Recommender Systems by means of Information Retrieval

Alberto Costa
LIX, École Polytechnique
91128 Palaiseau, France
costa@lix.polytechnique.fr

Fabio Roda
LIX, École Polytechnique
91128 Palaiseau, France
roda@lix.polytechnique.fr

ABSTRACT

In this paper we present a method for reformulating the Recommender Systems problem in an Information Retrieval one. In our tests we have a dataset of users who give ratings for some movies; we hide some values from the dataset, and we try to predict them again using its remaining portion (the so-called "leave-n-out approach").

In order to use an Information Retrieval algorithm, we reformulate this Recommender Systems problem in this way: a user corresponds to a document, a movie corresponds to a term, the active user (whose rating we want to predict) plays the role of the query, and the ratings are used as weigths, in place of the weighting schema of the original IR algorithm.

The output is the ranking list of the documents (" users") relevant for the query ("active user"). We use the ratings of these users, weighted according to the rank, to predict the rating of the active user. We carry out the comparison by means of a typical metric, namely the accuracy of the predictions returned by the algorithm, and we compare this to the real ratings from users. In our first tests, we use two different Information Retrieval algorithms: LSPR, a recently proposed model based on Discrete Fourier Transform, and a simple vector space model.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering, Retrieval models; H.3.4 [**Systems and Software**]: Performance evaluation.

General Terms

Algorithms, Measurement, Experimentation.

Keywords

Recommender Systems, Information Retrieval, Reformulation.

WIMS '11, May 25-27, 2011 Sogndal, Norway

1. INTRODUCTION

Recommender Systems (RS) have received a considerable interest from the scientific community in the last decade and they represent by this time a stable field of research. A number of interesting approaches have been proposed by many authors and in particular, Collaborative Filtering [19, 17, 20, 11] gained a great popularity and it is nowadays a well known method. It was remarked [1] that Collaborative Filtering shares fundamental aspects with Information Retrieval (IR) and there is somehow a continuity between these two fields of research. This work belongs to this stream. In fact we are developing a RS which makes use of concepts and tools used elsewhere in an IR context and, believing that the underlying structure could also provide an interesting framework for RS algorithms, we looked for experimental evidence of this intuition.

In our approach first we move from the Recommender Systems domain to the Information Retrieval one. To do this, we consider each user as a document and each movie as a term (even if we can use this approach not only for the movies): in this way, as in IR a document is a set of terms, in the RS field a user is characterized by a set of movies (for which the user has given a rating). Using this representation, the ratings of the users are the baseline for computing the weights of the terms, as explained in section 3.

Moreover, the active user becomes the query in this phase; the meaning of this is that in Information Retrieval we want the documents more similar to the query, and for the RS problem we want the users more similar to the active user.

At this point we can use one of the several existing IR algorithms to obtain the ranking list, that represents the set of users more similar to the active user, ordered by decreasing similarity.

Finally, as explained at the end of section 3, we use the ranking list to get the predicition for the active user; this last step brings us again in the Recommender Systems domain.

The evaluation of most works in this field is carried out using "artificial" datasets provided by well-known research groups, such as GroupLens [7, 18], or by Netflix (`http://www.netflixprize.com`) [3, 2]. This approach ensures somehow a standard method to evaluate results. Hence, we have implemented our in-house algorithm using both Least Spectral Power Ranking (LSPR) model, presented in [6], and an algorithm based on vector space model, as conceived in the 1960s by Salton [15, 16]; we tested it by means of a standard dataset provided by GroupLens. Basically, we have compared it with the "community" which constitutes

the benchmark to overcome, in order to show the feasibility of the approach.

This paper is organised as follows. In section 2 we introduce the problem in a more formal way, and in section 3 we describe the algorithm itself. After that, in section 4 we report the experimental results obtained when running this algorithm. Finally, in section 5 we discuss the results.

2. DESCRIPTION OF THE PROBLEM

We can formulate our problem as follows. We have:

- a set U of users, $|U| = n$;
- a set I of items (movies, songs, restaurants...), $|I| = m$;
- a gain function G which expresses the utility of an item for a user

Utility is expressed by a numeric value representing a rating (the higher the better) varying on a chosen interval V, more formally the function G is defined as:

$$\text{G: } U\text{x}I \rightarrow \text{V}$$

We want to maximize the users' utility by recommending good items and advising against bad ones. The problem is that we do not know "a priori" all the values of G, hence we have to predict users' ratings. This ability is normally tested in an almost empirical way showing that the system is able to predict a set of known ratings.

