Recommender System Based on Random Walks and Text Retrieval Approaches

Max Chevalier, Taoufiq Dkaki, Damien Dudognon, Josiane Mothe

Institut de Recherche en Informatique de Toulouse
Université de Toulouse, France
FirstName.Surname@irit.fr

Abstract. This paper presents the approaches IRIT developed for the VLNetChallenge regarding recommender systems in the context of video lectures. The first task aims at recommending newly acquired lectures after viewing an "old" lecture. We use random walk algorithms based on a graph composed of author, category, event, and lecture nodes and associated relationships. The second task aims at recommending 10 lectures from three lectures extracted from a sequence of lectures. We use the categories associated to lectures in addition to the lecture pairs (lectures viewed in a same session).

1 Introduction

IRIT participated to the two tasks of the VLNetChallenge.

Regarding the cold start task, which aims at recommending newly acquired lectures after viewing an "old" lecture, we first built a graph from the data collection. Graph nodes are lectures and associated meta-data (authors, events and categories). Graph links correspond to the various types of relationships (links between lectures, between events and between categories as well as cross-type links). Relationships were weighted differently according to the nature of the links. The resulting graph was used in random walk algorithms. The best results on the test collection have been obtained when the graph weights are significantly more important for the lecture pairs and the authors-lectures relationships than for the remaining relationships.

Regarding the pooling lecture task, we first considered the lecture contents only; this method showed poor results. We then consider the lecture categories. Since many lectures are not linked to categories, we first defined a way to deal with this problem. Then, we use the frequency of lecture visits, lecture pairs and the categories they belong to.

2 Data preparation

To begin with, we uploaded the CSV data provided to the participants in a PostgreSQL database [15]. For each lecture, we extracted the categories, events and authors associated with it.

We also indexed lectures using the Solr search engine [14]. We used as content the *name*, *description* and *slide_titles* fields of each lecture. Indexing is based on a "bag of words" approach. To build the Solr index, the stopwords were not removed and we did not use any stemming heuristic similar to the Porter Stemmer [8]. Avoiding pre-processing steps allows us to store all the lectures in the same index, regardless of their language. The retrieval model used in Solr combines Boolean Model [7] and Vector Space Model [11]. The documents are first selected by Boolean Model and then are scored using Vector Space Model. The scoring function implemented in Solr is derived from the VSM score, based on the Cosine similarity [10].

Solr was used in the two tasks. In the cold start task, Solr was used to build two matrices that reflect the lecture similarities based on content. For the first one, we used MoreLikeThis from Solr to calculate the similarities between each lecture pairs. For a given document, the MoreLikeThis module generates a query based on the representative terms of the document. These terms are selected depending on several parameters which are: their length, their frequency in the document and their frequency in the overall collection. The second matrix was built differently: for each lecture, we calculate its similarities with all the other lectures, considering its title as a query; lectures were favored if recent.

In the pooled sequences task, Solr was used to retrieve the most similar lectures from a given lecture.

3 Cold start task

The cold start task aims at predicting “which of the newly acquired lectures at the site should be recommended after viewing some of the 'older' lectures” [12].

To complete this task, we first built a graph from the data in which nodes and relationships are typed. In addition we weighted some of the relationships. Then we applied two random-walk models to compute document similarities and predict which new lectures should be recommended. Section 3.1 explains the way the graph is built and section 3.2 explains the way it is used.

3.1 Graph generation

From the data, we built a graph $G=\{N,R\}$ where N is a set of nodes and R a set of relationships between nodes.

The set of nodes N is defined as: $N=\{A, C, E, L\}$ where:

- A is a set of author nodes,
- C a set of category nodes,
- E a set of event nodes, and
- L is a set of lecture nodes.

The set of relationships R is defined as:

$R=\{CR, ER_{ei,ej}, AR_{li,aj}, DR_{li,cj}, TR_{li,ej}, LR_{li,lj}\}$ where:

- CR is a relationship defined between two categories.

