Noname manuscript No.
(will be inserted by the editor)

Bridging Memory-Based Collaborative Filtering and Text Retrieval

Alejandro Bellogín · Jun Wang · Pablo Castells

Received: date / Accepted: date

Abstract When speaking of information retrieval, we often mean text retrieval. But there exist many other forms of information retrieval applications. A typical example is collaborative filtering that suggests *interesting* items to a user by taking into account other users' preferences or tastes. Due to the uniqueness of the problem, it has been modeled and studied differently in the past, mainly drawing from the preference prediction and machine learning view point. A few attempts have yet been made to bring back collaborative filtering to information (text) retrieval modeling and subsequently new interesting collaborative filtering techniques have been thus derived.

In this paper, we show that from the algorithmic view point, there is an even closer relationship between collaborative filtering and text retrieval. Specifically, major collaborative filtering algorithms, such as the memory-based, essentially calculate the dot product between the user vector (as the query vector in text retrieval) and the item rating vector (as the document vector in text retrieval). Thus, if we properly structure user preference data and employ the target user's ratings as query input, major text retrieval algorithms and systems can be *directly* used without any modification. In this regard, we propose a unified formulation under a common notational framework for memory-based collaborative filtering, and a technique to use any text retrieval weighting function with collaborative filtering preference data. Besides confirming the rationale of the framework, our preliminary experimental results have also demonstrated the effectiveness of the approach in using text retrieval models and systems to perform item ranking tasks in collaborative filtering.

Keywords Collaborative filtering · Recommender systems · Text retrieval models

A. Bellogín · P. Castells
Escuela Politécnica Superior, Universidad Autónoma de Madrid

J. Wang
Department of Computer Science, University College London

1 Introduction

In the Information Retrieval (IR) community, information retrieval often means text retrieval by default, either intentionally or unintentionally. This might be due to historical reasons (Singhal 2001) or simply because text retrieval has been the most predominant information retrieval application. But, nonetheless, there exist many other forms of information retrieval applications. A typical example is collaborative filtering, which aims at finding information items a target user is likely to *like* by taking into account other users' preferences or tastes. Unlike text retrieval, collaborative filtering (CF) does not necessarily need textual descriptions of information items and user needs. It makes personalized recommendations by aggregating the opinions and preferences of previous users. Originally, the idea of collaborative filtering was derived from heuristics and coined by Goldberg et. al when developing an automatic filtering system for electronic mail (Goldberg et al 1992). Due to the uniqueness of the problem, it has been modeled and studied differently since then, mainly drawing from the preference prediction and machine learning view point (Breese et al 1998).

Despite the differences, the underlying goal of information filtering (and thus, collaborative filtering) is, nonetheless, essentially equivalent to other information retrieval tasks. They are all aimed at fulfilling user information needs (Belkin and Croft 1992). In fact, there are well known links between content-based recommender systems and text retrieval. In content-based recommender systems, information items are represented by a set of features. Usually these features are textual, such as tags or genres in movie recommendation, news content and URLs in news recommendation, and so on (Adomavicius and Tuzhilin 2005). Then, the match between a user profile and documents can be established by the techniques employed in text retrieval (Xu et al 2008).

Unfortunately, less research has been conducted to establish the possible equivalences between not only the task principles but also structures used in information retrieval and those in collaborative filtering. In (Wang et al 2008) and (Wang et al 2006), the authors find interesting analogies between implicit (frequency-based) CF and IR, introducing the concept of binary relevance into CF and applying the Probability Ranking Principle of IR to it. But the attempts at unification have been partial, and are focused purely on the model level. System level comparison and unification have rarely been studied. Moreover, standard neighborhood-based CF formulations (as the main class of memory-based algorithms in CF) are not yet fully integrated within the IR theories, and, thus, we cannot interpret them by means of IR techniques or methods (Desrosiers and Karypis 2011).

In this paper, we address those issues by taking both the algorithmic and systematic view points. We show that there is a close relationship between the two formulations. We found that the major collaborative filtering algorithms, such as the memory-based, essentially employ the same algorithmic framework as those of text retrieval. More specifically, to predict a user's preference, the memory-based collaborative filtering essentially calculates the dot product between the user vector (similar to the query in text retrieval) and the item vector (similar to the document vector as in text retrieval) –the vector elements either be similarity measures, ratings or some other weights depending on the specific weighting scheme that has been employed, as we shall describe in Section 3.3. In this regard, the main contribution of our work is a common notational framework for IR and memory-

based CF, which enables to define a technique to use any text retrieval weighting function with CF preference data. We demonstrate that if we properly structure user preference data and use the target user's ratings as query input, major text retrieval algorithms and systems, such as the *Terrier*[1] and *Lemur*[2] IR platforms, can be *directly* used for the collaborative filtering task without any modification.

Moreover, based on this unified framework, we evaluate how well IR methods perform against standard CF techniques in two different recommendation tasks, namely, item ranking (returning a ranking for each user) and rating prediction (predicting the rating for each user-item pair). We believe a distinction between these two tasks is needed because they have different requirements, for example, in the item ranking task the score predicted by the algorithm does not have to live in any specific interval, while in the rating prediction task the final performance depends heavily on the output range of the recommender. Our experiments show that IR methods perform particularly well in the item ranking task (which is equivalent to the classic ad-hoc IR task), while they are competitive in the rating prediction, obtaining results statistically equivalent to those of the state-of-the-art.

The remainder of the paper is organized as follows: we begin by placing our work into the broader context of collaborative filtering and information retrieval models in Section 2. We then formalize in Section 3 a new algorithmic framework of collaborative filtering and use it to build up a recommendation system where the state-of-the-art retrieval algorithms can be employed. We then provide an empirical evaluation for the item ranking and rating prediction tasks in Section 4, and, finally, Section 5 concludes our work by pointing out the future directions.

2 Related Work

Text retrieval techniques have been widely used in many different areas such as web retrieval (Agichtein et al 2006), image and video retrieval (Sivic and Zisserman 2003), or content-based recommendation (Xu et al 2008). Among many techniques, one of the most commonly used are the term weighting function: the system indexes documents for a vector representation of the documents and queries in the term space (Salton et al 1975). The vector space model (VSM) measures the similarity of a document with regard to the query as the similarity between the two vectors, using for instance the cosine similarity. The simplicity and flexibility of this model have resulted in many different applications. Probabilistic models have also been proposed to calculate the probability of relevance using document terms (Robertson and Sparck Jones 1988), while others estimate text statistics in the documents and queries and then build up the term weighting function through them (Amati and Van Rijsbergen 2002). In fact, it has been shown that most IR models can be unified and represented as a parametric term weighting form (Metzler and Zaragoza 2009).

More or less in parallel, collaborative filtering (CF) techniques were independently developed (Breese et al 1998). CF algorithms can be grouped into two general classes: the memory-based (or heuristic-based) and the model-based. The model-based approaches use the collection of ratings to learn a model, which is then

[1] http://terrier.org/

[2] http://www.lemurproject.org/

used to make rating predictions. Different approaches have been proposed, based on methods such as probabilistic CF (Yu et al 2004), neural networks (Pazzani and Billsus 1997), Markov chains (Salakhutdinov and Mnih 2008), or maximum entropy models (Pavlov et al 2004). Recently, mixture models proved to be effective in terms of reducing the RMSE (Rooted Mean Squared Error) as shown in the Netflix competition (Takács et al 2008). Examples include probabilistic Latent Semantic Indexing (Hofmann 2004) and Matrix Factorization (MF) methods (Koren et al 2009).

We focus here on the memory-based algorithms as they are heavily used in practice. They predict ratings based on the entire collection of previously rated items by the user. This type of recommenders typically predict the votes of the active user based on some partial information regarding that user and a set of weights calculated from the entire dataset, where they can reflect distance, correlation, or similarity. This can be done in two ways, known as user-based or item-based recommendation, depending on how the *neighborhood* is defined, as a set of users or items very similar to the target user (or item). Neighborhood-based CF methods typically compute the predicted ratings by using a weighted average between the neighbor's rating and the neighbor's similarity with respect to the target user (or item) (Adomavicius and Tuzhilin 2005).

Connections to IR model elements can be already found in the state-of-the-art of CF approaches. Breese et al (1998) use the IR cosine formula as the similarity function, according to how they rate, as a special case of vector similarity. This can be seen as a first attempt to relate memory-based CF with IR, though it is only valid for the user-based approach. More importantly, in (Soboroff and Nicholas 2000), the authors present a formal relationship between neighborhood-based CF and text retrieval, by using the generalized vector space model. They, however, do not evaluate or implement their approach, and besides, although the model proposed could make use of the rating matrix, they suggest to use content-based profiles in order to overcome sparsity, and thus, this model would not rely only on explicit ratings. Finally, another problem in which IR and CF have been identified and techniques from the former have been used in the latter is the scalability problem: in (Cöster and Svensson 2002), the authors propose the use of *inverted indexes* for CF in order to speed up the execution time. In that work, different heuristics are used in order to efficiently search in a user's neighborhood; in this way, memory-based algorithms can perform fast rating predictions by using disk based inverted files. Despite this identification, no formal theory is presented, and the authors simply cast the CF problem into an IR setting. A similar approach applied to personalized IR is proposed in (Pickens et al 2010). Furthermore, in (Cohen and Lewis 1999), the use of inverted files is able to accelerates any application involving (sparse) matrix multiplications, which is very common in recommendation techniques such as MF.

