Movie Recommender: *Semantically Enriched* Unified Relevance Model for Rating Prediction in Collaborative Filtering

Yashar Moshfeghi, Deepak Agarwal, Benjamin Piwowarski, and Joemon M. Jose

Department of Computing Science, University of Glasgow, Glasgow, UK
{yashar,agarwal,bpiwowar,jj}@dcs.gla.ac.uk

Abstract. Collaborative recommender systems aim to recommend items to a user based on the information gathered from *other* users who have similar interests. The current state-of-the-art systems fail to consider the underlying semantics involved when rating an item. This in turn contributes to many false recommendations. These models hinder the possibility of explaining *why a user has a particular interest* or *why a user likes a particular item*. In this paper, we develop an approach incorporating the underlying semantics involved in the rating. Experiments on a movie database show that this improves the accuracy of the model.

1 Introduction

Recommender systems help users find items they might like in large databases, thus reducing the information overload problem that such users face. Amongst recommender systems, collaborative filtering systems are the most successful and widely used [1]. In collaborative filtering approaches, items are recommended to a user based on the information gathered from other users who have similar interests. This subject became an important area of research [2] from the mid 90's onwards. Over the last decade, much work in both industry and academia has been carried out to improve such applications [3]. Examples of recommender systems are Amazon.com for books, CDs and other products [4], MovieLens for movies [5], and Jester for recommending jokes [6].

Schematically, collaborative filtering systems aim to predict the rating of an unrated item for a user. The rating is typically a numerical value on a scale between 1 (dislike) and 5 (like). Two distinct approaches can be distinguished that aim at predicting ratings, namely memory-based and model-based. To predict a rating, model-based techniques make predictions based on a hypothetical user and/or item model. Memory-based approaches use past ratings and comparisons between users and/or items. In this work, we extend a memory-based approach because it allowed us to easily include semantic information.

Data sparsity is a known issue with any of these models. Rating data is typically sparse, firstly because users don't exhaustively judge all the items they could. Secondly, there are unpopular or unviewed items. Finally, in special

M. Boughanem et al. (Eds.): ECIR 2009, LNCS 5478, pp. 54–65, 2009.

cases (new item and/or new user), only a few ratings are available. Memory-based approach can traditionnaly be classified either as item or user based. In user-based systems, the prediction of the rating of an item for a given user will depend upon the ratings of the *same* item by similar users. Similarly, in item-based systems the predicted rating depends upon the ratings of other similar items by the *same* user.

Both user and item-based approaches only use a restricted part of the available information, because they either restrict the considered ratings to those on the same item (user-based) or for the same user (item-based). This issue is even more important when data is sparse. In this paper we extend the work presented in [7] and propose a *unified approach* that makes use of all the possible data, i.e. both similar items and similar users when predicting a new rating in order to alleviate the sparsity problem.

Besides data sparsity, we postulate that collaborative systems fail to predict accurate ratings because most of them don't take into account semantic information. More precisely, the underlying reasons for two users giving the same rating to an item might be completely different. For instance, one user might be interested in a movie because of its genre, while another because of one of its actors. Following this idea, some works have proposed to use semantic information to improve the performance of their recommendation by using a latent semantic model [8], or an ontology [9]. The work presented in [1] considers both approaches. The authors in [10] used ontology languages such as OWL [11] to classify items and used support vector machine techniques to predict the class of one object.

In this paper, we first define a methodology to compute the similarity of users and items in semantic spaces. We then describe two ways to integrate this information into the Wang et al. approach [7], which we chose because of its potential to easily integrate different sources of evidence. The paper is organised as follows. Section 2 describes state-of-the-art works in collaborative filtering with a particular emphasis on the work upon which ours is based. Our approach is then described in Section 3. Section 4 outlines our methodology. Section 5 gives the experimental methodology and the results. We then conclude and discuss future work.

2 Background

In this section, we begin by presenting collaborative filtering work, focussing on memory-based ones. We then present the approach used in [7], in order to be in position to present our approach in the subsequent section.

2.1 Related Work

Research in collaborative filtering started with memory-based approaches [7]. In this type of approach all ratings are stored as-is into memory. To describe how a rating is predicted, we first need to define the similarity between two users: Two users are similar if they give the same ratings to the same items. To predict

the rating of an item i for a user u, this method simply averages the ratings $r_{i,u'}$ (same item, different user) weighting them by the similarity between u and u'. Because of the emphasis on the similarity between users, this approach is referred to as user-based collaborative filtering [1].

