceed to scan products using the camera of her smartphone and add them to the inShopnito shopping cart. During checkout, inShopnito can be used to redeem loyalty points and vouchers to get a discount. To provide the customer with a complete overview of her shopping history, the MSA stores this information securely on the smartphone where it can be used for recommendation purposes. In Section 3, we expand on the recommender components of the inShopnito MSA and backend, and propose a practical solution that preserves the privacy of the customer while still offering advanced personalized services.

Recommender systems initially face the cold start problem, because nothing is known about the user [7]. Usually, a user's actions can be tracked over time. As more information about the user becomes available, the quality of the recommendations increases. With inShopnito, each shopping session is associated with a different, anonymous user identifier. Thus, a server-side recommender system will always have to address the cold start problem, whether it is the customer's first store visit, or her hundredth. Client-side recommenders pose their own set of challenges [3].

Related research [4] into privacy-preserving, personalized ad delivery has proposed a coarse-grained filtering of ads based on the personal information that customers choose to disclose. Subsequently, a further filtering can be performed at client side based on the purchase details stored on the customer's smartphone. Compared to existing solutions, the hybrid recommender strategy of inShopnito goes further than a filtering of information, by analyzing individual shopping carts and comparing them with the purchases of similar customers at the backend.

3. HYBRID RECOMMENDATIONS

The hybrid strategy combines five recommendation approaches. For each approach, preserving the customer's privacy is of crucial importance. Figure 1 provides a schematic overview of these approaches and the data they use.

3.1 Explicit Profile Recommendations

Through an explicit profile on her smartphone, the customer can specify her preferred product categories, as shown in Figure 2(a). These categories, grouping individual products that are typically located in the same section of the store, allow customers to quickly express their interests and filter out irrelevant product groups. Product categories such as pet supplies, garden tools, car/motorcycle supplies, toys, or baby products, are not relevant for every customer. This explicit profile is created automatically based on the customer's purchases, but can be altered at her own discretion. Although the explicit profile contains only category preferences and no details regarding individual products, disclosing the explicit profile is an optional feature for privacy reasons. In addition, customers can opt to disclose some demographic data such as age, municipality, and gender, in order to further filter the product categories such as shaving

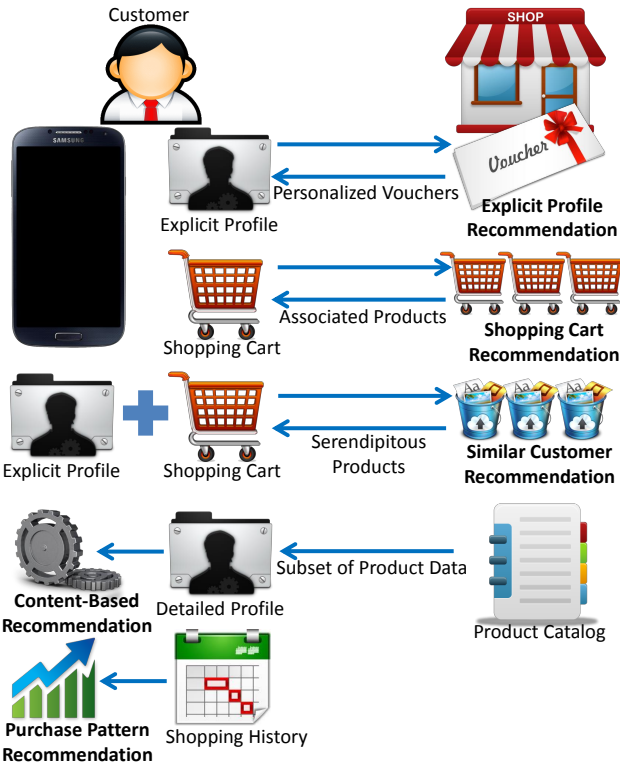


Figure 1: Schematic overview of the different recommendation techniques.

products, make-up, personal care products, etc.

If the customer opts to disclose (parts of) her explicit profile and demographic data, this information is sent to the store backend and used to personalize the *coupons and vouchers* she receives. These targeted vouchers have benefits for the retailers (the vouchers are more effective) as well as for the customers (more relevant vouchers are offered). For privacy reasons, these explicit profiles are only used during the current sessions and removed from the store backend after issuing the vouchers.

3.2 Shopping Cart Recommendations

During every shopping session, the content of the shopping cart is sent to the store backend for analysis. For privacy reasons, no uniquely identifying reference to the customer is stored at server side. Only the content of the individual shopping carts, together with the date of the purchase, are stored. The date provides useful information regarding trends in purchasing behavior, or seasonal products. A detailed timestamp with the exact moment of the purchase (hours/minutes) would have no extra value for the recommender and is omitted because this might induce a privacy risk. If a customer's time of purchase is known (e.g., by observation) and the exact timestamp would be stored, linking the content of the shopping cart to the customer's identity would be possible.

Analysis of the content of the shopping carts of different customers provides insight into the shopping habits of the customers and reveals which products are often bought together. For instance, customers will buy both pasta and

bolognese sauce if they intend to prepare spaghetti or lasagna. Product association rules are used to discover which products belong together. Interesting recommendations are products that are not yet added to the shopping cart, but are often bought in combination with the products that are in the shopping cart. So, if bolognese sauce is in the shopping cart, pasta is a good recommendation. More generally, the best recommendation is the product, Y, with the highest probability to be bought, given the current content of the shopping cart, X. Here, X can be a single product or a set of products that the customer wants to buy.

However, highly popular products will always be bought in combination with a large variety of other products, even though no direct link (e.g., a recipe) exists between them. The probability that the customer will buy these popular products is always high, regardless of the content of the shopping cart. In order to take into account the general popularity of products, this probability, $P(X, Y|X)$, is normalized by dividing it by the probability of buying Y, if the content of the shopping cart is different from X. The products with the highest normalized probability are recommended to the customer, as illustrated in Figure 2(b).

$$\underset{X \subset Cart}{\text{Max}} \frac{P(X,Y|X)}{P(!X,Y|!X)} = \underset{X \subset Cart}{\text{Max}} \frac{\frac{P(X,Y)}{P(X)}}{\frac{P(!X,Y)}{P(!X)}} \quad (1)$$

In addition to these automatically derived combinations of products using product association rules, domain knowledge helps to recommend the best matching products. For the shopping cart recommendations, the domain knowledge consists of a set of recipes. If the customer's shopping cart already contains several products that match the ingredients of a certain recipe, the missing ingredients are recommended and the recipe is suggested to try out. Since these recommendations do not require a user profile with an extensive purchase history, they can help to overcome the *cold start problem*.

3.3 Similar Customer Recommendations

Storing the customers' individual consumption behavior on a central server induces a privacy risk and is therefore undesirable. With inShopnito, each shopping session has a different, anonymous user identifier, and successive shopping sessions cannot be linked (Section 2). Because collaborative filtering is based on calculating the similarity between the historical consumption behavior of individual users (or products), a traditional collaborative filtering approach is not possible in this situation.

As an alternative, customers are compared based on their explicit profile, which is voluntarily disclosed and contains only data about product categories but not of individual product purchases. Calculating the similarity between customers based on their explicit profile might be less accurate than based on their complete consumption behavior; but this approach induces no privacy risk. Based on this explicit shopping profile, customers are partitioned into groups of similar customers, just as the neighborhoods of similar users in the traditional collaborative filtering approach. Each group is represented by a bucket that contains all products that have been bought by the customers of that group.

After every visit to the store, the content of the customer's shopping cart is added to the bucket of the customer's group. If two customers have a similar explicit profile, their shopping carts will end up in the same bucket. Since the bucket contains only product information and no link to the identity of the customer, the purchasing history of an individual customer cannot be deduced if the bucket groups purchases of many customers. Analysis of the products in the bucket of the customer allows to generate recommendations based on what people who like similar products have bought in the past. The products that are most popular with other customers of the group are recommended, with the exception of products that are already in the shopping cart of the customer. The popularity of products within a group is normalized with respect to the general popularity of a product. These recommendations, based on the purchases of similar customers, aim to offer more *serendipitous recommendations* to the customers, just as collaborative filtering algorithms do.

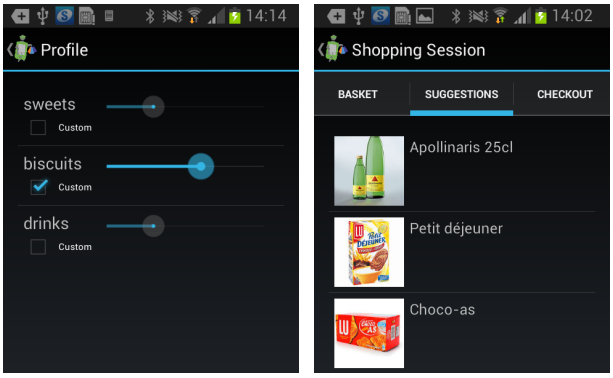
This recommendation technique can also be combined with the approach that compares shopping carts (Section 3.2). Individual shopping carts (without a reference to the customer's identity) can be stored per group of similar customers, as defined by the explicit profile. Subsequently, product association rules can be applied on the groups of similar customers, instead of on the complete population of customers. This partitioning of customers according to their preferences can help to refine the product association rules.

3.4 Content-based Recommendations

For privacy reasons, detailed historical information about purchases cannot leave the secured environment of the customer's smartphone. As a result, these detailed purchase data can only be exploited if the recommendation algorithm runs on the customer's smartphone. In this customer-centric personalization approach [1], each user has its own mobile recommendation engine. Storing this detailed user profile securely on the smartphone also has advantages. For instance, the user profile can be shared amongst different shops without the privacy risk that one retailer abuses these purchase data for commercial profits.

Since only purchase data of the target user (i.e. the user for who recommendations are calculated) are available on the smartphone, a content-based recommendation algorithm is the most worthwhile solution to process this detailed profile.

Content-based recommendation algorithms determine the products that best match the user's profile, based on a description of the product characteristics [6]. Since the detailed profile cannot leave the customer's smartphone, the product descriptions have to be transferred to the smartphone for comparison with the detailed profile. However, the complete product catalog of the store and the corresponding descriptions can be quite extensive for processing on a smartphone. Therefore, only products and descriptions of categories that are relevant for the customer are sent to the smartphone to reduce the data traffic. Recommendations for pet supplies may be irrelevant for customers who have never bought any pet supplies in the past. They may not have pets, or buy their supplies through other channels. The explicit profile is used to determine which categories are relevant and have to be considered. The resulting subset of the product catalog has to be downloaded only once, the first time that the customer visits the store. From then on, updates of the catalog are sufficient to keep track of new products, changed descriptions, and products that are not



(a) (b)

Figure 2: Screenshots of the mobile application: (a) the explicit user profile (b) the personalized suggestions.

available anymore.

Different types of content-based recommendation algorithms can be used, but with the limitation that the computational requirements must fit within the available resources of the smartphone. Our approach uses the InterestLMS algorithm of the Duine recommender framework [10]. Content-based algorithms often suffer from over-specialization [7], since they recommend only products similar to those already bought by the customers. In certain application domains, items should not be recommended if they are too similar to something the user has already seen, such as a different news article describing the same event. For shops however, various situations exist in which customers are interested in similar products: cheaper or discounted products of a different brand, new or similar food products to replenish their house stock, or alternatives for products that are out of stock.

3.5 Purchase Pattern Recommendations

The last type of recommendations focuses on the repetitive purchase behavior of customers [2]. Specific products, such as toothpaste or coffee, are used on a regular basis, and as a result, need to be replenished regularly. Patterns in the purchase behavior can be detected, and used to predict the next purchase of a certain product. E.g., if one tube of toothpaste is bought every month, predicting the next purchase of toothpaste is obvious.

Based on the shopping history (i.e. the time and amount of the last purchase), the recommender estimates if the customer needs to buy a certain product. If this is the case, and the customer has not yet added the product to the shopping cart, it will be recommended. So, the aim of the purchase pattern recommendations is to *remind customers* to buy products that they might forget but probably need because of their *repetitive consumption behavior*.

4. CONCLUSIONS

The growing importance of privacy in online services emphasizes the need for privacy-preserving recommender systems, not the least in the domain of shopping. Traditional collaborative filtering algorithms, which rely on a central storage and comparison of detailed user profiles, may induce a privacy risk. But limiting the disclosed customer data introduces a trade-off between the accuracy of the recommendations and the privacy of the customer. Therefore, we present a privacy-preserving, hybrid strategy that combines client-side and server-side recommendation techniques. At server-side, the recommender is based on information that customers opt to disclose, and performs an analysis of the shopping cart using product association rules, and a comparison with the shopping carts of similar customers. At client-side, detailed customer information is used for content-based recommendations and suggestions based on purchase patterns.

5. ACKNOWLEDGMENTS

This research was funded by the IWT-SBO Project MobCom: A Mobile Companion (https://www.mobcom.org)

6. REFERENCES

[1] G. Adomavicius, Z. Huang, and A. Tuzhilin. Personalization and recommender systems. *Tutorials in Operations Research, Informs*, pages 55–107, 2008.

[2] H. Baumgartner. Repetitive purchase behavior. In A. Diamantopoulos, W. Fritz, and L. Hildebrandt, editors, *Quantitative Marketing and Marketing Management*, pages 269–286. Gabler Verlag, 2012.

[3] L. N. Cassel and U. Wolz. Client side personalization. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, pages 8–12, 2001.

[4] M. Hardt and S. Nath. Privacy-aware personalization for mobile advertising. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 662–673, New York, NY, USA, 2012. ACM.

[5] IBM Research Security Team. Specification of the Identity Mixer Cryptographic Library v. 2.3.4. Technical report, 2012.

[6] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[7] P. Lops, M. Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011.

[8] V. Pez. Negative effects of loyalty programs: An empirical investigation on the french mobile phone sector. Technical report, Université Paris-Dauphine, 2007.

[9] A. Put, I. Dacosta, M. Milutinovic, B. De Decker, S. Seys, F. Boukayoua, V. Naessens, K. Vanhecke, T. De Pessemier, and L. Martens. inshopnito: An advanced yet privacy-friendly mobile shopping application. In *Proceedings of the IEEE 10th World Congress on Services (SERVICES 2014)*. IEEE, 2014.

[10] Telematica Instituut / Novay. Duine Framework, 2009. Online available at `http://duineframework.org/`.

[11] S. Worthington and J. Fear. The hidden side of loyalty card programs. *The Austalian centre for retail studies*, 2009.

A User-centered Music Recommendation Approach for Daily Activities

Ricardo Dias Ricardo Cunha
INESC-ID, Instituto Superior Técnico,
Universidade de Lisboa
Lisboa, Portugal
{ricardo.dias,
ricardo.d.cunha}@tecnico.ulisboa.pt

Manuel J. Fonseca
Faculdade de Ciências,
Universidade de Lisboa
Lisboa, Portugal
mjf@di.fc.ul.pt

ABSTRACT

The number of songs available on the Internet has grown steadily over the last decade, with the recent growth being due mainly to streaming services. As a consequence, it is extremely difficult for users to find the appropriate music that suit their needs, in particular, while using systems that do not have any previous information about them. This is further exacerbated while selecting appropriate songs for daily activities, like shopping, running or sleeping. In this paper we describe Improvise, a personalized music recommendation solution for daily activities, whose approach associates music content (acoustic features) with activities (context). Each activity is characterized by determining intervals for each content feature, which are then used to filter out songs to be suggested to users. While the initial intervals are generic enough to provide recommendations for different activities without having previous knowledge about the user's tastes, our approach also considers users' feedback to personalize the recommendations for each user and activity. This is done by adapting the intervals according to the feedback from users. Preliminary evaluation shows that we are on the good path to achieve the goal of developing a solution to effectively recommend songs for daily activities, and able to adjust to individual user's tastes, increasing their satisfaction.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Design, Algorithms, Human Factors, Experimentation

Keywords

User-centered, Music Recommendation, Content, Context, Daily Activities, Cold-Start

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

1. INTRODUCTION

Over the last decade, due to the increasingly easy access to online music streaming services, people have gained the opportunity to listen to millions of songs over the Internet and almost everywhere. However, the opening to this broad spectrum of songs lead people to feel paralyzed and doubtful [17], as it gets harder for them to filter out the songs they would enjoy the most, specially while performing other activities, such as practicing sports, driving or studying [11]. Due to the large number of songs that users have access, it is hard for them to select the most appropriate songs for their activities.

To reduce the burden of choosing among too many songs, researchers have focused on creating recommender systems that can automatically generate recommendations that fit users' preferences. Celma's extensive work on recommendation [5] classifies recommender systems into five typical categories: *Demographic Filtering, Collaborative Filtering, Context Aware Filtering, Content-Based Filtering* and *Hybrid Methods*. Recently, hybrid and context-aware approaches have gained relevance amongst researchers, as they agree that listening patterns can be influenced by different factors, such as temporal properties [7], location, emotions and the activity a listener is engaged in [21].

Concerning music selection for daily activities, several approaches have been proposed. In [21], the authors created a mobile system that is able to detect what activity the user is performing and select the appropriate music for it, based on the time of the day, accelerometer data, and audio from the the microphone. Lifetrak [15] is a context-aware playlist generator that automatically chooses music in real-time based upon the location, the pace of movement, the current time, and other phenomena in the users environment. It uses a simple learning mechanism to adjust the ratings of songs for a particular context based on users feedback when a song is being played. However, these approaches are still very impersonal with little control, lacking users involvement through the whole steps of the recommendation process, which could improve the users satisfaction and confidence [9, 20].

In this work we describe Improvise, a user-centered recommendation approach for daily activities. Improvise is a recommendation model developed by characterizing activities in terms of content features, by defining their boundaries.

The set of intervals generated for each feature and activity are then used to filter out songs to be suggested. We developed a generic model with this approach (to be used initially by all users) by gathering data from several users. Individual personalization of this model is created over time by taking into account user's feedback for each activity, and consequently adapting the intervals based on the features from the new selected songs.

Preliminary experimental evaluation revealed that the initial generic model was able to suggest songs to daily activities for different users, without any knowledge about them. Moreover, by using the user feedback (selected songs from the recommendation list) to personalize the recommendation model, increased the number of adequate songs for each activity in 25% (while compared to the generic model) and the users satisfaction.

We highlight as main contributions the following: i) a user-centered personalized solution to perform recommendations for daily activities; ii) an alternative approach for solving the cold-start problem.

In the next section we discuss the related work, presenting research made in music recommender systems. Section 3 describes the proposed solution for music recommendation, and in Section 4 we present the experimental user evaluation and discuss the preliminary results of our work. Finally, in Section 5 we present future work and conclude this research.

2. RELATED WORK

Several approaches for music recommendation have been developed so far. Depending on the type of data used to perform the recommendations, we can categorize the methods in different groups [5], either if they use demographic data [22], listening habits and ratings [10, 6], content information from songs [4], the context when they were listened [12, 19], or any combination of the previous [18, 16]. Although collaborative filtering solutions have been the most widely researched techniques in the past, nowadays, a huge effort has been put on techniques that capture the context around the listening activity [16], because they can provide insights about the reasons that lead users to listen to certain songs.

Content-based approaches use the description of songs to compute similarities and recommend songs similar to the user favorite ones or to chosen seeds. These approaches solve the *early-rater* and popularity bias problems, as all the items are considered to be of equal importance [5] (without human intervention). However, a potential problem of these approaches is the *novelty* problem. Assuming that the *similarity function* works accurately, one might assume that a user will always receive items too similar to the ones in his profile. To cope with this problem, recommenders should use other factors to promote the *diversity* and *novelty* of the recommended items. In the solution proposed by Cano [4], acoustic features of songs (*timbre, meter and rhythm patterns*) were used for recommendation. Daniel M. [13] used *lyric feature analysis* to find similar items that describe *race conflicts* and *social issues*. In [3], Cai recommends music based only on emotion.

Context can be defined as any information that can be used to characterize the listening process [1], such as, the place where we are listening to the music, the time of day, the activity we are performing, etc. Context aware recommendation systems (CARS) use *context information* to describe and characterize the songs or artists we listen to. For example, Su et al. [19] improved Collaborative Filtering (CF) methods combining user grouping by location, motion, calendar, environment conditions and health conditions, while using content analysis to assist the system in the selection of the appropriate songs. On the other hand, Park [14] developed a modified User-based CF method (called Session-based CF), where users were replaced by sessions, adding a temporal dimension to CF recommendations. In [12], Liu et al. took the change in the interests of users over time into consideration and added time scheduling to the music playlist. Baltrunas et al. [2] introduced a new context-aware recommendation approach called user micro-profiling, where the user profile is split into several sub-profiles, each one representing the user in a particular context. The authors stated that the choice of songs during the day is influenced by contextual conditions, such as, the time of day, mood or the current *activity* listeners perform. In [23], the authors presented a novel and improved statistical model for characterizing user preferences in consuming social media content. By taking into account information about listening sessions of individual users, they have arrived at a new session-based hierarchical graphical model that enhanced individual user experience.

In short, there have been some effort from researchers to create automatic mechanisms that characterize users preferences through the use of different sorts of data, like temporal patterns, emotions, or choices behind song selection for particular activities. On the other hand, content features have been extensively used for recommendation and playlist generation because of the benefits they present. Despite that, there has been little work on engaging users through the recommendation process, giving them the control over how profiles are created and managed. We intend to tackle this gap by developing a recommendation approach based on user input and feedback as well as on content features, for creating a customizable recommendation model for each user.

3. IMPROVISE

In this research we describe Improvise, a personalized recommendation system able to suggest songs that fit the users needs while performing daily activities. To achieve this goal we followed a user-centered approach, taking advantage of users' input and feedback to develop a generic model capable of recommending songs to everyone, even without any previous knowledge about them. Activities were characterized using content-based features. Five activities were considered based on the existing related work [21]: *walking, relaxing, running, sleeping and shopping.*

3.1 Approach Overview

Figure 1 shows the different steps for creating the recommendation models, explaining the recommendation process. First, we associate songs with activities by using the user input gathered through a web-application (Figure 1-1). This allowed us to analyze what songs were more suitable for each activity (based on the users preferences) and thereby create

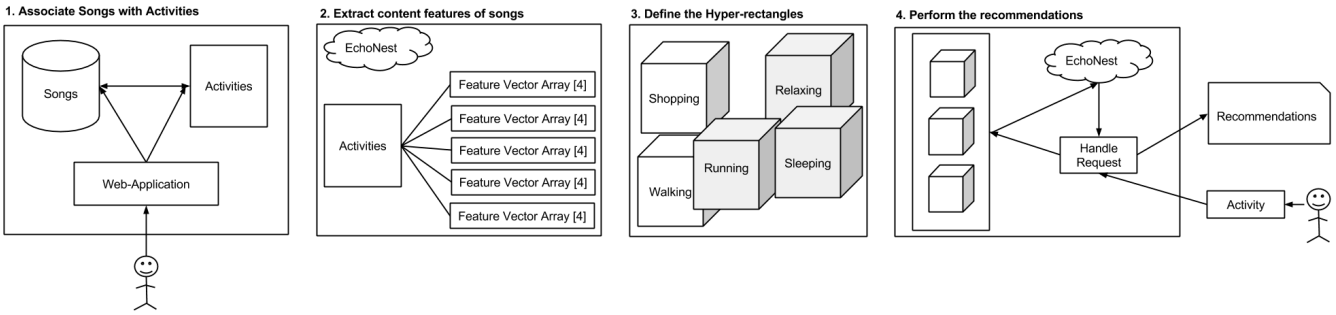


Figure 1: The different steps in Improvise to generate recommendations.

the broad and generic recommendation model. To this end, we then gathered content features from the songs, using the *EchoNest*[1] service (Figure 1-2). The values for each feature were collected to characterize each activity, by inferring intervals of values for each feature. These intervals define a set of *hyper-rectangles* (Figure 1-3), which Improvise uses to generate recommendations by filtering out songs from the *EchoNest* service (Figure 1-4).

To personalize the recommendation model we consider the songs selected by the users as appropriate for each activity (user's feedback), extract the content properties for these new songs and recalculate the hyper-rectangles for each activity, repeating steps 2 and 3. By using this approach our solution adapts the recommendations to the users' tastes and preferences over time.

In the following sections we provide details for the different steps described here.

3.2 Association between songs and activities

To develop a music recommendation system able to suggest songs suitable for different activities is necessary to capture the users' tastes and preferences for those activities. For example, to understand the reasons *"why do users select certain songs for running, and others for relaxing?"*. Although different criteria can influence this selection, some conceptual properties are shared among users for the same activities, such as, familiarity or distraction. However, these sort of more subjective features are difficult to extract and encode. On the other hand, content-based features have been used for some time in retrieval and recommendation systems [4, 13], presenting some advantages: they can be automatically extracted and used to compute the similarity between songs; they help solving the problem of cold-start for new songs. Therefore, they constitute a good approach to characterizing activities and empower a recommendation solution.

To associate songs with activities and thus characterize them by using content-based features, we took a user-centered approach. To that end, we developed a web-application to collect songs that users enjoy listening to, while performing each activity (see Figure 1-1). The users selected songs first by filtering genres, then artists, and finally by songs (see Figure 2). Regarding genres, we adopted the taxonomy used by the majority of digital music services (like Musicovery[2]), showing only 15 different genres (Rock, Electro, Pop, R&B, Rap, Metal, Classic, etc.). After selecting one or more genres suitable for each activity, users could choose from 30 artists in maximum (twice the number of genres). Next, users could finally choose songs from the artists previously selected (a maximum of 100 songs were shown). Top artists and songs were used for the selection, gathered through the online service EchoNest. The result of this process was an association between activities and a set of songs suitable to be listened by different users.

To collect this data we sent emails to contacts and spread the link for the application through social networks, namely, *Facebook* and *Google+*, to reach as many users as possible. 98 subjects used the application, providing a total of 251 answers for all the activities. Despite the fact that some users did not provide feedback about their tastes for all activities, the distribution was uniform: 55 answers for the activity *walk*, 53 for *running*, 47 for *sleeping*, 48 for *relaxing* and 48 for *shopping*. This resulted in associating *8,518* songs with the activities.

To characterize the activities we extracted content information from the songs selected by users using the *Echonest* service. For performance issues, we opted to use only the top-100 preferred songs for each activity.

After extracting all the features offered by EchoNest, we performed an evaluation to measure how discriminative each feature was in this characterization. This evaluation was performed using the *CfsSubsetEval* attribute evaluator along with the *best-fit* search method from Weka[3] [8]. The following four features were selected as the most discriminative: *accousticness*, *energy*, *loudness* and *tempo*. Therefore, for each song we created a 4-feature vector describing its content, and for each activity an array of feature vectors of the songs associated with them (see Figure 1-2).