Other approaches, such as (Wang et al 2008) and (Wang 2009) have presented analogies between frequency-based CF and IR models. For instance, in (Wang et al 2008) the authors introduce the concept of binary relevance into CF and apply the Probability Ranking Principle of IR to it. Wang (2009) adapts the language modeling framework to frequency-based CF, introducing a risk-averse model that penalizes the less reliable scores. However, none of these approaches can be easily translated from frequency-based to memory-based CF, which is typically based on ratings. No definitive method exists for transforming implicit preference data into

explicit ratings, so it is unclear whether this mapping can make any sense (Hu et al 2008), since they inherently represent different information gathered from the user (for example, negative preferences can only be elicited using explicit data).

Additionally, Deshpande and Karypis (2004) use a linear algebra formulation to describe their item-item collaborative filtering algorithm, an approach related with the algebraic framework proposed in Section 3.2. Moreover, Rölleke et al (2006) describe several concepts and retrieval models from IR in a general matrix framework, where the main operation is the dot product. As we have mentioned before, we obtain an equivalent algorithmic framework between memory-based CF and IR models by means of the dot product function. Our approach could be considered as an extension to the field of collaborative filtering of that work, where personalized IR were also considered as well as algebraic formalizations.

3 A New Algorithmic Framework

We can obtain a new insight into the current collaborative filtering approaches by reformulating them using a vector space model (Salton et al 1975). To do that, we need to acknowledge the fact that the input data and the immediate goal are different: in IR, we have queries and documents represented by terms; whereas in memory-based CF, we have a set of ratings, which are used to infer the preferences of users towards items. Apart from that, the objective of IR systems is to generate a document ranking for each query, while, in general, standard CF models tend to predict a rating as *accurately* as possible.

3.1 Representing items and users in a common space

Formally, the user's ratings can be regarded as a query. Each of the rated items may be considered as a query term. In the next sections, we propose how to represent rating information into different spaces, depending on the elementary components serving as a basis for the space: items or users.

3.1.1 Item space

If we have a space with as many dimensions as items, we could use the following vector to represent a user:

$$\mathbf{u}_I = (r_{u1}, \cdots, r_{uk}, \cdots, r_{un}) \tag{1}$$

where $\mathbf{u}_I$ denotes user's preferences in the item space. We use subscript I to indicate this user is in the item space. The value r_{uk} represents user $\mathbf{u}$'s rating of item[3] k, where $k \in \{1, n\}$. It is equal to 0 when item k is not rated by the user, or any other value not within the natural range of ratings in the system. Thus, if R is the maximum rating value, each coordinate of these vectors live in the interval $[1, R]$, and thus, $\mathbf{u}_I \in [1, R]^n$. In practice, the ratings are usually normalized with

[3] For the sake of clarity, when the candidate user or item is ambiguous, we will note it as $\mathbf{u^k}$ or $\mathbf{i^k}$, denoting we are referring to the user or item k in the collection; otherwise, we will simply represent it as $\mathbf{u}$ or $\mathbf{i}$.

respect to the users' or items' mean, which would increase the computational complexity of the algorithm but, at the same time, such normalization would allow the method to predict a default rating for any user and item (alleviating in this way the sparsity problem).

By contrast, the item representation is different from the typical representation in text retrieval, which is easily represented using the same space as the query, i.e., the term space. In collaborative filtering we do not have a common feature space and a reference dimension should be selected. In this case, taking into account Eq.(1), the equivalence holds if we project an item into the same feature space as users (queries) by using its similar items. That is,

$$\mathbf{i}_I = (s_{i1}, \cdots, s_{ik}, \cdots, s_{in}) \tag{2}$$

where s_{ik} is the similarity between the candidate item $\mathbf{i}$ and item k. Since similarities usually range in the interval $[-1, 1]$, then these vectors would fulfil $\mathbf{i}_I \in [-1, 1]^n$. In practice, it has proven useful to select top-N similar items as opposed to using all the similar items. We have to note that the use of top-N similar items simply would produce more sparse item (or similarity) vectors, but user and item vectors would still be of equal length, since they are represented in the item space.

Using item-item similarities as the item coordinates in the item-based vector space produces an equivalence between the memory-based rating estimation in CF and a common retrieval function in the vector space IR model, as we will see next. Not only does the similarity-based projection find motivation in just getting this equivalence, but we shall show in Section 3.2 that there is a formal algebraic basis to this representation.

Now, let us see how the retrieval function equivalence is derived. Given the vector representations of Eq.(1) and Eq.(2), a query-document (user-item) similarity value can be obtained by comparing the corresponding vectors, using for example the conventional inner product:

$$\text{Score}(\mathbf{u}_I, \mathbf{i}_I) = \sum_k r_{uk} \cdot s_{ik} \tag{3}$$

The system can produce a ranked recommendation output by the decreasing order of the above retrieval function between $\mathbf{u}_I$ and $\mathbf{i}_I$, providing an answer to the item ranking task. For the rating prediction task, the similarity score can be normalized to produce the rating prediction for an unspecified item.

$$\hat{r}(u, i) = \frac{\sum_{k=1}^{n} r_{uk} \cdot s_{ik}}{\sum_{\forall k: r_{uk} \neq 0} s_{ik}} = \frac{\text{Score}(\mathbf{u}_I, \mathbf{i}_I)}{\sum_{\forall k: r_{uk} \neq 0} s_{ik}} = \frac{\text{Score}(\mathbf{u}_I, \mathbf{i}_I)}{\text{Score}\left(\delta\left(\mathbf{u}_I\right), \mathbf{i}_I\right)} \tag{4}$$

where function $\delta\left(\mathbf{u}_I\right)$ is a binary version of the user $\mathbf{u}_I$. It can be easily seen that Eq.(4) is indeed an item-based collaborative filtering approach, but rather in a vector space formulation.

3.1.2 User space

In the previous section, we have shown that it is possible to represent users and items in an item space. Similarly, we can do the same but in the user space. Let

us define now the user's preference vector as follows:

$$\mathbf{u}_U = (s_{u1}, \cdots, s_{uk}, \cdots, s_{um}) \tag{5}$$

the value s_{uk} in this situation represents the similarity between the current user $\mathbf{u}$ and user k. As before, typically only the top-N similar users are considered. Likewise subscript U indicates it is defined in a common space of users. In contrast with the user vectors presented in the previous section, now we would have $\mathbf{u}_U \in [-1, 1]^m$. Again, Section 3.2 will provide an algebraic generalization which particularizes to the expression above in order to project users into the user-based vector space.

Items are projected in this user space by considering how each user (one for each dimension) rated each item:

$$\mathbf{i}_U = (r_{1i}, \cdots, r_{ki}, \cdots, r_{mi}) \tag{6}$$

Thus, $\mathbf{i}_U \in [1, R]^m$. In order to compute rating predictions given a specific user-item pair, we need to normalize their dot-product properly, as follows:

$$\hat{r}(u,i) = \frac{\sum_{k=1}^{m} r_{ki} \cdot s_{uk}}{\sum_{\forall k: r_{ki} \neq 0} s_{uk}} = \frac{\text{Score}(\mathbf{u}_U, \mathbf{i}_U)}{\sum_{\forall k: r_{ki} \neq 0} s_{uk}} = \frac{\text{Score}(\mathbf{u}_U, \mathbf{i}_U)}{\text{Score}\left(\mathbf{u}_U, \delta\left(\mathbf{i}_U\right)\right)} \tag{7}$$

As in the previous case, we can see that Eq.(7) is indeed a user-based collaborative filtering approach, but in a vector space formulation. Again, if we just want to provide a ranked recommendation, it would be sufficient to generate a ranking by decreasing order of the function $\text{Score}(\mathbf{u}_U, \mathbf{i}_U)$ for all different $\mathbf{i}_U$ in the system (not previously evaluated by the user).