Lack of scalability is one of the shortcomings of user-based approaches. Because user profiles change rapidly as users interact with the system, the neighbourhood construction phase must be an online process. Therefore, for very large data sets, the latency for computing the recommendation can be unacceptable. Item-based collaborative filtering, which is similar to the user-based but swapping the role of the user and the item, was proposed as a solution to the scalability problem in user-based collaborative filtering [12]. In item-based approaches such as [12], an item representation can remain the same when users interact with the system. Therefore, similarity between items can be pre-computed and used when the system wants to predict a new item's rating.

As was stated in the introduction, data sparsity is an issue for user or item-based systems, regardless of the sparsity origin. In general, the number of ratings that need to be predicted is usually very large compared to the number of ratings already obtained. This can be even more important, for instance when a user has unusual preferences [13]. Overcoming this problem is not an easy issue, and is beyond the scope of this paper.

The data sparsity problem has been tackled by dimensionality reduction techniques. Latent Semantic Indexing is a typical example of such an approach, and has been used in collaborative filtering [14,15]. A downside of addressing sparsity issues by reducing the dimension is that it also discards important information that would have been useful for predicting ratings [16,17].

Another type of approach is model-based collaborative filtering. Algorithms such as [8,18] predict ratings by learning a model from the collection of ratings. There are several model-based collaborative recommendation approaches proposed in the literature, such as the Bayesian model [19], probabilistic relational model [20], linear regression model [21] and maximum entropy model [22]. Other work used unsupervised clustering of user records [23], or supervised classification models [24].

The advantages of model-based approaches is that they overcome the scalability issue by separating the offline tasks of creating user models from the real-time task of recommendation generation, and that data sparsity is usually handled. However, this is at the cost of either tuning of a significant number of parameters which makes them difficult to be used in practical scenarios [7] or lower recommendation accuracy [1], which has to do with the fact that as in dimensionality reduction techniques, machine learning techniques tend to remove noise, which in the case of collaborative filtering can be a signal useful for rating (e.g. preferences shared by a few users).

2.2 Unified Collaborative Filtering Model

We now focus on the work of Wang et al. [7]. Note that we changed sligthly the notations in order to make them clearer and consistent with our methodology.

Wang et al. [7] were inspired by the probabilistic models of relevance used in information retrieval. They presented three different models. The first two are probabilistic versions of a user-based and item-based memory-based system. The third is an unified model that uses three sources of information: user's rating for different items, other users' rating for the same item, and ratings from different but similar users for other but similar items. The last point makes the approach different to standard memory-based approaches.

We now describe the method more precisely. Let R be the random variable that takes values between 1 and M where M is the number of rating grades. Let U and I be discrete random variables over the sample space of respective users and items. The prediction is then equal to the estimation of the expectation of the the rating R given the user u and the item i. For simplicity, the events $R = r$, $U = u$, and $I = i$ will be respectively denoted r, u, and i. Using these notations, for a given user u and a given item i, we have:

$$\mathbb{E}(r|u,i) = \sum_{r=1}^{M} rP(r|i,u) = \sum_{r=1}^{M} rP(u,i|r)P(r) / \sum_{r=1}^{M} P(u,i|r)P(r) \qquad (1)$$

It is easy to evaluate $P(r)$ since it can be estimated by the ratio of the number ratings equal to r to the total number of ratings made so far.

The problem is now to evaluate the probability to have a given user u and a given item i knowing that the rating is r. It is postulated that a rating determines a density over the space of users and items. This density is estimated with a standard Gaussian kernel. In order to simplify the problem, they also postulated that the kernel defined over the users and items could be decomposed as the product of two separate kernels. This gives:

$$P(u,i|r) \;=\; \frac{1}{|S_r|} \sum_{(u',i') \in S_r} K_U(r_u - r_{u'}) K_I(r_i - r_{i'})$$

where S_r is the set of user-item couples with a rating r.