$CR(c_i,c_j)$ $= 1$ if categories c_i and c_j have a hierarchical relationship in the database;

$= 0$ otherwise.

- ER is a relationship between two events. As for CR, $ER(e_i,e_j)$ is either 0 or 1, based on the hierarchical relationship defined between events e_i and e_j using parent_id attribute.

- AR is a relationship between a lecture and an author.

- DR is a relationship between a lecture and a category.

- PR is a relationship between a lecture and an event.

For those three relationships, when the items are associated in the data set, the relationship is weighted 1; 0 otherwise.

- LR is a relationship between two lectures. We defined two types of LR relationships. They can be either content based similarities or deduced from pairs of lectures. Lecture pairs were provided to participants; the deduced LR_P relationships were weighted considering the frequency of each pair and the number of views associated to its related lectures. Lecture similarities were calculated as described in section 2 and conduced to weighted LR_S relationships. LR_P and LR_S relationships were fused considering a linear combination, such as:

$$LR(l_i, l_j) = \beta * LR_{P(l_i, l_j)} + \gamma * LR\_S(l_i, l_j)$$

where l_i and l_j are two lectures. In the experiments, β=1.5 and γ=0.05. These values have been obtained through manual tuning.

Finally, each type of relationships receives a relative weight. For example, $\mathrm{AR}(l_i, a_j)$ receives a relative weight of 3 between l_i and a_j if the lecture and the author are linked. Figure 1 depicts the various types of relationships that link nodes.

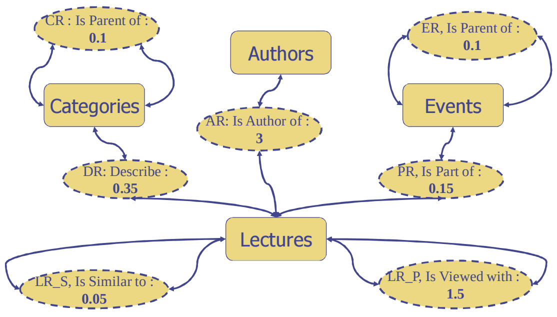

Fig. 1. Nodes and relationships between nodes.

3.2 Random walks

We considered two random walk algorithms: Katz [6] and Random-Forest based Algorithm [5] that consider route accessibility and relative forest accessibility [4]. More details on these methods are presented by Fouss *et al.* [5]. In this latter paper, more methods are also discussed.

Katz. The method proposed by Katz to compute similarities takes into account both direct and indirect links between items [6]. The similarity matrix is defined as:

$$K = \alpha A + \alpha^2 A^2 + \cdots + \alpha^n A^n + \cdots = (I - \alpha A)^{-1} - I \quad (1)$$

where A is the adjacency matrix, I the identity and α constant.
A is the adjacency matrix generated from the complete graph (rows and columns of the matrix are the nodes of the various types) and thus represents direct links between the graph's nodes. A^n represents the indirect links through paths of length n. Both direct and indirect links are taken into account but a coefficient of attenuation is used: α^n represents the attenuation in importance of the links of length n, K exists provided that the attenuation coefficient α is less than the in-

verse of the spectral radius of A. In our experiments, we use α =0.05. This value should have been tuned; but we did not for time reasons.

Random-forest based algorithm (RFA). In RFA, the similarity matrix S between the nodes of a graph G is based on relative forest accessibility. Let $\mathcal{F}(G) = \mathcal{F}$ be the set of all spanning forests of graph G. A spanning forest is any subgraph of G that is cycle free and includes every vertex of G. For any two nodes i and j of G, $\mathcal{F}_{ij}$ denote the subset of $\mathcal{F}$ where i and j belong to the same tree. The relative forest accessibility of i and j is defined as $s_{ij} = \varepsilon(\mathcal{F}_{ij})/\varepsilon(\mathcal{F})$. ε is the weight function defined in [4]. For unweighted graphs $\varepsilon(\mathcal{F}_{ij})/\varepsilon(\mathcal{F})= |\mathcal{F}_{ij}|/|\mathcal{F}|$

[4] demonstrates $(I + L)^{-1}$ exists for any undirected weighted graphs and that :

$$S = (I + L)^{-1} \qquad (2)$$

where L is the laplacian matrix from the adjacency matrix A generated from the complete graph G (see section 3.1).