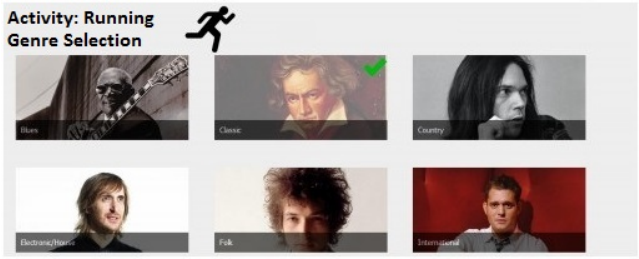
3.3 Generic Recommendation Model

The association detailed in the previous subsection allows us to describe each activity through a set of feature-vectors, representing each vector a song chosen by the users. To use this information in the suggestion of songs we need to convert it into a simpler representation to facilitate the rec-

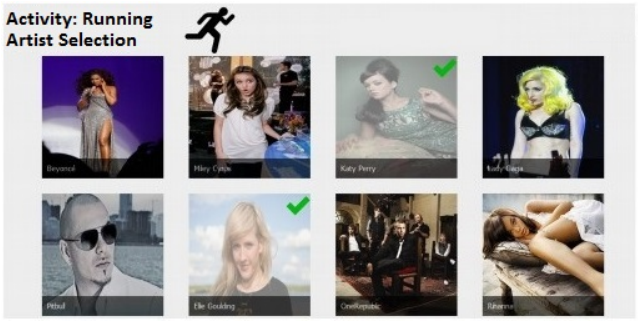
[1]http://the.echonest.com/

[2]http://musicovery.com

[3]http://www.cs.waikato.ac.nz/ml/weka/



(a) Genre selection.



(b) Artist selection.

(c) Songs selection.

Figure 2: Web-application developed to associate songs with activities.

ommendation process.

Typically, a recommendation system is often seen as a suggestion of items similar to a feature vector that represents the users' tastes. However, this approach is very restrict, since it will tend to recommend the same set of songs every time. Based on this and on the mechanism we are using to characterize each activity, we decided to use an interval approach for the recommendation.

To this end, we defined a set of intervals delimiting the value that each feature could take, instead of considering a single point in space (computed using a clustering algorithm, for instance). Moreover, this approach not only increases the range of songs that we can suggest for each activity, but also gives the possibility of using different sub-intervals to restrict the filtering process. This set of intervals define what we labeled as the *hyper-rectangle* (see Figure 1-3). A hyper-rectangle has four dimensions, and is defined by intervals with a maximum and a minimum value that each feature can take within each activity. The size and position of these rectangles differ between activities and for each user, pro-

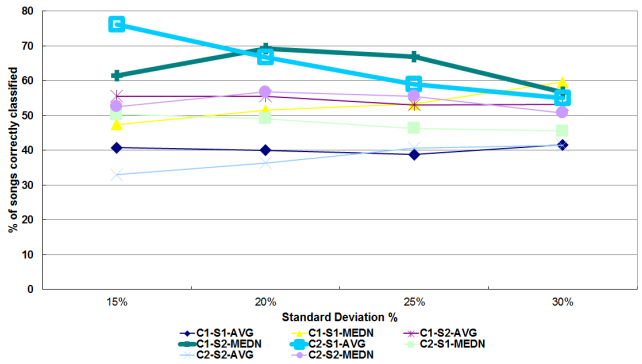


Figure 3: Method for determining the hyper-rectangle limits.

viding an adaptable recommendation mechanism as detailed later in the paper. The hyper-rectangles represent the backbone of Improvise. To recommend songs using them, we search for songs within the limits of the hyper-rectangles, using the *EchoNest* service.

Our generic recommendation approach consists in creating five generic hyper-rectangles, one for each activity, based on the songs collected through the web-application developed (see Section 3.2). This generic model is therefore capable of suggesting songs for each activity to any user, without having previous knowledge about him/her. This way we have a simple approach that provide an answer to the *cold-start* problem.

To calculate the intervals for each feature and thus define the hyper-rectangles, we started by testing two different methods. The first method (*M1*) used the average minus the standard deviation for finding the minimum of the interval and the average plus the standard deviation to find its maximum. The second method (*M2*) used the 10% percentile as the minimum value of the interval and the 90% percentile for its maximum. To evaluate the quality of the two methods, we searched for songs within the intervals defined, using the minimum and maximum values of the intervals for each feature. The results of these tests lead us to conclude that the methods were not adequate since the intervals generated were too wide, with a considerable overlap between them, blurring the differences between the recommendations for the different activities.

Therefore, we created two new methods: the first based on *M1* using a percentage of the standard deviation, with values of 15, 20, 25 and 30%; and the other method, similar to the previous one, but using the median instead of the average (*M4*). The limits for the hyper-rectangles were generated in two different ways: one using the top-100 songs selected by users, and the other using only the top-20. These variants, generated a set of 8 different data sets that were used to assess the quality of the proposed methods for the hyper-rectangle calculus. The datasets were used for training a Random Forest classifier with the goal of evaluating the quality of the limits generated (the adequacy of the songs to the activity). Figure 3 depicts the accuracy values of the classifiers. In this figure, C1 and C2 encode the datasets

used to train the Random Forest classifiers: C1 represents the top-20 songs dataset, and C2 the top-100. Notice that C1 is a subset of C2, as these songs were those selected by users. The labels S1 and S2 encode the datasets used for determining the intervals of the hyper-rectangles: S1 indicates that the top-20 songs were used, while S2 represents the usage of the top-100 songs dataset. Again, S1 is a subset of S2. Finally, AVG stands for the average, while MEDN for the median.

We used these two dataset divisions (S1 and S2) to understand if it would be beneficial to have a wider (100) or narrower (20) history of songs for generating the intervals. Although the best result was achieved with the average ±15% of the standard deviation, the number of songs suggested by this method was smaller while compared to others. Therefore, we chose the second best method, which corresponds to the usage of the median ±20% of the standard deviation. In this case the median and standard deviation were calculated using the top-100 songs chosen by the users. The training instances used by the classifier were the top 20 songs chosen by the users for each activity (*C1-S2-MEDN*).

In summary, to create the generic recommendation model we defined a set of five hyper-rectangles, one for each activity, using the top-100 songs and the median ± 20% of the standard deviation as the method to determine their intervals. Thus, without previous knowledge about a user's preferences, we can generate recommendations suitable for him/her and for the activity at hand (see Figure 1-4).

3.4 Personalized Recommendation Model

To personalize the recommendations for each activity we incorporate the user feedback, expressed by selecting the songs she/he considered adequate for the activity. This is then materialized by adjusting the intervals for each activity based on the songs listened.

While the method for determining the hyper-rectangles in the personalized model is the same as in the generic approach (top-100 songs and the median ± 20% of the standard deviation), the list of songs used is different. This list starts with the top-100 songs chosen by all users (and used to create the generic model) and is updated with the new songs selected by the users. These are added to the end of the list replacing the oldest ones, as they represent less preferred songs.

When the list of songs used to generate the intervals no longer contains songs used for the generic model, the process follows a FIFO order (*First In First Out*). This approach constantly personalizes the recommendation model by considering the user feedback and by adjusting to his/her current tastes and preferences, over time. New songs remain more time in the list used to determine the new intervals. This design allows us to perform a more personalized recommendation, taking advantage of the current tastes and preferences of the users.

4. EVALUATION

We conducted two user-centric experiments to evaluate both recommendation approaches offered by Improvise, the generic and the personalized model. To that end, we developed a web application where users could select the songs they consider appropriate for each activity. By counting the number of suitable songs we could measure the effectiveness of Improvise in suggesting songs for daily activities.

In the following sections we describe our objectives, the participants, the evaluation procedure, the main results and the discussion about them.

4.1 Goals and Tasks

The main goal of Improvise is to recommend and suggest songs to be listened while doing activities, such as, running, relaxing or shopping. To validate both the generic and the personalized solution, we divided the evaluation into two phases.

The first phase consisted in evaluating the generic model to understand if it was flexible enough to recommend music that fit the preferences of any potential user. In the second phase, the songs selected by each user during the first evaluation (feedback) were used to individually personalize Improvise and to generate new recommendations for each activity. The main objective consisted in understanding if personalized suggestions were better than those generated using the generic model. Finally, a second interaction with the personalized model was conducted to assess the impact in personalization over time. Here, the personalized recommendation model suffered a second personalization by taking into account the new feedback collected during the previous session.

The main task for both phases consisted in selecting the appropriate songs for each activity from a list of songs suggested by our solution.

4.2 Participants

During the first phase of the evaluation, ten users participated in the experiment. Eighty percent of the subjects were male, with ages between 22 and 29 years old (90%), being graduate or undergraduate students from the university campus. All of them reported listening to music for different activities during the day.

In the first iteration of the second experiment all the ten previous subjects participated in the tests using their personalized version of the hyper-rectangles for each activity, created based on their feedback from the first phase. Due to time restrictions, only five of the ten users were able to participate in the second iteration of the second phase. Here, we used a new personalized version of the recommendation model, created using the feedback provided in the previous session.

4.3 Procedure

To evaluate the proposed solution we developed a web application for users to interact with the recommendation technique (see Figure 4). For both experiments, the evaluation started first with users answering a small questionnaire with demographic information to characterize them (*e.g.* age, gender, music listing information, etc.). Then users selected the appropriate songs for each activity, and at the end they filled a satisfaction questionnaire. Notice that the activities

(a) Activity selection. (b) Song selection.

Figure 4: Application developed for evaluating the recommendation model.

were simulated, as the users were not actually performing them, we just mentioned their names.

To evaluate the adequacy of the generic and personalized models, we presented 50 songs for each activity, from which users should select those they consider correctly assigned to the activity. Songs were presented (album cover, song and artist name) one at a time, with the possibility of playing a 30 seconds sample.

After selecting the songs for each activity, users were asked to answer a satisfaction questionnaire to express their agreement with the suitability of the suggested songs for the activity in question.

4.4 Results

Overall, users were satisfied with the recommendations performed by both the generic and the personalized model. Moreover, the effectiveness of the personalized model was confirmed by a steady increase in the number of songs considered suitable for each activity by the users, from the generic model to the personalized model.

On average users selected eleven to twelve songs, for each activity, from the list suggested by the generic model, corresponding to 24% of the total of songs recommended.

For the first iteration of the personalized model, as depicted in Figure 5, users selected on average more than 15 songs

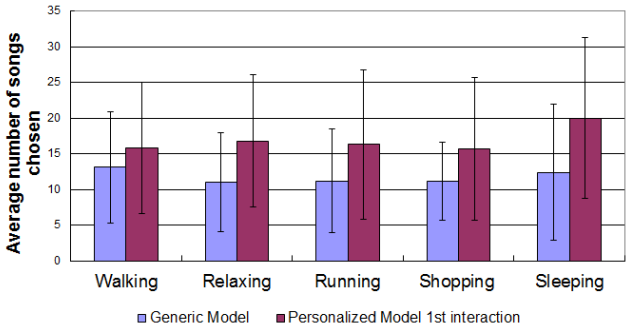


Figure 5: Comparison between the generic and the personalized model in terms of the number of songs selected for each activity. Error bars denote standard deviation.

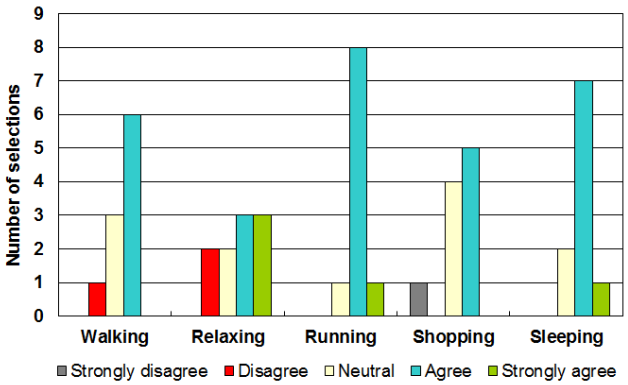


Figure 6: Results of the user satisfaction regarding songs suggested by the generic model.

per activity (30%). This corresponds to an average increase of 25% over the number of songs selected using the generic model. *Sleeping* is the activity that presents the best results and the highest improvement for the personalized model.

The satisfaction questionnaire used to collect users opinion about the quality of the suggested songs was composed by a five point likert scale, with answers as *strongly disagree, disagree, neutral, agree and strongly agree*. Figure 6 depicts the results for the generic model. Overall, more than half of the users agreed or strongly agreed with the suggested songs, for four of the five activities. Only the *Shopping* activity did not achieve this value.

Figure 7 depicts the total number of songs considered appropriate for the various activities. As we can see, there is a steady increase in the number of correct songs, from the generic model to the second iteration of the personalized model. Indeed, this corresponds to an increase of 31% (on average) for all users, revealing that our model is able to fit the tastes of the different users over time.

The growth in the number of songs from the first to the second iteration of the personalized model was around 10%. Detailed data on the behavior of the three models, for the five users who participated in the three test sessions, is depicted in Figure 8. As we can see, overall, the personalized

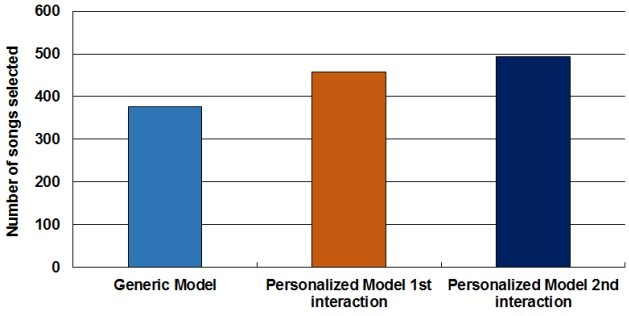


Figure 7: Evolution of the total number of songs selected for the various activities using the different recommendations models.

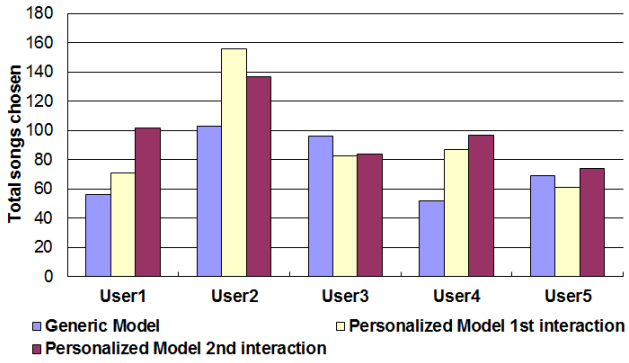


Figure 8: Results of the evaluation of the personalized model for each user.

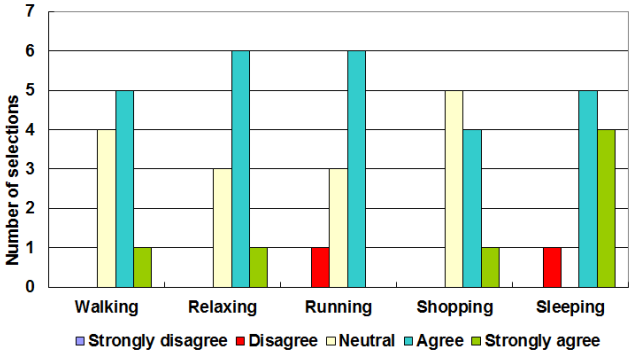


Figure 9: Results of the user satisfaction regarding songs suggested by the personalized model during the first iteration.

models suggested more songs suitable for the different activities than the generic model (Users 1, 2, 4 and 5). Only for User 3 the personalized model suggested less adequate songs. Two other special cases are worth mentioning: for User 2 the second iteration with the personalized model performed worse than in the first iteration; and for User 5, the personalized model required a second iteration to outperform the generic model.

Figure 9 depicts the satisfaction results for the first iteration of the personalized model. Similarly to what happened with the generic model, more than half of the users agreed or strongly agreed with the suggested songs, for four of the five activities. But, for the personalized model we have more strongly agree answers. The *Shopping* activity still has the worst results, but are better than in the generic model.

To get a better understanding of the improvement provided by the personalized model, we grouped the users' answers about the generic and the personalized model in negative (strongly disagree and disagree), neutral and positive (agree and strongly agree) opinions. We found an increase of 13% in the number of positive opinions from the generic to the personalized model, showing that the personalized suggestions are more inline with users' preferences.

4.5 Discussion

From these preliminary results, we can conclude that our work is on the good path to create an approach able to effectively suggest songs for daily activities and flexible enough to adapt the recommendation list to the users' tastes and preferences over time, supporting both "unknown" and "known" users.

Results for the number of songs chosen for each activity show that users selected more songs while using the personalized model than while using the generic model. This confirms that our solution can effectively suggest songs for different users and activities, and adapt to their preferences. Moreover, the second iteration with the personalized model reinforced these results. Satisfaction results were also in agreement with the reported increase in the number of songs selected. Users were overall happy and satisfied with the recommendations performed.

In a particular case the personalized model required two iterations to get adjusted to the user, showing that for some users our model needs more time to "learn" the users preferences. In another case, for which we did not find any evidence for it, the user preferred more songs from the generic model than from the personalized ones.

Although these results are very promising, showing that our approach can deal with the cold-start problem by providing a generic model that can suggest songs for any user without knowing anything about them, we would like to mention some constraints that prevent us from state stronger claims. First, we cannot draw any statistical significance from the results due to the small number of users involved in the preliminary evaluation. In a near future we plan to perform an evaluation with a larger number of users. Second, users were not performing the activities for which we suggested songs. More evaluation is required to clarify if this affected the result.

5. CONCLUSIONS AND FUTURE WORK

Nowadays, a huge amount of songs is available to millions of users around the world. With millions of artists and songs on the market, it is difficult for users to find songs that please them. This problem is even worse when trying to select songs for different activities.

In this paper we described Improvise, an adaptable solution for recommending songs for daily activities. Improvise is a user-centered approach that relies on the hyper-rectangle concept, determined using content from songs. We described the rationale behind the calculus of the hyper-rectangles for a generic recommendation model and also the creation of a personalized solution. Preliminary results show that the generic model was successful in recommending songs to users. But more relevant is the flexibility of the solution in adapting the recommendation to different users for each activity, increasing not only the number of songs selected, but also their satisfaction.

Regarding future work, we plan to explore two paths. The first is to explore new and different methods for determining the hyper-rectangles, like for instance to consider more than one hyper-rectangle for each activity. This can capture more

diverse and sparse preferences and tastes, promoting new recommendations and user satisfaction. The second path is to use a larger number of songs to determine the hyper-rectangles that characterize the user profile, since at the moment we are only using the top-100 songs preferred by the users as detailed in Section 3. Although, using more songs could bring a more accurate and detailed calculus of the hyper-rectangles, a study to determine the best number of songs is also planned.

In summary, we can report that Improvise suggests songs suitable for daily activities to users with different tastes and preferences. Moreover, the proposed model is flexible enough to constantly adapt its recommendations accordingly to the user feedback, while providing an answer to the cold-start problem.

6. ACKNOWLEDGMENTS

This work was supported by national funds through Fundação para a Ciência e Tecnologia, under INESC-ID multi-annual funding - PEst-OE/EEI/LA0021/2013 and LaSIGE Strategic Project - PEst-OE/EEI/UI0408/2014. Ricardo Dias was supported by FCT, grant reference SFRH/BD/70939/2010.

7. REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *Proc. of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999.

[2] L. Baltrunas, X. Amatriain, and V. Augusta. Towards Time-Dependant Recommendation based on Implicit Feedback. In *Context Aware Recommendation Systems (CARS - RecSys)*, 2009.

[3] R. Cai, C. Zhang, C. Wang, L. Zhang, and W.-Y. Ma. Musicsense: contextual music recommendation using emotional allocation modeling. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 553–556, New York, NY, USA, 2007. ACM.

[4] P. Cano, M. Koppenberger, and N. Wack. An industrial-strength content-based music recommendation system. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in Information Retrieval*, 2005.

[5] O. Celma. *Music Recommendation and Discovery.* Springer, 1st edition, 2010.

[6] W. W. Cohen and W. Fan. Web-collaborative filtering: recommending music by crawling the Web. *Computer Networks*, 33, 1999.

[7] R. Dias and M. J. Fonseca. Improving music recommendation in session-based collaborative filtering by using temporal context. In *ICTAI*, 2013.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 2009.

[9] B. P. Knijnenburg, S. Bostandjiev, J. O'Donovan, and A. Kobsa. Inspectability and control in social recommenders. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, 2012.

[10] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, J. Riedl, and H. Volume. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 1997.

[11] T. Leong, S. Howard, and V. Frank. Choice: Abdicating or exercising. In *CHI*, 2008.

[12] N. Liu, S. Lai, and C. Chen. Adaptive music recommendation based on user behavior in time slot. *Journal of Computer Science and Network Security*, 9(2):219–227, 2009.

[13] D. McEnnis and S. J. Cunningham. Sociology and music recommendation systems. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 185–186, 2007.

[14] S. E. Park, S. Lee, and S.-g. Lee. Session-Based Collaborative Filtering for Predicting the Next Song. *First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, 2011.

[15] S. Reddy and J. Mascia. Lifetrak: Music in tune with your life. In *Proceedings of the 1st ACM International Workshop on Human-centered Multimedia*, 2006.

[16] M. Schedl and P. Knees. Personalization in Multimodal Music Retrieval. In *9th International Workshop on Adaptive Multimedia Retrieval*, 2011.

[17] B. Schwartz. *The Paradox of Choice: Why More Is Less.* Harper Perennial, 2005.

[18] B. Shao, M. Ogihara, D. Wang, and T. Li. Music recommendation based on acoustic features and user access patterns. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(8):1602 –1611, nov. 2009.

[19] J. Su, H. Yeh, and P. Yu. Music recommendation using content and context information mining. *Intelligent Systems, IEEE*, (IEEE Intelligent Systems), 2010.

[20] K. Verbert, D. Parra, P. Brusilovsky, and E. Duval. Visualizing recommendations to support exploration, transparency and controllability. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, 2013.

[21] X. Wang, D. Rosenblum, and Y. Wang. Context-aware mobile music recommendation for daily activities. In *Proc. of the 20th international conference on Multimedia*, 2012.

[22] B. Yapriady and A. L. Uitdenbogerd. Combining demographic data with collaborative filtering for automatic music recommendation. Springer, 2005.

[23] E. Zheleva, J. Guiver, E. Mendes Rodrigues, and N. Milić-Frayling. Statistical models of music-listening sessions in social media. *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010.

Exploiting Social Tags in Matrix Factorization Models for Cross-domain Collaborative Filtering

Ignacio Fernández-Tobías, Iván Cantador
Escuela Politécnica Superior
Universidad Autónoma de Madrid
28049 Madrid, Spain
{ignacio.fernandezt, ivan.cantador}@uam.es

ABSTRACT
Cross-domain recommender systems aim to generate or enhance personalized recommendations in a target domain by exploiting knowledge (mainly user preferences) from other source domains. Due to the heterogeneity of item characteristics across domains, content-based recommendation methods are difficult to apply, and collaborative filtering has become the most popular approach to cross-domain recommendation. Nonetheless, recent work has shown that the accuracy of cross-domain collaborative filtering based on matrix factorization can be improved by means of content information; in particular, social tags shared between domains. In this paper, we review state of the art approaches in this direction, and present an alternative recommendation model based on a novel extension of the SVD++ algorithm. Our approach introduces a new set of latent variables, and enriches both user and item profiles with independent sets of tag factors, better capturing the effects of tags on ratings. Evaluating the proposed model in the movies and books domains, we show that it can generate more accurate recommendations than existing approaches, even in cold-start situations.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *information filtering*. G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra – *singular value decomposition.*

General Terms
Algorithms, Performance, Experimentation.

Keywords
Recommender systems, collaborative filtering, cross-domain recommendation, social tagging.

1. INTRODUCTION
Recommender systems [2] have been successfully used in numerous domains and applications to identify potentially relevant items for users according to their preferences (tastes, interests and goals). Examples include suggested movies and TV programs in Netflix[1], music albums in Last.fm[2], and books in Barnes&Noble[3].

Even though the majority of recommender systems focus on a single domain or type of item, there are cases in which providing the user with *cross-domain recommendations* could be beneficial. For instance, large e-commerce sites like Amazon[4] and eBay[5] collect user feedback for items from multiple domains, and in social networks users often share their tastes and interests on a variety of topics. In these cases, rather than exploiting user preference data from each domain independently, recommender systems could exploit more exhaustive, multi-domain user models that allow generating item recommendations spanning several domains. Furthermore, exploiting additional knowledge from related, auxiliary domains could help improve the quality of item recommendations in a target domain, e.g. addressing the cold-start and sparsity problems [7].

These benefits rely on the assumption that there are similarities or relations between user preferences and/or item attributes from different domains. When such correspondences exist, one way to exploit them is by *aggregating knowledge* from the involved domain data sources, for example by merging user preferences into a unified model [1], and by combining single-domain recommendations [3]. An alternative way consists of *transferring knowledge* from a source domain to a target domain, for example by sharing implicit latent features that relate source and target domains [15][17], and by exploiting implicit rating patterns from source domains in the target domain [9][14].

In either of the above cases, most of the existing approaches to cross-domain recommendation are based on collaborative filtering, since it merely needs rating data, and does not require information about the users' and items' characteristics, which are usually highly heterogeneous among domains.

However, inter-domain links established through content-based features and relations may have several advantages, such as a better interpretability of the cross-domain user models and recommendations, and the establishment of more reliable methods to support the knowledge transfer between domains. In particular, social tags assigned to different types of items –such as movies, music albums, and books–, may act as a common vocabulary between domains [6][17]. Hence, as domain independent content-based features, tags can be used to overcome the information heterogeneity across domains, and are suitable for building the above mentioned inter-domain links.

In this paper, we review state of the art cross-domain recommendation approaches that utilize social tags to exploit knowledge from an auxiliary source domain for enhancing collaborative filtering rating predictions in a target domain.

[1] Netflix online movies & TV shows provider, http://www.netflix.com
[2] Last.fm music discovery service, http://www.lastfm.com
[3] Barnes&Noble retail bookseller, http://www.barnesandnoble.com

[4] Amazon e-commerce website, http://www.amazon.com
[5] eBay consumer-to-consumer website, http://www.ebay.com

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

Specifically, we focus on several extensions of the matrix factorization technique proposed in [6], which incorporates latent factors related to the users' social tags. By jointly learning tag factors in both the source and target domains, hidden correlations between ratings and tags in the source domain can be used in the target domain. Hence, for instance, a movie recommender system may estimate a higher rating for a particular movie tagged as *interesting* or *amazing* if these tags are usually assigned to books positively rated. Also, books tagged as *romantic* or *suspenseful* may be recommended to a user if it is found that such tags correlate with high movie ratings.