In order to cope with data sparsity, they used bandwidth parameters that are learned automatically from data. The idea is that if users (or items) are far from each other for the same rating, then the bandwidth will be high, and it will be low in the opposite case. The bandwidth parameters are learned automatically (given a training set of users, items and associated ratings) with a variation of the EM algorithm. Including the bandwidths h_{U-R} (for Users in a Rating space) and h_{I-R} (for Items in a Rating space) in the above formula gives:

$$P(u,i|r) = \frac{1}{|S_r|} \sum_{(u',i' \in S_r)} K(\frac{r_u - r_{u'}}{h_{U-R}}) K(\frac{r_i - r_{i'}}{h_{I-R}}) \qquad (2)$$

In order to use a kernel, it is necessary to define a distance, and hence, first to define the vector space where users and items lie. Wang et al. [7] followed the standard memory-based approach: A user is represented in a space where dimensions are the different items and where the components of the vector are

the ratings he or she made (a rating of 0 is usually used when the user did not rate the item).

Several different distances can be used in a vector space, including the standard Euclidian one. However, Wang et al. [7] chose a metric based on cosine similarity. For two users u and u' the distance is defined as $1 - \cos(u, u')$. This distance actually projects a couple of users in a one dimensional space. More precisely, it projects the couple into the space defined by the cosine distance between two users. Other metrics could be considered (e.g. the Pearson correlation coefficient), but the cosine distance has been shown to perform better [25]. Including the distance in formula (1) and using a Gaussian kernel gives the final formula used to predict a rating:

$$\mathbb{E}(r|u,i) = \frac{\sum_{u',i' \in S} r_{u',i'} e^{-\frac{1-\cos(r_u, r_{u'})}{h_{U-R}^2}} e^{-\frac{1-\cos(r_i, r_{i'})}{h_{I-R}^2}}}{\sum_{u',i' \in S} e^{-\frac{1-\cos(r_u, r_{u'})}{h_{U-R}^2}} e^{-\frac{1-\cos(r_i, r_{i'})}{h_{I-R}^2}}} \tag{3}$$

The results of their experiments illustrate that the unified model performed better than user and item-based. The main reason is that data sparsity is addressed by taking much more data into account than simple item or user based models.

3 Approach

Like most work in collaborative filtering, the work of Wang et al. [7] relies on a distance based on the cosine distance between two users (or two items) in the standard rating space. A number of questions remain unanswered by the rating representation, for example "*why does the user like this item?*", and "*What are the underlying interests of two users that make them similarly rate some items?*".

Our approach addresses these questions by considering semantic spaces. Each item – whether it is a movie, music, book, product, game, etc. – can be characterised by semantic traits. For example, the genre of a movie can be chosen among different types as comedy and drama. To compute the similarity between two items or users, we not only consider the rating space, but also compute the similarity in semantic spaces. These spaces need to be defined in relation to the type of items we are dealing with (e.g. movies, books, etc.). For example, in the case of movies, people are usually interested in specific genres, follow the movies of specific actors, or have favourite directors. Each of these dimensions contain valuable information that can be used as an important source of information to decide whether an item should be recommended to the user or not. We associate to each dimension a so-called *semantic space*.

Our two objectives are to improve prediction accuracy and to address the problem of data sparsity, since even with a few ratings we might know for example that a user likes comedy or horror movies. This is also the underlying motivation of [1].

In this work, we use a movie database. We extracted the information needed to define the different semantic spaces from the IMDb website[1]. We considered the following semantic spaces: the genre, the actors, and the director. Other information such as title or plot summary for a movie could also be used, but this is beyond the scope of this work, since they do not consist of discrete categories like the ones in the previous list. We also discarded information such as language, because all the rated movies were in English and this does not convey any information.

4 Methodology

In this section, we describe how we construct a semantic space. For simplification purposes, we focus on the movie genre semantic space. Its use for other semantic spaces is straightforward. We then show two ways to integrate our work into the approach of Wang et al. [7].

4.1 Construction of a Semantic Space

In order to construct a semantic space for the genre of a movie, a first intuitive idea is to build two vector spaces for each semantic space (one for the users, one for the items), and define the representation of users and items, as to be able to compute distances in those spaces.