RFA which can be seen as a rough Laplacian regularization is closely related to the similarity measure associated to the pseudo-inverse of graphs Laplacian L^+(see [4] for more details). L^+ is a valid kernel that preserves the Euclidian commute time distance in graphs. We did not experiment the similarity measure based on L^+ in the context of VLNetChallenge for lack of time to solve a technical problem.

3.3 Implementation issues

All experiments were conducted on Linux computers with a 2.9 GHz Intel Core2 Duo processor P9700 and 6 GB of RAM.

The graphs we handled in the context of VLNetChallenge contain around 15 000 nodes. The approaches we explored are then based on inverting matrices ($\mathcal{O}(n^3)$) of size 15 000 x 15 000. Our attempt to use Scilab [16] (with memory stack set to the maximum) was unfruitful and ended with a stack overflow error after more than 20 hours of running time. After shifting to atlas [17] the Automatically Tuned Linear Algebra Software, the running time was about 20 minutes.

3.4 Results

When considering the preliminary results on the training collection (based on 20% of the final collection), our method obtained from 0.1434 to 0.22465, depending both on the random walk method used and on the weight used for the relationships. The best results have been obtained for RFA using the weights presented in bold font in Figure 1. These weights have been obtained through a rough manual-tuning that used the entire training collection.

When considering the final collection, our method is ranked 9 over 58 submissions without nil results or errors. We obtained a score of 0.24044 while the best result is 0.35857. Interesting enough, when considering the approaches better than ours, we can see that the results decrease from the preliminary results to the final results. One hypothesis could be that those approaches over learnt on test data.

4 Pooled sequences

In this task participants “are asked to recommend a ranked list of ten lectures that should be recommended after viewing a set of three lectures” [13].

To complete this task, we followed an empirical approach according to our knowledge mainly acquired in Information Retrieval field. This knowledge has been transposed and adapted to recommender systems. We tried two approaches that are related to the work we presented in [2]: one was based on **lecture content only**; the second one considered the **categories** associated to lectures and **lecture pair frequency**.

4.1 Content-based approach

In this approach, we considered the lecture content only. We used Solr search engine [14] as explained in section 2. For each lecture of a given triplet, we search for the 50 most similar documents. Then we fused the three retrieved document list using CombSum function [8] that consists in the sum of the document’s individual scores.

When applied to the training collection, the results were slightly above 0.04. Indeed when analyzing the learning data set, we identify that users read lectures related to various topics to complete their knowledge. This variety of topics cannot be captured with a standard content similarity-based measure. For this reason, we did not continue with this content-only approach. Thanks to the work done in the cold-start task, we decided to particularly study lecture pair frequency (importance of LR_P in section 3.1) and categories.

4.2 Category-based approach

Rather than considering the lecture content only, we concentrated on the categories of the lectures. The first issue to solve was the fact that many lectures were not associated with any category. For those lectures, we first associated them with a category considering the hierarchy of events. Once the lectures are associated with a category, we then consider the lectures that have been visited with one of the lectures of the target triplet within close categories in the category hierarchy.

Association of categories to lectures. Some of the lectures are not associated with any category; for those lectures, we applied two algorithms. First for any lecture that is not in categories_lectures, we browsed the lecture-event hierarchy using a bottom up approach and associated the current lecture to the category or categories associated to the closest event (considering the hierarchy). When such a parent does not exist, we associated the category (or categories) of the most similar lectures or events, based on its content or description.