Enrich et al. [6] presented several recommendation models that exploit different sets of social tags when computing rating predictions, namely tags assigned by the active user to the item for which the rating is estimated, and all the tags assigned by the community to the target item. Despite their good performance, these models do have difficulties in cold-start situations where no tagging information is available for the target user/item.

In this paper, we propose a method that expands the users' and items' profiles to overcome these limitations. More specifically, we propose to incorporate additional parameters to the above models, separating user and item latent tag factors in order to capture the contributions of each to the ratings more accurately. Furthermore, by modeling user and item tags independently we are able to compute rating predictions even when a user has not assigned any tag to an item, or for items that have not been tagged yet. For such purpose, we adapt the gSVD++ algorithm [10] – designed to integrate content metadata into the matrix factorization process– for modeling social tags in the cross-domain recommendation scenario.

Through a series of experiments in the movies and books domains, we show that the proposed approach outperforms the state of the art methods, and validate the main contribution of this work: A model that separately captures user and item tagging information, and effectively transfers auxiliary knowledge to the target domain in order to provide cross-domain recommendations.

The reminder of the paper is structured as follows. In section 2 we review state of the art approaches to the cross-domain recommendation problem, focusing on algorithms based on matrix factorization, and on algorithms that make use of social tags to relate the domains of interest. In section 3 we provide a brief overview of matrix factorization methods for single-domain recommendation, and in section 4 we describe their extensions for the cross-domain recommendation case. In section 5 we present and discuss the conducted experimental work and obtained results. Finally, in section 6 we summarize some conclusions and future research lines.

2. RELATED WORK

Cross-domain recommender systems aim to generate or enhance personalized recommendations in a target domain by exploiting knowledge (mainly user preferences) from other source domains [7][19]. This problem has been addressed from various perspectives in several research areas. It has been faced by means of user preference aggregation and mediation strategies for the cross-system personalization problem in user modeling [1][3][16], as a potential solution to mitigate the cold-start and sparsity problems in recommender systems [5][17][18], and as a practical application of knowledge transfer techniques in machine learning [9][14][15].

We can distinguish between two main types of cross-domain approaches: Those that *aggregate* knowledge from various source domains to perform recommendations in a target domain, and those that *link* or *transfer* knowledge between domains to support recommendations in the target domain.

The knowledge aggregation methods merge user preferences (e.g. ratings, social tags, and semantic concepts) [1], mediate user modeling data exploited by various recommender systems (e.g. user similarities and user neighborhoods) [3][16], and combine single-domain recommendations (e.g. rating estimations and rating probability distributions) [3]. The knowledge linkage and transfer methods relate domains by common information (e.g. item attributes, association rules, semantic networks, and inter-domain correlations) [5][18], share implicit latent features that relate source and target domains [15][17], and exploit explicit or implicit rating patterns from source domains in the target domain [9][14].

Cross-domain recommendation models based on latent factors are a popular choice among knowledge linkage and transfer methods, since they allow automatically discovering and exploiting implicit domain relations within the data from different domains. For instance, Zhang et al. [20] proposed an adaptation of the matrix factorization model to include a probability distribution that captures inter-domain correlations, and Cao et al. [4] presented a method that learns similarities between item latent factors in different domains as parameters in a Bayesian framework. Aiming to exploit heterogeneous forms of user feedback, Pan et al. [15] proposed an adaptive model in which the latent features learned in the source domain are transferred to the target domain in order to regularize the matrix factorization there. Instead of the more common two-way decomposition of the rating matrix, Li et al. [14] used a nonnegative matrix tri-factorization to extract rating patterns –the so-called *codebook*– in the source domain. Then, rather than transferring user and item latent factors, the rating patterns are shared in the target domain and used to predict the missing ratings.

Despite the ability of matrix factorization models to discover latent implicit relations, there are some methods that use tags as explicit information to bridge the domains. Shi et al. [17] argued that explicit relations established through common social tags are more effective for such purpose, and used them to compute user-user and item-item cross-domain similarities. In this case, rating matrices from the source and target domains are jointly factorized, but user and item latent factors are restricted so that they are consistent with the tag-based similarities.

Instead of focusing on sharing user or item latent factors, Enrich et al. [6] studied the influence of social tags on rating prediction. More specifically, the authors presented a number of models based on the well-known SVD++ algorithm [11], to incorporate the effect of tag assignments into rating estimations. The underlying hypothesis is that information about how users annotate items in the source domain can be exploited to improve rating prediction in a different target domain, as long as a set of common tags between the domains exists. In all the proposed models, tag factors are added into the latent item vectors, and are then combined with user latent features to compute rating estimations. The difference between these models is the set of tags considered for rating prediction. Two of the proposed models use the tags assigned by the user to a target item, and the other model takes the tags of the whole community into account. We note that the first two models require the active user to tag, but not rate the item in the target domain. In all the models, the transfer of knowledge is performed through the shared tag factors in a collective way, since these factors are learned jointly for the source and the target domains. The results reported in the movies and books domains confirmed that shared knowledge can be effectively exploited to outperform single-domain rating predictions.

The model we propose in this paper follows the same line as Enrich et al. [6], in the sense that tags are directly integrated as latent factors into the rating prediction process, as opposed to Shi's and colleagues' approach [17], which estimates the ratings using only user and item factors. The main difference of our model with the approaches presented in [6] is the way in which the rating matrix is factorized. Rather than using a single set of tag factors to extend the item's factorization component, we introduce additional latent variables in the user component to separately capture the effect of tags utilized by the user and the tags assigned to the item. For this purpose, we adapt the gSVD++ algorithm [10], which extends SVD++ by introducing a set of latent factors to take item metadata into account for rating prediction. In this model, both user and item factors are respectively enhanced with implicit feedback and content information, which allows improving the accuracy of rating predictions.

3. OVERVIEW OF MATRIX FACTORIZATION METHODS

Since the proposed cross-domain recommendation model is built upon a matrix factorization collaborative filtering method, in this section we provide a brief overview of the well-known standard rating matrix factorization technique, and the SVD++ and gSVD++ algorithms, which extend the former by incorporating implicit user feedback and item metadata, respectively.

3.1 MF: Standard rating matrix factorization

Matrix factorization (MF) methods [8][12] are a popular approach to latent factor models in collaborative filtering. In these methods, the rating matrix is decomposed as the product of low-rank matrices of user and item latent features. In its most basic form, a factor vector $p_u \in \mathbb{R}^k$ is assigned to each user u, and a factor vector $q_i \in \mathbb{R}^k$ to each item i, so that ratings are estimated as:

$$\hat{r}_{ui} = q_i^T p_u + b_{ui} \tag{1}$$

where the term b_{ui} is a baseline estimate that captures the deviation of user and item ratings from the average, and is defined as:

$$b_{ui} = \mu + b_u + b_i \tag{2}$$

The parameter μ corresponds to the global average rating in the training set, and b_u and b_i are respectively the deviations in the ratings of user u and item i from the average. The baseline estimates can be explicitly defined or learned from the data. In the latter case, the parameters of the model are found by solving the following regularized least squares problem:

$$\min_{b_*, p_*, q_*} \sum_{(u,i)\in\mathcal{R}} \left(r_{ui} - \mu - b_u - b_i - q_i^T p_u\right)^2 + \lambda\left(b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2\right) \tag{3}$$

In this formula, the parameter λ controls the amount of regularization to prevent high model variance and overfitting. The minimization can be performed by using gradient descent over the set $\mathcal{R}$ of observed ratings [8]. This method is popularly called SVD, but it is worth noticing that it is not completely equivalent to the *singular value decomposition* technique, since the rating matrix is usually very sparse and most of its entries are actually not observed.

For simplicity purposes, in the following we omit the baseline estimates. They, nonetheless, can be easily considered by adding the b_{ui} term into the rating estimation formulas.

3.2 SVD++: Adding implicit user feedback to the rating matrix factorization method

The main motivation behind the SVD++ algorithm, proposed by Koren [11][13], is to exploit implicit additional user feedback for rating prediction, since it is arguably to use a more available and abundant source of user preferences.

In this model, user preferences are represented as a combination of explicit and implicit feedback, searching for a better understanding of the user by looking at what items she rated, purchased or watched. For this purpose, additional latent factors are combined with the user's factors as follows:

$$\hat{r}_{ui} = q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j\right) \tag{4}$$

In the previous formula, $p_u \in \mathbb{R}^k, q_i \in \mathbb{R}^k, y_j \in \mathbb{R}^k$ represent user, item, and implicit feedback factors, respectively. $N(u)$ is the set of items for which the user u provided implicit preference, and k is the number of latent features.

Similarly to the SVD algorithm, the parameters of the model can be estimated by minimizing the regularized squared error loss over the observed training data:

$$\min_{p_*, q_*, y_*} \sum_{(u,i)\in\mathcal{R}} \left[r_{ui} - q_i^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j\right)\right]^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2 + \sum_{j\in N(u)} \|y_j\|^2\right) \tag{5}$$

Again, the minimization problem can be efficiently solved using stochastic gradient descent.

3.3 gSVD++: Adding item metadata to the rating matrix factorization method

The gSVD++ algorithm [10] further extends SVD++ considering information about the items' attributes in addition to the users' implicit feedback.

The model introduces a new set of latent variables $x_g \in \mathbb{R}^k$ for metadata that complement the item factors. This idea combined with the SVD++ algorithm leads to the following formula for computing rating predictions:

$$\hat{r}_{ui} = \left(q_i + |G(i)|^{-\beta} \sum_{g\in G(i)} x_g\right)^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j\right) \tag{6}$$

The set $G(i)$ contains the attributes related to item i, e.g. *comedy* and *romance* in the case of movie genres. The parameter β is set to 1 when the set $G(i) \neq \emptyset$, and 0 otherwise. We note that in the previous formula, both user and item factors are enriched with new uncoupled latent variables that separately capture information about the users and items, leading to a symmetric model with four types of parameters. Again, parameter learning can be performed by minimizing the associated squared error function with gradient descent:

$$\min_{p_*, q_*, x_*, y_*} \sum_{(u,i)\in\mathcal{R}} \left[r_{ui} - \left(q_i + |G(i)|^{-\beta} \sum_{g\in G(i)} x_g\right)^T \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j\in N(u)} y_j\right)\right]^2 + \lambda\left(\|p_u\|^2 + \|q_i\|^2 + \sum_{g\in G(i)} \|x_g\|^2 + \sum_{j\in N(u)} \|y_j\|^2\right) \tag{7}$$

The use of additional latent factors for item metadata is reported to improve prediction accuracy over SVD++ in [10]. In this paper, we adapt this model to separately learn user and item tag factors, aiming to support the transfer of knowledge between domains.

4. TAG-BASED MODELS FOR CROSS-DOMAIN COLLABORATIVE FILTERING

In this section, we first describe the tag-based cross-domain collaborative filtering models presented in [6], which are an adaptation of the SVD++ algorithm, and next introduce our proposed model, which is built upon the gSVD++ algorithm.

4.1 Adaptation of SVD++ for Tag-based Cross-domain Collaborative Filtering

The main hypothesis behind the models proposed in [6] is that the effect of social tags on ratings can be shared between domains to improve the rating predictions in the target domain. In that work, three different adaptations of the SVD++ algorithm were explored that utilize tags as implicit user feedback to enhance the item factors, as opposed to user factors like in the original model.

The first of the algorithms proposed by Enrich et al. is the **UserItemTags** model, which only exploits the tags $T_u(i)$ that the active user u assigned to the target item i:

$$\hat{r}_{ui} = p_u^T \left(q_i + \frac{1}{|T_u(i)|} \sum_{t \in T_u(i)} y_t \right) \quad (8)$$

We note here that if the user has not tagged the item, i.e., $T_u(i) = \emptyset$, then the model corresponds to the standard matrix factorization technique. Also, even though the tag factors y_t are only combined with the item factors q_i, the user and item factorization components are not completely uncoupled, since the set $T_u(i)$ still depends on the user u.

An improvement over the model was also presented in [6], based on the observation that not all the tags are equally relevant (i.e. discriminative) to predict the ratings. The proposed alternative is to filter the tags in the set $T_u(i)$ that are not relevant according to certain criterion. In that work, the Wilcoxon rank-sum test is performed for each tag to decide if the mean rating significantly changes in the presence/absence of the tag in the dataset. In this model, rating predictions are computed in an analogous manner:

$$\hat{r}_{ui} = p_u^T \left(q_i + \frac{1}{|TR_u(i)|} \sum_{t \in TR_u(i)} y_t \right) \quad (9)$$

Here, the set $TR_u(i) \subseteq T_u(i)$ only contains those tags for which the p-value of the abovementioned test is $p < 0.05$. This method was called as **UserItemRelTags**.

As noted by the authors, the previous methods are useful when the user has tagged but not rated an item. However, these methods do not greatly improve over the standard matrix factorization technique in the cold-start situations where new users or items are considered. Aiming to address this limitation, a last approach was proposed, the **ItemRelTags** model:

$$\hat{r}_{ui} = p_u^T \left(q_i + \frac{1}{|TR(i)|} \sum_{t \in TR(i)} n_t y_t \right) \quad (10)$$

Now, the set $TR(i)$ contains all the relevant tags assigned by the whole community to the item i, with possible repetitions. Tags that appear more often contribute with more factors to the prediction, and n_t is the number of times tag t was applied to item i. In this case, the normalization factor is $|TR(i)| = \sum_{t \in TR(i)} n_t$.

We note that the set $TR(i)$ does not depend on the user, and that the user and item components of the factorization are fully uncoupled. This has the advantage that tag factors can also be exploited in the rating predictions for new users for whom tagging information is not available yet, improving over the standard matrix factorization method. The ItemRelTags model, however, does not take into account the possibility that the user has tagged different items other than the one for which the rating is being estimated. In such cases, it may be beneficial to enrich the user's profile by considering other tags the user has chosen in the past as evidence of her preferences. In the next subsection, we propose a model that aims to exploit this information to generate more accurate recommendations.

Similarly to the SVD++ algorithm, all of the above models can be trained by minimizing the associated loss function with stochastic gradient descent.

4.2 Adaptation of gSVD++ for Tag-based Cross-domain Collaborative Filtering

Although the previous recommendation models can successfully transfer tagging information between domains, they suffer from some limitations. The UserItemTags and UserItemRelTags models cannot do better than the standard matrix factorization if the user has not tagged the item for which the rating is being estimated, while the ItemRelTags model does not fully exploits the user's preferences expressed in the tags assigned to other items.

In this paper, we propose to adapt the gSVD++ algorithm by introducing an additional set of latent variables $x_s \in \mathbb{R}^k$ that enrich the user's factors and better capture the effect of her tags in the rating estimation. Specifically, we distinguish between two different sets of tags for users and items, and factorize the rating matrix into fully uncoupled user and item components as follows:

$$\hat{r}_{ui} = \left(p_u + \frac{1}{|T_u|} \sum_{s \in T_u} n_{us} x_s \right)^T \left(q_i + \frac{1}{|T_i|} \sum_{t \in T_i} n_{it} y_t \right) \quad (11)$$

The set T_u contains all the tags assigned by user u to any item. Respectively, T_i is the set of tags assigned by any user to item i, and plays the role of item metadata $G(i)$ in the gSVD++ algorithm. As in the ItemRelTags model, there may be repeated tags in each of the above tag sets, which we account for by considering the number of times a tag appears in T_u or T_i, respectively. In (11), n_{us} is the number of items on which the user u applied tag s, and n_{it} is the number of users that applied tag t to item i. As previously, tag factors are normalized by $|T_u| = \sum_{s \in T_u} n_{us}$ and $|T_i| = \sum_{t \in T_i} n_{it}$, so that factors x_s and y_t do not dominate over the rating factors p_u and q_i for users and items with a large number of tags.

In the proposed model, which we call as **TagGSVD++**, a user's profile is enhanced with the tags she used, since we hypothesize that her interests are better captured, and that transferring this information between domains can be beneficial for estimating ratings in the target domain. Likewise, item profiles are extended with the tags that were applied to them, as in the ItemRelTags model.

The parameters of TagGSVD++ can be learned from the observed training data by solving the following unconstrained minimization problem:

$$\min_{p_*,q_*,x_*,y_*} \sum_{(u,i)\in\mathcal{R}} E(p_u, q_i, \{x_s\}_{s\in T_u}, \{y_t\}_{t\in T_i})$$
$$= \min_{p_*,q_*,x_*,y_*} \sum_{(u,i)\in\mathcal{R}} \frac{1}{2}\left[r_{ui} - \left(p_u + \frac{1}{|T_u|}\sum_{s\in T_u} n_s x_s\right)^T \left(q_i + \frac{1}{|T_i|}\sum_{t\in T_i} n_t y_t\right)\right]^2 \quad (12)$$
$$+ \frac{\lambda}{2}\left(\|p_u\|^2 + \|q_i\|^2 + \sum_{s\in T_u}\|x_s\|^2 + \sum_{t\in T_i}\|y_t\|^2\right)$$

The factor $1/2$ simplifies the following derivations with no effect on the solution. As in the previous models, a minimum can be found by stochastic gradient descent. For completeness, in the following we list the update rules of TagGSVD++ taking the derivatives of the error function in (12) with respect to the parameters:

$$\frac{\partial E}{\partial p_u} = -e_{ui}\left(q_i + \frac{1}{|T_i|}\sum\nolimits_{t\in T_i} n_{it} y_t\right) + \lambda p_u$$
$$\frac{\partial E}{\partial q_i} = -e_{ui}\left(p_u + \frac{1}{|T_u|}\sum\nolimits_{s\in T_u} n_{us} x_s\right) + \lambda q_i$$
$$\frac{\partial E}{\partial x_a} = -e_{ui}\frac{n_{ua}}{|T_u|}\left(q_i + \frac{1}{|T_i|}\sum\nolimits_{t\in T_i} n_{it} y_t\right) + \lambda x_a \qquad \forall a \in T_u$$
$$\frac{\partial E}{\partial y_b} = -e_{ui}\frac{n_{ib}}{|T_i|}\left(p_u + \frac{1}{|T_u|}\sum\nolimits_{s\in T_u} n_{us} x_s\right) + \lambda y_b \qquad \forall b \in T_i$$

where the error term e_{ui} is $r_{ui} - \hat{r}_{ui}$. In the training phase, we loop over the observed ratings simultaneously updating the parameters according to the following rules:

$$p_u \leftarrow p_u - \alpha\left[\lambda p_u - e_{ui}\left(q_i + \tfrac{1}{|T_i|}\textstyle\sum_{t\in T_i} n_{it} y_t\right)\right]$$
$$q_i \leftarrow q_i - \alpha\left[\lambda q_i - e_{ui}\left(p_u + \tfrac{1}{|T_u|}\textstyle\sum_{s\in T_u} n_{us} x_s\right)\right]$$
$$x_a \leftarrow x_a - \alpha\left[\lambda x_a - e_{ui}\tfrac{n_{ua}}{|T_u|}\left(q_i + \tfrac{1}{|T_i|}\textstyle\sum_{t\in T_i} n_{it} y_t\right)\right], \forall a \in T_u$$
$$y_b \leftarrow y_b - \alpha\left[\lambda y_b - e_{ui}\tfrac{n_{ib}}{|T_i|}\left(p_u + \tfrac{1}{|T_u|}\textstyle\sum_{s\in T_u} n_{us} x_s\right)\right], \forall b \in T_i$$

The learning rate α determines to what extent the parameters are updated in each iteration. A small learning rate can make the learning slow, whereas with a large learning rate the algorithm may fail to converge. The choice of both the learning rate and the regularization parameter λ is discussed later in section 5.3.

5. EXPERIMENTS

We have evaluated the proposed TagGSVD++ algorithm (section 4.2) in a cross-domain collaborative filtering setting, by empirically comparing it with the single-domain matrix factorization methods (section 3) and the state-of-the-art cross-domain recommendation approaches described in section 4.1.

5.1 Dataset

We have attempted to reproduce the cross-domain dataset used in [6], aiming to compare our approach with those presented in that paper. For the sake of completeness, we also describe the data collection process here.

In order to simulate the cross-domain collaborative filtering setting, we have downloaded two publicly available datasets for the movies and books domains. The MovieLens 10M dataset[6] (ML) contains over 10 million ratings and 100,000 tag assignments by 71,567 users to 10,681 movies. The LibraryThing dataset[7] (LT) contains over 700,000 ratings and 2 million tag assignments by 7,279 users on 37,232 books. Ratings in both of the datasets are expressed on a 1-5 scale, with interval steps of 0.5.

Since we were interested in analyzing the effect of tags on rating prediction, we only kept ratings in MovieLens on movies for which at least one tag was applied, leaving a total of 24,564 ratings. Also following the setup done by Enrich et al., we considered the same amount of ratings in LibraryThing, and took the first 24,564 ratings. We note, however, that the original dataset contained duplicate rows and inconsistencies, i.e., some user-item pairs had more than one rating. Hence, we preprocessed the dataset removing such repetitions and keeping only the repeated ratings that appeared first in the dataset's file. We also converted the tags to lower case in both datasets. Table 1 shows the characteristics of the final datasets.

Table 1. Details of the datasets used in the experiments after preprocessing.

	MovieLens	**LibraryThing**
Users	2,026	244
Items	5,088	12,801
Ratings	24,564	24,564
Avg. ratings per user	12.12	100.67
Rating sparsity	99.76%	99.21%
Tags	9,529	4,598
Tag assignments	44,805	72,943
Avg. tag assignments per user	22.16	298.95
Ratio of overlapping (shared) tags	13.81%	28.62%

5.2 Evaluation methodology

As mentioned above, we have compared the performance of the proposed model against the single-domain matrix factorization baselines from section 3, and the state-of-the-art tag-based algorithms described in section 4.1. All these methods are summarized next:

MF The standard matrix factorization method trained by stochastic gradient descent over the observed ratings of both movies and books domains.

SVD++ An adaptation of MF to take implicit data into account. In our experiments, the set $N(u)$ contains all the items rated by user u.

gSVD++ An extension of SVD++ to include item metadata into the factorization process. In our experiments, we have considered as set of item attributes $G(i)$ the tags T_i assigned to item i by any user. Note that, as tags are content features for both movies and books, this method is suitable for cross-domain recommendation, since knowledge can be transferred through the metadata (tag) factors. This differs from the proposed TagGSVD++ in that users are modeled as in SVD++ by considering rated items as implicit feedback instead of their tags. Also, normalization of the implicit data factors on the user component involves a square root; see equations (6) and (11).

UserItemTags A method that expands an item i's profile with latent factors of tags that the target user assigned to i. Its parameters are learned by simultaneously factorizing the rating matrices of both source and target domains.

UserItemRelTags A variation of the previous method that only takes relevant tags into account, as determined by a Wilcoxon rank-sum test.

[6] MovieLens datasets, http://grouplens.org/datasets/movielens
[7] LibraryThing dataset, http://www.macle.nl/tud/LT

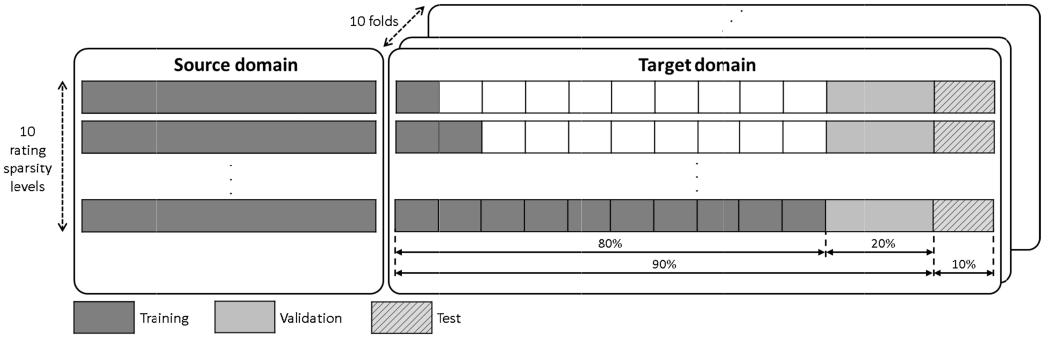


Figure 1. Data splitting done for cross-validation. Training data consists of source domain ratings and portions of the target domain, marked in dark.

ItemRelTags Instead of tags assigned by the user, this method exploits all relevant tags applied by the whole user community, and is thus able to compute rating predictions even if the user has not tagged the target item.

We evaluated all these recommendation methods in two settings, using MovieLens as source domain and LibraryThing as target domain, and vice-versa. In both cases, we evaluated the methods through 10-fold cross-validation, i.e., we shuffled the target ratings and split them into 10 non-overlapping folds. In each fold, we left out one part, 10% of the ratings, as *test* set to estimate the performance of the methods. The rest 90% of the ratings were used as a *training* set to learn the models, and a *validation* set to find the optimal values of the models' parameters. Specifically, we randomly chose 80% of these remaining ratings, and combined them with the source domain ratings to build the models. The final 20% left was used for the validation set to select the best number of factors k, learning rate α, and regularization λ. Figure 1 depicts the split of the data into training, validation, and test sets.

As in [6], we also wanted to investigate how the number of available ratings in the target domain affects the quality of the recommendations. For such purpose, we further split the training data from the target domain into 10 portions to simulate different rating sparsity levels. First, in order to evaluate the performance of the methods in cold-start situations, we used only 10% of the target training ratings, i.e., $0.1 \cdot 0.8 \cdot 0.9 \cdot 24{,}564 = 1{,}768$ ratings (see Table 1). Then, we incrementally added additional 10% of the ratings to analyze the behavior of the methods with an increasingly larger amount of observed rating data. In each sparsity level, the full set of source domain ratings was also used to build the models.

Since all the methods are designed for the rating prediction task, we measured their performance as the accuracy of the estimated ratings. Specifically, we computed the Mean Absolute Error (MAE) of each model in the different settings described above:

$$MAE = \frac{1}{|\mathcal{R}_{te}|} \sum_{(u,i) \in \mathcal{R}_{te}} |r_{ui} - \hat{r}_{ui}|$$

where $\mathcal{R}_{te}$ contains the ratings in the test set we left out for evaluation.