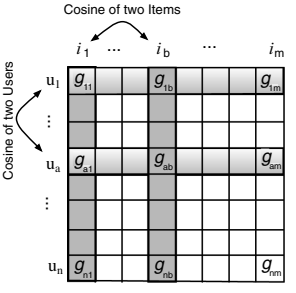
The different genres a movie belongs to can be extracted from IMDb website. Examples of genres are comedy, horror, drama. In order to represent a movie in a vector space, we define a the representation of a movie i as a vector $g_i^{'} = (0, \ldots, 0, 1, 0, 0, 1, \ldots, 1, \ldots)$ where each dimension corresponds to a genre. A component is set to 1 if the movie belongs to this genre, and 0 otherwise. Note that a movie can belong to several different genres.

From the representation of a movie, we construct the vector representing a user by giving more importance to the genres of the movies the user liked, i.e. those for which the rating is high. More formally, we define the vector of a user in the genre space as $g_u^{'} = \sum r_{u,i} g_i^{'}$ where $r_{u,i}$ is the rating of the item i for user u.

It appears to be logical to then use the cosine in the above defined genre space. However, this straightforward approach has some shortcomings related to data sparsity. Consider the case where users will judge similarly items that belong to comedy and to comedy-drama. That is, a given user will either like both genres or dislike both. If this is the case, then considering the defined genre space will not capture this information since two items belonging to respectively comedy and comedy-drama will have a similarity of 0. This might not be a problem with genres since there are not so many of them, but this might prove of more importance with for example actors or directors which are much more numerous and for which data sparsity might have important effects.

[1] The Internet Movie Database (IMDb, http://www.imdb.com/) is an online database of information related to movies, actors, television shows, production, etc.

To capture the interpendance between different genres, we chose to represent users and items in a similarity space. More precisely, users are represented as vectors in a space where each dimension corresponds to an item i and the vector component is the similarity between item i and the user u. The vector can be rewritten for such a user as $g_u = \left(\ldots, \cos\left(g_u^{'}, g_i^{'}\right), \ldots\right)$. We can define a similar vector space for items.



To illustrate the latter step of the space construction, consider the matrix on the left. Rows correspond to users, items to column, and cells to the cosine similarity between a user and an item in the space of $g^{'}$. We use the rows as the vector representation of users, and columns as the vector representation of items, in order to compute the cosine similarity used in the density estimation formulas. Given the cosine distance between two users or two items, we can then construct a distance based on it, defined as $2 - 2\cos(g_u, g_{u'})$ for users and $2 - 2\cos(g_i, g_{i'})$ for items similar to Wang et al. [7].

This method solves the interdependancy problem, i.e. the "comedy versus comedy-drama problem": A comedy movie will be similar to the comedy-drama movie in the newly defined space since *the comedy and the comedy-drama movies are rated similarly by different users*. Symmetrically, two users will be similar in the genre space if they rate similarly the same genres (and not the same items).

4.2 Integrating Semantic Spaces in the Unified Approach

The next step is to use the semantic information, i.e. the distance we defined in the previous section. We consider two methods for combining semantic and rating spaces, namely the Kernel Multiplication and Linear Combination of Predictions. The following two subsections will describe each of these approaches.

Kernel Multiplication. As mentioned in Section 2.2, the rating for a given user and item can be estimated from Formula 1. However, to estimate $P(u, i|r)$ we consider the rating and various semantic spaces. Therefore, we equate $P(u, i|r)$ and $P(r_u, r_i, g_u, g_i|r)$. Similar to the Wang et al. [7] approach, we define a cosine based kernel density estimator. The final rating prediction $\mathbb{E}(r|u, i)$ is defined by:

$$\frac{\sum_{u',i' \in S} r_{u',i'} e^{-\frac{1-\cos(r_u, r_{u'})}{h_{U-R}^2}} e^{-\frac{1-\cos(r_i, r_{i'})}{h_{I-R}^2}} e^{-\frac{1-\cos(g_u, g_{u'})}{h_{U-G}^2}} e^{-\frac{1-\cos(g_i, g_{i'})}{h_{I-G}^2}}}{\sum_{u',i' \in S} e^{-\frac{1-\cos(r_u, r_{u'})}{h_{U-R}^2}} e^{-\frac{1-\cos(r_i, r_{i'})}{h_{I-R}^2}} e^{-\frac{1-\cos(g_u, g_{u'})}{h_{U-G}^2}} e^{-\frac{1-\cos(g_i, g_{i'})}{h_{I-G}^2}}} \tag{4}$$

Linear Combination of Predictions. If we consider the Genre space, for a given user g_u and given Item g_i, the estimation of the rating will be:

$$\mathbb{E}_G(r|u,i) = \sum_{r=1}^{|R|} rP(r|g_i, g_u) \tag{5}$$

This same estimation can be done with rating, actor and director spaces. A linear combination of the estimated rating for a given user and item in each of the Rating, Genre, Actor, and Director spaces is the final estimation.

