Structured Collaborative Filtering

Alejandro Bellogín
Universidad Autónoma de Madrid
Madrid, 28049 Spain
alejandro.bellogin@uam.es

Jun Wang
University College London
Malet Place, London WC1E 6BT, UK
j.wang@cs.ucl.ac.uk

Pablo Castells
Universidad Autónoma de Madrid
Madrid, 28049 Spain
pablo.castells@uam.es

ABSTRACT

In a general collaborative filtering (CF) setting, a user profile contains a set of previously rated items and is used to represent the user's interest. Unfortunately, most CF approaches ignore the underlying structure of user profiles. In this paper, we argue that a certain class of interest is best represented jointly by several items, drawing an analogy to "phrases" in text retrieval, which are not equivalent to the separate meaning of their words. At an alternative stance, we also consider the situation where, analogously to word synonyms, two items might be substitutable when representing a class of interest. We propose an approach integrating these two notions as opposing poles on a continuum spectrum. Upon this, we model the underlying structure in user profiles, drawing an analogy with text retrieval. The approach gives rise to a novel structured Vector Space Model for CF. We show that item-based CF approaches are a special case of the proposed method.

Categories and Subject Descriptors

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

General Terms

Algorithm, Performance, Experimentation

Keywords

Recommender systems, Collaborative filtering, synonymy

1. INTRODUCTION

Most Collaborative Filtering (CF) algorithms disregard the fact that the rating data shows some structure. For example, in an item-based approach, the candidate item's preference is commonly predicted by independently evaluating its similarity to each individual items in the profiles and then averaging them together to obtain the final score. However, the items in the system may have explicit relations among them, such as sharing the director (for movies), or belonging to the same genre (movies, music, books). Dealing properly with this structure is a challenge for the community. Most of the works assume some kind of clustering is performed on the users or the items, and then, predictions are performed using only these subsets of the collection [4, 1]. Obviously, this would increase the sparsity of the system, which turns out to be a serious problem for real-world systems. On the other hand, some works propose to use clustering for avoiding the sparsity, although it is not clear how the structure is kept in this situation [8].

Recently, different papers have explicitly linked CF algorithms with Information Retrieval (IR) techniques, and apply them successfully for recommendation. In [7], the authors find an analogy between implicit CF and IR, applying the Probability Ranking Principle from IR to CF. More recently, [2] proposes a framework in which any IR scoring function could be used with CF rating data. In [3], the authors reformulate the recommendation problem and use algorithms from IR, namely a model based on Discrete Fourier Transform and the Vector Space Model (VSM).

In the proposed approach we adapt, upon the already found analogies between IR and CF, the extended Boolean retrieval model presented in [6]. In that work, the authors introduce a generic model in which intermediate systems between Boolean and VSM models appears naturally. In particular, the authors acknowledge that the structure of the query should be altered in order to distinguish between compulsory terms (phrases) and alternative words (synonynms), whereas the Boolean strategy requires a very strict interpretation of such structure, the VSM model completely loses this distinction and the terms are considered independent of each other.

The proposed approach defines a method for providing structure to user profiles in CF. The extended model is applied to two particular tasks: a) *user profile expansion*, which consists of propagating user ratings to other similar (or synonym) items; and b) user profile decomposition into area-specific subprofiles, as an enhanced structure enabling recommendation performance improvements. We report empiric results confirming that structured user profiles in CF outperform the standard item-based approach, which is a particular case of our model with plain profiles and no synonymy expansion.

2. STRUCTURED FILTERING MODEL

The main idea of this work is inspired by the Salton's classic paper [6]. The novelty here lies in the use of the analogy between IR and CF proposed in [2] to address the structure problem in CF. In the next subsections, we propose how structure could be added to CF, by adapting the methods proposed in [2] so that more advanced models such as [6] could be used.

2.1 A Vector-Space Representation

As already proposed in [2], we can obtain a new insight into the current CF approaches by reformulating them using a Vector Space Model [5]. Formally, a user profile can be regarded as a query. Each of the rated items in the profile is considered as a query term. Therefore, we could use a vector to represent a user profile as follows:

$$\mathbf{Q}^u = (i_1, r^u_{i_1}; \ldots; i_k, r^u_{i_k}; \ldots; i_n, r^u_{i_n}) \tag{1}$$

where $\mathbf{Q}^u$ denotes a user profile u. $r^u_{i_k}$ represents the rating

Table 1: "AND" and "OR" representations of a user interest

(a) using Boolean retrieval.