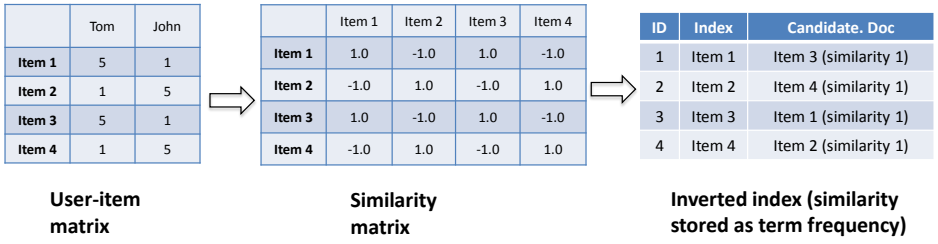
Usually, the ratings are normalized with respect to the users' mean (Adomavicius and Tuzhilin 2005). Equivalences with this kind of techniques can also be found. We first need a vector of user's rating averages, i.e. $\mathbf{m} = (\bar{r}_1, \cdots, \bar{r}_k, \cdots, \bar{r}_m)$, where $\bar{r}_k$ is the average rating of user k. Thus, the predicted rating can be calculated using a slight modification of the previous formula for only considering known rating values:

$$\hat{r}(u,i) = \bar{r}_u + \frac{\text{Score}(\mathbf{u}_U, \mathbf{i}_U - \mathbf{m_i})}{\text{Score}\left(\mathbf{u}_U, \delta\left(\mathbf{i}_U\right)\right)} = \bar{r}_u + \frac{\sum_{\forall k: r_{ki} \neq 0} s_{uk} \cdot (r_{ki} - \bar{r}_k)}{\sum_{\forall k: r_{ki} \neq 0} s_{uk}}$$

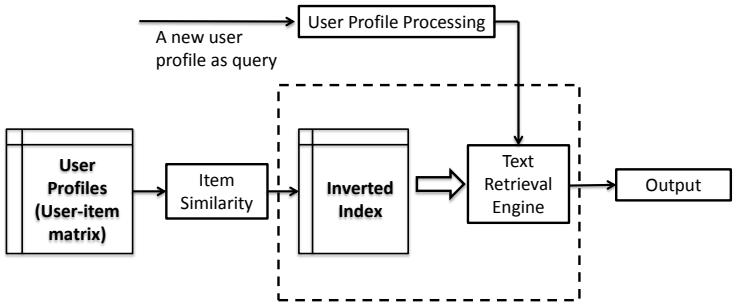
where $\mathbf{m_i}$ is a transformed version of vector $\mathbf{m}$ that is zero wherever $\mathbf{i}$ contains an unknown rating value.

3.2 Algebraic generalization

As mentioned in the previous sections, we now show that item-item and user-user similarity arise as special cases of a more general way to represent items in the item-based vector space and users in the user-based vector space, respectively. In this section, we provide a formal motivation from linear algebra in which different recommender formulations fit as an IR VSM retrieval function (the dot product), such as memory-based CF and matrix factorization methods.



(a) Similarity stored in inverted index



(b) The algorithmic framework

Fig. 1 Collaborative filtering as text retrieval.

First, let us assume we have an orthonormal basis in a common vector space of dimension l in which we can express users and items, that is, $\exists\{e_i\}_1^l$ such that

$$\mathbf{u^v} = \lambda_1^v e_1 + \cdots + \lambda_l^v e_l = (\lambda_1^v, \cdots, \lambda_l^v)$$
$$\mathbf{i^j} = \mu_1^j e_1 + \cdots + \mu_l^j e_l = (\mu_1^j, \cdots, \mu_l^j)$$

Now, we take matrix $X = (r_{vj})$ as the rating matrix (where unknown values are, as before, considered zero), and r_{vj} as the rating given by user $\mathbf{u^v}$ towards item $\mathbf{i^j}$. Let us take the space of items as the common space; an analogous formulation can be similarly derived from the user space and will be discussed later. Although the item space is not generally orthonormal, we now present a method to obtain an orthonormal basis by means of the spectral theorem.

In this situation, the item covariance matrix is defined as $C_I = cov(X)$, since each column (items) is considered as the random variable. More specifically, the element a_{ij} of that matrix is $a_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)]$ where X_i denotes the i-th vector (related with item $\mathbf{i}$) and $\mu_i = E(X_i)$. Since the covariance matrix is a positive semidefinite, symmetric matrix, the spectral theorem grants that there is an orthonormal basis of eigenvectors of dimension n of C_I. This yields the following basis for items: $\mathbf{i^j} = \mu_1^j e_1 + \cdots + \mu_n^j e_n = (\mu_1^j, \cdots, \mu_n^j)$. In order to obtain a user representation in this space, we make use of the following identity in the original item space, $\mathbf{u^v} = r_{v1}\mathbf{i^1} + \cdots + r_{vn}\mathbf{i^n}$. We can then replace the previously obtained representations for items as follows:

$$\begin{aligned}\mathbf{u^v} &= r_{v1}\mathbf{i^1} + \cdots + r_{vn}\mathbf{i^n} = r_{v1}\sum_j \mu_j^1 e_j + \cdots + r_{vn}\sum_j \mu_j^n e_j \\ &= (r_{v1}\mu_1^1 + \cdots + r_{vn}\mu_1^n, \cdots, r_{v1}\mu_n^1 + \cdots + r_{vn}\mu_n^n) \\ &= (r_{v1} + \cdots + r_{vn}) \cdot C_I\end{aligned}$$

As we can observe from the previous equations, C_I is indeed the change of basis matrix from the user space to the item space. Moreover, the matrix $\hat{X} = X \cdot C_I$ provides the new representations of each user in the item space.

Furthermore, in this common space we can compare user vectors against item vectors using distances based on, for example, the cosine, the Euclidean distance, or the inner (dot) product of those vectors, as follows:

$$\mathbf{u^v} \cdot \mathbf{i^j} = \sum_{p=1}^{n} \mu_p^j (r_{v1}\mu_p^1 + \cdots + r_{vn}\mu_p^n) = \sum_{p=1}^{n} \mu_p^j \sum_{k=1}^{n} r_{vk}\mu_p^k = \sum_{k=1}^{n} r_{vk} \sum_{p=1}^{n} \mu_p^j \mu_p^k \quad (8)$$

The distance between the target user and item could, hence, be used as the retrieval system score for this user-item pair. Indeed, if we encapsulate $\sum_{p=1}^{n} \mu_p^j \mu_p^k$ as s_{jk}, then, we obtain an equivalent formulation to the one presented in Eq.(3), that is, an item-based CF. Note that the similarity term defined in this way would be equivalent to the inner product between the vectors representing both items. Furthermore, if the original rating matrix (X) is pre-processed appropriately, the term s_{jk} can be made equivalent to the cosine distance —by normalizing the item vectors to norm 1— or to the Pearson correlation —by centering item ratings on their mean value, and normalizing the item vectors. However, we have to note that this pre-processing could be computationally expensive.

The derivations in this section thus show that the formulations proposed in Section 3.1.1 and Section 3.1.2 are special cases of a more general algebraic formulation where users and items are represented in the same space. When the common space is the space of items, it has as many dimensions (n) as items in the collection. If the user space is taken instead, an m-dimensional space would be derived by using $C_U = cov(X^{\mathrm{T}})$ instead of C_I at the beginning of the procedure, where C_U is the user covariance matrix and X^{T} is the transpose matrix.

It is also interesting to highlight how this procedure may introduce the notion of subspace in CF. This concept arises, for instance, when users and items are represented in an item-space and $n < m$, i.e. there are less items than users. In this case, there must be *some linear dependency* between users in terms of the item space components, although this situation may occur also when $n \geq m$. It has been proposed recently (Desrosiers and Karypis 2011) that, when there are less items than users (i.e., such a subspace must exist), item-based algorithms tend to perform better, in general, than user-based methods. Similarly, when users and items are represented in the user space and there are less users than items, linear dependencies occur between items in terms of user-based coordinates. In the same way, in the latter situation, user-based algorithms are generally more effective than item-based CF (Desrosiers and Karypis 2011). These results may suggest that the existence of a subspace, and in particular, the dimension of such subspace, is key for the performance of memory-based algorithms, since the more linearly dependent

elements exist in the vector space, more information would be contained in the (user or item) covariance matrix, and thus, the better are the recommendations generated by the algorithm.

Finally, note that this is not the first attempt to use algebraic formulation in CF, like the item-based approach described in (Deshpande and Karypis 2004). Besides, in the IR literature we can find theoretical methods where queries and items are formally represented in the same space. Wong et al (1985) propose a generalized vector space model, and Demartini et al (2009) introduces a common space for entities and documents in the context of expert finding and entity ranking. Additionally, the CF literature also provides alternative approaches to represent users and items in a common space, namely as a space of latent features and the use of matrix factorization techniques (Hofmann 2004; Koren et al 2009). Therefore, the formulation presented in Eq.(8) could be further generalized by assuming that users and items live in an l-dimensional common space; in that way,

$$\mathbf{u^v} \cdot \mathbf{i^j} = \sum_{p=1}^{n} \mu_p^j (r_{v1}\mu_p^1 + \cdots + r_{vn}\mu_p^l) = \sum_{p=1}^{l} \mu_p^j \sum_{k=1}^{n} r_{vk}\mu_p^k \tag{9}$$

This generalization would provide an alternative basis to draw equivalences to the vector space model. However, these different techniques introduce additional non-trivial operations in the formulation (the ones involved in factorization and latent analysis, which in turn lend themselves to alternative instantiations and variants), whereby the equivalence is less direct and straightforward. Nonetheless, from that point on, the remainder of our formulation would be applicable: the user and item matrices produced by matrix factorization are the projection matrices (providing thus the λ's and μ's). Hence, it is worth highlighting that both memory-based and matrix factorization CF approaches fit nicely in the same framework, the difference being the transformation matrices: in memory-based CF, these are the covariance matrix and the ratings matrix; in matrix factorization CF, they are the item and user matrices resulting from the factorization.

In particular, specific adaptations to the approaches found in the literature could be derived. For instance, the standard rating prediction computation using matrix factorization techniques (Koren et al 2009) is based on the product of some projection matrices. Using the above formulation and the one presented in (Koren et al 2009), the p-th component of the vector product $q_j^T p_v$ (where the vectors q_i and p_u are associated with the item i and user u) can be derived as an instantiation of Eq.(9) by computing $\mu_p^j r_{vk} \mu_p^k$. In this situation, the l dimensions of the space represent the dimensionality of the latent factor space.