This approach gave disappointing results, performing consistently worse than the previous one. We therefore do not present the results in the experimental section. We used a naive linear combination approach by considering the steps of 0.1 to create different combinations, and the linear combination seem to give better results when the user/item rating to predict is in a low density region of the space.

5 Experiment

5.1 Test Collection

We performed an evaluation using the Movielens data set, collected by the Grouplens through the Movielens web site [2]. The dataset spans a period from September 1997 to April 1998, and contains ratings of 943 users for 1682 movies (items). Each user has rated at least 20 movies. The rating scale takes values from 1 (the lowest rating) to 5 (the highest rating).

5.2 Evaluation Protocols

Evaluation Procedure. For testing, 400 users from the test collection were selected randomly as training users. The remaining users were assigned to the test set, and used to evaluate models. In the test set, each user's ratings were randomly split into two sets of equal size, one for observed items and the other for held-out items. Held-out items for a user are discarded when predicting, and are only used for evaluation purposes, i.e. to compare the predicting rating with the observed one. The semantic spaces are constructed as explained in Section 4.1. For each space, the user and item bandwidths were calculated using the EM algorithm as described in [7]. We then evaluated the performances of the model for test set users: The ratings for the held-out items were predicted and compared to their real value. To ensure result consistency, we repeated the experiments twenty times for each model, each time with a different training/testing split.

Sparsity. In order to investigate the effect of data sparsity on the performance of our collaborative filtering methods, we randomly removed some ratings from the training (or test) set as described below. We first ensured (*item sparsity*) that each user in the observed set had not rated more than a given number of

items (5, 10, 20, 30 and 40). We then ensured (*user sparsity*) that no item in the held-out set was rated by more than a given number of users (20, 30, 50 and 100) in the observed set With respect to notation, $5- > 20$ means that the user sparsity is 5 and the item spasity is 20.

Complexity. The computational complexities of creating distance matrix, estimating the bandwidth parameter with EM algorithm and predicting the rating are $O(AB^2 + B^2A)$, $O(ABN^2)$ and $O(kN^2)$ respectively where A is the number of items, B is the number of users, k is the number of spaces and N refers to the top-N nearest neighbour. This was derived from estimations given in [7]. In order to decrease the complexity of calculation (Formula 4), we consider the top-50 nearest users for the test user and top-50 nearest items for the test item. For complexity reasons, the distance we used was the distance in the rating space. While it would in theory be better to use the nearest items/users with respect to the kernel K, in practice this gave us good results.

Metrics. We evaluated our approach and hypotheses using the standard metrics of collaborative filtering, namely the mean absolute error MAE and the mean squared error MSE. MAE corresponds to the average absolute deviation between observed and predicted ratings, while MSE corresponds to the average squared deviation. MSE will penalise systems for having a comparatively small number of big deviations rather than those having a big number of small deviations. For both measures a smaller value indicates a better performance. In the formula below N is the number of test ratings, $r_{u,i}$ is the actual rating and $\widehat{r}_{u,i}$ is the estimated rating.

$$MAE = 1/N \times \sum_{u,i} |r_{u,i} - \widehat{r}_{u,i}| \qquad MSE = 1/N \times \sum_{u,i} |r_{u,i} - \widehat{r}_{u,i}|^2 \tag{6}$$

Evaluation Methodology. We exhaustively tested the performance of models with different combinations of kernels associated to the Rating (R), Genre (G), Actor (A), Director (D) spaces. Over the $2^4 - 1$ different possibilities, the RGAD, combinations of all kernels, overall outperformed all other methods.