5.3 Results

As previously mentioned, we reserved 20% of the target domain training data in each fold for validating the models and finding the best model parameters, in order not to overestimate the performance of the methods.

For hyperparameter optimization, with each method and sparsity level in the target domain, we performed a grid (stepsize) search on the validation set for the values of the learning rate α, the amount of regularization λ, and the number of latent features k. To get an idea of the typical values found for the parameters, Table 2 shows the average best values for each method.

Table 2. Average values of the best parameters found.

	ML → LT			LT → ML		
	k	α	λ	k	α	λ
MF	41	0.020	0.009	43	0.020	0.009
SVD++	41	0.020	0.007	43	0.020	0.006
gSVD++	43	0.019	0.001	43	0.020	0.004
UserItemTags	46	0.019	0.003	46	0.020	0.010
UserItemRelTags	39	0.017	0.008	41	0.020	0.017
ItemRelTags	40	0.017	0.001	46	0.020	0.006
TagGSVD++	40	0.013	0.036	46	0.019	0.045

From the table, we observe that there is not a large difference in the optimal number of factors and learning rates between configurations. In contrast, we note that the amount of regularization needed for the proposed TagGSVD++ method is relatively large, e.g. compare $\lambda = 0.036$ of TagGSVD++ with $\lambda = 0.009$ of MF. This may be due to the additional set of latent variables for tags that our model uses; more complex models are able to account for greater variance in the data and tend to overfit more easily, thus requiring more regularization. In order to analyze how the available information in the target domain affects the stability of the model, Figure 2 shows the optimal value for the regularization parameter for different sparsity levels.

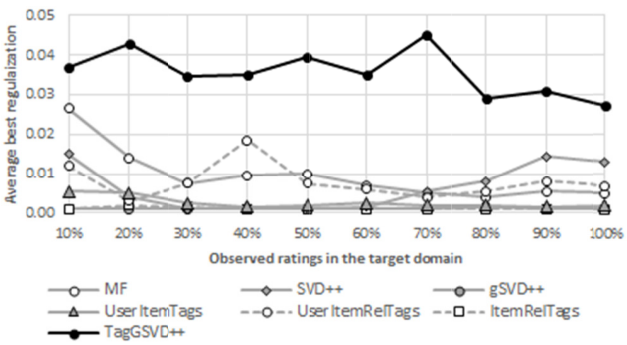


Figure 2. Optimal values for the regularization parameter using MovieLens as source domain and LibraryThing as target domain.

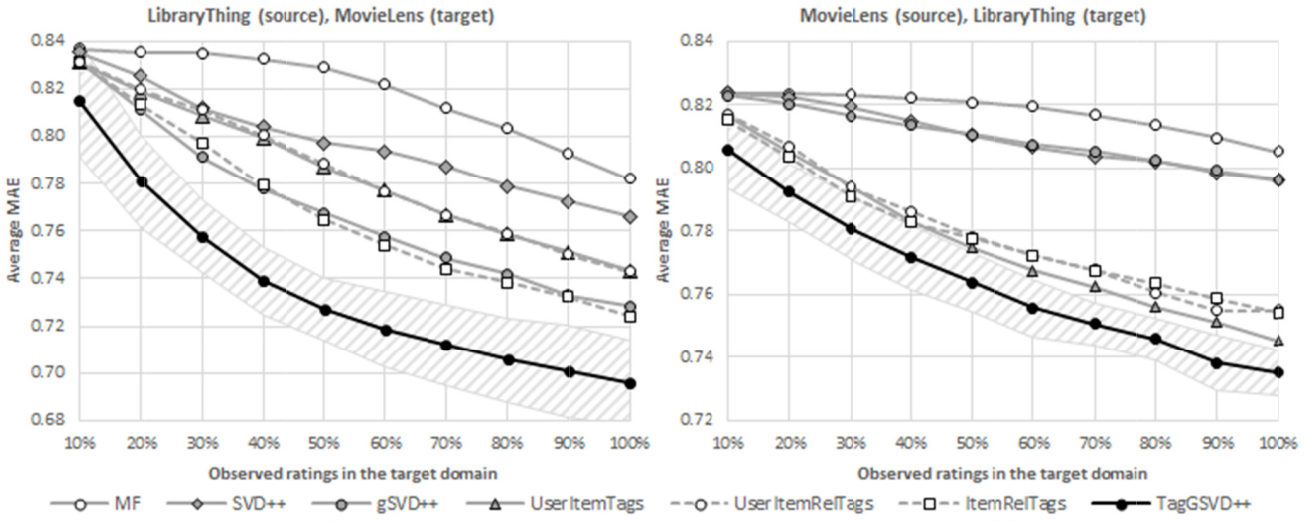


Figure 3. Average prediction error over the 10 folds for different amounts of observed ratings in the target domain. The striped area represents the range of values within two standard deviations from the mean. (a) Results using LibraryThing as source domain and MovieLens as target domain. (b) Results using MovieLens as source domain, and LibraryThing as target domain.

We note that the gSVD++, upon which our model is defined, also introduces additional latent variables and yet requires a lower regularization. We argue that the differences between gSVD++ and TagGSVD++ regularizations are caused by the $N(u)$ and $T_u$sets, see equations (6) with $G(i) = T_i$ and (11). In Table 1 we see that, on average, the number of tags applied by a user is much larger than the number of rated items. This results in more variables actually taking part in the rating predictions, and hence in a more complex model that requires more regularization to prevent overfitting.

Once we found the best parameters for each method and sparsity level, we ran the models separately in the test set of each fold. The final performance was estimated as the average MAE over the 10 folds. Figure 3a shows the results obtained using LibraryThing as source domain and MovieLens as target domain. All the differences with respect to the TagGSVD++ algorithm are statistically significant as determined with a Wilcoxon signed rank test at the 95% confidence level. It can be seen that the proposed TagGSVD++ method is able to consistently outperform the rest of the methods for all sparsity levels in the target domain, also in the cold-start setting when only 10%-20% of the ratings are available. We also note that cross-domain methods always achieve better accuracy than single-domain MF, although SVD++ effectively exploits implicit feedback and remains competitive until the 50% sparsity level. Then, as the sparsity decreases, cross-domain models provide greater improvements. This indicates that even if plenty of target domain rating data is available, it is still beneficial to transfer knowledge from the source domain.

The results using MovieLens as source domain and LibraryThing as target domain are shown in Figure 3b. As before, the difference in MAE between TagGSVD++ and the rest of the methods is statistically significant, according to the Wilcoxon signed rank test with 95% confidence level. Again, TagGSVD++ is the best performing method for all rating sparsity levels, followed by the cross-domain methods. We now observe that the values of MAE are in general larger than in the previous case, which seems to indicate that the transfer of knowledge is not as effective in this setting. This observation is in accordance with the results reported in [6], where the authors argue that this may be caused by differences in the ratio of overlapping tags between the domains. Only 13.81% of the tags in MovieLens are shared in LibraryThing (see Table 1), and thus less latent tag factors learned in the source domain can be used in the target to compute rating predictions.

Figure 4 shows the average rating prediction error for users with different amounts of observed ratings and tag assignments, using LibraryThing as source domain and MovieLens as target domain. We see that our model achieves the best improvements in cold-start situations, where few ratings and tag assignments are available in the target domain. We also note that the performance degrades for users with more than 20 ratings (respectively, 100 tag assignments), when enough target domain data is available. Nonetheless, in these cases, TagGSVD++ is still able to exploit the learned tag factors to compute more accurate predictions.

6. CONCLUSIONS AND FUTURE WORK

Cross-domain recommendation has potential benefits over traditional recommender systems that focus on single domains, such as alleviating rating sparsity in a target domain by exploiting data from a related source domain, improving the quality of recommendations in cold-start situations by inferring new user preferences from other domains, and by personalizing cross-selling strategies to provide customers with suggestions of items of different types.

Despite these advantages, cross-domain recommendation is a fairly new topic with plenty of research opportunities to explore. One of the major difficulties that arises is how to link or relate the different domains to support the transfer of knowledge. Due to the common heterogeneity of item attributes across domains, collaborative filtering techniques have become more popular than content-based methods. However, recent work [6][17] has concluded that more reliable and meaningful relations can be established between the domains by exploiting certain content information, such as social tags.

In this paper, we have adapted a novel extension of the well-known SVD++ algorithm to separately model the effect of user and item tags in the observed ratings. By introducing a new set of latent variables that represent tags in the user profile, our TagGSVD++ method is able to transfer knowledge from a source domain more effectively, providing accurate rating predictions in

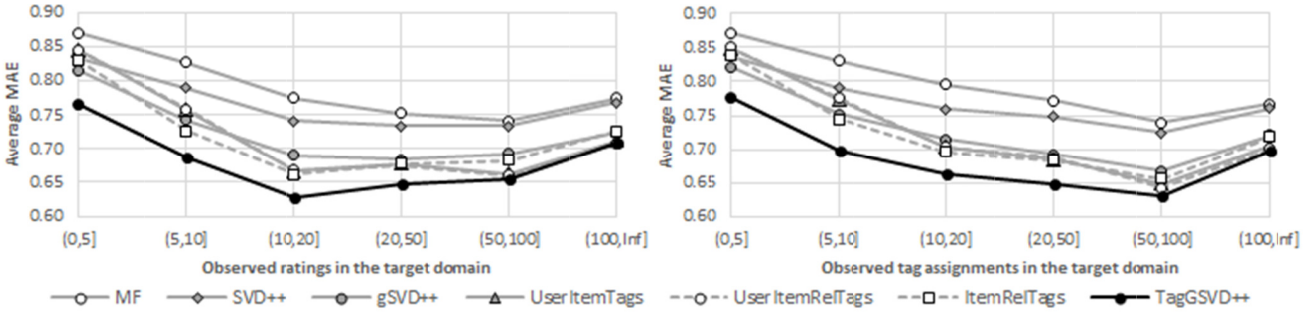


Figure 4. Average rating prediction error for users with different amounts of observed ratings (left) and tag assignments (right), using LibraryThing as source domain and MovieLens as target domain.

the target domain, even in cold-start situations. From our experiments in the movies and books domains, we conclude that exploiting additional tag factors, and decoupling user and item components in the factorization process improves the transfer of knowledge and the accuracy of the recommendations.

In the future, we plan to further investigate the effect of tags in the quality of recommendations. In particular, we want to study how the recommendation performance depends on the number of shared tags between domains. Increasing the overlap by grouping tags with similar semantics but expressed differently in the domains could favor the transfer of knowledge.

In our experiments we altered the amount of observed rating data in the target domain, but it would also be interesting to evaluate the methods varying the number of available ratings in the source domain. Moreover, we will perform a more exhaustive evaluation with other datasets including more cross-domain recommendation methods from the state of the art, such as [17].

7. ACKNOWLEDGEMENTS

This work was supported by the Spanish Government (TIN2013-47090-C3-2).

8. REFERENCES

[1] Abel, F., Helder, E., Houben, G.-J., Henze, N., Krause, D. 2013. Cross-system User Modeling and Personalization on the Social Web. *User Modeling and User-Adapted Interaction* 23(2-3), pp. 169-209.

[2] Adomavicius, G., Tuzhilin, A. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, pp. 734–749.

[3] Berkovsky, S., Kuflik, T., Ricci, F. 2008. Mediation of User Models for Enhanced Personalization in Recommender Systems. *User Modeling and User-Adapted Interaction* 18(3), pp. 245-286.

[4] Cao, B., Liu, N. N., Yang, Q. 2010. Transfer Learning for Collective Link Prediction in Multiple Heterogeneous Domains. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 159-166.

[5] Cremonesi, P., Tripodi, A., Turrin, R. 2011. Cross-domain Recommender Systems. In *Proceedings of the 11th IEEE International Conference on Data Mining Workshops*, pp. 496-503.

[6] Enrich, M., Braunhofer, M., Ricci, F. 2013. Cold-Start Management with Cross-Domain Collaborative Filtering and Tags. In *Proceedings of the 14th International Conference on E-Commerce and Web Technologies,* pp. 101-112.

[7] Fernández-Tobías, I., Cantador, I., Kaminskas, M., Ricci, F. 2012. Cross-domain Recommender Systems: A Survey of the State of the Art. In *Proceedings of the 2nd Spanish Conference on Information Retrieval*, pp. 187-198.

[8] Funk, S. 2006. Netflix Update: Try This At Home. `http://sifter.org/~simon/journal/20061211.html`

[9] Gao, S., Luo, H., Chen, D., Li, S., Gallinari, P., Guo, J. 2013. Cross-Domain Recommendation via Cluster-Level Latent Factor Model. In *Proceedings of the 17th and 24th European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 161-176.

[10] Garcia Manzato, M. 2013. gSVD++: Supporting Implicit Feedback on Recommender Systems with Metadata Awareness. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 908-913.

[11] Koren, Y. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 426-434.

[12] Koren, Y., Bell, R., Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42(8), pp. 30-37.

[13] Koren, Y., Bell, R. 2011. Advances in Collaborative Filtering. *Recommender Systems Handbook*, pp. 145-186.

[14] Li, B., Yang, Q., Xue, X. 2009. Can Movies and Books Collaborate? Cross-domain Collaborative Filtering for Sparsity Reduction. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 2052-2057.

[15] Pan, W., Xiang, E.W., Liu, N.N., Yang, Q. 2010. Transfer Learning in Collaborative Filtering for Sparsity Reduction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 210-235.

[16] Shapira, B., Rokach, L., Freilikhman, S. 2013. Facebook Single and Cross Domain Data for Recommendation Systems. *User Modeling and User-Adapted Interaction* 23(2-3), pp. 211-247.

[17] Shi, Y., Larson, M., Hanjalic, A. 2011. Tags as Bridges between Domains: Improving Recommendation with Tag-induced Cross-domain Collaborative Filtering. In *Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization*, pp. 305-316.

[18] Tiroshi, A., Berkovsky, S., Kaafar, M. A., Chen, T., Kuflik, T. 2013. Cross Social Networks Interests Predictions Based on Graph Features. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pp. 319-322.

[19] Winoto, P., Tang, T. 2008. If You Like the Devil Wears Prada the Book, Will You also Enjoy the Devil Wears Prada the Movie? A Study of Cross-Domain Recommendations. *New Generation Computing* 26, pp. 209-225.

[20] Zhang, Y., Cao, B., Yeung, D.-Y. 2010. Multi-Domain Collaborative Filtering. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pp. 725–732.

A Travel Recommender System for Combining Multiple Travel Regions to a Composite Trip

Daniel Herzog
TU München
Boltzmannstr. 3
85748 Garching
Germany
herzogd@in.tum.de

Wolfgang Wörndl
TU München
Boltzmannstr. 3
85748 Garching
Germany
woerndl@in.tum.de

ABSTRACT

Internet-based services are available to recommend destinations and activities for organized trips. Only few systems support travelers when creating composite trips consisting of multiple destinations or activities. The idea in this work is to select travel regions that maximize the value of the composite trip for the user while still respecting her limitations in time and money. The value of a travel region can be determined by the similarity between a specified user query and the cases in a travel region database. The recommendation algorithm needs to find a decent routing between the regions while still satisfying diversity of the whole trip. We developed an algorithm based on an approximation for the knapsack problem and extended it to recognize dependencies between the regions while calculating best combinations. It is able to determine the optimal duration of stay per region and its performance improves when benefiting from the hierarchical structure of our travel database. In an expert study, we verified the results of our approach. The study proves that our algorithm for composite trips delivers good recommendations which satisfied an expert user more than baseline algorithms. Regions in the composite trip fit together better and a decent routing between the regions can be ensured. Nevertheless, the algorithm leaves room for improvement by combining less similar regions in a composite trip, thus leading to a higher diversity of the recommendation.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Dynamic programming

General Terms

Algorithms

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

Keywords

case-based recommender system, tourist trip design problem, travel recommendation, knapsack problem, similarity

1. INTRODUCTION

Recommender systems are an established technology in online shops to provide users with a list of recommended items during their shopping experience. In comparison to recommending single items, travel recommendation turns out to be a more complicated issue since tourists have to deal with both limited budget and time frame. If users call for an individual trip composed of multiple regions, the task becomes even more complex. An exemplary situation illustrates the difficulty: A person is looking for a trip for four weeks in September with a budget of 2000 Euros; she likes nature and hiking with some cultural highlights as a plus. A high number of possible routes could be packaged to a composite trip and the constraints in time and money have to be taken into account. This is a special case of the so called tourist trip design problem (TTDP) [14]. TTDP is an extension of the orienteering problem (OP): A score which determines the value for the user is assigned to every location in a sequence. The goal is to maximize the sum of the scores of the selected locations while still meeting the given limitations [16].

Recommender systems use different algorithms to find a suitable list of recommendations in a feasible time. A subarea of content-based recommender systems is case-based recommender systems. Case-based recommenders rely on items structured as cases using a well defined set of features and feature values. Those cases are compared to a user query or her profile and the most similar cases are delivered as a recommendation [13].

These recommenders are already applied for travel recommendation. In order to recommend a longer trip, the cases have to be combined to maximize the value while still respecting the users' limitations. The simplest approach to determine the value of a composite recommendation is to add the values of all single parts of the proposed trip. This approach does not consider dependencies between the parts, for instance travel activities which cannot be executed at the same time as other activities. Furthermore, determining the maximum score of all possible combinations is not computationally feasible. This is the reason why these algorithms need to work with heuristics.

This paper aims to investigate possible solutions for combining multiple travel regions to a composite trip for in-

dependent travelers. The more specific research problem is whether an algorithm which recognizes dependencies between items and is based on a hierarchically structured data model can lead to better recommendations of a sequence of items. The rest of this paper is organized as follows. Section 2 looks at related work in the field of travel recommendation. In Section 3 we present the underlying data model we developed for testing our algorithm. Section 4 explains the main idea, the single steps and the implementation of our algorithm. This algorithm is evaluated in Section 5. Section 6 concludes our work and presents future work.

2. RELATED WORK

Research already focuses on travel recommendation and bundling k items to a recommendation package. Related work is [7] who developed the TAST model which represents the interests of tourists as well as extracted features of regions like the location or travel seasons. Travel packages are recommended to the user based on this model combined with additional information like prices. [17] developed a composite recommender system with access to one or more underlying recommender systems and therefore devises two algorithms for generating top-k packages as recommendations, both with high quality and one being much faster than the other. Furthermore, [1] explains how to bundle recommendation packages by regarding relationships and associations between the entities. Hence, the best recommendations for a given set of keywords can be determined. Early pruning and terminations strategies are used to develop an efficient algorithm.

[14] introduces the TTDP, shows the connection to the OP and presents a fast algorithm that produces solutions of better quality by using a guided local search meta-heuristic. [5] tries to solve the TTDP by presenting a cost-aware travel tour recommender, while Trip-Mine is looking for optimal trips by satisfying the user's travel time constraints [8].

[9] shows that case-based recommenders can be used to bundle travel items. The developed recommendation methodology Trip@advice stores recommendation sessions as cases. Furthermore, the user can give direct feedback to invoke suggested query changes during the recommendation process. [12] confirms that case-based reasoning and making association rules are solutions for recommending tourism travel packages. [3] devises a hybrid algorithm that additionally includes collaborative filtering in the recommendation process. The bundling of trips to packages in [15] is based on an object orientation solution which faces the high variation in travel requirements.

[10] developed DieToRecs, a travel recommender system which offers personalized interaction during the recommendation process to create individual bundles of trips. [11] shows the application of automatically collected constraints and preferences which can be added to user queries in order to improve the results of complex trip recommender systems. Further approaches make use of geo-location and pictures of the travel entities [4].

Our developed algorithm combines multiple travel regions to a composite trip. It is based on an approximation for the knapsack problem and extends the existing approaches by taking dependencies between single regions into account. The algorithm considers the fact that the value of each region in a composite trip is dependent on the presence or absence of other regions in the same recommendation. Furthermore, it calculates the optimal duration of stay per region in the composite trip and benefits from the hierarchical structure of the underlying data model.

3. UNDERLYING DATA MODEL

In this paper, we aim to recommend trips composed of multiple regions for individual traveling. We have developed a travel database with realistic data in order to test the proposed algorithm. The data model is composed of several layers ordered in a hierarchical manner as explained in this section.

In our model, a region is always a subregion of another region while the world is the parent region of every region. Regions contain the necessary information for recommendation:

- How good a region matches traveler types such as *Free spirits* or *Cultural explorer*, on a 5-point Likert scale
- Minimum and a maximum recommended duration of stay
- Minimum and optimal budget a traveler has to spend per pay
- Recommended periods of traveling in the region as 5-point Likert scale per month
- Security for travelers (crime level), on a 5-point Likert scale

If a region lacks specified attributes, it inherits the values from the parent region. We also model the connections between regions by specifying the necessary effort (time and cost) in one number to travel from one region to another. The connection cost between neighboring regions is 0. Regions are part of a country or can be spread over several countries. For example, larger countries such as the USA are divided in several travel regions, while smaller countries such as the Netherlands, Belgium and Luxembourg are combined into one region. Furthermore, regions contain one or more routes, i.e. concrete itineraries to visit a region. At this time, our systems recommends regions only, but in future work, routes can be considered for recommendation as well. The model can also be extended with additional attributes such as spoken languages in regions. Regions can be assigned to the countries they belong to and store additional information like visa requirements, but we do not use countries for recommendation at this time.

Figure 1 illustrates the data model by showing an example with several layers of regions. USA Southwest, USA Pacific Northwest and Western Canada are subregions which can be recommended by our algorithm. When developing and testing our algorithm, our database was composed of a total of 152 regions with 124 leaves in the region tree which can be recommended in a trip. The data was complied based on various Internet sources.

A user who asks for a recommendation chooses one traveler type which expresses her expectations towards trips. The offered traveler types such as *Free spirits* or *Cultural explorer* are inspired by a market segmentation tool of the Canadian Tourism Commission[1]. Based on the described

[1]http://en-corporate.canada.travel/resources-industry/explorer-quotient

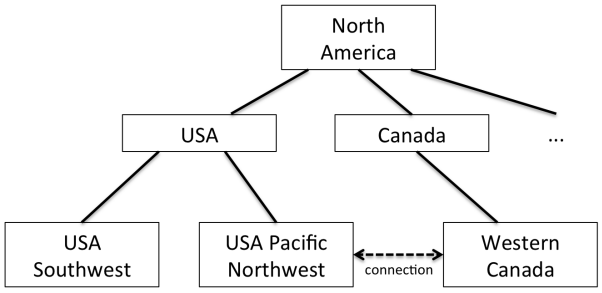


Figure 1: Data example from region tree

preferences and suggested activities for each traveler type, we rated each region in the database to express how it matches the traveler types on a 5-point Likert scale.

4. ALGORITHM FOR THE RECOMMENDATION OF COMPOSITE TRIPS

In this section, we present our algorithm to recommend trips composed of multiple travel regions. This algorithm can serve as a basis for a fully functional travel recommendation application.

4.1 Basic idea and goals

Composite trip recommendation can be seen as a special case of the knapsack problem to find best combinations of multiple travel items [17]. The underlying idea is to combine as many single travel items like regions, routes or activities as necessary to maximize the benefit for the user while still respecting existing limitations like time and money. The problem can be described formally as:

maximize

$$\sum_{i=1}^{n} f_i(x_i, X_i)$$

subject to

$$\sum_{i=1}^{n} d_i \cdot x_i \leq D$$

and

$$\sum_{i=1}^{n} b_i \cdot x_i \leq B$$

with

$$x_i \in \{0, 1\}$$

where $f_i(x_i, X_i)$ is the value function for item i and X_i is the set of items upon which the value of item i depends. x_i is either 0 or 1 which means that each travel item can be added to the travel sequence only once (or not). d_i is item i's recommended duration of stay and b_i is item i's recommended daily budget. D is the maximum possible duration of stay and B is the maximum budget the user can spend on her trip. In this paper, travel items are represented by regions from all over the world.

The first challenge that arises is to determine the value a single region offers to the user. This value is dependent on the user's requirements and preferences for her trip. Hence, a travel recommendation algorithm needs to collect information like the user's traveling type or preferred activities, her intended period of traveling and her monetary and time limitations. The algorithm needs a predefined way to calculate the value by using this information. To decide whether a region will be considered for a trip, we calculate a rating (see below).

In our case, the problem gets more complicated than the standard knapsack problem because the value of a region is not only determined by the user query. Rather, it depends on the presence or absence of other regions in the recommended composite trip. This extension of the knapsack problem is called the Oregon Trail Knapsack Problem [2]. Imagine a travel sequence that recommends visiting Germany, Austria and Switzerland. If the user could spend more time and money, the system can add further regions to this trip. Depending on the user's preferences, additional regions in or close to Central Europe should be recommended. Only exceptional circumstances legitimate an additional region far from Central Europe because the effort of visiting this region during this trip is disproportional. The value of the composite trip itself is lower than the sum of the region values because the distance between regions has a negative impact on the composite trip.

In addition, a region's value in a composite trip influences the diversity of the sequence. For instance, if the recommender accepts different activities as user input, the algorithm should focus on a decent coverage of all demanded activities. Downgrading the values of similar regions, when combining them in a recommendation, allows avoiding situations where only a few activities are served. If a user wants to go skiing and swimming, a recommender should not only recommend regions for one of these activities, but both.

In this paper, regions instead of routes are recommended. Regions in our data model are not limited to specific activities, diversity is mainly expressed by the the duration of stay in each recommended region. Every additional week a user stays in a region leads to a lower value for the user because the probability that she explored this region enough is increasing. Hence, another goal of the algorithm is determining the optimal duration of stay per region in the travel sequence.

To conclude, a region reaches its maximum value for a given query when regarding it exclusively, and not within a composite trip. Other regions in the composite trip can decrease this value because of increasing distances and lack of diversity. This negative impact can be calculated by a penalty function. Hence, the value function for the composite trip recommendation problem extends the value functions of the Oregon Trail Knapsack Problem by a penalty function [2]. The resulting value function can be described formally as:

$$f_i(x_i, X_i) = x_i \cdot v_i - \sum_{e \in X_i} (t(i, e) \cdot x_i \cdot v_i \cdot [x_e > 0])$$

where v_i is the value of region i for the specified user query and $t(i, e)$ is the penalty function for two regions i and e. $[x_e > 0]$ represents a Boolean value whether item i is in the knapsack. The algorithm needs to provide an implementation of $t(i, e)$.