In this paper we use a dataset provided by GroupLens. Basically there are 100 000 ratings (from 1 to 5) given by $n = 943$ users on $m = 1682$ movies and each user rated at least 20 movies. We represent this dataset with a matrix $D^{m\text{x}n}$, where D_{ij} is the rating given by the user j for the movie i (0 value is used if no rating is available). Our aim is to predict the rating of a user (called "active user") for each movie (using the informations of the matrix D), minimizing the differences between the predicted ratings and the real ones.

3. MODEL

This section describes the core concepts of our framework, where we use both the LSPR model and the vector space model as IR algorithms.

Basically in the LSPR model the query is viewed as a spectrum and each document as a set of filters, with one filter for each document term, whereas the vector space model views terms as basis vectors, documents and queries as vectors of the same space.

Usually in Information Retrieval some weighting schemes are used for the terms of the documents and for the terms in the query; the basic choice is to use the TF-IDF weighting schema for the former, and the IDF weighting schema for the latter. In order to use the IR algorithms for the Recommender Systems, it is necessary to modify these weighting functions. Since each user becomes a document, and each movie becomes a term, there is a similarity between the matrix D and the well-known term-document matrix. At this point, consider an active user $k \in U$, for which we want to predict the rating for the movie $h \in I$. Starting from D, we compute a new matrix WU (that plays the role of the normalized TF-IDF weights matrix in Information Retrieval) as follows:

$$WU_{ij} = \begin{cases} 0 & \text{if } D_{ij} \cdot D_{ik} = 0 \\ 1 - \frac{|D_{ij} - D_{ik}|}{4} & \text{otherwise.} \end{cases} \quad (1)$$

This means that the more the rating of a user for a movie is similar to the rating of the active user, the more its weight (from 0 to 1).

The column k in this matrix is not considered, because it is 0 for the movies not rated by the active user, 1 otherwise.

After that, we compute the weights for the active user (i.e. the IDF weights for the query); we save these informations in the column k of the matrix WU, using the following formula:

$$WU_{ik} = \begin{cases} 0 & \text{if } n_i = 0 \text{ OR } D_{ik} = 0 \\ \log_2\left(\frac{n}{n_i}\right) & \text{otherwise,} \end{cases} \quad (2)$$

where n_i is the number of users that have rated the movie $i \in I$, and n is the total number of users.

Now we are ready to use the Information Retrieval algorithm: the query is represented by the column k of WU, while the documents of the collection are the columns $j \neq k$ of the same matrix with $WU_{hj} \neq 0$. The output of the model is the ranking list of the documents, ordered by increasing relevance. This means that the collection is the set of users that have rated the movie h, and the output is the same set of users ordered from the more to the less "similar" to the active user.

The last operation is to predict the rating. To do this, we use the ratings of the users in the ranking list, weighted by their rank, so that the smaller the rank of the user is, the more his rating is considered. Suppose the ranking is given by the list of users R, where $|R|$ is the number of retrieved users, the rank of each user is from 0 (first) to $|R| - 1$ (last), and $D_{h,j(r)}$ is the rating for the movie h of the user with rank r. The predicted rating is computed as:

$$p_{hk} = \frac{\sum_{r=0}^{|R|-1} \left(1 - \frac{r}{|R|}\right) \cdot D_{h,j(r)}}{\lambda}, \quad (3)$$

where λ is the normalization term, computed as

$$\lambda = \sum_{r=0}^{|R|-1} \left(1 - \frac{r}{|R|}\right) = \frac{|R| + 1}{2}. \quad (4)$$

Figure 1 summarizes the algorithm. Basically, the rows from 1 to 14 represent the operation described by equation (1), while the rows from 15 to 23 implement the equation (2). Finally there is the call to the Information Retrieval algorithm, and the prediction of the rating, according to equations (3) and (4).

4. EVALUATION OF THE ALGORITHM

We can find in literature different approaches to evaluate the performance provided by a Recommender System. However a full examination of all these methods is out of the scope of this work [8]. We adopt a simple and well-known metric which helps us to understand the outcome of our algorithm. We used the accuracy computed as the square root of the averaged squared difference between each prediction and the actual rating (the root mean squared error or "RMSE").