Results. Table 1 shows the MAE and MSE values for the RGAD and R (baseline) models. We only present the results for the RGAD model as it performed better than any other combination. Those results which performed better than the baseline is presented in bold. The results are consistence because we always have a small variance. Paired t-test was performed on the MAE of the predicted rating for RGAD and the baseline approach for different sparsity combinations, and those combinations which had P-values less than 0.05 were denoted in the table by a star (*).

The main conclusions are that the RGAD model performed consistently better than the R (baseline) model, except for low user sparsity values (i.e. where only a few users judged the item). This result was contrary to our expectations, and shows that the semantic information we extract is also somehow sensitive to sparsity issues.

Table 1. MAE, MSE and rating genre, actor, and director bandwidths for different sparsity configuration (user-item sparsity)

	Baseline (MAE, MSE)	RGAD (MAE, MSE)	h^2_{U-R}, h^2_{I-R}	h^2_{U-G}, h^2_{I-G}	h^2_{U-A}, h^2_{I-A}	h^2_{U-D}, h^2_{I-D}
5->20	3.041, 11.402	3.048, 11.436	0.708, 0.537	0.074, 0.144	0.634, 0.671	0.661, 0.781
5->30	2.958, 10.966	2.965, 10.994	0.798, 0.613	0.076, 0.146	0.692, 0.754	0.729, 0.87
5->50	2.811, 10.2	2.82, 3.078	0.779, 0.619	0.07, 0.138	0.674, 0.749	0.717, 0.877
5->100	2.795, 10.137	2.806, 10.184	0.788, 0.644	0.079, 0.142	0.682, 0.758	0.693, 0.871
10->20	1.307, 3.078	1.313, 3.102	1.523, 1.206	0.083, 0.152	1.171, 1.288	1.268, 1.467
10->30	1.286, 2.967	1.293, 3.004	1.522, 1.235	0.083, 0.149	1.195, 1.298	1.274, 1.472
10->50	1.301, 3.081	1.307, 3.108	1.493, 1.217	0.076, 0.145	1.141, 1.266	1.208, 1.44
10->100	1.3, 2.883	1.308, 2.901	1.47, 1.22	0.077, 0.143	1.121, 1.254	1.201, 1.418
20->20	1.028, 1.828	**1.013, 1.803**	1.71, 1.411	0.071, 0.136	1.127, 1.354	1.198, 1.55
20->30	1.04, 1.898	**1.026, 1.871**	1.693, 1.418	0.067, 0.133	1.085, 1.335	1.158, 1.534
20->50	1.053, 1.974	**1.04, 1.949**	1.67, 1.417	0.063, 0.128	1.054, 1.316	1.182, 1.513
20->100	1.051, 1.976	**1.04, 1.951**	1.635, 1.406	0.062, 0.124	1.014, 1.289	1.142, 1.486
30->20	0.973, 1.586	**0.954*, 1.552**	1.693, 1.418	0.067, 0.133	1.085, 1.335	1.158, 1.534
30->30	0.978, 1.625	**0.96*, 1.588**	1.711, 1.441	0.052, 0.12	0.941, 1.297	1.035, 1.515
30->50	0.978, 1.648	**0.961*, 1.612**	1.693, 1.447	0.052, 0.118	0.91, 1.273	1.055, 1.496
30->100	0.983, 1.687	**0.967*, 1.648**	1.671, 1.44	0.053, 0.118	0.88, 1.256	1.019, 1.476
40->20	0.95, 1.483	**0.928*, 1.441**	1.737, 1.44	0.051, 0.123	0.885, 1.297	1.012, 1.522
40->30	0.948, 1.502	**0.927*, 1.459**	1.718, 1.447	0.048, 0.119	0.844, 1.265	0.98, 1.495
40->50	0.947, 1.525	**1.307*, 0.928**	1.69, 1.448	0.046, 0.114	0.799, 1.238	0.933, 1.473
40->100	0.947, 1.538	**0.929*, 1.495**	1.68, 1.445	0.046, 0.113	0.787, 1.224	0.893, 1.456

Analysing the bandwidths is not easy since they depend strongly on the variation of the distance of the top-50 users/items. However, we note that almost all values (except for rating) increase abruptly when user sparsity is more than 5. An increase means that the neighbouring items or users taken into account is smaller, and hence it means that for a sparsity of 5, there is not enough information for the models to perform well, whatever the space. Moreover, the less sparse the data is, the more the semantic space bandwidths decrease, which means that the rating space alone is enough to predict more and more the ratings. This suggests that semantic information is particularly interesting for a medium sparsity collection.