After determining the best regions and the durations of

stay for the final composite trip, an optimal routing should be ensured. The problem of combining regions to a composite trip is part of the orienteering problem. Basically, the orienteering problem extends the knapsack problem by charging time and money for the routing between the travel items [14]. Hence, finding the shortest and cheapest routing between regions reduces loss of time and money when executing the algorithm.

We have developed a travel sequence recommender based on the explained data model (cf. Section 3). The algorithm exploits the hierarchical structure. For instance, if the user wants to travel through Europe but already explored Scandinavia, there is no need to execute any calculations for other continents or Scandinavian regions. As a consequence strictly respecting the hierarchical structure promises improvements in the algorithm's runtime.

The basic idea is implemented in an algorithm that aims to achieve these targets. It is composed of three phases which are gradually executed on the available database:

1. Reduce number of regions
2. Rate regions
3. Calculate the best combination of regions

First, the approach reduces the number of considered regions for the recommendation process. Users can explicitly exclude regions in their query in our approach. If one or more regions are excluded, all subregions are removed from consideration as well, utilizing the hierarchical structure of our travel database. Reducing the number of regions means fewer items for the next two steps and therefore, the algorithm's runtime is improved.

In the following, we describe the other two steps in more detail.

4.2 Rate regions

The remaining travel items of the pruned region tree are rated in the following step. In our case, only the leaves are considered for recommendation, reducing the number of items in the rating phase. This behavior can be adapted according to the recommender's use case.

Travel regions in our scenario can be rated in several ways. For case-based recommender systems, the similarity between the user query and the case - the region - can be calculated in order to determine the value of a region for a specified user query. [13] explains that the assessment of similarity in case-based recommender systems involves combining the individual feature level similarities for relevant features. Relevant features in our travel recommender are the traveling type served by a region, recommended traveling periods etc. (cf. Section 3). Budget and duration of stay are not taken into account at this step because they are used as the constraints for the knapsack algorithm (cf. Section 4.3). The similarity between single features of the query and the case can be calculated with the following formula:

$$sim_{feature}(f_q, f_c) = 1 - \frac{|f_q - f_c|}{max(f_q, f_c)}$$

[13], where f_q is the feature expectation defined in the user query and f_c is the actual feature value in the case. The feature level similarities determine the similarity between the query and the case by weighting all features:

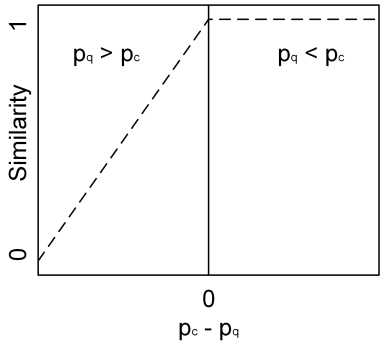


Figure 2: Asymmetric similarity metric (adapted from [13])

$$Similarity(q, c) = \frac{\sum_{i=1...n} w_i \cdot sim_i(q_i, c_i)}{\sum_{i=1...n} w_i}$$

[13]. In our case, the traveling type and the traveling period are weighted higher than other features like crime level because the assumption is that the first two features are more important for the decision.

[13] presents two examples of similarity metrics which consider deviation of a case feature from the user query. A symmetric similarity metric reduces the similarity by the same value if the query feature is lower or higher than the case feature. An asymmetric metric prefers either higher or lower values. For example, the price of an item usually corresponds to an asymmetric metric. If a user is prepared to pay 1000 euros for an item, a price of 800 euros satisfies her requirement better than a price of 1200 euros.

For our recommender, we use the following asymmetric similarity metric. All features of the regions are rated in a range between 0 and 1. For example, the perfect month to visit a region is rated with 1, while the worst possible rating is 0 (the particular month is not recommended at all). The deviation of the query from a case reduces the overall similarity if the case feature is rated lower than the expected value which usually is 1, the best value. If a user expects a region only to fit somewhat with regard to a certain feature, the query feature will be rated with a value lower than 1. A higher value of the case feature always leads to a similarity of 1 since no user would complain about a feature being better served by a region than expected. Figure 2 illustrates this metric. The similarity reaches the maximum when the case feature is rated greater or equal 1.

At the end of step 2, regions with a low rating are removed from consideration in order to reduce the number of items for step 3. In our implementation, we exclude regions with a rating lower than 0.7 on the scale from 0 to 1. This approach also prevents composite trips including regions which do not satisfy the user's demands but could be cheap enough to be considered in a recommendation only for the use of free capacities.

4.3 Calculate the best combination of regions

In step 3, the remaining regions in the recommendation process are combined in a way that maximizes the value for the user while still respecting her limitations in time and money.

The algorithm extends the classic knapsack problem in order to achieve two additional objectives. First, the value of a composite trip should be decreased according to the distances of regions. After testing several variants, we implemented a penalty function $t(i, e)$ which decreases the total value of the composite trip by a percentage depending on the connection costs between the regions i and e. This implementation is evaluated in the expert study in Section 5.

The second objective is the determination of the optimal duration of the stay per region while calculating the best combination of regions. We assume that travelers stay longer in regions with a significant higher value than in other regions of the same composite trip. On the other hand, simply recommending the maximum possible duration of stay in the best-rated region of phase two is not the best solution either because this could decrease the diversity of the recommended trip when visiting only a low number of regions.

Therefore, we suggest regarding every week of every region separately. The value of staying in a region for an additional week is lower than in the week before. We decided to decrease the value after every week by a constant between 5 and 10 percent. The lower the maximum duration of traveling, the higher the deduction. This ensures diversity even for short trips. Splitting regions in one week blocks means that the algorithm is executed on an increased number of items which extends the runtime. Nevertheless, this approach allows determining the optimal durations of stay. If diversity is only determined by the duration of stay in each region, there is no need to extend the penalty function $t(i, e)$ because deductions are already stored in the one week blocks. If, for instance, routes characterized by activities are recommended, the penalty function can be extended by a formula which calculates the diversity of routes.

The knapsack problem itself can be solved by different approximation. We decided to implement a dynamic programming solution which promises good results by splitting the problem into smaller subproblems [6]. This approach iterates over the number of available regions n as well over all limits, in our case the budget B and maximum duration of stay D. The runtime of this algorithm is $O(n \cdot B \cdot D)$. When executing, it creates a three dimensional matrix with $n \cdot B \cdot D$ subproblems. The maximum possible value for a composite trip can then be derived from this matrix. We store the selected regions in every entry of the matrix to access the regions of our final recommendation after the algorithm has terminated.

Shortest path algorithms can be applied on the final recommendation in order to optimize the routing. For our scenario, we implemented a simple approach for finding the best sequence of regions in the trip. For each added region in a knapsack with at least one region, the algorithm calculates at which position the new region includes the lowest additional costs. This region is then inserted at the identified position.

4.4 Implementation

We developed a Java 1.7 application in order to implement the algorithm and to test it in a real scenario. The travel data of our data model is stored in a NoSQL database. The application accesses regions, connections and the corresponding data by parsing an online provided JSON file.

The application offers a simple user interface which allows specifying individual queries for composite trip recommendation. The user can enter one predefined traveling type, her preferred month of traveling, her budget and average spending as well as the maximum possible duration of traveling. Furthermore, regions can be excluded from the recommendation process.

Table 1: Recommended composite trip for an example query

North Argentina and Paraguay	2 weeks	EUR 560
Bolivia	3 weeks	EUR 525
Peru	3 weeks	EUR 630
Sum	8 weeks	EUR 1715

Table 1 illustrates the outcome of an example query[2]. Imagine, a user sees herself as cultural explorer who wants to travel in August. She has a budget of 2000 euro and usually spends a low amount of money when traveling. Her maximum time of traveling is eight weeks. As she already knows Europe and Asia very well, she excludes these regions in her query. Every query is composed of this information and it is automatically extended by an expectation of the lowest possible crime rate. In this example, the best recommendation is composed of three different regions with total costs of 1715 euro. The costs already include local transport within the region and to the borders. The selected regions can be visited overland by passing mutual borders. This is why there are no connection costs in this case. Additional costs would occur if the recommendation demands traveling to a non-neighboring country or region. For future work, we suggest extending the model by specific costs for other means of transportation like buses or trains in order to calculate the overall costs more accurately. The long-distance travel from the starting point of the trip to the first region is never included.

5. EXPERT STUDY

The research question of this paper is whether an algorithm which recognizes dependencies between items and is based on a hierarchically structured data model can improve travel recommendations. We developed an algorithm that is able to consider distances between single regions and ensure diversity by calculating optimal durations of stay. We conducted an expert study to evaluate its performance. The expert had to be familiar with the available traveler types and has knowledge in the travel domain.

5.1 Procedure of the study

Our user study is composed of 56 sample queries. Basically, the queries are selected randomly but we tried to define 7 queries per traveler type. Furthermore, each query in all of the 8 traveling type groups changes only one or two features compared to the precedent query like an increased budget or less time to travel. This allows to understand how a recommendation changes if you modify certain features.

The developed Java application executes each query and presents the best-rated recommendation, based on the explained procedure. In addition, two additional (baseline) algorithms deliver two further recommendations. All of these

[2]The stated cost covers the minimum a traveler would need to spend in these regions, the cost is (much) higher when the user specifies average or high amount of spending

three recommendations are presented in a random order in order to prevent the expert from relating the recommendations to the algorithms. The recommendations are presented to the expert like in table 1, with the duration of stay per region and the total costs of the trip.

The expert rated all recommendations in these four categories on a 5-point Likert scale: 1. General satisfaction with the recommendation, 2. the diversity of the recommendation, 3. how well the single recommendation items fit together and 4. if the routing between the singles items is reasonable.

5.2 Comparative algorithms

Two further algorithms deliver recommendations based on two different approaches: A baseline algorithm that is a standard implementation to solve the knapsack problem, and a top-k algorithm that selects regions sequentially from an ordered list of rated regions. Both algorithms are used for comparison in the expert study.

Related works already implement variations of the classical knapsack problem [17] or refer travel package recommendation to the orienteering problem [14]. The baseline algorithm works like our presented travel recommendation algorithm but does not include our extensions. Thus, the values of items in a composite trip are not dependent on the presence or absence of other items and no weekly calculated penalties are considered. This algorithm allows evaluating the influence our extensions have on the quality of composite trip recommendations.

The top-k algorithm ranks all available regions by their value in a list. It is able to implement weekly calculated penalties and to decrease the total value by the distances to be covered, but it tries to find the best combination of regions in a simpler approach. The algorithm inserts every region from top to bottom in the ordered list into the recommendation sequence. If a region cannot be inserted because of the constraints with regard to money or time, the next region in the list is checked. Hence, the algorithm goes through the list of regions exactly once. The top-k algorithm allows determining if a simpler approach of the knapsack problem can lead to similar results as our more complex travel recommendation algorithm.

5.3 Results

Figure 3 illustrates the performance of the three algorithms in each of the four categories. Our travel recommendation algorithm resulted in the highest overall satisfaction to the expert (∅: 4.02, σ: 0.82). Furthermore, it is best-rated in the categories *regions fit together* (∅: 4.29, σ: 0.76) and *routing* (∅: 4.16, σ: 0.95). In terms of diversity, our travel recommendation algorithm (∅: 3.11, σ: 0.91) is rated somewhat lower than the two comparative algorithms. In this category, the baseline algorithm is ranked first place (∅: 3.57, σ: 0.84). The top-k algorithm is ranked second in every category. The results show that in 3 out of 4 categories, our extensions lead to significant improvements. Some improvements can also be observed when applying the top-k algorithm over the baseline, but the more complex knapsack based travel recommendation algorithm performs better than both.

The expert study delivers further insights into our travel recommendation algorithm. 29 queries asked for trips with a maximum duration of less or equal than six weeks, 27 queries

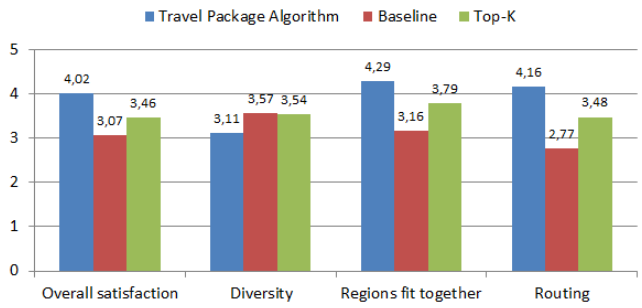


Figure 3: Evaluated travel recommendation algorithms

for more than 6 weeks. For higher durations of stay, the difference in the overall satisfaction with the recommendations is comparable, but our travel recommender algorithm rates better in how regions fit together (∅: 4.41, σ: 0.84) and in terms of the routing (∅: 4.26, σ: 1.06). On the other hand, queries with a higher maximum duration of stay are rated lower in terms of diversity (∅: 3.00, σ: 0.92).

25 queries asked for recommendations with a maximum budget per week of lower or equal than 500 euro. This does not have a significant impact on the quality of the recommendation except in terms of routing which is a bit better for lower budget inputs (∅: 4.28, σ: 0.94) than for higher inputs (∅: 4.06, σ: 0.96).

Our travel recommendation algorithm offers the possibility to limit the recommendation process to preselected regions (step 1 in the process). In the expert study, 16 of the 56 queries had some constraints on the regions. These queries deliver recommendations that satisfied the expert less than average (∅: 3.75, σ: 1.00). Furthermore, single regions fit together less (∅: 3.81, σ: 0.83) and the routing gets worse (∅: 3.75, σ: 1.06) in these cases.

6. CONCLUSION

Recommending composite trips can be understood as a knapsack problem with extensions. Single recommendation items like regions or routes offer value to the user, depending on the similarity between the items and the user query. Multiple regions can be combined to a composite trip which respect the user's limitations in time and money. The total value for the user can not be determined by summing up the single values of all travel stages. Regions in a composite trip are dependent on the presence or absence of other regions in the same recommendation. The distance between regions decreases the possible value for a specified user query because of the costs of the connection.

In this paper, we present a travel recommendation algorithm for composite trips that addresses the Oregon Trail Knapsack Problem and applies the problem to the travel domain. The recommendations generated by our algorithm satisfies an expert user more than comparative algorithms. It is able to combine regions and can determine the optimal duration of stay per region. We showed that the recommended regions fit together and with the availability of connection costs between regions, a decent routing can be ensured. A high maximum duration of stay allows choosing among a higher number of regions and this improves the selection of regions and allows better routing. A low max-

imum budget avoids high connection costs and thus also promises better routing. In terms of diversity, our algorithm performs below average. We integrated mechanisms which penalize long stays in the same region. Nevertheless, the recommended trips offer less diversity than the baseline algorithms. Further effort is necessary in order to understand the preferences of potential users and to find out how a better diversity can be ensured without recommending regions which fit less together. One possibility is extending the penalty function in our algorithm.

Restrictions are useful for users when they want to ignore specific regions in the recommendation process. Reducing the number of regions for the recommendation process improves the algorithm's runtime but also reduces the quality of recommendations. Regional constraints made it more difficult to find regions which match the user's preferences. Moreover, a limited choice of regions makes it more difficult to find regions which fit together and allow a decent routing.

Future work is to extend the system by recommending routes or concrete itineraries in addition to travel regions. In addition, an improved version of the algorithm could also take possible individual activities of travelers such as hiking or shopping into account. In this case, different routes with similar activities could result in additional deduction of the rating and thus reducing the value of this trip. Furthermore, we plan to conduct a more extensive user study with potential users in order to test the recommendations of the travel recommendation algorithm for composite trips.

7. REFERENCES

[1] A. Angel, S. Chaudhuri, G. Das, and N. Koudas. Ranking Objects Based on Relationships and Fixed Associations. In *EDBT '09 Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 910–921, 2009.

[2] J. J. Burg, J. Ainsworth, B. Casto, and S.-D. Lang. Experiments with the "Oregon Trail Knapsack Problem". *Electronic Notes in Discrete Mathematics*, 1:26–35, 1999.

[3] J.-H. Chen, K.-M. Chao, and N. Shah. Hybrid Recommendation System for Tourism. In *2013 IEEE 10th International Conference on e-Business Engineering (ICEBE)*, pages 156–161. Ieee, Sept. 2013.

[4] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *HT '10 Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 35–44, 2010.

[5] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen. Cost-aware travel tour recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, pages 983–991, New York, New York, USA, 2011. ACM Press.

[6] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[7] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized Travel Package Recommendation. In *2011 IEEE 11th International Conference on Data Mining*, pages 407–416. Ieee, Dec. 2011.

[8] E. H.-C. Lu, C.-Y. Lin, and V. S. Tseng. Trip-Mine: An Efficient Trip Planning Approach with Travel Time Constraints. In *2011 IEEE 12th International Conference on Mobile Data Management*, pages 152–161. Ieee, June 2011.

[9] F. Ricci, D. Cavada, N. Mirzadeh, and A. Venturini. Case-based travel recommendations. In *Destination recommendation systems: behavioural foundations and applications.*, pages 67–93. CABI, 2006.

[10] F. Ricci, D. R. Fesenmeier, N. Mirzadeh, H. Rumetshofer, E. Schaumlechner, A. Venturini, K. W. Wöber, and A. H. Zins. DieToRecs: a Case-based Travel Advisory System. In *Destination recommendation systems: behavioural foundations and applications.*, pages 227–239. CABI, 2006.

[11] A. Savir, R. Brafman, and G. Shani. Recommending improved configurations for complex objects with an application in travel planning. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 391–394, New York, New York, USA, 2013. ACM Press.

[12] M. Schumacher and J.-P. Rey. Recommender systems for dynamic packaging of tourism services. In R. Law, M. Fuchs, and F. Ricci, editors, *ENTER*, pages 13–23. Springer Vienna, 2011.

[13] B. Smyth. Case-based recommendation. *The adaptive web*, 4321:342–376, 2007.

[14] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. V. Berghe, and D. V. Oudheusden. a Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides. *Applied Artificial Intelligence*, 22(10):964–985, Oct. 2008.

[15] C. Tan, Q. Liu, E. Chen, H. Xiong, and X. Wu. Object-oriented Travel Package Recommendation. *ACM Transactions on Intelligent Systems and Technology*, 5(3):26, 2014.

[16] P. Vansteenwegen and D. V. Oudheusden. The mobile tourist guide: an OR opportunity. *OR Insights*, 20(3):21–27, 2007.

[17] M. Xie, L. V. Lakshmanan, and P. T. Wood. Breaking out of the box of recommendations: from items to packages. In *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, pages 151–158. ACM Press, 2010.

Linked Open Data-enabled Strategies for Top-N Recommendations

Cataldo Musto
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
cataldo.musto@uniba.it

Pierpaolo Basile
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
pierpaolo.basile@uniba.it

Pasquale Lops
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
pasquale.lops@uniba.it

Marco de Gemmis
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
marco.degemmis@uniba.it

Giovanni Semeraro
Dept. of Computer Science
Univ. of Bari Aldo Moro, Italy
giovanni.semeraro@uniba.it

ABSTRACT

The huge amount of interlinked information referring to different domains, provided by the Linked Open Data (LOD) initiative, could be effectively exploited by recommender systems to deal with the *cold-start* and *sparsity* problems.

In this paper we investigate the contribution of several features extracted from the Linked Open Data cloud to the accuracy of different recommendation algorithms. We focus on the *top-N* recommendation task in presence of binary user feedback and cold-start situations, that is, predicting ratings for users who have a few past ratings, and predicting ratings of items that have been rated by a few users.

Results show the potential of Linked Open Data-enabled approaches to outperform existing state-of-the-art algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords

Content-based Recommender Systems; Top-N recommendations; Implicit Feedback; Linked Open Data; DBpedia

1. INTRODUCTION

Recently, novel and more accessible forms of information coming from different open knowledge sources represent a rapidly growing piece of the big data puzzle.

Over the last years, more and more semantic data are published following the Linked Data principles[1], by connecting information referring to geographical locations, people, companies, book, scientific publications, films, music, TV and radio programs, genes, proteins, drugs, online communities, statistical data, and reviews in a single global data space, the *Web of Data* [2].

This information, interlinked with each other, forms a global graph called *Linked Open Data cloud*, whose nucleus is represented by `DBpedia`[2].

A fragment of the Linked Open Data cloud is depicted in Figure 1.

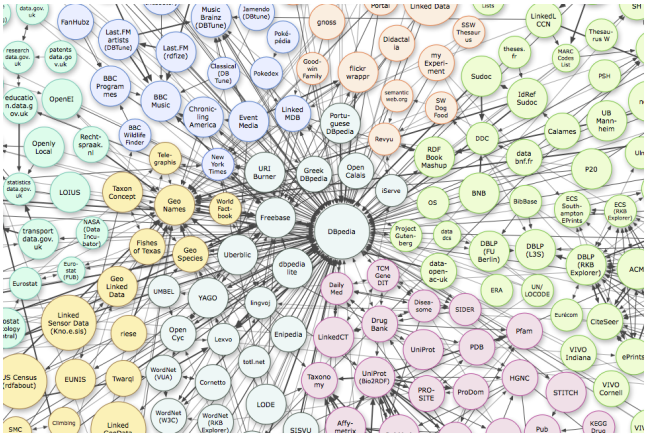


Figure 1: Fragment of the Linked Open Data cloud (as of September 2011).

Using open or pooled data from many sources, often combined and linked with proprietary big data, can help develop insights difficult to uncover with internal data alone [4], and can be effectively exploited by recommender systems to deal with classical problems of cold-start and sparsity.

On the other hand, the use of a huge amount of interlinked data poses new challenges to recommender systems researchers, who have to find effective ways to integrate such knowledge into recommendation paradigms.

This paper presents a preliminary investigation in which we propose and evaluate different ways of including several kinds of Linked Open Data features in different classes of recommendation algorithms. The evaluation is focused on the *top-N* recommendations task in presence of binary user feedback and cold-start situations.

[1] http://www.w3.org/DesignIssues/LinkedData.html

[2] http://dbpedia.org

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

This paper extends our previous work carried out to participate to the Linked Open Data-enabled Recommender Systems challenge[3] [1], by presenting results for new tested algorithms, along with the various combinations of features. Results show the potential of Linked Open Data-enabled approaches to outperform existing state-of-the-art algorithms.

2. RELATED WORK

Previous attempts to build recommender systems that exploit Linked Open Data are presented in [17], where a music recommender system uses DBpedia to compute the *Linked Data Semantic Distance*, which allows to provide recommendations by computing the semantic distance for all artists referenced in DBpedia.

In that work, the semantics of the DBpedia relations is not taken into account, differently from the approach described in [5], where properties extracted from DBpedia and LinkedMDB [12] are exploited to perform a semantic expansion of the item descriptions, suitable for learning user profiles.

In [15], DBpedia is used to enrich the playlists extracted from a Facebook profile with new related artists. Each artist in the original playlist is mapped to a DBpedia node, and other similar artists are selected by taking into account shared properties, such as the genre and the musical category of the artist.

DBpedia is also used in [16] to capture the complex relationships between users, items and entities by extracting the paths that connect users to items, in order to compute recommendations through a *learning to rank* algorithm called *SPRank*. *SPRank* is a hybrid recommendation algorithm able to compute *top-N* item recommendations from implicit feedback, that effectively incorporates ontological knowledge coming from DBpedia (content-based part) with collaborative user preferences (collaborative part) in a graph-based setting. Starting from the common graph-based representation of the content and collaborative data models, all the paths connecting the user to an item are considered in order to have a relevance score for that item. The more paths between a user and an item, the more that item is relevant to that user.

The increasing interest in using Linked Open Data to create a new breed of content-based recommender systems is witnessed by the success of the recent Linked Open Data-enabled Recommender Systems challenge held at the European Semantic Web Conference (ESWC 2014). The contest consisted of 3 tasks, namely rating prediction in cold-start situations, top-N recommendation from binary user feedback, and diversity. Interestingly, top-N recommendation from binary user feedback was the task with the highest number of participants. The best performing approach was based on an ensemble of algorithms based on popularity, Vector Space Model, Random Forests, Logistic Regression, and PageRank, running on a diverse set of semantic features [1]. The performance of the single methods were aggregated using the Borda count aggregation strategy. Most of the techniques used in the contest are presented in this paper.

Similarly to the best performing approach, the second best performing one was based on the same ingredients [18]. Indeed, it combined different base recommenders, such as collaborative and content-based ones, with a non-personalized recommender based on popularity. Content-based strategies leveraged various features sets created from DBpedia. Additional Linked Open Data sources were explored, such as British Library Bibliography[4] and DBTropes[5], even though they did not provide meaningful features with respect to those derived from DBpedia. The results of the individual recommenders were combined using stacking regression and rank aggregation using Borda.

[3] challenges.2014.eswc-conferences.org/index.php/RecSys

3. METHODOLOGY

Section 3.1 describes the set of different features extracted from the Linked Open Data cloud, while Section 3.2 presents different kinds of recommendation algorithms, i.e. those based on vector space and probabilistic models, those based on the use of classifiers, and graph-based algorithms, which are fed in different ways by the features extracted from the Linked Open Data cloud.

3.1 Features extracted from the Linked Open Data cloud

The use of Linked Open Data allows to bridge the gap between the need of background data and the challenge to devise novel advanced recommendation strategies.

There are two main approaches to extract Linked Open Data features to represent items:

1. use of the *Uniform Resource Identifier* (URI)
2. use of *entity linking* algorithms.

The first approach directly extracts DBpedia properties for each item by using its *Uniform Resource Identifier* (URI). URIs are the standard way to identify real-world entities, and allow to define an entry point to DBpedia.

However, DBpedia provides a huge set of properties for each item, hence a proper strategy to select the most valuable ones is necessary. We could manually identify and select a subset of domain-dependent properties, or we could take into account a subset of the most frequent ones.

Referring to the book domain, in which we performed the evaluation, we selected the 10 properties in Table 1, which are both very frequent and representative of the specific domain.

Starting from these properties, further resources could be recursively added. For example, starting from a book, we could retrieve its author through the property
`http://dbpedia.org/ontology/author`
and then retrieve and link other resources by the same author, or other genres of works by the same author.

As an example, the resulting representation obtained for the book *The Great and Secret Show* is provided in Figure 2. The book is linked to its author (*Clive Barker*), to the genre (*Fantasy literature*), and to the Wikipedia categories (*British fantasy novels* and *1980s fantasy novels*). Furthermore, other books by Clive Barker are reported, such as *Books of Blood* and *Mister B. Gone*.

The second approach to extract LOD features uses *entity linking* algorithms to identify a set of Wikipedia concepts occurring in the item description. Next, those Wikipedia concepts can be easily mapped to the corresponding DBpedia nodes.