	Query items		user interest representation	
	a	b	a OR b	a AND b
item 1	1	1	1	1
item 2	1	0	1	0
item 3	0	1	1	0
item 4	0	0	0	0

(b) using extended Boolean retrieval ($p = 2$).

	Query items		user interest representation	
	a	b	a OR b	a AND b
item 1	1	1	1	1
item 2	1	0	$1/\sqrt{2}$	$1-1/\sqrt{2}$
item 3	0	1	$1/\sqrt{2}$	$1-1/\sqrt{2}$
item 4	0	0	0	0

of item k by this user, where $k \in \{1, n\}$. It is equal to 0 when item k is not rated by u. In practice, the ratings are normalized with respect to the users' mean or items' mean.

By contrast, the item representation is different to the text retrieval. In CF, we do not have a common feature space. Thus, we should project each item into the same feature space as queries by using its similar items. That is

$$\mathbf{I}^j = (i_1, s^j_{i_1}; \ldots; i_k, s^j_{i_k}; \ldots; i_n, s^j_{i_n}) \tag{2}$$

where s_{i_k} is the similarity between the candidate item j and item k. In practice, it has proven useful to select top-n similar items as opposed to use all the similar items.

Given the vector representations of Eq. (1) and Eq. (2), a query-item similarity value may be obtained by comparing the corresponding vectors, using for example the conventional vector product:

$$\text{Sim}(u, i) = \text{Sim}(\mathbf{Q}^u, \mathbf{I}^j) = \sum_{k=1}^{n} s^j_{i_k} \cdot r^u_{i_k} \tag{3}$$

The system could provide a ranked recommendation output in decreasing order of the computed similarities between $\mathbf{Q}^u$ and $\mathbf{I}^j$. In practice, it is also needed to predict the user's rating of an unspecified item. The similarity score should be normalized to produce a rating in the proper range.

$$\hat{r}^u_j = \frac{\sum_{k=1}^{n} s^j_{i_k} \cdot r^u_{i_k}}{\sum_k s^j_{i_k}} = \sum_{k=1}^{n} \bar{s}^j_{i_k} \cdot r^u_{i_k} \tag{4}$$

where $\bar{s}^j_{i_k} = \frac{s^j_{i_k}}{\sum_k s^j_{i_k}}$. It can be easily seen that Eq. (4) is indeed an item-based CF approach, but rather in a vector space formulation.

2.2 An Extended Vector-Space Representation

As we discussed previously, the user profile represented by independently rated items is problematic. We shall illustrate it by considering the following Boolean retrieval example. Suppose we have a class of user interest, and it is represented jointly by two items a and b. In a Boolean retrieval model, we would use an "AND" rule, meaning that a candidate item should be similar to both of them. By contrast, if a class of user interest is represented by either of two items, An "OR" rule may be applied. Table 1(a) illustrates the results from the Boolean Retrieval model for the four different types of items. Note that for simplicity, the binary similarity is assumed.

From the table, we can see that the Boolean retrieval is too rigid in terms of producing the ranking score, either too loose or tight. And in practice, the interest representation is between "AND" and "OR". An extension of the Boolean model was given in [6]. The idea was to apply a more discriminative ranking formula by calculating the distance towards the most desired point in the vector space. Let us define how we represent the query and documents in the extended model by using a regular expression, where n is the number of terms in the collection:

$$\begin{aligned}
\mathbf{Q} &:= \begin{array}{c}\text{and}\\ \text{or}\end{array} (p_1) \left[\begin{array}{c}\text{and}\\ \text{or}\end{array} (p_2)[Q]^+, cw \right] \\
\mathbf{Q} &:= (\text{qw}_1, \ldots, \text{qw}_n) \\
\mathbf{D}^i &:= (\text{dw}_{i,1}, \ldots, \text{dw}_{i,n})
\end{aligned}$$

In this definition, a clause query is a combination of simple queries (as in VSM, where qw_i is the weight given for that query to the term t_i) by means of connectives and clause weights (cw). Besides, each connective has an associated p-value, in this case, we denote p_2 as the inner value and p_1 as the outer one. Documents are simply represented as in the standard VSM, where $\text{dw}(i, j)$ is the weight between document D_i and term t_j. Using this notation, the similarity between a query and a document can be computed recursively until a simple query is found, by using cw to weight the importance of the similarity between the subqueries and the document (more details in [6]). Depending on the connective, similarity between a document and a simple query is calculated as follows:

$$sim(\mathbf{Q}_{or(p)}, \mathbf{D}^i) = [\frac{(\text{qw}_1\, \text{dw}_{i,1})^p + \ldots + (\text{qw}_n\, \text{dw}_{i,n})^p}{(\text{qw}_1)^p + \ldots + (\text{qw}_n)^p}]^{1/p}$$
$$sim(\mathbf{Q}_{and(p)}, \mathbf{D}^i) = 1 - \left[\frac{(\text{qw}_1(1-\text{dw}_{i,1}))^p + \ldots + (\text{qw}_n(1-\text{dw}_{i,n}))^p}{(\text{qw}_1)^p + \ldots + (\text{qw}_n)^p}\right]^{1/p}$$

where $p \in [1, +\infty)$ is the corresponding connective p-value. Note that when $p = 1$ we have $sim(\mathbf{Q}_{or}, \mathbf{D}_i) = sim(\mathbf{Q}_{and}, \mathbf{D}_i)$.

In this way, depending on the value of p we can generalize different retrieval systems. For instance, when $p = \infty$ a standard Boolean scoring is performed, for $p = 1$ a VSM is obtained, and the rest of the values produce intermediate extended retrieval models.

Furthermore, as stated in [6], each p-value has a *semantic* interpretation in IR, depending on the connectives used. If $p = \infty$ and the connective is an "AND", then a strict **phrase** has to be matched, i.e., the document is not retrievable unless all phrase components are present. If, on the other hand, the connective is an "OR", then a strict **thesaurus** feature is used, i.e., every term is substitutable one for another. When the p-value is lower, these constraints are less strict, in the sense that, for the "AND" connective, the presence of every term is worth more than the presence of only some of them, but they are not compulsory; similarly, for the "OR" connective, the presence of several terms from a given class is more important than the presence of only one term. Finally, when $p = 1$ both connectives are equivalent and the distinction between phrase and thesaurus disappears, thus only the presence or absence of the terms is considered, i.e., the terms are independent of each other.

Therefore, in this paper, by adapting the extended retrieval model to CF, we would be able to analyze how phrase and thesaurus features perform in the user-item space. In the next sections, we describe such adaptation and propose different applications for it.

2.3 Modelling Structured User Profiles

We have seen in the previous section that documents need no further modification. Thus, in the context of CF, this means that only the user profile (query) has to be expressed as a set of clause queries by using different connectives, and the items (documents) may be represented as in Section 2.1.

Since the equivalence presented in that section corresponds with a VSM in IR, the representation of the constituent elements in the extended model is simply as follows, where n now is the number of items in the system:

$$\begin{aligned}\mathbf{Q}^u &= \text{and}(1)\,[\text{and}(1)((i_1, r^u_{i_1}), \ldots, (i_n, r^u_{i_n}), 1)] \quad (5)\\ \mathbf{I}^j &= \left(s^j_{i_1}, \ldots, s^j_{i_n}\right)\end{aligned}$$

In this situation, the query is defined as an "AND" rule with $p = 1$. As we have already noted, for this value of p, the similarity formula of an "AND" rule is equivalent to that of an "OR" rule, so we can define the queries with or(1) instead of and(1). Besides, we have to note that Eq. (4) is not completely equivalent with the formulation presented herein, since standard item-based CF normalizes with respect to the similarity values, which are encoded in the document vector, while in IR the model is normalized using the query weights. Both representations, nonetheless, could be equivalent with a slight modification of the extended model.

2.4 Exploiting Structured Profiles

Once an extended user profile representation is available for recommendation by adapting the extended retrieval model, we envision two main applications. First, performing expansions over the user profiles, in a similar way as IR researchers include synonyms and related words when expanding queries. Second, providing structure to the profiles based on implicit similarities found between the items in the system. In the following, we explain how these two applications may be performed in CF.

2.4.1 Profile expansion

In any recommendation system, there are some items which tend to occur very frequently together, such as movie series (e.g., Lord of the Ring, Star Wars, or Star Trek), or movies by some particular director. These movies could be considered close *synonyms*, in the sense that people tend to like them all or none of them. If this assumption holds, once one item belonging to a particular group is rated in a profile, the rating could be propagated to the rest of synonyms in the system, by using an "OR". In this way, the rating sparsity would be reduced, since each user would contain in her profile additional items, implicitly derived through the synonym relation.

A very important parameter in this setting would be the expansion size, that is, how many synonyms are incorporated into the user profile. As a first attempt, we can expand all the user profiles in the same way. However, as in query expansion, it seems clear that some users could benefit more from the expansion than others. Query performance techniques, which have been frequently applied to query expansion, can be very useful in order to determine the strength of the expansion on a per user-basis. Another important parameter is the p-value, which, for instance, could be set based on the co-occurrence strength between each pair of items, or any function which manipulates those co-occurrences, such as normalization by the average, minimum, or maximum value.