Furthermore, the latent semantic model proposed in (Hofmann 2004) could also be seen as another example of this formulation, by defining μ_p^j as $p(i_j|z_p)$ (according to the notation used in that paper) and $\sum_{k=1}^{n} r_{vk}\mu_p^k$ as $p(z_p|u_v)p(u_v)$. Here, the l dimensions represent the different hidden variables z introduced in the model.

3.3 Indexing and retrieval

From an information retrieval view point, the candidate item can be regarded as a document; and all its similar items construct its document representation,

Table 1 Different weighting schemes used in Information Retrieval. $\mathrm{qf}(t)$ denotes the frequency of term t in the query, $\mathrm{tf}(t,d)$ the frequency in document d, N is the number of documents, $\mathrm{df}(t)$ is the number of documents that contain t, $\mathrm{dl}(d)$ is the document length of d, and $\bar{\mathrm{dl}}$ is the average document length. Besides, k_1, k_3, b, λ,, and μ are parameters of the different models.

Method	w_t^q	w_t^d
Binary	1 if $t \in q$	1 if $t \in d$
TF	$\mathrm{qf}(t)$	$\mathrm{tf}(t,d)$
TF-IDF	$\mathrm{qf}(t)$	$(1+\log(\mathrm{tf}(t,d)))\log\left(\frac{N}{\mathrm{df}(t)}\right)$
BM25	$\frac{(k_3+1)\,\mathrm{qf}(t)}{k_3+\mathrm{qf}(t)}$	$\log\left(\frac{N-\mathrm{df}(t)}{\mathrm{df}(t)}\right)\frac{(k_1+1)\,\mathrm{tf}(t,d)}{k_1\left((1-b)+b\cdot\mathrm{dl}(d)/\bar{\mathrm{dl}}\right)+\mathrm{tf}(t,d)}$
Language Model (Jelinek-Mercer)	$qf(t)$	$(1-\lambda)p(t\|d)+\lambda p(t\|\mathcal{C})$
Language Model (Dirichlet)	$qf(t)$	$\frac{tf}{\mathrm{dl}(d)+\mu}+\mu\frac{p(t\|\mathcal{C})}{\mathrm{dl}(d)+\mu}$

Table 2 Weighting schemes under the unified framework for item-based CF. The rating from the (query) user u is denoted as r_{uk}, the similarity between the target item and item k is s_{ik}, N is the number of items, N_k is the number of items similar to item k, $\mathrm{il}(i)$ is the number of similar items of the target item, and $\bar{\mathrm{il}}$ is the average il.

Method	w_k^u	w_k^i
Binary	1 if rated	1 if similar
TF	r_{uk}	s_{ik}
TF-IDF	r_{uk}	$s_{ik}\log\left(\frac{N}{N_k}\right)$
BM25	$\frac{(k_3+1)r_{uk}}{k_3+r_{uk}}$	$\log\left(\frac{N-N_k}{N_k}\right)\frac{(k_1+1)s_{ik}}{k_1\left((1-b)+b\cdot\mathrm{il}(i)/\bar{\mathrm{il}}\right)+s_{ik}}$
Language Model (Jelinek-Mercer)	r_{uk}	$(1-\lambda)p(k\|i)+\lambda p(k\|\mathcal{C})$
Language Model (Dirichlet)	r_{uk}	$\frac{s_{ik}}{\mathrm{il}(i)+\mu}+\mu\frac{p(k\|\mathcal{C})}{\mathrm{il}(i)+\mu}$

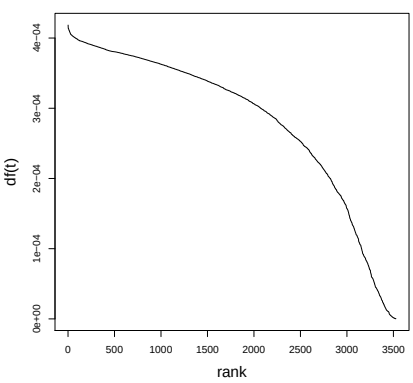
as shown in Eq.(3). In text retrieval, term frequency (TF) is the basic element, whereas in memory-based collaborative filtering, and in particular, in item-based CF, similarity is employed for document representation and ratings are used for query representation –in the user space, the situation is obviously the other way around. Thus, the standard item-based scoring approach in Eq.(3) is, *computationally*, equivalent to the simplest TF text retrieval model. It should be emphasized that the underlying physical meaning of term frequency and item similarity is different –in particular, they are not supposed to represent the same concept, since the equivalence is at the formal level– and their discussions are given in Section 3.4.

It would be, thus, straightforward to employ any text retrieval engine/system to index and perform a collaborative filtering task. It will immediately make many information retrieval techniques ready to use. An algorithmic framework of indexing and retrieval for collaborative filtering is presented in Figure 1. Note that the figure illustrates the item space representation, while the procedure for the user space representation can analogically be derived.

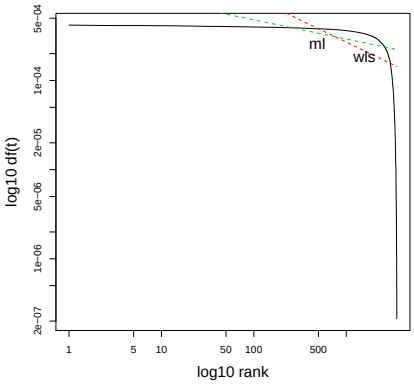
More specifically, we can incorporate the formulation used in (Metzler and Zaragoza 2009) and generalize the score function presented in Eq.(3) as follows:

$$\mathrm{Score}(\mathbf{u},\mathbf{i}) = \sum_k w_k^u \cdot w_k^i \tag{10}$$

Hence, we can use any IR retrieval model as a memory-based collaborative filtering technique by using any weighting scheme among those defined in Table 2,



(a) Raw similarity distribution where the frequency is plotted as a function of frequency rank for the items (terms) in the system.



(b) Zipf fit using weighted least-squares (wls) and maximum likelihood (ml) estimators. Both axis are in logarithmic scale.

Fig. 2 Distribution of similar items in Collaborative Filtering.

where $\text{qf}(t)$ has been replaced from the original IR models (Table 1) with r_{uk}, and $\text{tf}(t,d)$ with s_{ik}[4]. One difference, however, lies in the problem of memory-based CF where a normalization is needed to obtain the rating prediction, as shown in Eq.(4). The prediction task is done by querying any text retrieval engine twice. Firstly, obtaining $\text{Score}(\mathbf{u},\mathbf{i})$ as in Eq.(4) by using $\mathbf{u}$ as a query. Secondly, obtaining $\text{Score}(\delta(\mathbf{u}),\mathbf{i})$ by querying the binary query $\delta(\mathbf{u})$. It should be noted, however, that both scores can be computed algorithmically at the same time, with no added cost.

In the following paragraphs, we present how we can make use of such framework, as well as the analogies and differences with respect to text retrieval. Although more complex uses of inverted index in CF can be done, like in (Cöster and Svensson 2002), we want to focus on presenting the basic equivalences between our framework and text retrieval. Actually, the equivalence in representation presented above induces an equivalence when creating the inverted index. It is important to note that, however, the distribution of similar terms does not follow a Zipf distribution as it is usually assumed in IR (Manning et al 2008). We present in Figure 2 the distribution of similar items (terms in the item space representation) in a standard CF collection. We can observe how this distribution is clearly different from a Zipfian distribution, which has implications for the way the queries are processed (probably more accesses to the index are required for each query). Besides, it could also affect the index compression effectiveness, although this should not be critical because the vocabulary size in CF (number of items or users, depending on the representation) is much smaller than that in IR (Manning et al 2008).

In general, an inverted index in text retrieval is composed of a *vocabulary* and its occurrences or *postings*, that is, the list of all the texts where each word or term

[4] We write the TF-IDF equation where a logarithm is applied to the term frequency because it helps to reduce the impact of the very frequent terms (Croft et al 2009). However, since similarities and ratings are not large numbers, we have decided to keep its original formulation in Table 2 (Baeza-Yates and Ribeiro-Neto 1999).

appeared (Baeza-Yates and Ribeiro-Neto 1999); additionally, the term frequency in each document can also be stored. In our framework these basic structures can be interpreted in the CF space as follows. For each item, **i** (document) we store the items (now acting as terms) most similar to **i**. Since in our model, term frequencies are equivalent to the similarity values between items, it makes sense to store them also in the inverted index, as shown in Figure 1a.

Once the inverted index has been built, a typical text retrieval engine needs two more operations in order to generate a document ranking for a particular query. These operations are called query transformation and query processing (or scoring) (Manning et al 2008). As we can see in Figure 1b, we perform a separate process of the user's preferences, similar to the query transformation in text retrieval, in a user basis. In this process, and different from what is typical in text retrieval, we cannot perform the same operation as in documents, despite the fact they live in the same space, since the components of the user and item vector have different semantics, and thus, they have to be computed separately. However, once this process has been carried out, the standard scoring can be performed: for each term in the query, the system retrieves all the documents containing that term, computes a score for that term and that document, and repeats for the rest of documents. In this critical step, the system can weight the importance of each term in a document (with respect to the query) using several models, some of them are presented in Table 1. In our case, since we have encoded the similarities as term frequencies and the ratings as query frequencies, this process can be performed with no difference with respect to the text retrieval process, i.e., any actual text retrieval engine could be used based on the term and query frequencies defined above.