[4] http://bnb.data.bl.uk/

[5] http://skipforward.opendfki.de/wiki/DBTropes

Property	Description	Frequency
http://dbpedia.org/ontology/wikiPageWikiLink	Link from a Wikipedia page to another Wikipedia page. This property allows to take into account other Wikipedia pages which are somehow related.	523,321
http://purl.org/dc/terms/subject	The topic of a book.	38,627
http://dbpedia.org/property/genre	The genre of a book.	12,488
http://dbpedia.org/property/publisher	The publisher of a book.	9,798
http://dbpedia.org/ontology/author	The author of a book.	8,669
http://dbpedia.org/property/followedBy	The book followed by a specific book.	6,351
http://dbpedia.org/property/precededBy	The book preceded by a specific book.	5,978
http://dbpedia.org/property/series	The series of a book.	3,196
http://dbpedia.org/property/dewey	The Dewey Decimal library Classification.	1,930
http://dbpedia.org/ontology/nonFictionSubject	The subject of a non-fiction book (e.g.: history, biography, cookbook, ...).	966

Table 1: DBpedia properties selected for the book domain.

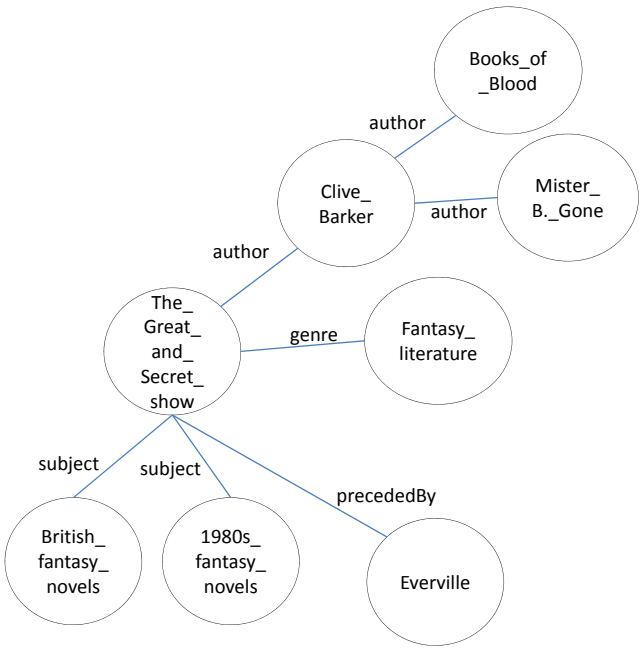


Figure 2: Properties of the book *The Great and Secret Show* by *Clive Barker*.

Several techniques can be adopted, such as `Explicit Semantic Analysis` [8] or `Tagme` [7].

In this work we adopt `Tagme`, that implements an anchor disambiguation algorithm to produce a Wikipedia-based representation of text fragments, where the most relevant concepts occurring in the text are mapped to the Wikipedia articles they refer to. `Tagme` performs a sort of feature selection by filtering out the noise in text fragments, and its main advantage is the ability to annotate very short texts.

As an example, the resulting representation obtained for the book *The Great and Secret Show* is provided in Figure 3. Interestingly, the technique is able to associate several concepts which are somehow related to the book, and which could be useful to provide accurate and diverse recommendations, as well.

Figure 3: Tagme representation of the book *The Great and Secret Show* by *Clive Barker*.

All these features are used in different ways by the different recommendation algorithms presented in the following section. Details are reported in Section 4.2.

3.2 Recommendation Algorithms

We tested three different classes of algorithms for generating *top-N* recommendations, by using several combinations of features extracted from the Linked Open Data cloud.

3.2.1 *Algorithms based on the Vector Space and Probabilistic Models*

Most content-based recommender systems rely on simple retrieval models to produce recommendations, such as keyword matching or Vector Space Model (VSM).

VSM emerged as one of the most effective approaches in the area of Information Retrieval, thanks to its good compromise between effectiveness and simplicity. Documents and queries are represented by vectors in an n-dimensional vector space, where n is the number of index terms (words, stems, concepts, etc.).

Formally, each document is represented by a vector of weights, where weights indicate the degree of association between the document and index terms.

Given this representation, documents are ranked by computing the distance between their vector representations and the query vector. Let $D = \{d_1, d_2, ..., d_N\}$ denote a set of documents or corpus, and $T = \{t_1, t_2, ..., t_n\}$ be the dictionary, that is to say the set of words in the corpus. T is obtained by applying some standard natural language processing operations, such as tokenization, stopwords removal, and stemming. Each document d_j is represented as a vector in a n-dimensional vector space, so $d_j = \{w_{1j}, w_{2j}, ..., d_{nj}\}$, where w_{kj} is the weight for term t_k in document d_j. The

most common weighting scheme is the TF-IDF (Term Frequency-Inverse Document Frequency).

In content-based recommender systems relying on VSM, the *query* is the *user profile*, obtained as a combination of the index terms occurring in the items liked by that user, and recommendations are computed by applying a vector similarity measure, such as the *cosine* coefficient, between the user profile and the items to be recommended in the same vector space.

However, VSM is not able to manage either the latent semantics of each document or the position of the terms occurring in it. Hence, we proposed an approach able to produce a *lightweight* and *implicit* semantic representation of documents (items and user profiles). The technique is based on the *distributional hypothesis*, according to which *"words that occur in the same contexts tend to have similar meanings"* [11]. This means that the meaning of a word is inferred by analyzing its usage in large corpora of textual documents, hence words are semantically similar to the extent that they share contexts.

The gist of the technique is presented in [14], in which a novel content-based recommendation framework, called *enhanced Vector Space Model* (eVSM), is described. eVSM adopts a latent semantic representation of items in terms of *contexts*, i.e. a *term-context* matrix is adopted, instead of the classical term-document matrix adopted in the VSM. The advantage is that the *context* can be adapted to the specific granularity level of the representation required by the application: for example, given a word, its context could be either a single word it co-occurs with, a sentence, or the whole document.

The use of fine-grained representations of contexts calls for specific techniques for reducing the dimensionality of vectors. Besides the classical Latent Semantic Indexing, which suffers of scalability issues, more scalable techniques were investigated, such as Random Indexing [22], adopted in the eVSM model.

Random Indexing in an incremental method which allows to reduce a vector space by projecting the points into a randomly selected subspace of enough high dimensionality. The goal of using eVSM is to compare a vector space representation which adopts very few dimensions for representing items, with respect to a classical VSM.

As an alternative to VSM, we used the BM25 probabilistic model [19], one of the most dominant retrieval paradigm today. The ranking function for matching a query q (user profile) and an item I is:

$$R = \sum_{t \in q} \frac{n_t \cdot (\alpha + 1)}{n_t + \alpha \cdot (1 - \beta + \beta \frac{|I|}{avgdl})} \cdot idf(t) \qquad (1)$$

n_t is frequency of t in the item I, α and β are free parameters, $avgdl$ is the average item length, and $idf(t)$ is the IDF of feature t:

$$idf(t) = log \frac{N - df(t) + 0.5}{df(t) + 0.5} \qquad (2)$$

$df(t)$ is the number of items in which the feature t occurs, N is the cardinality of the collection.

For all the previous models we explicitly managed negative preferences of users by adopting the *vector negation* operator proposed in [23], based on the concept of *orthogonality* between vectors.

Several works generally rely on the Rocchio algorithm [21] to incrementally refine the user profiles by exploiting positive and negative feedback provided by users, even though the method needs an extensive tuning of parameters for being effective.

Negative relevance feedback is also discussed in [6], in which the idea of representing negation by subtracting an unwanted vector from a query emerged, even if nothing about *how much to subtract* is stated. Hence, vector negation is built on the idea of subtracting exactly the *right* amount to make the unwanted vector irrelevant to the results we obtain.

This removal operation is called *vector negation*, which is related to the concept of *orthogonality*, and it is proposed in [23].

3.2.2 Algorithms based on Classifiers

The recommendation process can be seen as a binary classification task, in which each item has to be classified as interesting or not with respect to the user preferences.

We learned classifiers using two algorithms, namely *Random Forests* (RF) [3] and *Logistic Regression* (LR).

RF is an ensemble learning method, combining different tree predictors built using different samples of the training data and random subsets of the data features. The class of an item is determined by the majority voting of the classes returned by the individual trees. The use of different samples of the data from the same distribution and of different sets of features for learning the individual trees prevent the overfitting.

LR is a supervised learning method for classification which builds a linear model based on a transformed target variable.

3.2.3 Graph-based Algorithms

We adopted *PageRank with Priors*, widely used to obtain an authority score for a node based on the network connectivity. Differently from PageRank, it is biased towards the preferences of a specific user, by adopting a non-uniform personalization vector to assign different weights to different nodes [13].

In order to run the PageRank, we need to represent data using a graph model. To this purpose, users and items in the dataset are represented as nodes of a graph, while links are represented by the positive users' feedback. The graph may be enriched in different ways, for example exploiting entities and relations coming from `DBpedia`: in this case the whole graph would contain nodes representing *users*, *items*, and *entities*, and edges representing items relevant to users, and relations between entities. This unified representation allows to take into account both *collaborative* and *content-based* features to produce recommendations.

In the classic PageRank, the prior probability assigned to each node is evenly distributed ($\frac{1}{N}$, where N is the number of nodes), while *PageRank with Priors* is biased towards some nodes, i.e. the preferences of a specific user (see Section 4.2).

4. EXPERIMENTAL EVALUATION

The goal of the experiments is to evaluate the contribution of diverse combinations of features, including those extracted from the Linked Open Data cloud, to the accuracy of different classes of recommendation algorithms.

The experiments that have been carried out try to answer to the following questions:

1. Which is the contribution of the Linked Open Data features to the accuracy of *top-N* recommendations algorithms, in presence of binary user feedback and cold-start situations?

2. Do the Linked Open Data-enabled approaches outperform existing state-of-the-art recommendation algorithms?

4.1 Dataset

The dataset used in the experiment is `DBbook`, coming from the recent Linked-Open Data-enabled Recommender Systems challenge. It contains user preferences retrieved from the Web in the book domain. Each book is mapped to the corresponding `DBpedia` URI, which can be used to extract features from different datasets in the Linked Open Data cloud.

The training set released for the *top-N* recommendation task contains 72,372 binary ratings provided by 6,181 users on 6,733 items. The dataset sparsity is 99.83%, and the distribution of ratings is reported in Table 2.

The test set contains user-item pairs to rank in order to produce a *top-5* item recommendation list for each user, to be evaluated using F1@5 accuracy measure.

4.2 Experimental setup

Each recommendation algorithm is fed by a diverse set of features.

Besides `TAGME` and `LOD` features, algorithms may also use `BASIC` features, i.e. number of positive, number of negative, and total number of feedbacks provided by users and provided on items, ratio between positive, negative and total number of feedbacks provided by users and provided on items and `CONTENT` features, obtained by processing book descriptions gathered from Wikipedia. A simple NLP pipeline removes stopwords, and applies stemming. For books not existing in Wikipedia, `DBpedia` abstracts were processed.

For all the methods, the 5 most popular items are assigned as liked to users with no positive ratings in the training set. Indeed, 5.37 is the average number of positive ratings for each user in the dataset (see Table 2).

Algorithms based on the Vector Space and Probabilistic Models.

Recommender systems relying on VSM and probabilistic framework index items using `CONTENT`, `TAGME` and `LOD` features, and use as *query* the *user profile* obtained by combining all the index terms occurring in the items liked by that user.

Items in the test set are ranked by computing the similarity with the user profile. For VSM and eVSM the cosine measure is adopted, while Equation 1 is used for the probabilistic model. According to the literature [20], parameters α and β are set to 1.6 and 0.75, respectively.

Algorithms based on Classifiers.

Classifiers based on Random Forests and Logistic Regression are trained with examples represented using `CONTENT`, `TAGME` and `LOD` features, and labeled with the binary ratings provided by users. The value of each feature is the number of times it occurs in each item, normalized in the [0,1] interval.

The LR classifier always includes `BASIC` features in the training examples, while these did not provide valuable results for RF.

The RF classifier used 1,500 trees to provide a good trade-off between accuracy and efficiency.

For Logistic Regression we adopted the implementation provided by Liblinear[6], while for Random Forests we adopted the implementation provided by the Weka library[7].

Top-N recommendations are produced by ranking items according to the probability of the class.

Graph-based Algorithms.

PageRank with Priors is performed (for each single user) using graphs with different sets of nodes. Initially, only users, items and links represented by the positive feedback are included; next, we enriched the graph with the 10 properties extracted from `DBpedia` (see Section 3.1). Then, we ran a second level expansion stage of the graph to retrieve the following additional resources:

1. internal wiki links of the new added nodes
2. more generic categories according to the hierarchy in `DBpedia`
3. resources of the same category
4. resources of the same genre
5. genres pertaining to the author of the book
6. resources written by the author
7. genres of the series the book belongs to.

This process adds thousands of nodes to the original graph. For this reason, we pruned the graph by removing nodes which are neither users nor books and having a total number of inlinks and outlinks less than 5. This graph eventually consisted of 340,000 nodes and 6 millions links.

The prior probabilities assigned to nodes depend on the users' preferences, and are assigned according to the following heuristics: 80% of the total weight is evenly distributed among items liked by users (0 assigned to disliked items), 20% is evenly distributed among the remaining nodes. We ran the algorithm with a damping factor set to 0.85.

We adopted the implementation of PageRank provided by the Jung library[8].

The PageRank computed for each node is used to rank items in the test set.

4.3 Results

Figure 4 shows the results of VSM and probabilistic models. The *paired t-test* is used for testing the significance.

The first interesting outcome is that the worst configurations are always obtained using `LOD` data alone, and the best ones always contain `TAGME` features. In detail, the best VSM configuration is obtained combining `TAGME` and `LOD` features, which is significantly better than using `LOD` features alone ($p < 0.0001$) and `CONTENT` alone ($p < 0.05$). The combination of `CONTENT` and `LOD` features outperforms the models using just `CONTENT` ($p < 0.01$) or just `LOD` features ($p < 0.001$).

[6] www.csie.ntu.edu.tw/~cjlin/liblinear/
[7] www.cs.waikato.ac.nz/ml/weka/
[8] jung.sourceforge.net

Property	Value
Statistics about users	
Avg. ratings provided by users	11.71 (5.37 positive, 6.34 negative)
# of users who provided only negative ratings	520 (8.41%)
# of users having a number of positive ratings below the avg.	3,804 (61.54%)
# of users having more negative than positive ratings	3,343 (54.09%)
Statistics about items	
Avg. ratings received by items	10.75 (4.93 positive, 5.82 negative)
# of items with no positive ratings	1,839 (27.31%)
# of items having a number of positive ratings below the avg.	6,447 (95.75%)
# of items having more negative than positive ratings	4,046 (60.09%)

Table 2: Distribution of ratings for the DBbook dataset.

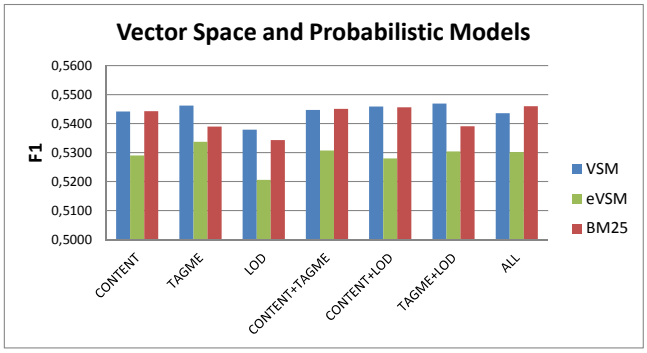


Figure 4: Results of the VSM and probabilistic models using different combinations of features.

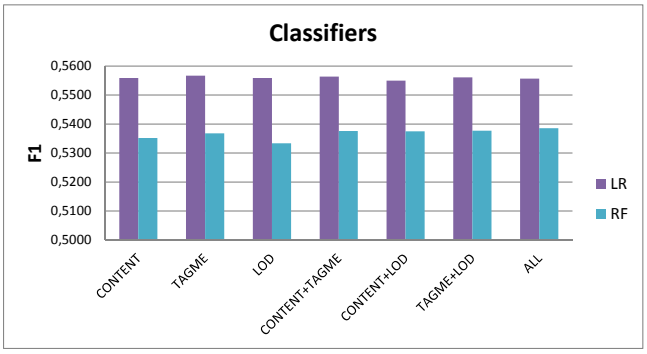


Figure 5: Results of the classifiers using different combinations of features.

The best configuration for eVSM adopts `TAGME` features alone, and is significantly better than all the configurations but the one combining `CONTENT` and `TAGME` features ($p = 0.13$). This could mean that the entity linking algorithm is able to select the most important features in the book descriptions, while `CONTENT` features introduce noise.

For BM25, the best configuration with `ALL` the features significantly outperforms all the others but the one combining `CONTENT` and `LOD` features ($p = 0.53$).

Surprisingly, there is no statistical difference between the best performing configuration for VSM and the best one for BM25.

A final remark is that eVSM performance is not comparable to the other methods, even though it is worth noting that it represents items using very low-dimensional vectors (dimension=500), compared to VSM, which uses vectors whose dimensionality is equal to the number of items (6,733).

Figure 5 presents the results obtained by the classifiers.

We note that Logistic Regression always outperforms Random Forests, and provides better results than the vector space and probabilistic models, regardless the set of adopted features.

The best result using Logistic Regression is obtained with `TAGME` features alone. This configuration significantly outperforms the one including `CONTENT` and `LOD` features ($p < 0.05$), while it is not different with respect to the other configurations. This is probably due to the high sparsity of the feature vector used to represent each training example (220,000 features).

Random Forests classifiers outperform eVSM, but they are worse than vector space and probabilistic models. The best result is obtained using `ALL` features. Since Random Forests classifiers are able to automatically perform feature selection, this was an unexpected result which deserves further investigations.

Finally, Figure 6 presents the results obtained by the PageRank with Priors algorithm.

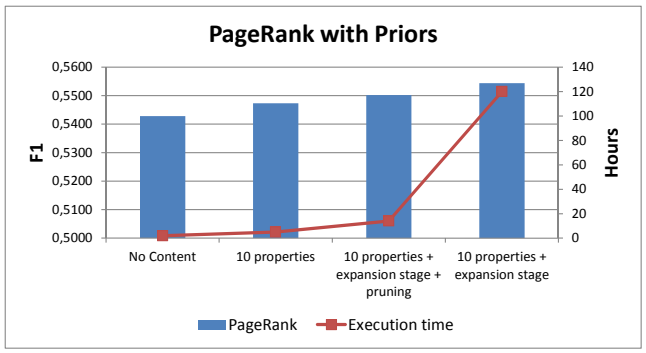


Figure 6: Results of the PageRank with Priors using different combinations of features.

When using PageRank with Priors, we observe the impact of the graph size on both the accuracy and execution time. Starting with a graph not including content information, we observe the worst performance and the lowest execution time (2 hours on an Intel i7 3Ghz 32Gb RAM - the algorithm

is performed for each user with different weights initially assigned to the nodes).

Enriching the graph with the 10 selected DBpedia properties leads to an improvement of accuracy ($p < 0.001$), and to a 5 hours execution time. Running the expansion stage and pruning of nodes as described in Section 4.2, the time needed to run the algorithm increases to 14 hours and produces a slight accuracy improvement ($p < 0.001$). Results using the graph with no pruning procedure are not different from the previous method ($p = 0.09$), but its time complexity is not acceptable. This call for a more efficient implementation of the algorithm.

To complete the empirical evaluation, we compare the best performing configuration of each algorithm in each class, with some state-of-the-art algorithms.

More specifically, we report the performance of *user-to-user* and *item-to-item collaborative filtering*, besides two non-personalized baselines based on *popularity* and *random* recommendations.

Furthermore, we report the results for two algorithms for *top-N* recommendations from implicit feedback: an extension of matrix factorization optimized for Bayesian Personalized Ranking (*BPRMF*) [9] and *SPRank* [16], able to exploit LInked Open Data knowledge bases to compute accurate recommendations.

Except for SPRank, we used the implementations available in MyMediaLite 3.10 [10], using the default parameters.

The analysis of results in Figure 7 unveils the difficulty of collaborative filtering algorithms to deal with the high sparsity of the dataset (99.83%), and with the high number of users who provided only negative preferences, or more negative than positive ratings. It is unexpected the better performance of *BPRMF* compared to *SPRank*, differently from previous results obtained on the MovieLens and Last.fm datasets [16]. It is also surprising the better performance of simple algorithms based on the vector space and probabilistic models with respect to matrix factorization.

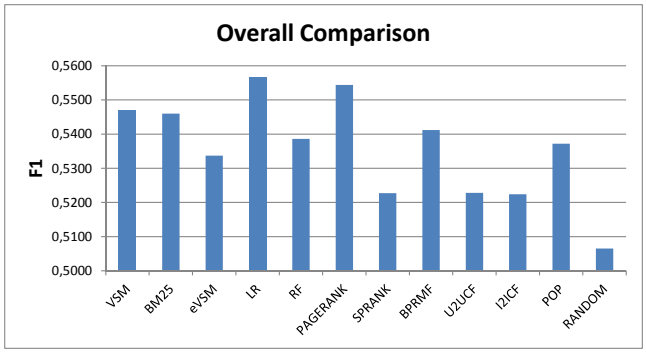


Figure 7: Comparison with other state-of-the-art approaches.

4.4 Discussion

The analysis of the previous results allows to conclude that TAGME and LOD features have the potential to improve the performance of several recommendation algorithms for computing *top-N* recommendations from binary user feedback.

However, in order to generalize our preliminary results, it is necessary to further investigate:

- the effect of different levels of sparsity on the recommendation accuracy: to this purpose, it is needed to assess the extent to which LOD features are able to improve the performance of recommendation algorithms for different levels of sparsity
- the accuracy on other datasets to generalize our conclusions: further experiments on different target domains are needed. Indeed, different item types, such as books, movies, news, songs have different characteristics which could lead to different results. Moreover, experiments on a much larger scale are needed
- the effect of the selection of domain-specific DBpedia properties to feed the recommendation algorithms: it is needed to assess the effect of the selection of specific sets of properties on the performance of the recommendation algorithms. Indeed, DBpedia contains a huge number of properties, and their selection could have a strong influence on the accuracy of the recommendation methods. Our preliminary experiments leverage 10 DBpedia properties which are both frequent and representative of the specific domain, but a subset of these properties, or a different set of features could lead to different results.

As future work, we will study the effect of enriching the graph-based representation with DBpedia nodes extracted from the Tagme entity linking algorithm.

Indeed, using entity linking to access DBpedia knowledge is innovative and avoids the need of explicitly finding URIs for items, a complex process which may hinder the use of the Linked Open Data. Hence, the use of entity linking algorithms represents a novel way to access the DBpedia knowledge through the analysis of the item descriptions, without exploiting any explicit mapping of items to URIs.

Furthermore, starting from the preliminary evaluation carried out in [1], we will thoroughly investigate the potential of using the wealth of relations of LOD features to produce not only accurate, but also *diversified* recommendation lists.

Acknowledgments

This work fulfils the research objectives of the project "VINCENTE - A Virtual collective INtelligenCe ENvironment to develop sustainable Technology Entrepreneurship ecosystems" (PON 02_00563_3470993) funded by the Italian Ministry of University and Research (MIUR).

5. REFERENCES

[1] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro. Aggregation strategies for linked open data-enabled recommender systems. In *European Semantic Web Conference (satellite Events)*, 2014.

[2] C. Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, 24(5):87–92, 2009.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] M. Chui, J. Manyika, and S. V. Kuiken. What executives should know about open data. *McKinsey Quarterly, January 2014*, 2014.

[5] T. Di Noia, R. Mirizzi, V. C. Ostuni, and D. Romito. Exploiting the web of data in model-based recommender systems. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Proceedings of the ACM Conference on Recommender Systems '12*, pages 253–256. ACM, 2012.

[6] M. D. Dunlop. The effect of accessing nonmatching documents on relevance feedback. *ACM Trans. Inf. Syst.*, 15:137–153, 1997.

[7] P. Ferragina and U. Scaiella. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *IEEE Software*, 29(1):70–75, 2012.

[8] E. Gabrilovich and S. Markovitch. Wikipedia-based semantic interpretation for natural language processing. *Journal of Artificial Intelligence Research (JAIR)*, 34:443–498, 2009.

[9] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, editors, *10th IEEE International Conference on Data Mining*, pages 176–185. IEEE Computer Society, 2010.

[10] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: a free recommender system library. In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the ACM Conference on Recommender Systems '11*, pages 305–308. ACM, 2011.

[11] Z. S. Harris. *Mathematical Structures of Language*. Interscience, New York„ 1968.

[12] O. Hassanzadeh and M. P. Consens. Linked movie data base. In C.Bizer, T. Heath, T. Berners-Lee, and K. Idehen, editors, *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009*, volume 538 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[13] T. H. Haveliwala. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

[14] C. Musto. Enhanced vector space models for content-based recommender systems. In *Proceedings of the ACM Conference on Recommneder Systems '10*, pages 361–364. ACM, 2010.

[15] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, and F. Narducci. Leveraging social media sources to generate personalized music playlists. In C. Huemer and P. Lops, editors, *E-Commerce and Web Technologies - 13th International Conference, EC-Web 2012*, volume 123 of *Lecture Notes in Business Information Processing*, pages 112–123. Springer, 2012.

[16] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, editors, *Proceedings of the ACM Conference on Recommender Systems '13*, pages 85–92. ACM, 2013.

[17] A. Passant. dbrec - Music Recommendations Using DBpedia. In *International Semantic Web Conference, Revised Papers*, volume 6497 of *LNCS*, pages 209–224. Springer, 2010.

[18] P. Ristoski, E. L. Mencia, and H. Paulheim. A hybrid multi-strategy recommender system using linked open data. In *European Semantic Web Conference (Satellite Events)*, 2014.

[19] S. E. Robertson, S. Walker, M. H. Beaulieu, A. Gull, and M. Lau. Okapi at TREC. In *Text REtrieval Conference*, pages 21–30, 1992.

[20] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[21] J. Rocchio. Relevance Feedback Information Retrieval. In Gerald Salton, editor, *The SMART retrieval system - experiments in automated document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[22] M. Sahlgren. An introduction to random indexing. In *Proc. of the Methods and Applications of Semantic Indexing Workshop at the 7th Int. Conf. on Terminology and Knowledge Engineering*, 2005.

[23] D. Widdows. Orthogonal negation in vector spaces for modelling word-meanings and document retrieval. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 136–143, 2003.