6 Conclusion

In this paper we investigated the effect of adding semantics to the unified collaborative recommendation model as presented in [7]. We extended this work by taking into account semantic information, such as the genre of a movie. We proposed two methods to include semantic information, by including semantic information directly within the density estimator, and by defining different submodels that we then combine linearly. The analysis of the experiments shows that the kernel combination method performed better than linear combination of prediction. Among the different variants of the first approach, we found that

the approach that integrated all the semantic subspaces (director, genre, actor and rating) was significantly better than the baseline method using only ratings. For future work, we will investigate further methods to exploit semantic information in the cases where data sparsity is high. We will also consider other forms of use of the semantic information: the proposed method considers the rating to be the same if the user/items are similar in *all* spaces. However, it might be interesting to consider models where similarity in *some* spaces only is required.

Acknowledgement

This research was partly supported by the EC under the MIAUCE project (Ref: IST-033715).

References

1. Mobasher, B., Jin, X., Zhou, Y.: Semantically Enhanced Collaborative Filtering on the Web. LNCS, pp. 57–76. Springer, Heidelberg (2004)
2. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: CSCW Conference, pp. 175–186 (1994)
3. Adomavicius, G., Tuzhilin, A.: 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(6), 734–749 (2005)
4. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing 7(1), 76–80 (2003)
5. Miller, B.N., Albert, I., Lam, S.K., Konstan, J.A., Riedl, J.: Movielens unplugged: experiences with an occasionally connected recommender system. In: IUI Conference, pp. 263–266 (2003)
6. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4(2), 133–151 (2001)
7. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unified relevance models for rating prediction in collaborative filtering. TOIS Journal 26(3), 1–42 (2008)
8. Hofmann, T.: Collaborative filtering via gaussian probabilistic latent semantic analysis. In: ACM SIGIR Conference, pp. 259–266 (2003)
9. Bradley, K., Rafter, R., Smyth, B.: Case-based user profiling for content personalisation. In: Brusilovsky, P., Stock, O., Strapparava, C. (eds.) AH 2000. LNCS, vol. 1892, pp. 62–72. Springer, Heidelberg (2000)
10. Farsani, H., Nematbakhsh, M.: A Semantic Recommendation Procedure for Electronic Product Catalog. International Journal of Applied Mathematics and Computer Sciences 3, 86–91 (2006)
11. Antoniou, G., van Harmelen, F.: Web Ontology Language: OWL. Handbook on Ontologies 2, 45–60 (2004)
12. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. TOIS Journal 22(1), 143–177 (2004)
13. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Proceedings of the Eighteenth National Conference on Artificial Intelligence, pp. 187–192 (2002)

14. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: Electronic Commerce Conference, pp. 158–167 (2000)
15. Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: ICML Conference, pp. 713–719 (2005)
16. Huang, Z., Chen, H., Zeng, D.: Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. TOIS Journal 22(1), 116–142 (2004)
17. Xue, G.R., Lin, C., Yang, Q., Xi, W., Zeng, H.J., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: SIGIR Conference, pp. 114–121 (2005)
18. Marlin, B.: Modeling user rating profiles for collaborative filtering. Advances in Neural Information Processing Systems 16, 627–634 (2004)
19. Chien, Y., George, E.: A bayesian model for collaborative filtering. In: Proceedings of the 7 International Workshop on Artificial Intelligence and Statistics (1999)
20. Getoor, L., Sahami, M.: Using probabilistic relational models for collaborative filtering. In: Workshop on Web Usage Analysis and User Profiling (1999)
21. Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: WWW Conference, pp. 285–295 (2001)
22. Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. Computer Linguististic 22(1), 39–71 (1996)
23. OConner, M., Herlocker, J.: Clustering items for collaborative ltering. In: The ACM SIGIR Workshop on Recommender Systems (August 1999)
24. Billsus, D., Pazzani, M.J.: Learning collaborative information filters. In: ICML Conference, pp. 46–54 (1998)
25. Herlocker, J.L.: Understanding and improving automated collaborative filtering systems. Doctoral Thesis, University of Minnesota (2000)