Algoritm: RecSys-to-IR
Input: data set D, active user k, movie h, IR algorithm
Output: prediction p_{hk}

```
1  for each i ∈ I
2        do
3           n_i ← 0
4           for each j ∈ U|j ≠ k
5                 do
6                    if (D_ij · D_ik = 0)
7                       then
8                             WU_ij ← 0
9                       else
10                            WU_ij ← 1 − |D_ij − D_ik| / 4
11                            n_i ← n_i + 1
12                   end if
13          end for
14 end for
15 for each i ∈ I
16       do
17          if (n_i = 0)
18             then
19                   WU_ik ← 0
20             else
21                   WU_ik ← log(n / n_i)
22          end if
23 end for
24 Call the IR algorithm, and get the ranking list R
25 p_hk ← Round ( 2 · (Σ_{r=0}^{|R|−1} (1 − r/|R|) · D_{h,j(r)}) / (|R|+1) )
26 return p_hk
```

Figure 1: The prediction algorithm.

Let $\bar{U}$ be the set of active users, $\bar{I}_k$ be the set of movies rated by an active user $k \in \bar{U}$; let p_{ik} denote the predictions generated by a certain algorithm for the movie i and the active user k, while r_{ik} is the correspondig real rating. RMSE is defined by

$$\text{RMSE} = \sqrt{\frac{\sum_{k\in\bar{U}}\sum_{i\in\bar{I}_k}(r_{ik}-p_{ik})^2}{\sum_{k\in\bar{U}}\sum_{i\in\bar{I}_k}1}}. \qquad (5)$$

In our tests, we round the p_{ik} values to the closest integer number, because the real ratings are integers. As mentioned above, the evaluation of RMSE is typically performed using the "leave-n-out" approach [4], where a part of the dataset is hidden and the rest is used as a training set for the Recommender Systems, which tries to predict properly the withheld ratings.

We calculate the predictions with LSPR and the basic vector space algorithms using data from the training set and we compare the prediction against the real rating in the test set.

Moreover, we employ the *community average* for a certain item (that is the average of the ratings given by all the users who vote the item) as a benchmark, with the aim of measuring how much our algorithm can improve the simple *community recommendation*. Thus, we also compute the RMSE of the community recommendation with respect to the actual ratings provided by the users.

In order to explain the concepts of RMSE and *community average*, consider the following example: we have in our dataset 5 items, and 5 users who rate some of these items (from 1 to 5, 0 means no rating); table 1 shows these ratings.

item\user	1	2	3	4	5
1	0	3	0	0	2
2	0	5	4	0	2
3	2	1	0	3	3
4	1	3	3	2	0
5	0	2	2	4	3

Table 1: Ratings

Furthermore, there are 2 active users for whom we want to predict some ratings, as shown in table 2.

item\active users	a	b
1	0	1
2	5	3
3	0	0
4	2	0
5	0	3

Table 2: Active users

The aim of a Recommender Systems algorithm is to predict the ratings of the active users; in our example, the ratings of the user a for the items 2 and 4, and the ratings of the user b for the items 1, 2 and 5, trying to find values closer to the real ones reported in table 2.

According to the notation used in equation (5), we have $\bar{U} = \{a, b\}$, $\bar{I}_a = \{2, 4\}$, $\bar{I}_b = \{1, 2, 5\}$. Moreover, $r_{a2} = 5$, $r_{a4} = 2$, $r_{b1} = 1$, $r_{b2} = 3$, $r_{b5} = 3$.

To show what is the *community average*, consider p_{a2}: it is the average of the ratings of the users which vote the item 2 (rounded to the closest integer, since the ratings are integer). This means that

$$p_{a2} = \text{Round}\left(\frac{5+4+2}{3}\right) = 4.$$

In a similar way, we obtain $p_{a4} = 2$, $p_{b1} = 3$, $p_{b2} = 4$, $p_{b5} = 3$. Now we have all the elements to compute the RMSE, and we obtain the following result

$$\text{RMSE} = \sqrt{\frac{6}{5}} = 1.095.$$

The results reported in table 3 refer to the analysis we performed using the dataset from GroupLens[1] described above. We used five couples (training set, test set) which share the same composition (80%/20% spilts of the orginal data into training and test data) as suggested by the guidelines of GroupLens itself.

In table 4 we express this result in relative terms by providing the *rate of improvement* with respect to the average of the ratings by the community: LSPR overcomes the community by 8.4% on average, while the vector space model decreases the RMSE by 7.6%.