3.4 Discussion

The formulations proposed in previous sections allow us to represent users and items in the same space, as Salton did in (Salton et al 1975) with the vector space model, where documents and queries are defined as term vectors, and each component encodes the frequency of that term in a particular document or query. We discuss here some fine details and variants in the fundamental terms and model components, such as the cosine retrieval function, IDF, and BM25 parameters.

Although the cosine formula is usually normalized using the norms of both vectors, the score is rank-equivalent if the query norm is omitted. Besides that, many other different weighting functions of IR can be used such as those summarized in Table 1. This allows us to fully take advantage of the techniques and insights obtained from the IR research. As shown in Table 2, we establish a correspondence between the TF technique in IR and the basic item-based collaborative filtering. Moreover, while the CF technique based on language models is used to smooth the similarity, the new CF models based on TF-IDF and BM25 explore the discriminative power of an item, since the item's similarity is penalized if it is similar to many items.

It should be noticed though that some specific parameters may not be necessarily suitable for the collaborative filtering problem. For instance, the k_1 and k_3 parameters in BM25 introduce a nonlinear function and are designed to deal with the term frequency –by reducing the effect of a highly frequent term. In collabo-

rative filtering, these parameters work well for frequency data (Wang et al 2008), but they may not be suitable for rating data. For rating data, a larger value could be chosen to remove this effect as follows

$$\frac{(k_3+1)r_k}{k_3+r_k} = \frac{r_k}{\frac{k_3}{k_3+1}+\frac{r_k}{k_3+1}} \approx r_k \text{ when } (k_3+1) >> r_k$$

When such a value is selected for parameter k_3, the BM25 method is equivalent to the rest of methods presented in Table 2 for weight w_k^u (i.e., the query vector in the item space representation); however, we prefer to maintain the original formulation to allow for further derivations where such parameter could be exploited. In Section 4.3.1, we will analyze in more detail the rest of the parameters of the BM25 function, namely, k_1 and b.

Inverse Document Frequency (IDF) is a very important function in IR (Baeza-Yates and Ribeiro-Neto 1999). The IDF of a term is based on counting the number of documents which contain that particular term. Different works have tried to relate it with information theory metrics (Aizawa 2003) or different probabilistic theories (Rölleke and Wang 2008). The intuition behind this function is that a term which occurs in many documents is not a good discriminator, and should be given less weight (Spärck Jones 1972). In its original formulation, it was a measure of the statistical specificity of the term.

In rating-based collaborative filtering, however, this function has been barely used. In (Breese et al 1998), the Inverse User Frequency function appears in order to not to take into account universally liked items when computing similarities between users. Authors report an average improvement of 2.4%, however other authors indicate that the use of this weighting factor degrades performance (Jin et al 2004). Herlocker et al (1999) introduces a similar weighting factor in the similarity function, which takes into account the variance in ratings of the items, however no significant effect on accuracy was found.

In our framework, the use of IDF appears in a natural way, and therefore we can check whether it is so beneficial for recommendation as in IR. Specifically, in the item space model it represents that items similar to many items should be penalized. In the user space, on the other hand, a user obtains a higher IDF when she has rated a lot of items, which a priori can be seem as counterintuitive, since the more ratings the system has from a user, the more information, and better defined, the user is; however, it may also be interpreted as a user more likely to have inconsistent information (noise) in her preferences, and thus, it would be interesting to dampen the influence of such kind of users.

It is interesting to note that, in our model, we do not obtain an equivalence with the Inverse User Frequency mentioned above, since in that paper the authors adopted IR formalisms for CF, but they considered users as documents, items as words, and ratings as word frequencies. In this way, they could use standard distances in IR, such as cosine, for computing similarities between users, or extending those distances by using the IDF.

Finally, the language modeling formulation also includes a parameter related with the amount of smoothing applied to weighting functions. In this way, when the parameter (λ or μ, depending on the model) is small, more emphasis on relative term weights is assumed (Zhai and Lafferty 2001). Besides, it is worth noting that when using the Dirichlet smoothing in the item space representation, we may obtain a deterministic behavior if a fixed top-N similar items are considered.

Table 3 Statistics about the datasets used in the experiments.

Dataset	Users (m)	Items (n)	Ratings	Sparsity
Movielens 100K	943	1,682	100,000	6.30%
Movielens 1M	6,040	3,900	1,000,000	4.24%
Movielens 10M	71,567	10,681	10,000,000	1.31%

This is because the probability derived by the Dirichlet LM model is related with the average document length (Zaragoza et al 2003; Losada and Azzopardi 2008). However, this is not the case in the user space representation, where a situation more similar to the one in IR (with documents of different lengths) arises.

Furthermore, it is not clear whether more smoothing or less should be required in collaborative filtering for rating data, although for frequency data it has been observed that lower values are preferred (Wang et al 2006, 2008). We have used the following estimations for the probabilities defined in Table 2:

$$p(k|i) = \frac{s_{ik}}{\sum_k s_{ik}}, \quad p(k|\mathcal{C}) = \frac{s(k,\mathcal{C})}{\sum_k s(k,\mathcal{C})}$$

where $s(k,\mathcal{C})$ is the accumulated similarity of item k in the collection.

4 Empirical Evaluation

Memory-based CF is generally used for predicting an unknown rating for a user-item pair, given a set of ratings as input. Historically, this type of algorithms has been evaluated using error metrics such as mean average error (MAE) or root mean square error (RMSE), focused on the evaluation of the algorithm's predictive accuracy. Different from frequency-based CF (Wang et al 2008; Wang 2009), in this work, we aim to bring memory-based CF a step closer to IR by incorporating not only IR models into the CF framework, but also the evaluation methodology. Indeed, there is a growing consideration in the field that such metrics have limitations as direct proxies of the user satisfaction one may ultimately wish to measure. In this perspective, ranking quality evaluation metrics from the IR field have started to be increasingly adopted (McLaughlin and Herlocker 2004; Shani and Gunawardana 2011; Koren 2008).

Because of this we evaluate our approach using two different tasks: item ranking and rating prediction. While the latter is based on error metrics, the former uses precision-based metrics. The item ranking task is more similar to how IR algorithms are usually evaluated. Besides that, we have less constraints with respect to the output of the algorithms, and since we are dealing with different weighting functions, it is very desirable to have this property. The second task, however, measures how close is the predicted rating to the real one, which is very sensitive to even slight variations of the original formula, and thus, it would limit the number of variations introduced in our models that can be tested.

4.1 Experimental setup

Our experiments have been carried out using three public datasets. First, we use the *Movielens 100K* dataset to tune the different parameters needed by the models and to analyze their sensitivity. Then, we use *Movielens 1M* and *Movielens 10M* datasets to validate our approach, where the tuned parameters for the smaller dataset are used. In the three cases, we perform a 5-fold cross validation. For *Movielens 100K* we use the splits contained in the public package. These splits retain 80% of the data (as a whole) for training, and the rest for testing. For the other datasets we repeat this procedure in order to generate equivalent data splits. Statistics about these datasets are summarized in Table 3.

The item ranking task was performed by computing a score for each user for every item contained in the test set. We take as relevant items those contained in the test set rated by the user with a value greater or equal than 4 (in a 5-rating scale). In the case of non-binary metrics (such as the normalized discounted cumulative gain, or nDCG), we use the real rating value as the relevance level, that is, 4 or 5, although further transformations could be considered to smooth those values and consider them as an utility function (Vargas and Castells 2011; Breese et al 1998). We also include the Mean Average Precision (MAP) which is known to be more robust and stable than other metrics when aggregating results from different queries (Manning et al 2008), along with recall, mean reciprocal rank (MRR) (Voorhees 1999), and bpref (Buckley and Voorhees 2004).

All the metrics for this task are calculated using the trec_eval program[5]. We have to note that, like (Wang et al 2008) and (McLaughlin and Herlocker 2004), the rated items in the test users represent only a fraction of the items that the user truly liked, and therefore, the measured metrics may underestimate the true metric values.

On the other hand, the rating prediction task is evaluated using only the rated items by each user, since we need to know the real rating values in order to use error metrics.

Finally, we have used Pearson's correlation (Adomavicius and Tuzhilin 2005) as similarity measure, a neighborhood of 50 users for the user-based methods, and a dimension of 50 with 100 iterations for the factor model in the matrix factorization algorithm.

4.2 Generating recommendations using our framework

In this section, we explain the exact process we have to follow to generate recommendations using the framework described in Section 3. We also compare such process against the traditional one by memory-based CF techniques. In particular, we shall provide an instantiation of Figure 1(b) where the modules for user profile processing, item similarity, and retrieval engine are specified.