2.4.2 Inferring profile structure

It often makes sense to find subprofiles within user profiles. This is a very recurrent idea in recommender systems, with different motivations, such as when two users A and B have very similar movie tastes, but very different in music. Their music dissimilarity should not obscure their movie similarity, so that A and B can get area-specific recommendations from each other based on their area-specific common tastes. Preference clusters can be created automatically and used in many ways (see [4, 8] among others).

In this view, user profile vectors could be decomposed into a soft "OR" of cohesive subprofiles or *phrases* (music tastes, movie tastes, sports, touristic routes, etc.), where subprofiles are also soft "OR"s of item ratings, and the outer "OR" should have a higher p than the inner "OR". Different options arise when defining this structure with respect to how the clusters are defined and how to set the inner and outer p values.

Clusters within a user profile can be created using predefined clusters over the item space, and these clusters can be found using classic clustering algorithms (such as K-means) or any other alternative strategy. Pearson correlation can be used as distance between items, and external features such as genre or actor information apart from ratings may be taken as input data.

Inner p-values can be set in many different ways: they can be fixed or depend on the average (or minimum, maximum) similarity in the cluster or its centroid, using an additional step function discretizing those values, in such a way that the higher the cohesion in the cluster, the higher the inner p-value. The outer p-value can have a fixed, low value since we want to retrieve documents matching any of the subprofiles, however if we want to boost those documents matching more than one subprofile, a higher p-value should be used instead. Actually, the outer "OR" rule may be replaced by an "AND" rule in order to check if documents matching every subprofile are more likely to be relevant to the user or not.

3. EXPERIMENTS

The experiments have been carried out using the publicly available dataset called *Movielens 100K*[1]. This dataset contains 943 users, 1682 items and 100000 ratings. We performed a 5-fold cross validation using the splits contained in the public package, these splits retain the 80% of the data for training, and the rest for testing.

The evaluation was performed as explained in [2], that is, for each user, a ranking is generated by predicting a score for every item in the test set. We then measure the performance of this ranking, using the *trec_eval* program.

In the next sections, we present the results obtained in the two tasks where the extended retrieval model has been applied: expanding the user profiles and inferring profile structure.

3.1 User profile expansion

In this experiment, we analyze the recommendation performance variations that result from introducing different amounts of profile expansion, compared to a plain item-based recommender baseline without expansion ($S = 0$). Besides, as noted in Section 2.4.1, we consider the expanded items as synonyms, and thus, we include an infinity inner p-value, in order to have the standard Boolean behaviour for the "OR" connective. As shown in Table 2(a), we may observe a significative improvement when user profiles are expanded with similar items, and the more synonyms are used in the expansion, the better the performance. It is worth noting that changing the order of the connectives (bottom row) leads to poor results, showing that profiles built this way make little sense.

The baseline in the table is the method presented in Section 2.1, which corresponds to the standard item-based CF approach. We can see that properly using the extended model leads to outperforming the baseline formulation. In contrast with [2], we have not yet experimented with different normalization techniques, in order to keep the extended model as close to the original as possible. The results suggest, nonetheless, that a combination of both methods might result in higher, more significant performance improvements,

[1]Available at http://www.grouplens.org/node/73

Table 2: Results. Outer and inner connectives in the user profile representation are presented in this order.
(a) Performance values for the profile expansion experiment. S is the expansion size.

Method	P@1	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@50	MRR
Baseline	0.002	0.004	0.005	0.007	0.002	0.003	0.009	0.027
$S = 1, \text{and}(1), \text{or}(\infty)$	0.129	0.121	0.120	0.114	0.099	0.099	0.126	0.243
$S = 2, \text{and}(1), \text{or}(\infty)$	0.149	0.138	0.130	0.119	0.113	0.110	0.128	0.260
$S = 5, \text{and}(1), \text{or}(\infty)$	0.190	0.166	0.155	0.141	0.140	0.134	0.146	0.304
$S = 10, \text{and}(1), \text{or}(\infty)$	0.197	0.171	0.161	0.147	0.146	0.140	0.151	0.313
$S = 5, \text{or}(\infty), \text{and}(1)$	0.004	0.005	0.005	0.005	0.002	0.003	0.010	0.028

(b) Performance values for the dynamic user profile expansion.

Method	P@1	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@50	MRR
Threshold found by median	0.183	0.167	0.157	0.145	0.137	0.132	0.158	0.302
Threshold found by average	0.186	0.165	0.158	0.146	0.137	0.133	0.157	0.303

(c) Performance values for structured profiles.