Frequency of lecture pairs. For each lecture of the current triplet, we search for the 100 most visited lectures with the current lecture. We then calculate the lecture score (3). The score of the retrieved lecture l_i is computed as its frequency times the distance between categories. Indeed, this distance between categories allows the system to identify recommendations close in sibling categories. In that way, we emphasize the selection of information in close categories in order to simulate the user behavior according to what we have extracted from the training data set analysis.

$$\text{Score}\ (l_j) = \text{Frequency}\ (l_j) * \text{similarity}\ (\text{category}\ (l_j), \text{category}\ (l_i)) \qquad (3)$$

When a lecture has more than one category, we use the most general category. This treatment is repeated for the three lectures of the triplet and the three lists are fused using CombSum. The distance between categories is inspired from our previous work detailed in [1].

We then ranked the retrieved lectures by decreasing scores. The recommendations are the top 10 lectures. Using this method, it occurs that we obtained less than 10 recommendations. In those cases, we then add lectures to the recommended list.

Completing the recommended list. When less than 10 lectures are recommended using the previous method, we complete the list by considering the lecture content rather than the lecture visits. For each lecture, we search for the 10 most similar lectures. For each lecture, we search for the 100 lectures the **most visited** with the current lecture and calculate the score of the lectures using the same method as previously. When this process fails to complete the list, it is completed with the lectures the most visited thanks to the frequency of lecture views.

4.3 Results

Considering the training set, using our method, we obtained from 0.04453 to 0.18725 (depending on the approach used).

Regarding the complete set, we are ranked 12th with the score of 0.18943. The best score being 0.62415.

The results we obtained show that the visits on lectures has a great importance; more than the content itself.

5 Conclusion

In this paper, we describe the methods we developed for the two tasks defined in VLNetChallenge. With regard to the cold start task, our method was not over trained. We tried various values for the different parameters. A more systematic tuning could help improving the results. With regard to the pooled sequence task, we identified that content only approach is not sufficient. Furthermore, we think that categories could have been used more. For example, for a given triplet, we could have kept only those retrieved lectures that share a category with any lecture of the triplet.

In the two tasks, we also identified the importance of the frequency of lecture pairs. As a conclusion, we expect that combining various dimensions in recommender systems can improve recommendation quality.

References

1. G. Cabanac, M. Chevalier, C. Chrisment, and C. Julien, An Original Usage-based Metrics for Building a Unified View of Corporate Documents, International Conference on Database and Expert Systems Applications (DEXA), Springer-Verlag, LNCS 4653, pp. 202-212, 2007.
2. L. Candillier, M. Chevalier, D. Dudognon, and J. Mothe, Diversity in Recommender Systems: bridging the gap between users and systems, Centrics, to appear 2011.
3. P. Y. Chebotarev and E. V. Shamis. The matrix-forest theorem and measuring relations in small social groups. Automation and Remote Control, Vol. 58, N°9, pp. 1505–1514, 1997.
4. P. Y. Chebotarev and E. V. Shamis, On proximity measures for graph vertices, Automation and Remote Control, Vol. 59, N°10, pp. 1443–1459, 1998.
5. F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation, IEEE Transactions on Knowledge and Data Engineering, 2006.
6. L. Katz. A new status index derived from sociometric analysis. Psychmetrika, 18(1), pp 39–43, 1953.
7. F. W. Lancaster, E. G. Fayen, Information Retrieval On-Line, Melville Publishing Co., Los Angeles, California, 1973.
8. J. H. Lee. Analyses of multiple evidence combination, Proceeding of SIGIR'97, pp. 267–276, 1997.
9. M. F. Porter, An algorithm for suffix stripping, Program, pp. 130–137, 1980.
10. G. Salton and M. J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1983.
11. G. Salton, A. Wong, and C. Yang, A vector space model for automatic indexing, Commu- nications of the ACM, 18(11), pp. 613–620, 1975.
12. http://tunedit.org/challenge/VLNetChallenge/task_1, June 2011.
13. http://tunedit.org/challenge/VLNetChallenge/task_2, June 2011.
14. http://wiki.apache.org/solr/, June 2011.
15. http://www.postgresql.org/, 2011.
16. http://www.scilab.org, August 2011.
17. http://math-atlas.sourceforge.net, August 2011