In our framework, the first step in the process is to calculate all the similarities between items or users, depending on the space representation. In order to simplify the explanation, we present only the item-based approach, the user-based being analogous. We compute all the similarities between items and store them in an

[5] http://trec.nist.gov/trec_eval/

index using memory-efficient structures, as presented in Figure 1(a). This allows reusing these intermediate results for different offline experiments. Moreover, since the index can be updated using classic IR techniques (Tomasic et al 1994; Brown et al 1994); whenever a new incoming item (or, in general, a rating) is added to the system, the index could be updated. In that case, a trade-off between the cost of these updates (and the consequent IDF and related weights precomputations) and the advantage of using this framework also for online experiments should be considered.

Once the index is built, the queries can be processed. In our framework, this translates to the transformation of the observed user interest for items into a term vector, which in the item space approach consists of the ratings given to each item in the system. Then, the user is submitted –as a query– to a standard retrieval engine against the aforementioned index. Basically, this engine retrieves those documents which contains at least one query term (i.e., it finds the items rated by that user), and obtains a score for each document. Depending on the selected retrieval method (see Table 2), we would obtain the same scores as those provided by the standard collaborative filtering techniques or different ones. After that, a ranking is generated according to these generated scores.

In this process, any IR indexing technique could be plugged in into our model, along with any other IR model apart from those considered in Tables 1 and 2. As a matter of fact, we have built our framework on top of *Lucene*[6]'s index layer, and we have used some of the models defined in the *Terrier* platform, by simply redefining what term and query frequency, document length, etc. mean in CF according to the rating data available.

Once the item similarities have been (pre-)computed and stored in the index, a score is calculated by using Eq.(4) for every target item. By contrast, most of the implementations of memory-based CF algorithms compute item or user similarities on demand, storing for each user a very limited number of weights (Desrosiers and Karypis 2011). However, due to the highly sparse data, this caching mechanism is not very efficient, and these similarities are often used only once in the system.

When, instead of a predicted rating, a ranking is required, a score has to be computed for every other item in the system (except for those already rated by the user), and these predicted scores are aggregated and sorted accordingly. The difference on which candidate items are considered for generating the final ranking seems to be crucial for the efficiency of memory-based CF methods. For example, if we compare the average time required to obtain the whole ranking for each user when using our framework, against the time required by a public implementation of a memory-based CF algorithm (provided in the Mahout[7] library), we find improvements of up to 25% when all the similarities are pre-computed, from 139.60 ms when using our framework to 186.67 ms, which is consistent with the results reported in (Cöster and Svensson 2002). We have to note that this method, although saves time, requires a large amount of memory that sometimes is not available (Sarwar et al 2001). In that case, when the whole item-item matrix is not pre-computed, our framework achieves improvements of up to 75%, depending on the caching strategy.

[6] http://lucene.apache.org/

[7] http://mahout.apache.org

Table 4 Different normalization alternatives when calculating the score between a query and a document.

Query	**Document**	
	Normalization	No normalization
Normalization	n_{11}	n_{10}
No normalization	n_{01}	n_{00}

4.3 Results

We have evaluated our approach on two alternative base spaces: the user space and the item space. We test different weighting methods, more specifically, those shown in Table 2. As described in Sections 3.1.1 and 3.1.2, the rating prediction task requires a normalization of the score produced by the conventional dot-product, in order to obtain retrieval scores in the range of ratings. This is not required, however, in the item ranking task, which does not impose any constraint on the score range. For this reason, we have experimented with different normalization techniques along with different norms, namely L_1 and L_2.

Let us denote as n_{qd} the different normalization strategies which are presented in Table 4; that is, q is 1 in this notation when the norm of the query is used, or 0 otherwise, and similarly with d for the document norm. In general, when norm L_j is used, we would denote such normalization as $||n_{qd}||_j$[8]. As an example, note that Eq.(7) is equivalent to n_{10} in the user space (that is, where similarities are encoded in the query and ratings in the document), while Eq.(4) is the same as n_{01} (and equivalent in ranking to n_{11}) when the L_1 norm and the item space are used (i.e., the query encodes ratings and the document, similarities), that is, $||n_{10}||_1$ and $||n_{01}||_1$ respectively. Furthermore, normalization n_{11} with L_1 norm ($||n_{11}||_1$) is equivalent to the cosine similarity distance between the query and document vectors.

Finally, in order to obtain scores (predicted preferences) in the range of ratings, we have applied the different weighting methods only to the vector containing the similarity values, since if we weight both vectors indistinctly, the ratings' range gets modified, and the result will not live in a proper range. In particular, this means that, in the item space, the user vector will remain intact, whereas in the user-based representation, the weighting methods will not affect to the item vector.

4.3.1 Parameter sensitivity

Before presenting the results obtained, we discuss an analysis performed in *Movielens 100K* about the sensitivity of the different parameters used by the language modeling approaches and the BM25 method. For this reason, we have evaluated these models in the item space with varying parameters, in a similar way as it was done in (Zhai and Lafferty 2001), and using a different split to that of the subsequent evaluations in order to avoid overfitting.

Figure 3 shows the sensitivity of the smoothing parameters regarding the recommendation performance for the two language models evaluated: Dirichlet and Jelinek-Mercer. Here, we can observe some agreement between both approaches,

[8] For obvious reasons, when no normalization is used, i.e., n_{00}, this notation is simplified by removing the j index related to the L_j norm.

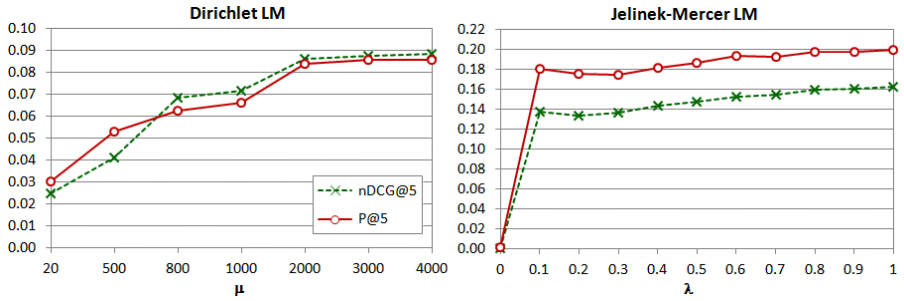


Fig. 3 Parameter sensitivity for the language models evaluated in the item-based approach.

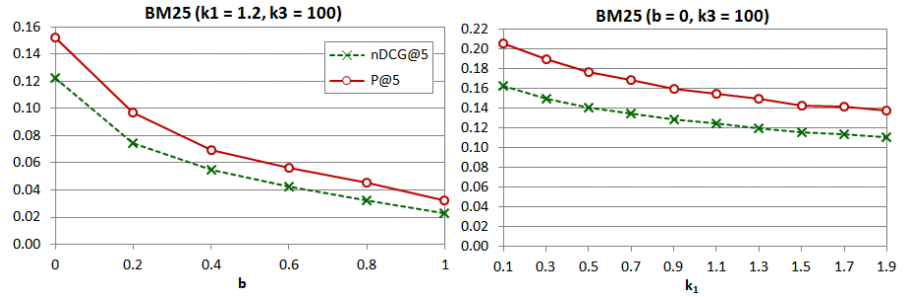


Fig. 4 Parameter sensitivity for the BM25 evaluated in the item-based approach.

so that best results are obtained when more smoothing is added from the collection. At the limit (the right end of the curve) the performance should be similar to the performance of coordination level matching, which corresponds to the count of matched terms in text retrieval. In collaborative filtering, this means the similarity between the target item and item k is more smoothed, and, in this case, the accumulated weight of item k in the collection is more emphasized.

For BM25, as we discussed in Section 3.4, larger values of k_3 are preferred for rating data, in order to obtain a similar representation of the query vector when compared against other IR methods. Thus, we set $k_3 = 100$ and performed a first test with different values of b, that is, the parameter controlling how much weight is given to the document (item) length. Then, with the best value found for parameter b, we performed another test with different values of k_1. Figure 4 shows, first, that best performance results are achieved when the document length is not used. This is probably due to the original linguistic point of view which introduced this parameter, by stating that longer documents tend to repeat information, instead of assuming other reasons for varying length, such as more elaboration or multitopicality (Spärck Jones et al 2000). In collaborative filtering, it seems this assumption is no longer valid, since a document with more terms represents an item similar to several other items in the collection, and it is not clear whether this kind of items should be penalized or not. This, in fact, would probably need to be an item-dependent parameter, since some items may get more benefit than others when using length normalization. Furthermore, this result agrees with what

Table 5 Performance results in the item space for the item ranking task (*Movielens 100K*).

Methods	P@5	P@10	nDCG@3	nDCG@5	nDCG@10	MAP
Item-based CF	0.0008	0.0008	0.0008	0.0007	0.0007	0.0139
TF $\|\|n_{01}\|\|_1$	0.0008	0.0008	0.0008	0.0007	0.0007	0.0139
MF	0.0812	0.0748	0.0570	0.0755	0.0739	0.0522
TF-IDF n_{00}	0.0831	0.0829	0.0585	0.0618	0.0653	0.0782
Dirichlet n_{00}	0.0858	0.0910	0.0984	0.0881	0.0952	0.0821
BM25 $\|\|n_{01}\|\|_2$	0.0925	0.0870	0.0772	0.0762	0.0758	0.0727
BM25 $\|\|n_{11}\|\|_1$	0.1320	0.1462	0.0790	0.0904	0.1040	0.1406
TF-IDF $\|\|n_{01}\|\|_2$	0.1594	0.1509	0.1409	0.1391	0.1389	0.1161
Jelinek-Mercer n_{00}	0.1969	0.1878	0.1662	0.1598	0.1563	0.1485
BM25 n_{00}	0.2052	0.1976	0.1619	0.1599	0.1619	0.1533
TF $\|\|n_{01}\|\|_2$	0.2195	0.2016	0.2069	0.2000	0.1963	0.1452

Table 6 Performance results in the user space for the item ranking task (*Movielens 100K*).