Method	P@1	P@3	P@5	P@10	NDCG@3	NDCG@5	NDCG@50	MRR
Baseline	0.002	0.004	0.005	0.007	0.002	0.003	0.009	0.027
K-means genre $\text{or}(1), \text{or}(2)$	0.013	0.016	0.017	0.018	0.010	0.011	0.032	0.061
K-means genre $\text{or}(2), \text{or}(2)$	0.005	0.006	0.006	0.006	0.003	0.004	0.011	0.030
K-means sim $\text{or}(1), \text{or}(2)$	0.008	0.011	0.014	0.016	0.007	0.009	0.023	0.047
K-means sim $\text{or}(2), \text{or}(2)$	0.006	0.006	0.006	0.008	0.004	0.004	0.012	0.031

since they may be used complementarily: while our method expands the profiles, the method described in [2] explores different techniques inspired by IR scoring functions.

In Table 2(b), we present further experiments with a dynamic profile expansion strategy. In this experiment, the parameter S is not fixed for all the users in the system, but depends on each user's characteristics. In particular, and as a first attempt, we assume that the size of the user profile might be an indicator of the demand for more or less items in the expanded model —in the same way as the query length may be used in IR for deciding whether or not to expand the query. In this context, we only expand those users which are below some threshold. In the table, we show the results when the threshold is the average or the median of the number of ratings in the system. These two methods obtain very similar results, and compared with the previous methods, they outperform static expansion when the cutoff value is high, e.g., P@10 and NDCG@50.

3.2 Inferring user profile structure

In order to compare whether structured profiles improve the performance, we compare against the baseline method presented in Section 2.1, which simply uses plain profiles (no structure). Furthermore, to provide structure to the item profiles, two clustering algorithms were used: K-means and X-means (as implemented by Weka library[2]). These algorithms have been trained on different data related with items, more specifically, on genre information and similarity values among items. Different values for the number of clusters with K-means were tried, and an optimal $K = 50$ was found and used in our experiments.

Table 2(c) shows the results of this experiment. We have tried a uniform p for all the subprofiles, which corresponds to the clusters found by the clustering algorithm. In the future, a different p depending on the intracluster similarity might be considered. Results with the X-means algorithm are not reported here because it always performed worse than K-means. As we can see in this table, structured profiles obtain better performance than plain profiles. Besides, we have found no significative differences when changing the inner p-value, since or(2) and or(5) gave the same result. Furthermore, clustering genre information about items provides better results than clusters generated using similarity between items. This may be due to the relatively higher amount of noise when using item similarity, in comparison with explicit external information such as item genres.

[2]Available at http://www.cs.waikato.ac.nz/˜ml/weka/

4. CONCLUSION AND FUTURE WORK

We have presented a method for incorporating structure into the user profiles of CF algorithms. Upon the work described in [2], we apply an extended retrieval model which allows for incorporating such structure. We report two applications for this approach: user profile expansion and inference of profile structure. Empiric results show that our methods introduce significant performance improvements over an item-based CF baseline, which can be simply represented in our framework as a method in which the items are considered independent from each other. Therefore, taking into account the structure of the user profiles seems to lead to better performing CF algorithms, while it opens up new possibilities and applications.

As future work, we plan to investigate a formal way for calculating automatically the p-values, instead of the fixed values used so far. This would provide insights about the real meaning in CF of the IR concepts of synonymy and phrase. More generally, the proposed model may provide a flexible ground for the exploration of further potential applications. The extraction of user subprofiles, for instance, is a recurrent problem in the recommender systems area, which may find a wide range of uses.

5. REFERENCES

[1] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, pages 245–248. ACM, 2009.

[2] A. Bellogín, J. Wang, and P. Castells. Text retrieval methods for item ranking in collaborative filtering. In *ECIR*, pages 301–306. Springer, 2011.

[3] A. Costa and F. Roda. Recommender systems by means of information retrieval. In *WIMS*, pages 57:1–57:5. ACM, 2011.

[4] M. O'Connor and J. Herlocker. Clustering items for collaborative filtering. In *ACM SIGIR Workshop on Recommender Systems*, 1999.

[5] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, 1987.

[6] G. Salton, E. A. Fox, and H. Wu. Extended boolean information retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.

[7] J. Wang, S. Robertson, A. de Vries, and M. Reinders. Probabilistic relevance ranking for collaborative filtering. *Information Retrieval*, 11(6):477–497, 2008.

[8] G. R. Xue, C. Lin, Q. Yang, W. Xi, H. J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR*, pages 114–121. ACM, 2005.