Content-Based Cross-Domain Recommendations Using Segmented Models

Shaghayegh Sahebi
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA
shs106@pitt.edu

Trevor Walker
LinkedIn
Mountain View, CA
twalker@linkedin.com

ABSTRACT

Cross-Domain Recommendation is a new field of study in the area of recommender systems. The goal of this type of recommender systems is to use information from other source domains to provide recommendations in target domains. In this work, we provide a generic framework for content-based cross-domain recommendations that can be used with various classifiers. In this framework, we propose an efficient method of feature augmentation to implement adaptation of domains. Instead of defining the notion of domain based on item descriptions, we introduce user-based domains. We define *meta-data features* as a set of features to characterize the fields that domains come from and introduce *indicator features* to segment users into different domains based on values of the *meta-data features*. We study an implementation of our framework based on logistic regression and perform experiments on a dataset from LinkedIn to perform job recommendations. Our results show promising performance in certain domains of the data.

1. INTRODUCTION

Recommender systems can help users to address the information overload problem by providing related items considering user's interests. So far, most of the recommender systems were focused on specific domains, recommending one type of item (such as books) to all categories of users. Recent research introduced cross-domain recommender systems that aim to take advantage of shared information among various domains [10]. In cross-domain recommendation, the goal is to use various source domain information to recommend items in target domains. Previous studies on collaborative filtering cross-domain recommender systems has shown an improvement of accuracy of recommendations, especially in the cold-start case [14]. Most of the work on cross-domain recommender systems and the definition of domains in them has been based on cross-domain collaborative filtering methods and ignored the domains that can be defined based on user specifications. Li [10] has categorized the domains in cross-domain recommendation into system, data, and temporal domains. These domains are related to, respectively, different datasets that a recommender system is built upon, various representation of user preferences (explicit or implicit), and various time points in which the data is gathered. Although this is a good classification of possible domains in recommender systems, it focuses on the type or domains that are defined based on items. In other words, usually the notion of domain is selected as a constant characteristic of items, systems, etc. For example, type of items (e.g. books, movies, etc.)[14], genre of items (e.g. for movies) [1], or indicators of various systems that the data is gathered from [6] are some of features that have been used as domain indicators. Joshi et. al. [5] have chosen domains based on meta-data features. These meta-data features can be selected from all unique subsets of features by experimenting (e.g. selecting the best performing features on a validation set) which is a time-consuming task. Choosing the proper domain indicator among features is still an open field of research. In this paper, we propose a framework for performing content-based cross-domain recommendation. We propose that in content-based and hybrid recommender systems, the domain notion can be extended to the type or domain of users. Here, the definition of domain can be determined based on the recommendation task and users' information, such as users' demographic data. For example, in a movie recommendation task, age of user can be an effective factor in deciding which movies match the best for her. As another example, in job recommendation, we expect the model parameters to be different for different job functions of users. For a designer, it is important to have matching skills with the job description, while for a network engineer, his certificates might have more importance.

We choose job recommender application in our experiment in this paper, although it is applicable to other recommender system domains. Having many different jobs listed online in various industries with different job descriptions, it is essential for people to find the job that best matches their abilities and specifications. Searching for the right job is a time-consuming task for a user. It needs spending a lot of effort on defining the criterion the user is looking for. Job recommender systems can address this problem by actively finding good job matches for the user, utilizing her profile information, search keywords, etc. Based on previous results in the job recommendation literature [8, 9] and our field experience, we believe that job recommendations can benefit from cross-domain information.

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

In this paper, our proposed framework can be utilized by various algorithms defined on any notion of domain from data attributes. Our work also differs from the existing literature in defining domains on the user profile side instead of item side. It can, of course, be used for domains defined on item-set features. We experiment with LinkedIn data for job recommendations. Our experiments lead to promising results for content-based cross-domain recommendations based on user job functions.

In Section 2, we briefly discuss related literature. In Section 3, we introduce our approach to content-based cross-domain recommendation. We present our dataset and experiment setup in Section 4 and we discuss the results in section 5.

2. RELATED WORK

2.1 Segmented Regression Model

Segmented regression or piece-wise regression [13] can be used as a classification method in which data features are partitioned into intervals using some breakpoints. In the final model, a separate model will fit each of the segments. This model is useful for approximating higher-degree models with multiple lower-degree models in smaller ranges. In this paper, we adopt this method to our problem of cross-domain recommendation.

2.2 Feature Augmentation in Domain Adaptation

Feature augmentation was introduced by Daumé [3] in domain-adaptation literature. In his paper, Daumé considers a source domain and a target domain separately. He augments each of these domains individually by copying the feature space three times: once copying all features for the general version, and once for each of the source and target versions. Eventually, the augmented source data will contain only general and source-specific versions and the augmented target data contains general and target-specific versions. After Daumé's paper, this method have been used in the domain-adaptation field, especially in Natural Language Processing (NLP)[2, 4, 5].

Our approach improves Daumé's model in the possibility of having multiple meta-data features for defining the domains (instead of having one dimension of source and target domains). In addition, each of the meta-data features can have multiple values and define multi-dimensional domains. Moreover, we can have separate sets of common (overlapping) and uncommon features in the main-effect and domain-specific models. Our model is extensible to incorporating cross-products of domain indicator features.

2.3 Job Recommendation

Despite of the importance of job recommender systems, there have not been many research on this subject. Rafter et al. [12] introduced CASPER, an intelligent online recruitment service. Keim [7] provided a multilayer framework to support the matching of individuals for recruitment processes. In [11] Hutterer used hybrid user profiling to enhance the job recommendation results. He incorporated explicit and implicit feedback of user in the user profile. Lee and Brusilovsky [8, 9] implemented and experimented with Proactive, which has multiple interfaces for various types of users. They showed that different users use various information resources to look for the perfect job.

3. OUR FRAMEWORK: SEGMENTED MODEL FOR CROSS-DOMAIN RECOMMENDATION

In cross-domain recommendation, we aim to build a model that can be general and flexible enough, to transfer the information in multiple related domains, and specific enough, to capture particular aspects of each individual domain. This means that we expect a trade off between the bias and the variance in our model. We want all models to be close to each other in particular dimensions (having less variance) and we want them to be biased towards each domain's specific distribution. For example, if we think of various job functions as different domains in job recommendation, we expect the user profile to have a good match to the job description in all domains (common feature of the domains). Also, we expect the skills feature to be more important for an artist than a university professor (domain-specific feature). If we consider one main model for all of the data present in various domains, we are going to have no variance in the model, but we will lose the bias we are looking for. On the other hand, if we treat each domain with a separate model, we will achieve the bias each domain is introducing, but we will have too much variance in the achieved models. In other words, we will lose the ability to transfer common information among different models. Our framework consists of two parts: the *main-effect model*, and *domain-specific models*. The main-effect model is to model the shared statistics among all domains. We have one domain-specific model per domain to capture the domain-specific characteristics. A general formulation of model can be seen in Equation 1.

$$\text{Final-Model} = \text{Main-Effect Model} + \sum_{i \in Domains} \text{Domain-Specific Model}_i \quad (1)$$

3.1 Cross-Domain Augmentation and Segmentation

As said before, we characterize the fields that the domains come from by some features called meta-data features. Each dataset has a set of features, such as user-related features, item-related features, and features that represent similarities between users and items, that we call them "base features". Meta-data features are a subset of base features, which specify aspects that we want to define the domains based on. Each domain is constructed based on the values of these meta-data features and their combinations. For example, if we want to recommend movies to users, base features are user features, such as age, education, language, etc, item features, such as movie genre, actors, director, etc, or the relationship between users and items, such as the similarity between each movie genre and genres that a user likes. We can choose some of these base features as meta-data features to define the notion of domain based on them. For example, we can choose the genre base feature as the meta-data feature. In this case, the defined domains will be action movies, drama movies, action-drama movies, etc. If we choose two meta-data features, the domains can be a combination of val-

ues for those meta-data movies. For example, if we choose genre of movie and age of user as meta-data features, the domains will be like: action-middle-age, drama-young, etc. In the case of user-based domains for job recommendation, we can choose some base features of users, such as job function, or job seniority of users, as meta-data features. For example, if we want to define the domains based on job function, people who have IT job function form one domain and people who have medical job function form another domain.

3.1.1 Augmentation

Each of the domain-specific models in Equation 1 works on one domain's data. As a result, we should split the dataset for each domain and provide each domain-specific model with the section of the dataset related to that domain. To address this splitting, we expand on the idea of segmented regression model. We propose to augment the feature space based on the domain definitions and copy each datapoint into the related domain's sub-space. The main space is then used for the *main-effect model* and each copy of the space is used for the related *domain-specific model*.

However, this augmentation has a problem: if we have k different domains (e.g. k different job functions), we need to partition the data space into 2^k separate segments (copies) to capture all of different settings of the dataset which takes too much space. Each one of these copies is for each subset of the possible combination of meta-data feature values (domains). For example, if we want to partition users based on the job function represented in their resume, and we have three different possible job functions (e.g. operations, education, and sales), we will have to partition the data into $2^3 = 8$ segments: people for whom the job function is in operations, the ones who have sales function, the ones who have education job function, the ones who have sales and operation functions, and etc. In addition to the space problem, segmenting the data into combinations of domains may lead to very sparse copies of the original dataset. Additionally, looking at all various combinations of the domains and their interactions might not be necessary for our purpose.

3.1.2 Indicator Features for Segmentation

To alleviate the problem indicated in Section 3.1.1, we expand on the idea of Segmented Regression Model and introduce *indicator features* for each domain. These features allow us to augment the feature space in a polynomial order, while being able to keep the main effect model and control the granularity of combinations among different domains. Suppose that each meta-data feature can take k different values and segment our data into k domains and suppose that we are choosing only one meta-data feature. For each of the k possible values in each of the domains, we define a binary *indicator feature*, representing if a data point falls into that specific domain or not. As a result, we will end up with k binary indicator features for the selected meta-data feature. Eventually, we will augment the feature space based on the indicator features in the following way: We keep the original feature space for the common features used in the main effect model. For each of the features falling into the domain-specific models, we augment the feature space by copying it k times for each of the meta-data feature values. Consequently, we have a polynomial expansion of space. Note that we do not consider combinations of meta-data feature values yet.

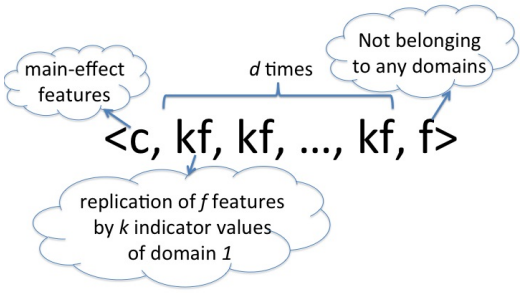


Figure 1: Augmented Feature Space with c Common Features in the Main Effect Model, f Features in Each of the d Domains, that Are Represented by Indicator Features with k Possible Values

Now, if we have d meta-data features to choose the domains from, each of which can take k values, and in each of the domain-specific models, f of the base-features exist, we should replicate this f dimensional space for $dk+1$ times. This number is polynomial in d (number of meta-data features) and k (number of values for each meta-data feature). While if we have not used the indicator features in segmented model, the f-dimensional feature space should have been replicated for d^k times. Considering having c common features for the *Main Effect Model*, we can represent the new feature space by Figure 1.

3.1.3 Challenges and Advantages

An advantage of this framework is its extensibility to higher order cross-products of values between and within domains. For example, if we want to consider the effect of interaction between two domains, we can extend the model to consider an indicator feature, representing cross-products of feature values in the domains. E.g. if we want to take into account the combination of every two feature values within each of the domains, we will end up with a space that has $O(c+f(dk+1)+f(dk^2+1))$ dimensions or is expanded for $dk+dk^2+2$ times. This gives us the ability to control the dimensionality of feature space while avoiding the sparsity in each of the segments.

Another challenge is choosing the features that should be in the main-effect model and the ones that should remain as domain-specific features. In other words, which features should be responsible for transferring the information among domains (controlling the variance) and which ones should provide the domain-specific bias? In our approach, the model can learn which features to use in the main effect model and which features to use in each of the domains using regularization. Since regularization imposes coefficient values to be as close to zero as possible, the less important coefficients of the model will have very small values and are removed from the model. While the model can choose between these sets of features, we can also initialize the main effect and

domain-specific features by the expert's domain knowledge.

Besides, in some of the cross-domain recommender problems, each domain has its own subset of a general feature set, which might differ in the size or type with other domains' feature sets. We expect our cross-domain solution to consider this problem and be extensible to domains with heterogeneous number and types of features.

3.2 Implementation Using Logistic Regression

Although the approach we presented here can be used in various classification algorithms, we used a straightforward classification algorithm to implement the model.

Suppose that f_{c_i} is the i^{th} common feature among the domains; $\mathcal{M}$ is the set of meta-data features; $\mathcal{V}_j$ is the set of values (domains) for the j^{th} meta-data feature; I_{ij} is the binary indicator feature for the domain i of meta-data feature j; f_{ijk} represents the k^{th} base feature specific to the i^{th} domain of meta-data feature j; and p is the probability of the model's outcome. Equation 2 shows the resulting segmented regression model with indicator features. As we can see, it has a simple representation that can be implemented easily for domain adaptation.

$$logit(p) = \sum_i w_{c_i} \times f_{c_i} + \sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{V}_j} I_{ij} \sum_k w_{f_{ijk}} \times f_{ijk} \quad (2)$$

Here w represent the weight (importance) of each feature in the model. In case we want to extend it to having two-way interaction effects of values of each meta-data feature (belonging to two domains), we will have:

$$\begin{aligned} logit(p) = &\sum_i w_{c_i} \times f_{c_i} + \sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{V}_j} I_{ij} \sum_k w_{f_{ijk}} \times f_{ijk} \\ &\sum_{j \in \mathcal{M}} \sum_{i \in \mathcal{V}_j} \sum_{l \in \mathcal{V}_j} I_{i,l,j} \sum_k w_{f_{i,l,j,k}} \times f_{i,l,j,k} \end{aligned} \quad (3)$$

In Equation 3 $I_{i,l,j}$ represents the binary indicator feature for the datapoint belonging to both i and l domains (or having both i and l values) of meta-data feature j. To decide which features should be in the common set of features and which should be in each of the domain-specific models, we use L_2 regularization.

4. EXPERIMENTAL SETUP

4.1 Data

The dataset we are using in this study is LinkedIn's job application data. It contains records of users and job features and a binary label indicating if the user has applied for the job or not. Some of base features are calculated similarities between the job and the user. For example, we use TF.IDF to calculate the similarity of job description with user's skills and store it as a feature in the user-job record. Some other features, from which we have chosen the meta-data features, are user-specific. For example, the past and current job functions of a user, past and current industries

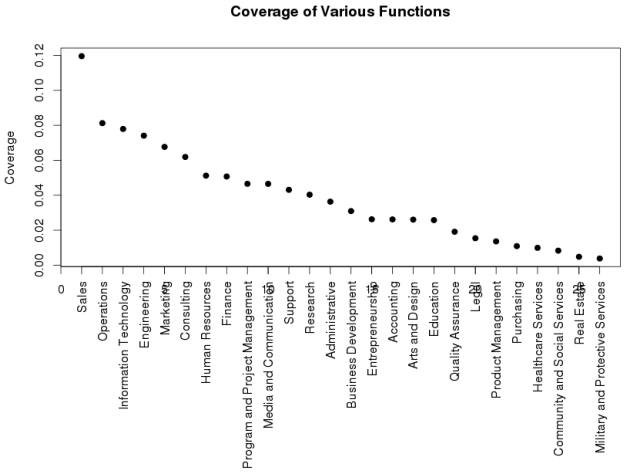


Figure 2: Coverage of User Current Job Functions in Offline Data

the user has worked at, or the geographical location of user. Meta-data features should have categorical values, so that we can extract binary indicator features from them. In case we want to use features with continuous values, we partition values into more than one category.

The offline dataset used in the following experiments consists of three million records of more than $150,000$ users. There is a one to ten ratio of positive job applications to negative job applications in the dataset. We use 100 user-job features as base features in the model.

We pick one meta-data feature (user's current job function) from user-specific features and split domains based on it. This feature is specifically related to what a user does in his/her job, e.g., a user can be an IT (job function) engineer in a bank. We choose this feature based on our experience that people working in various functions (e.g. arts and legal domain) have different requirements and definitions for a good job recommended to them.

Based on the LinkedIn data, current job functions of a user can have 26 different values. Each user can have multiple job functions at the same time. The distribution of user job functions is not uniform in the dataset: some functions are more covered in the dataset and some include less number of users. Figure 2 shows the coverage of current job functions in the data. As we can see in the picture, "Sales", "Operations", and "Information Technology" are among the most covered job functions in the data and "Community and Social Services", "Real Estate", and "Military and Protective Services" are the job functions with least coverage.

4.2 Implementation of Models for Job Recommendation

As we discussed in section 3, we can have two extremes of modeling users as our baselines: a) when there is only one main model for all of the users, and b) when there is a separate model for each segment of users and there is no shared

Figure 3: Three Experiment Settings with Different Granularities of Domain Definition

information among the models. In each of our studies, we compare our model to at least one of these two baselines.

To capture the effect of domain granularity on recommendation results, we experiment with three different settings for domains: segmenting on two domain indicator features versus all other domains (*two-vs-all*), segmenting on all indicator features of a domain (*all-indicators*), and segmenting on clusters of indicator features of a domain (*domain-clusters*). Figure 3 shows a graphical demonstration of user segmentation in each case for job functions.

For the *two-vs-all* model, we choose the two most covered values of the selected meta-data feature and define three indicator features based on that: the indicator feature that selects users with the most covered value, the indicator feature that selects users with the second most covered value, and the indicator feature for the rest of users. For example, for user's job function meta-data feature, we will have the following indicator features: I_S for users with "Sales" job function, I_O for users with "Operations" job function, and I_{Other} for all other users. The final model is presented in Equation 4. Here, fc_i shows the i^{th} common features among domains, f_{S_i}, f_{O_i}, and f_{Other_i} are features used in the "Sales", "Operations", and "Other" domains respectively, w_j is the weight for feature j, and p is the probability that the target user applies for the target recommended job. Note that since we have chosen only one meta-data feature, we do not need to present it in the model (e.g. $j \in M$ in Eq. 2). Also, note that by including I_{Other} as an indicator feature, we are capturing the effect of the interaction or cross-product of "Sales" and "Operations" domains.

$$logit(p) = \sum_i w_{c_i} \times fc_i + I_S \sum_i w_{f_{S_i}} \times f_{S_i} + I_O \sum_i w_{f_{O_i}} \times f_{O_i} + I_{Other} \sum_i w_{f_{Other_i}} \times f_{Other_i} \quad (4)$$

For the *all-indicators* model, we segment based on all values of the selected meta-data feature. If the meta-data feature can take k values, we will end up with k indicator features to segment all users into k different partitions. For user's job function meta-data feature we end up with 26 different indicator features. Our final model is shown in Eq. 5.

$$logit(p) = \sum_i w_{c_i} \times fc_i + \sum_{i \in 1..26} I_i \sum_j w_{f_{i,j}} \times f_{i,j} \quad (5)$$

Here, I_i shows the indicator feature for domain i (different values of job function); and $f_{i,j}$ represents the j^{th} base feature of domain i.

Figure 4: Clusters of User Job Function

For the last set of experiments (*domain-clusters*), we cluster the values of meta-data features into groups. Each group represents a cluster of domains. We use one indicator feature for each group. We use spectral clustering to group 26 different job functions into 8 clusters. The clusters are based on the user transition between job functions in the data. Figure 4 shows a tag-cloud representation of these clusters. Each color indicates one cluster. As we can see in the picture, functions like "Sales" and "Marketing" are clustered together and "Research" and "Education" functions fall in one cluster. We run the segmented model having one indicator per cluster. Suppose that $\mathcal{C}$ is the set of cluster indicators for a meta-data feature. The final model is shown in Equation 6.

$$logit(p) = \sum_i w_{c_i} \times fc_i + \sum_{i \in \mathcal{C}} I_i \sum_j w_{f_{i,j}} \times f_{i,j} \quad (6)$$

The final model we have in *domain-clusters* is similar to the *all-indicators* model, but domain indicators are representative of each cluster of job functions.

5. PERFORMANCE ANALYSIS

We experiment in the *two-vs-all* and *domain-clusters* settings for current job function of users as meta-data feature. We divide the data into 70% train and 30% test subsets.

We measure accuracy of the algorithms on the test set. To find the performance of algorithm in each domains of the data, we partition the test set into domains in the same way that we partitioned the training set and calculate the accuracy in each domains of the dataset. Our model is compared to at least one of the two baseline models: "one-for-all" and "independent" models. The "one-for-all" model only contains one model for all of the datapoints, ignoring the domain-specific models. The "independent" model trains a separate model for each of the domains independently. This model ignores the common information among the domains and treats them as independent from each other.

To dig deeper into the offline results, we look at the coefficient values obtained by the algorithm in each of the models.

Table 1: Accuracy of "two-vs-all" vs. "one-for-all" and "independent" models for job functions

Domain	One-for-All	Two-vs-All (Segmented)	Independent
Sales	96.28%	96.33%	95.01%
Operations	96.43%	96.49%	94.93%
Sales and Operations	96.54%	96.58%	NA
Other	96.44%	96.45%	96.44%

5.1 Two vs. All Model

As explained in Section 4, in this two-vs-all setting two most covered domains are compared to the rest of the domains. We compare our model with the two base models: "one-for-all" and "independent" models. The most covered job functions in the data are "Sales" (about 12% coverage) and "Operations" (about 8% coverage). The accuracy results for the "job function" meta-data feature are shown in Table 1. The "Sales and Operations" row represent the domain with users in both "Sales" and "Operations" domains and the "Other" row shows the users who are not in any of "Sales" or "Operations" domains. The first two rows show the users who are only in "Sales" or only in "Operations" domains respectively.

As we see in table 1, the segmented model has slightly higher accuracy than the base models. The difference is bigger for the "independent" base model, specially in the two most-covered domains. To understand how the models work differently, we look at the coefficients assigned to base variables of "two-vs-all" segmented model and "one-for-all" base model in Figure 5. In this picture, we can compare the coefficient values for these two models. The "two-vs-all" segmented model can have more than one coefficient for each variable: the variable might repeat in the main-effect part of the model or in each of the domain-specific parts of the model. To be able to compare the coefficient values, we use the average coefficient value of the main-effect and domain-specific parts of the "two-vs-all" segmented model. The red dots represent this average value and the bars around them represent the variance of these coefficients in the model. The blue dots are coefficient values in the "one-for-all" base model. As we can see in the picture, some of coefficients have a different value in the two models. For example, the similarity of user skills with job description has more importance in the "two-vs-all" segmented model. In addition, we can see that some of base variables existing in the "two-vs-all" segmented model, do not exist in the "one-for-all" base model. The reason is that these variables were removed automatically during the regularization process. For example, the similarity between user location and the location of the job is only present in the "two-vs-all" segmented model. Based on Figure 5, the "two-vs-all" model's coefficients have different variance in the main-effect and domain-specific models. Some of the coefficients vary more than the others. This can indicate that this model is able to capture the difference between different domains.

To understand if the "two-vs-all" segmented model is capturing the difference between each of the domains, we look at coefficients of variables in all four job function domains (Sales, Operations, Sales and Operations, and other) in Figure 6. The coefficient values are shown as stacked over each other in the picture. Since there are many base variables in the model and their names are not easily readable, we removed the names in figures of this section. As we can see, coefficient values for some of the variables are different for various domains. With a closer look we can find the differences in coefficient values. For example, the similarity of past positions of user to the job description is more important for the Sales domain than the Operations domain. The similarity of user skills to the job's required skills are more important for users in the Operations domain than Sales domain.

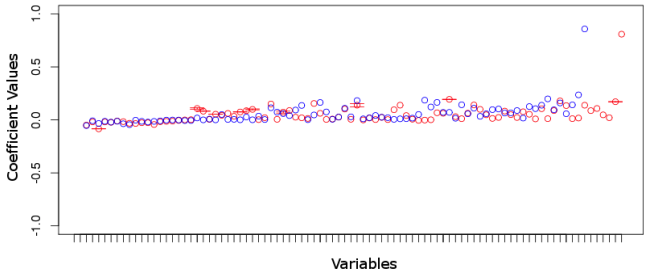


Figure 5: Coefficient Values for Two-vs-All Segmented Model (Red) Compared to the "One-for-All" Base Model (Blue)

Table 2: Accuracy of domain-clusters segmented model vs. "one-for-all" model for job functions

Domain	One-for-All	Domain Clusters (Segmented)
Cluster 1	96.57%	96.52%
Cluster 2	96.18%	96.26%
Cluster 3	96.62%	96.75%
Cluster 4	96.98%	97.09%
Cluster 5	97.58%	97.61%
Cluster 6	96.68%	96.85%
Cluster 7	96.56%	96.61%
Cluster 8	96.34%	96.36%

5.2 Domain Clusters Model

As explained in section 4, user job function meta-data features are grouped into 8 clusters. The accuracy results for the clusters in the "job function" meta-data feature is shown in Table 2. As we can see in this table, the accuracy of the models are very close to each other, for some clusters the baseline models work better than the domain-clusters segmented model and for others it is the reverse.

We compare coefficient values of the one-for-all baseline and domain-clusters model to have a more detailed insight of the results. Looking at the differences of coefficient values for each of the domains in the segmented model, we can understand how different features are more important for each of the domains. Figure 7 shows the coefficient values of domain-clusters model for the job function meta-data feature. As we can see in the picture, some of the coefficients are more important in some of the domains and less in others. For example, The similarity of user's previous searches to the job description is more important to users in cluster 2 (including sales, marketing, and similar job functions).