Methods	P@5	P@10	nDCG@3	nDCG@5	nDCG@10	MAP
User-based CF	0.0240	0.0282	0.0201	0.0195	0.0223	0.0309
TF $\|\|n_{10}\|\|_1$	0.0240	0.0282	0.0201	0.0195	0.0223	0.0309
MF	0.0812	0.0748	0.0570	0.0755	0.0739	0.0522
BM25 $\|\|n_{10}\|\|_2$	0.1487	0.1254	0.1421	0.1348	0.1273	0.0713
TF-IDF $\|\|n_{10}\|\|_2$	0.1488	0.1254	0.1420	0.1348	0.1272	0.0713
TF-IDF n_{00}	0.1546	0.1299	0.1409	0.1355	0.1296	0.0733
TF-IDF $\|\|n_{01}\|\|_2$	0.1571	0.1306	0.1391	0.1351	0.1295	0.0740
BM25 $\|\|n_{01}\|\|_2$	0.1575	0.1311	0.1393	0.1355	0.1299	0.0741
Jelinek-Mercer n_{00}	0.2279	0.1935	0.2011	0.1917	0.1773	0.0968
Dirichlet n_{00}	0.2279	0.1935	0.2011	0.1917	0.1773	0.0968
BM25 n_{00}	0.2279	0.1935	0.2011	0.1917	0.1773	0.0968

is described in adaptive filtering, where it is acknowledged that smaller values of b are sometimes advantageous for filtering tasks (Robertson 2002).

Figure 4 also shows that the smaller parameter k_1 is, the better performance is obtained. This constant determines how much weight reacts to increasing term frequencies (or similarity scale in our item space case) (Spärck Jones et al 2000). In this situation, when k_1 is small, it implies that the effect of TF is highly non-linear, even more than in traditional text retrieval (where typical values for k_1 are in the range of 1.2-2.0). This is due to the fact that TF in the item space is defined as the similarity between two items, ranging from -1 (completely dissimilar items) to 1 (perfect similarity), where 0 represents that no similarity is found, which is the most common situation. The situation in IR, by contrast, is different, since the TF is typically positive and unbounded.

In the next sections, we evaluate these three methods using the best parameters found in *Movielens 100K*, that is, $\lambda = 0.8$ and $\mu = 4000$ for the language modeling approaches, and $k_1 = 0.1, b = 0.0$, and $k_3 = 100$ for BM25. Then, we use these tuned parameters in the other datasets (*Movielens 1M* and *Movielens 10M*) to check the robustness of our approaches.

4.3.2 Item ranking task

In Tables 5 and 6, we present results from the item ranking task in *Movielens 100K*. We can see here that in both user and item spaces, several of our methods outperform the standard CF approaches, as well as a recommender based on matrix

Table 7 Performance results in the item space for the item ranking task (*Movielens 1M*).

Methods	P@5	P@10	nDCG@3	nDCG@5	nDCG@10	MAP
Item-based CF	0.0001	0.0001	0.0001	0.0001	0.0001	0.0011
TF $\|\|n_{01}\|\|_1$	0.0001	0.0001	0.0001	0.0001	0.0001	0.0011
MF	0.0623	0.0586	0.0592	0.0606	0.0602	0.0307
BM25 n_{00}	0.0044	0.0044	0.0032	0.0032	0.0032	0.0020
Dirichlet n_{00}	0.0076	0.0073	0.0052	0.0050	0.0050	0.0021
TF-IDF n_{00}	0.0121	0.0115	0.0077	0.0082	0.0082	0.0078
BM25 $\|\|n_{11}\|\|_1$	0.0369	0.0383	0.0222	0.0225	0.0239	0.0340
BM25 $\|\|n_{01}\|\|_2$	0.0530	0.0514	0.0424	0.0433	0.0433	0.0364
Jelinek-Mercer n_{00}	0.0581	0.0572	0.0441	0.0447	0.0453	0.0407
TF-IDF $\|\|n_{01}\|\|_2$	0.0799	0.0757	0.0733	0.0717	0.0699	0.0475
TF $\|\|n_{01}\|\|_2$	0.1225	0.1138	0.1192	0.1149	0.1102	0.0643

Table 8 Performance results in the user space for the item ranking task (*Movielens 1M*).

Methods	P@5	P@10	nDCG@3	nDCG@5	nDCG@10	MAP
User-based CF	0.0274	0.0252	0.0224	0.0232	0.0224	0.0139
TF $\|\|n_{10}\|\|_1$	0.0274	0.0252	0.0224	0.0232	0.0224	0.0139
MF	0.0623	0.0586	0.0592	0.0606	0.0602	0.0307
BM25 $\|\|n_{01}\|\|_2$	0.0983	0.0863	0.0964	0.0914	0.0883	0.0379
TF-IDF $\|\|n_{01}\|\|_2$	0.0984	0.0862	0.0963	0.0913	0.0882	0.0379
Dirichlet n_{00}	0.1013	0.0892	0.1020	0.0953	0.0921	0.0395
BM25 n_{00}	0.1013	0.0892	0.1020	0.0953	0.0921	0.0395
Jelinek-Mercer n_{00}	0.1013	0.0892	0.1020	0.0953	0.0921	0.0395
TF-IDF n_{00}	0.1013	0.0892	0.1020	0.0953	0.0921	0.0395
BM25 $\|\|n_{10}\|\|_2$	0.1038	0.0902	0.1049	0.0982	0.0942	0.0397
TF-IDF $\|\|n_{10}\|\|_2$	0.1041	0.0902	0.1051	0.0987	0.0944	0.0399

factorization denoted as MF and described in (Koren et al 2009). Besides, we can observe how the TF method, when normalized appropriately ($||n_{01}||_1$), is equivalent to one of the CF algorithms, although more cost-efficient, as explained in Section 4.2. It is worth noticing the poor performance of standard CF algorithms at top-N recommendation, something already illustrated in (McLaughlin and Herlocker 2004). This is particularly evident in the item space, where the standard item-based CF seems to be more sensitive to noise in the evaluation than other techniques. We have considered different similarity functions such as Pearson's correlation and cosine similarity with mean centering, but the precision results remained quite stable, with small variations depending on the similarity function.

We believe the main difference with respect to other studies reported in the literature is in the performance metric: error-based vs. ranking-based. Error-based metrics have been pervasive in field up to the last few years, whereas ranking-oriented metrics have started to be used relatively recently. For instance, MAE is used in (Sarwar et al 2001) whereas hit-rate is used in (Deshpande and Karypis 2004). In prior work (Bellogín et al 2011), we already observed an item-based method with low (i.e. good) RMSE and a close to zero precision, evidencing that a well performing method in terms of error metrics does not necessarily have a good performance in terms of precision. Here again, we shall see later in Section 4.3.3 that our item-based baseline performs reasonably well (better than user-based CF) in terms of MAE and RMSE (see Tables 11 and 12).

Table 5 shows the performance values of the framework in the item space, where users and items are represented using item coordinates (see Section 3.1.1).

For readability, only methods significantly different with respect to the baselines (item-based CF and matrix factorization) are presented. In this case, the n_{01} normalization strategy combined with the L_2 norm ($||n_{01}||_2$) produces the best results. Besides, BM25 and language modeling approaches obtain very good results with different normalization strategies.

In Table 5, we can check the assumption presented in Section 3.4, where IDF is assumed to capture the extent to which items may be similar to too many other items, penalizing them accordingly. Here, we can observe how TF-IDF obtains better performance than the baselines in two experimental conditions, whereas TF is better in only one of them. In the rest of the experiments (not presented in the table), TF-IDF significantly outperforms TF. Therefore, the use of IDF function seems to be beneficial for item-based recommendation, or at worst, it does not degrade the performance. Furthermore, all the different proposed methods statistically outperform the item-based CF baseline; therefore, the integration of IR weighting functions into rating-based CF by using text search engines not only speed up the computation but, more importantly, it provides better recommendation accuracy in terms of precision metrics.

On the other hand, Table 6 shows results when we use our framework to represent users and items in the user space. In this situation, we can see that normalizing only by the document vector (n_{01}) along with the L_2 norm –$||n_{01}||_2$– produces the best results. Strategy n_{10} also produces good results, although worse when compared with situations where no normalization is used. Interestingly, in this case it is clear that the L_2 norm produces better results than L_1, as opposed to the item-based approaches, where norm L_1 also provided good performance.

In general, the results in the user space representation involve more ties, mainly because the weighting methods are only applied to similarities, i.e., the query vector, and thus different methods are generating similar (and even equal) rankings for each user. Related with this fact, there is another difference with respect to the item-based approach. In the user-based situation there is no difference between using TF or TF-IDF methods, since they both leave the query vector intact. BM25 is the only weighting method which modifies the query vector, and thus, it produces slightly different rankings with respect to TF (or TF-IDF) methods.