5. DISCUSSION AND CONCLUSIONS

[1]The dataset can be found on `http://www.grouplens.org`.

set	LSPR	vector space	community
1	0.985	0.989	1.073
2	0.974	0.984	1.067
3	0.971	0.980	1.060
4	0.967	0.979	1.056
5	0.975	0.984	1.065

Table 3: RMSE

set	Improvement LSPR	Improvement vector space
1	8.2 %	7.8 %
2	8.7 %	7.8 %
3	8.4 %	7.5 %
4	8.4 %	7.3 %
5	8.5 %	7.6 %
mean	**8.4 %**	**7.6 %**

Table 4: Improvement over community average.

We can report that the algorithm already outperforms the community even if the gap is not prodigious. As a matter of fact, other algorithms recently proposed by different authors, like [9, 14], show RMSE values in the range 0.88 - 0.95 for the same dataset. However, the aim of this paper is to show that our new approach could be a basis for a more sophisticated algorithm, rather than presenting an algorithm already comparable with the state-of-the-art.

So, first of all, we plan to work on fine-tuning our algorithm, to extend this empirical evaluation and to compare it with some well known algorithms such as KNN or Slope-one [10].

We also plan to perform further experiments with a larger number of datasets and with a finer grain analysis of sensitivity of the algorithm with respect to the size of the data set. In parallel, we are also working on LSPR to improve its scalability and to include a number of optimization techniques.

Moreover, we plan to use other weighting schemes, as the well-known Okapi BM25, and other IR algorithms, for example probabilistic models like Terrier [13, 12]. The final aim of this work is to merge the results to obtain better performances, as already done in Information Retrieval [5].

Basically, we are still trying to better understand if our approach can provide a nice outcome in the Recommender Systems field and we consider the present work as a first answer, so our contribution is an experimental investigation of some possible relations between Information Retrieval and Recommender Systems. In this sense it seems very interesting that the results obtained with LSPR are better than the ones obtained with the vector space model, reflecting the behaviour of these models in the IR field, as reported in [6].

6. REFERENCES

[1] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *CACM*, 35:29–38, 1992.

[2] R. Bell and Y. Koren. Lessons from the Netflix Prize challenge. *SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[3] R. M. Bell, J. Bennett, Y. Koren, and C. Volinsky. The million dollar programming prize. *IEEE Spectr.*, 46(5):28–33, 2009.

[4] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical Report MSR-TR-98-12, Microsoft Research, 1998.

[5] H. Chen and D. R. Karger. Less is more: probabilistic models for retrieving fewer relevant documents. In E. N. Efthimiadis, S. T. Dumais, D. Hawking, and K. Järvelin, editors, *SIGIR*, pages 429–436. ACM, 2006.

[6] A. Costa and M. Melucci. An information retrieval model based on Discrete Fourier Transform. In H. Cunningham, A. Hanbury, and S. Rüger, editors, *Proceedings of the 1st Information Retrieval Facility Conference*, volume 6107 of *Lecture Notes in Computer Science*, pages 84–99, Vienna, 2010. Springer, Heidelberg.

[7] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA, 1999.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[9] L. Kozma, E. Ilin, and T. Raiko. Binary principal component analysis in the netflix collaborative filtering task. In U. Adali, J. Chanussot, C. Jutten, and J. Larsen, editors, *proceedings of the IEEE International Workshop on Machine Learning for Signal Processing*, Grenoble, France, 2009.

[10] D. Lemire and A. Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[11] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[12] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.

[13] I. Ounis, C. Lioma, C. Macdonald, and V. Plachouras. Research directions in terrier. *Novatica/UPGRADE Special Issue on Web Information Access, Ricardo Baeza-Yates et al. (Eds), Invited Paper*, 2007.

[14] L. G. D. A. Rivera. Sampling pca, enhancing recovered missing values in large scale matrices. Technical Report 80555S, Helsinki University of Technology, 2009.

[15] G. Salton. *Automatic information organization and retrieval*. Mc Graw Hill, New York, NY, 1968.

[16] G. Salton. Mathematics and information retrieval. *Journal of Documentation*, 35(1):1–29, 1979.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001.

[18] J. Schafer, J. Konstan, and J. Riedl. Recommender

systems in e-commerce. In *In Proceedings of the ACM Conference on Electronic Commerce*. ACM, 1999.

[19] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, chapter 9, pages 291–324. Springer, 2007.

[20] E. Vozalis and K. Margaritis. Analysis of recommender systems algorithms. In E. Lipitakis, editor, *The 6th Hellenic European Conference on Computer Mathematics & its Applications*, pages 732–745, Athens, 2003.