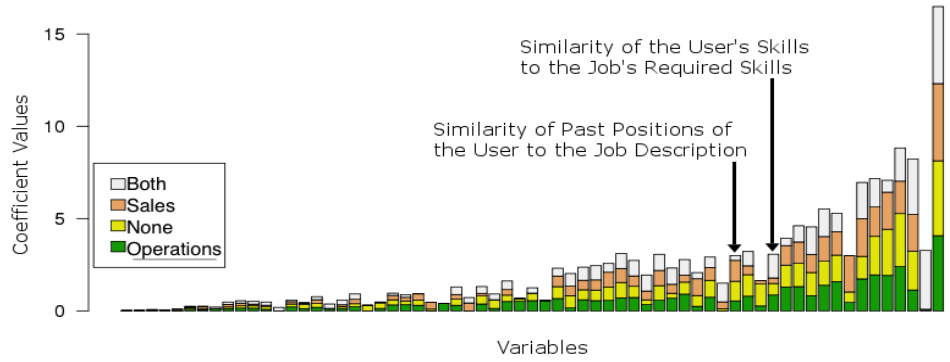


Figure 6: Coefficient Values for Different Domains in Two-vs-All Segmented Model

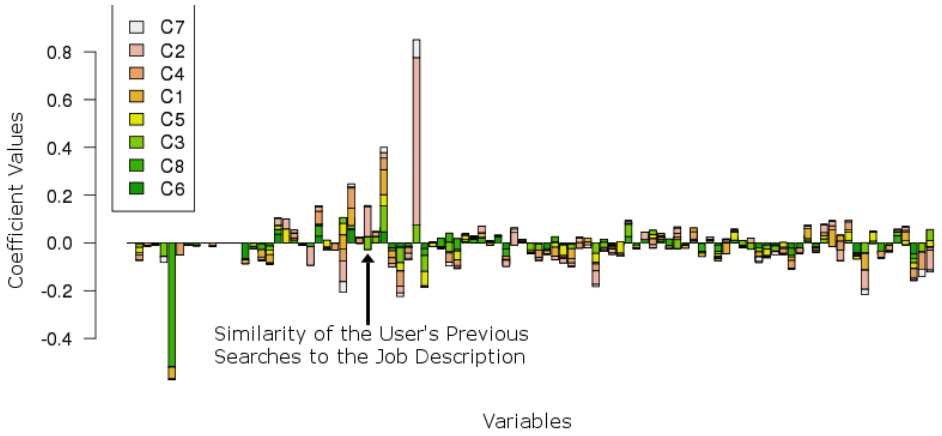


Figure 7: Coefficient Values for Domain Clusters Segmented Model for Job Functions

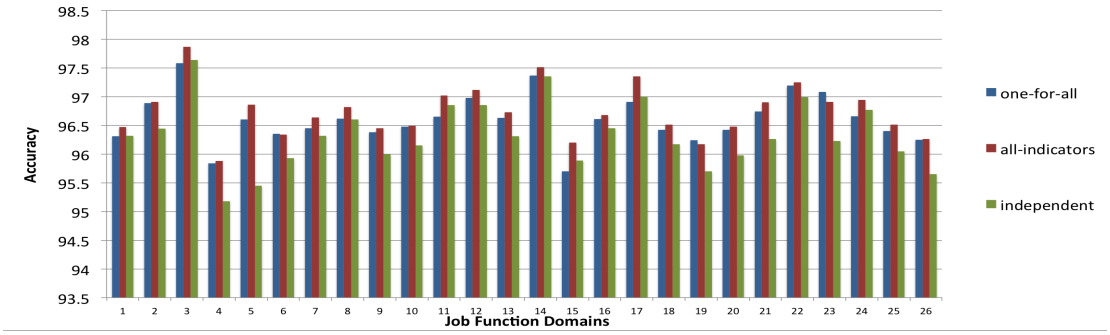


Figure 8: Accuracy of for All Indicators Segmented Model for Job Functions

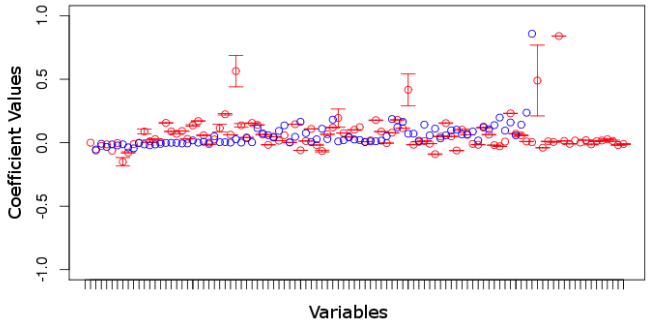


Figure 9: Coefficient Values for All-Indicators Segmented Model (Red) Compared to the "One-for-All" Base Model (Blue)

5.3 All Indicators Model

In this model, we pick all of the possible values for a meta-data feature as domain indicators: each indicator feature is representative of one of the values the meta-data feature can take. Since we choose job function as our meta-data feature, we will end up with 26 domains related to job functions, such as IT, sales, engineering, real estates, and marketing. We can see accuracy results of the all-indicators segmented model and the two one-for-all and independent baseline models in Figure 8. As we can see here, the all-indicators model is usually slightly better than the two other models. In some of the domains (such as Product Management (number 19) and Real Estate (number 23) domains) the one-for-all model has more accuracy than all-indicators model.

Comparing coefficients of one-for-all and all-indicators models in Figure 9, we can see that some of the base features have a very different coefficients in the model. Also, some of base features have a large variance in different domains of the all-indicators model. There are some base features that are removed from the one-for-all model by regularization, while they still play a role in the all-indicators model.

6. CONCLUSION AND FUTURE WORK

In this paper, we propose a framework for content-based cross-domain recommender systems. This framework is flexible enough to be implemented with various classifiers. The model in this framework can transfer common information among different domains while keeping them distinct. We define user-based domains based on users' meta-data features and implement our framework using logistic regression. The regularization in the model allows us to pick important features of each of the domains automatically, while keeping it flexible to accept expert knowledge in choosing the features. We experiment on job recommendations for LinkedIn users. Our results indicate slight improvement in recommendation accuracy in the offline setting. Furthermore, the experimental results are promising: i) different features have different coefficient values in each of the domains; and ii) coefficients are different in the cross-domain model compared to the one-for-all base model. As a result, we are hopeful that this model can be a good fit to our problem in the online experiments (A/B testing). We expect several directions for future work: implementing the framework based on various classifier algorithms, expansion of experiments of the model using different meta-data features, and experimenting on interaction of various possible domains. Automatic selection of meta-data features is another interesting direction of research.

7. REFERENCES

[1] S. Berkovsky, T. Kuflik, and F. Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, 18(3):245–286, 2008.

[2] J. H. Clark, A. Lavie, and C. Dyer. One system, many domains: Open-domain statistical machine translation via feature augmentation. In *Proceedings of the Tenth Biennial Conference of the Association for Machine Translation in the Americas*, 2012.

[3] H. Daumé III. Frustratingly easy domain adaptation. In *ACL*, volume 1785, page 1787, 2007.

[4] L. Duan, D. Xu, and I. Tsang. Learning with augmented features for heterogeneous domain adaptation. *arXiv preprint arXiv:1206.4660*, 2012.

[5] M. Joshi, M. Dredze, W. W. Cohen, and C. P. Rosé. What's in a domain? multi-domain learning for multi-attribute data. In *Proceedings of NAACL-HLT*, pages 685–690, 2013.

[6] M. Kaminskas and F. Ricci. Location-adapted music recommendation using tags. In *User Modeling, Adaption and Personalization*, pages 183–194. Springer, 2011.

[7] T. Keim. Extending the applicability of recommender systems: A multilayer framework for matching human resources. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, pages 169–169, 2007.

[8] D. Lee and P. Brusilovsky. Fighting information overflow with personalized comprehensive information access: A proactive job recommender. In *Autonomic and Autonomous Systems, 2007. ICAS07. Third International Conference on*, pages 21–21, 2007.

[9] D. Lee and P. Brusilovsky. Proactive: Comprehensive access to job information. *Journal of Information Processing Systems*, 8(4):721–738, December 2012.

[10] B. Li. Cross-domain collaborative filtering: A brief survey. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pages 1085–1086. IEEE, 2011.

[11] H. M. *Enhancing a Job Recommender with Implicit User Feedback*. PhD thesis, FakultŁt f§r Informatik der Technischen UniversitŁt Wien, 2011.

[12] R. Rafter, K. Bradley, and B. Smyth. Personalized retrieval for online recruitment services. In *In Proceedings of the 22nd Annual Colloquium on Information Retrieval*, 2000.

[13] H. Ritzema. Frequency and regression analysis (chapter 6). *Drainage principles and applications*, 16:175–224, 1994.

[14] S. Sahebi and P. Brusilovsky. Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In *User Modeling, Adaptation, and Personalization*, pages 289–295. Springer, 2013.

Preference Mapping for Automated Recommendation of Product Attributes for Designing Marketing Content

Moumita Sinha and Rishiraj Saha Roy
Adobe Research Labs, India
Bangalore, India - 560029.
{mousinha, rroy}@adobe.com

ABSTRACT

Identification of relevant product attributes is critical to the success of any marketing campaign. This task can be conceptualized as an attribute recommendation problem based on the *product's content or features*, where the goal of a solution would be to automatically recommend relevant features to the marketer for highlighting in a campaign. In this research, we try to solve this problem by using *preference mapping*, a powerful technique for associating feature preferences with users. We perform preference mapping with *sentiment scores* associated with product attributes mined from user reviews on the Web. As a result of this process, we are able to visualize a set of compared products and the appropriateness of the attributes on *the same two-dimensional space*, enabling us to easily recommend important features to a marketer. Finally, we show that expert recommendations or ratings for product features do not necessarily correlate with preference maps based on user sentiments.

Categories and Subject Descriptors

Information retrieval [**Retrieval tasks and goals**]: Recommender systems

General Terms

Algorithms, Experimentation, Human factors

Keywords

Preference Mapping, Sentiment Scores, Product Attributes

1. INTRODUCTION

Motivation. Product manufacturers are always faced with the dilemma of identifying which attribute(s) of their products they should highlight in their targeted marketing campaigns. For example, a digital camera has several defining aspects like power of zoom, size of display and image size in megapixels. A release of a new camera model by a manufacturer like Nikon will usually be followed by a marketing campaign to potential customers that will try to highlight certain aspects or attributes of the model. This attribute recommendation problem is critical to the success of the campaign. Focusing on features that do not appeal to users can result in a loss of large amount of ad spend and potential losses in product revenue for a manufacturer. In this paper, we address this challenge by proposing a principled technique called *preference mapping* [6], used in a novel way to automate the process of product attribute recommendation.

Related research. Alpert [1] presents one of the relatively early works emphasizing the importance of identifying relevant product attributes, and compares the effectiveness of direct and indirect questioning techniques. Cropper et al. [3] finds that a linear hedonic price function performs as well as a linear logit model in estimating consumer preferences for product attributes. But their analysis is based on simulations and does not draw connections between preferred attributes and campaign design. Zhang and Liu [12] try to identify product features that are associated with user sentiment by analyzing the contextual text associated with the mention of the product feature. While it could be meaningful to further scrutinize such attributes while designing product campaigns, the authors do not propose any method towards that end. Lehdonvirta [10] aims to discover product attributes that are likely to drive purchase decisions for virtual goods like online games and engaging activities on social media. However, the analysis presented by the author is purely from a sociological perspective and the author does not provide an algorithm for automating the above process. Recommendation algorithms similar to collaborative filtering have been used for designing campaigns, but they rely heavily on large amounts of existing customer preference data available with the advertiser [11]. On a related note, they are also known to have limitations such as data sparsity and model scalability, which leads to poor recommendations [2]. We provide a method for associating products with their marketable attributes that relate to each other based on publicly available sources. Such data sources may become accessible much before the advertiser receives direct information about customers' preferences based on product view or product purchase data. *Preference mapping* is an approach to identify customer preferences based on users' surveys of product attributes. Individual user differences are not averaged, but are directly incorporated into the mapping model and play vital roles in the preference fitting process [5]. As of date, the technique has only been used for understanding user preferences for diverse food items like lamb sausages [7], lager beer [6] and vanilla ice cream [4]. We believe that this

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

method has a far greater potential and can be readily extended to unexplored application areas.

Approach. In this research, after specifying our product and attribute set, we acquire sentiment scores of *user reviews* that mention attributes for the products in our set. Following this, we associate *user sentiments with the attributes* mentioned in the reviews (instead of the product as a whole) and average them over reviewers who have written reviews concerning the attributes. We perform *preference mapping* on this processed dataset involving products, attributes and average sentiment scores and generate a *biplot visualization* that can be used for attribute recommendation. Finally, we compare our recommendations with *expert opinion* and show that there is no perfect correlation with what experts believe to be good features and what consumers like in a marketed product.

Organization. The rest of this paper is organized as follows. In Sec. 2, we describe our method of applying preference mapping to this situation. Next, we describe our data in Sec. 3 followed by experimental results and discussion in Sec. 4. Finally, we summarize our contribution and provide directions for future work in Sec. 5.

2. METHOD

We analyze a set of products p and a set of product attributes k. Customers who have bought these products often go to the product or retailer website to provide feedback about the product in the form of textual reviews. Most of these reviews generally contain mentions of product attributes. Further, positive or negative sentiments usually accompany the above mentions of the attributes. In our approach, we collect reviews where each sentence talks about only one attribute. Appropriate anaphora resolution is performed for review sentences when the attribute name is not directly mentioned [8]. Each sentence in each review is then assigned a sentiment score. Since each sentence mentions exactly one attribute, the sentiment score associated with the sentence is assumed to be the score associated with the attribute. Note that the effectiveness of our algorithm is not affected by the scale or range of this sentiment scoring. Next, the scores are averaged over the reviewers for each attribute for each product.

A preference mapping is then performed with the reviewer-averaged scores of each of the various attributes for the different products. We now explain how this is performed. As the first step, sentiment scores for all the product attributes are scaled to the same range so that variances are comparable across attributes of each product. Consider $X = (X_1, X_2, \ldots, X_p)^T$ as the matrix of the reviewer-averaged scores for the p products (say, different camera models) and the k attributes (like battery life, size of display and shutter delay). Thus each X_i is a vector with its elements as X_{ij}, which is the reviewer-averaged sentiment score for attribute j of product i. The principal component (PC) transformation of the feature vector X is the linear transformation $Y = \Gamma^T(X - \mu)$ where $\mu = E(X)$ and $\Sigma = Var(X) = \Gamma\Delta\Gamma'$. The transformation is such that $Var(Y)$ is maximized and the following holds:

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$$

where, $Var(Y_j) = \lambda_j$, $j = 1, 2, \ldots, p$, $E(Y_j) = 0$ and $Cov(Y_j, Y_i) = 0$ when $i \neq j$.

Functions $Var(\cdot)$, $E(\cdot)$ and $Cov(\cdot)$ refer to the variance, expectation, and covariance functions, respectively, and the λ_j's represent the eigenvalues of the matrix X. These eigenvalues have the corresponding eigenvectors as $\gamma_1, \gamma_2, \ldots, \gamma_p$ (the number of eigenvectors is equal to the rank of the matrix X). Then the i^{th} PC for each product is the weighted sum of the scores of the product across the attributes, the weights being obtained from the i^{th} eigenvector. A biplot graph can be plotted for PC1 and PC2 with the weighted scores of each of the products and the eigenvector values for each attribute. The resultant graph provides an easily interpretable visualization that shows how products compare among each other based on customer reviews and the relative proximity of each attribute to their respective products with respect to associated positive user sentiment. Based on this multivariate visualization, marketing contents can be designed, highlighting favorable attributes for products. A schematic of the steps a marketer will undergo to utilize statistical analysis of social reviews to design product specific marketing campaigns is shown in Figure 1. Relevant steps have been explained in this section. Specific details about our dataset and experimental setup will be provided in the next section.

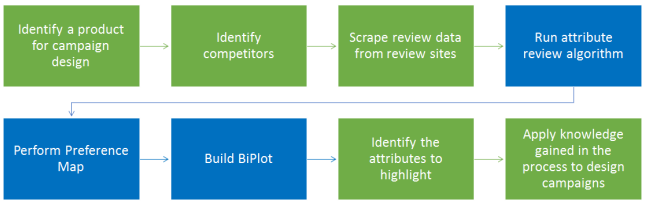


Figure 1: A schematic of the steps in our use case: The steps in green are part of the workflow, while those in blue are part of the proposed algorithm.

3. DATASET

We test our approach on a dataset consisting of 1309 reviews related to four digital camera models (Canon G3, Canon Powershot SD500, Canon S100, and Nikon Coolpix 4300), having a total of 13 distinct attributes. These attributes (or features) that we analyzed are: flash, zoom, battery, auto (quality of automatic mode), photo quality, view (quality of view through the viewfinder), delay (delay between photos), look, start (startup speed), color, night (quality of night photos), lens and resolution. The reviews are pre-processed to identify mentions of camera attributes within their texts. The 13 attributes are mentioned a total of 583 times in the product reviews that we collected.

Expert ratings. It is an interesting exercise to compare our attribute recommendation system to expert opinion. To this end, we went through popular digital camera review sites `dcresource`[1] and `imaging-resource`[2] for extracting expert ratings on the thirteen attributes for our

[1] `http://www.dcresource.com`, Accessed 11 July '14.
[2] `http://www.imaging-resource.com`, Accessed 11 July '14.

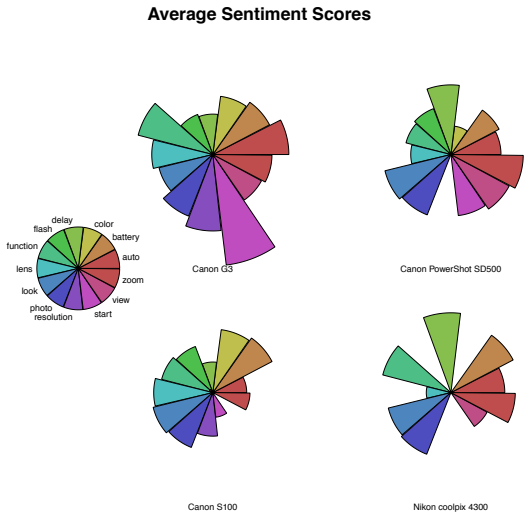


Figure 2: (Color online) Reviewer-averaged sentiment scores of attributes for our camera models.

four camera models. Since none of the popular camera review sites provide direct numeric ratings for attributes, we mapped expert opinion to a score of 1 or 2 depending upon the comments provided. For example, comments containing words like *exceptional*, *excellent* and *good* about an attribute were mapped to two, and *weak* and *worst* were assumed to be a one rating. The data that we collected has been made publicly available at `http://goo.gl/v8BGj4`.

4. EXPERIMENTS AND RESULTS

We assign a sentiment score to each sentence in each review in our dataset with the Alchemy API[3] and transfer the score to the attribute mentioned in the sentence. The higher the magnitude of the score, the stronger is the strength of the associated sentiment. Following this, the positive and negative sentiment scores of all the 52 ($= 13 \times 4$) camera-attribute pairs were averaged together over all the reviewers who mentioned the pair in his/her reviews, the neutral sentiments contributing zero to the sum. The missing observations are assumed to be neutral sentiments and hence the scores in such cases are assumed to be zero. These average sentiments for each camera over all attributes are shown in a radial chart in Figure 2. As a specific example, the battery of the Canon S100 was mentioned in 13 reviews, with seven, one, and five review(s) showing positive, negative and neutral scores respectively. While the numbers of positive and negative mentions seem comparable, the average positive and negative sentiment scores were found to be 1.3461 and 0.3569 respectively, indicating that the strength of the negative sentiment was not as strong as the positive sentiment. In our experiments, the two values were averaged to obtain 0.8515.

We now have a matrix with four rows (corresponding to each camera model) and thirteen columns (corresponding to each model attribute). The cells of this matrix are the reviewer-averaged sentiment scores associated with each camera and attribute pair. A principal component analysis (PCA) is then performed on this matrix of camera-attribute pairs. The PC1 and PC2 for this example, cumulatively explain 85% of the variability in the data. We then produce the biplot of the weighted scores of the products and the eigenvectors of each of the attributes, as shown in Figure 3.

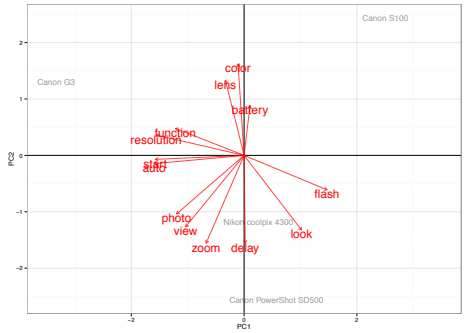


Figure 3: (Color online) A biplot of the weighted scores of products and eigenvector attributes. Attributes are in red and product names are in gray.

This graph provides a lot of information for design of marketing campaigns. First, in the graph, two attributes (in red) that are pointing towards the same direction, are attributes that tend to be highly positively correlated. A product that is in the same direction as an attribute, has a high value for this attribute. Thus, from the graph, we can conclude that attributes, which are closer and in the same direction as a product, are the ones that should be recommended for highlighting in marketing content for that particular model. For example, Canon G3 and Canon S100 received high sentiment scores on attributes like lens and color, while Nikon Coolpix 4300 and Canon PowerShot SD500 received high positive sentiments on low shutter delay and zoom quality. Thus, for example, lens and color should be recommended for designing marketing content in the campaign for Canon G3, rather than the zoom.

Second, this methodology also helps to contrast competing products simultaneously and provides *competitive intelligence* to the marketer. Thus, based on the given set of consumers' reviews, one can deduce that Nikon Coolpix 4300 and Canon PowerShot SD500 are similar with respect to the attributes studied, as compared to Canon G3 and Canon S100. For example, if Nikon Coolpix 4300 and Canon Powershot SD500 are competing products, then it is meaninful to recommend only discriminatory features that add value to a particular product for its campaign. It is more sensible to recommend flash for Nikon Coolpix 4300 (more closer to the model than Canon 500) than the zoom, which is approximately equidistant from the both the products.

Analysis of expert opinion. From the data collected on expert comments (Sec. 3), we find that many of the discussed attributes are rated as 2, which implies that these attributes are "excellent" or "good" (Table 1). We assume that high expert score is analogous to high positive sentiment.

Table 2 shows the Kendall-Tau rank correlation coefficients between the preference mapping technique and the plain average sentiment scores (which is the unweighted sum of the attributes as opposed to the weighted sum for each camera). For three cameras we have statistically significant (at 0.05 level) correlation between the methods and a moder-

[3] `http://www.alchemyapi.com`

Table 1: Proportion of Attributes Rated as Excellent/Good and Poor.

Camera	Excellent/Good	Poor
Canon G3	0.385	0.538
Canon S100	0.615	0.231
Canon Powershot SD500	0.385	0.538
Nikon Coolpix 4300	0.615	0.385

Expert ratings were not available for all the attributes. So the sum of the values in a row may not add up to one

Table 2: Correlation between ranks of the attributes based on average sentiment scores and preference mapping scores.

Camera	Kendall-Tau	p-Value
Canon G3	0.564	0.007
Canon S100	0.615	0.003
Canon Powershot SD500	0.641	0.002
Nikon Coolpix 4300	0.294	0.172

ate correlation for Nikon Coolpix 4300. This shows that our method has high correlation with the intuitive understanding of the importance of the attributes and helps in further refinement. We could not observe any direct relation between the predictions based on the preference mapping and the attributes highly rated by experts.

5. CONCLUSIONS AND FUTURE WORK

The *preference mapping* technique, as described by us in this research, recommends *potentially "valuable" attributes* of products to marketers for *highlighting in a marketing campaign*. Our method provides the marketer the ability to design marketing content that can potentially increase response rates. We have used *sentiment scores* for product attributes, extracted from *review texts* to identify product features to be highlighted in campaigns. By focusing on attributes that are *known to have received positive sentiments of customers*, the risk in the campaign is minimized. Moreover, the comparison with the experts' comments suggests that sometimes, what customers value more about a product may be different from attributes that experts consider of high quality. So, designing marketing content taking into account what a large section of consumers show positive sentiments towards may help in engaging more effectively with a larger section of the consumers. The sentiment score in our research is a continuous variable and PCA has been used to identify appropriate attributes that have high scores. If some or all the scores are categorical in nature, multi-factor analysis [9] is preferable over PCA. The proposed technology does not require large amounts of customer preference data to be available internally with the advertiser (for example, customers who have viewed the same product or customers who have bought the same product), from their own sales and browsing patterns. Rather, we use reviews that *directly reflect customer preferences*. The reviews can be collected from any external source with consumers' opinion. The other major strength of our approach is that it is more likely to be positively viewed by the future customer. Such an approach enables having an informed conversation with the potential customer and is likely to improve customer satisfaction.

As future work, we would like to cluster products using attribute sentiment scores as features and observe the correlation of the clustering output to the representation produced by our preference mapping technique. Also, the quality of the reviews can be improved by choosing relevant users by mapping them to specific customer segments. This can lead to better insights on the data and finer levels of control in the design of marketing content.

Acknowledgements

We thank Ritwik Sinha from Adobe Research Labs India for valuable inputs at various stages of this work.

6. REFERENCES

[1] M. I. Alpert. Identification of determinant attributes: A comparison of methods. *Journal of Marketing Research*, pages 184–191, 1971.

[2] Y. H. Cho, J. K. Kim, and S. H. Kim. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329–342, 2002.

[3] M. L. Cropper, L. Deck, N. Kishor, and K. E. McConnell. Valuing product attributes using single market data: a comparison of hedonic and discrete choice approaches. *The Review of economics and Statistics*, pages 225–232, 1993.

[4] L. Dooley, Y. S. Lee, and J. F. Meullenet. The application of check-all-that-apply (CATA) consumer profiling to preference mapping of vanilla ice cream and its comparison to classical external preference mapping. *Food quality and preference*, 21(4):394–401, 2010.

[5] K. Greenhoff and H. MacFie. Preference mapping in practice. In H. MacFie and D. Thomson, editors, *Measurement of Food Preferences*, pages 137–166. Springer US, 1994.

[6] J. X. Guinard, B. Uotani, and P. Schlich. Internal and external mapping of preferences for commercial lager beers: comparison of hedonic ratings by consumers blind versus with knowledge of brand and price. *Food Quality and Preference*, 12(4):243–255, 2001.

[7] H. Helgesen, R. Solheim, and T. NÃęs. Consumer preference mapping of dry fermented lamb sausages. *Food Quality and Preference*, 8(2):97–109, 1997.

[8] S. Lappin and H. J. Leass. An algorithm for pronominal anaphora resolution. *Comput. Linguist.*, 20(4):535–561, Dec. 1994.

[9] S. Lê, J. Josse, F. Husson, et al. Factominer: an r package for multivariate analysis. *Journal of statistical software*, 25(1):1–18, 2008.

[10] V. Lehdonvirta. Virtual item sales as a revenue model: identifying attributes that drive purchase decisions. *Electronic Commerce Research*, pages 97–113, 2009.

[11] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[12] L. Zhang and B. Liu. Identifying noun product features that imply opinions. In *HLT '11*, pages 575–580, 2011.