We should recall that, as already explained in Section 3.4, IDF captures different situations in the item-based and user-based formulation. Whereas the latter is related with penalizing users with lots of ratings (empirically found to have negative effects), the former penalizes those items that are similar to many other items (in the results this penalization turned out to be very useful).

Additionally, in order to further validate our results, we have evaluated the methods introduced herein with two larger datasets, namely the 1M and 10M versions of *Movielens*, using the optimal parameters obtained for *Movielens 100K*. Tables 7 and 8 show the results regarding the item ranking task in *Movielens 1M*. Here, we can see again that some methods outperform both baselines (MF and standard memory-based CF), although in the item space there are several methods which are not able to improve over the matrix factorization algorithm. In any case, TF with strategy $||n_{01}||_2$ proves to be the best performing method again in this space; in the user space, BM25 and TF-IDF methods obtain the best results, again with the L_2 norm, which confirms the adequacy of this norm for the item ranking task. A similar behavior is observed on the *Movielens 10M* dataset. As a

Table 9 Performance results in the item space for the item ranking task (*Movielens 10M*).

Methods	P@5	R@5	nDCG@5	MAP	MRR	bpref
Item-based CF	0.0001	0.0001	0.0001	0.0001	0.0002	0.0046
TF $\|\|n_{01}\|\|_1$	0.0001	0.0001	0.0001	0.0001	0.0002	0.0046
MF	0.0456	0.0103	0.0467	0.0162	0.1210	0.3303
Dirichlet n_{00}	0.0012	0.0002	0.0008	0.0002	0.0037	0.0052
TF-IDF n_{00}	0.0018	0.0002	0.0013	0.0006	0.0065	0.0189
BM25 n_{00}	0.0062	0.0007	0.0046	0.0089	0.0205	0.4187
BM25 $\|\|n_{11}\|\|_1$	0.0072	0.0003	0.0041	0.0064	0.0174	0.1367
BM25 $\|\|n_{01}\|\|_2$	0.0087	0.0004	0.0065	0.0060	0.0222	0.1981
Jelinek-Mercer n_{00}	0.0111	0.0007	0.0082	0.0063	0.0264	0.0919
TF-IDF $\|\|n_{01}\|\|_2$	0.0338	0.0035	0.0303	0.0142	0.0841	0.3731
TF $\|\|n_{01}\|\|_2$	0.0693	0.0080	0.0646	0.0258	0.1474	0.4795

Table 10 Performance results in the user space for the item ranking task (*Movielens 10M*).

Methods	P@5	R@5	nDCG@5	MAP	MRR	bpref
User-based CF	0.0124	0.0018	0.0102	0.0090	0.0425	0.4972
TF $\|\|n_{10}\|\|_1$	0.0124	0.0018	0.0102	0.0090	0.0425	0.4972
MF	0.0456	0.0103	0.0467	0.0162	0.1210	0.3303
BM25 $\|\|n_{01}\|\|_2$	0.0865	0.0272	0.0773	0.0381	0.2177	0.5983
TF-IDF $\|\|n_{01}\|\|_2$	0.0865	0.0272	0.0773	0.0381	0.2177	0.5983
Dirichlet n_{00}	0.0913	0.0279	0.0826	0.0388	0.2251	0.5800
BM25 n_{00}	0.0913	0.0279	0.0826	0.0388	0.2251	0.5800
Jelinek-Mercer n_{00}	0.0913	0.0279	0.0826	0.0388	0.2251	0.5800
TF-IDF n_{00}	0.0913	0.0279	0.0826	0.0388	0.2251	0.5800
TF-IDF $\|\|n_{10}\|\|_2$	0.0927	0.0275	0.0848	0.0382	0.2281	0.5705
BM25 $\|\|n_{10}\|\|_2$	0.0928	0.0277	0.0850	0.0382	0.2285	0.5716

complement to the common performance metrics, we also report recall, MRR, and bpref for *MovieLens 10M* in Tables 9 and 10, showing a similar trend.

The trend in performance is identical in the user space (Table 10) where most of the methods outperform the MF technique using any of the reported metrics. In Table 9, on the other hand, we can observe that in the item space only the TF method with $||n_{01}||_2$ normalization is able to outperform the MF technique in terms of precision, recall, nDCG and MAP metrics, the BM25 method with n_{00} and TF-IDF with $||n_{01}||_2$, however, outperform such technique in terms of MRR and bpref evaluation metrics. This situation is again similar to that obtained for the *Movielens 1M* dataset; we believe this reduction on performance may be due to that the parameters used in BM25 and LM methods are not the optimal for these datasets, since we are using those found optimal for the *Movielens 100K*, and thus, there is still further room for improvement in these datasets.

Finally, we have to note that the results presented in this section are very dependent on the selected normalization strategy. In general, normalizing only by the document vector gives the best results for user and item approaches, though sometimes no normalization at all also produces positive results.

4.3.3 Rating prediction task

Table 11 shows the results from the rating prediction task (using error metrics) for the same dataset. In this task the retrieval score must range between 1 and R (which in this dataset is 5). This is why, in this case, the only normalization

Table 11 Results for the rating prediction task (*Movielens 100K*). Letters denote statistically significant different groups, performing a Wilcoxon paired test for each metric with $p < 0.005$.

Item-based			User-based		
Method	MAE	RMSE	Method	MAE	RMSE
Item-based CF	0.8362^a	1.0439^a	User-based CF	0.9317^a	1.2021^a
MF	0.7281^b	0.9242^b	MF	0.7281^b	0.9242^b
BM25	0.8464^c	1.0706^c	BM25	0.9316^a	1.2020^a
TF-IDF	0.8362^a	1.0439^a	TF-IDF	0.9316^a	1.2020^a
Dirichlet	0.8394^d	1.0519^d	Dirichlet	0.9317^a	1.2021^a
Jelinek-Mercer	0.8399^d	1.0503^d	Jelinek-Mercer	0.9317^a	1.2021^a

Table 12 Results for the rating prediction task (*Movielens 1M*).

Item-based			User-based		
Method	MAE	RMSE	Method	MAE	RMSE
Item-based CF	0.8210^a	1.0255^a	User-based CF	0.9443^a	1.2138^a
MF	0.6747^b	0.8687^b	MF	0.6747^b	0.8687^b
BM25	0.8236^a	1.0408^c	BM25	0.9443^a	1.2138^a
TF-IDF	0.8256^c	1.0301^a	TF-IDF	0.9443^a	1.2138^a
Dirichlet	0.8284^d	1.0359^d	Dirichlet	0.9443^a	1.2138^a
Jelinek-Mercer	0.8290^d	1.0358^d	Jelinek-Mercer	0.9443^a	1.2138^a

techniques which make sense are $||n_{01}||_1$ (for item-based) and $||n_{10}||_1$ (for user-based), which provide equivalences with Eq.(4) and Eq.(7), respectively. We can see here that the proposed methods produce comparable (significantly similar) results to that of the standard collaborative-filtering baselines.

Additionally, in Table 12 we present the results regarding the *Movielens 1M* dataset. Here we can observe the same situation described above regarding the user-based methods. In the item-based space, on the other hand, the method that is not statistically different from the standard memory-based CF baseline is the BM25 method instead of TF-IDF, like in *Movielens 100K*.

It should be emphasized that, as stated in (McLaughlin and Herlocker 2004), it is difficult to define an algorithm that achieves good results in terms of both rating prediction accuracy (as measured by error metrics) and item ranking quality (measured by usual IR metrics). While the new algorithmic framework only obtains competitive results for error metrics with respect to memory-based baselines (where model-based techniques, such as MF, obtain much better prediction accuracy), the ability of employing text retrieval insights and techniques significantly speeds up the processes and also obtains much better performance in terms of rank quality metrics, as presented in the previous section.

4.4 Discussion

The analysis of the results revealed that it is possible to use IR models effectively as the basis for memory-based CF algorithms. Moreover, we have shown in previous sections that, from an algorithmic point of view, there is a close relationship between IR and memory-based CF formulation. In particular, this relationship enables us the definition of a framework in which several IR concepts can be translated into the memory-based CF domain, such as inverted indices, weighting functions, and IDF as a component of the latter.

We test our framework on two related but different recommendation tasks: rating prediction and item ranking. These experiments seem to indicate that it is more difficult to obtain good results in the rating prediction task. Moreover, many different weighting methods and normalization strategies can be used in our framework to properly scale the system's output scores, which introduces a decision to which the rating prediction accuracy is highly sensitive. Results, however, confirm that our methods are competitive in this task –the differences with respect to the memory-based baselines are not statistically significant– and, at the same time, our approach has better computational times.

Performance on the item ranking task, on the other hand, is clearly improved by the use of IR weighting methods and the introduction of the different normalization strategies in the scoring equations. Both in user- and item-based situations, strategy n_{01} (normalizing by the document norm only) leads to the best results, as well as the L_2 norm, which consistently achieves the best results in both approaches.

BM25 has proved to behave differently with respect to ad-hoc text retrieval. The parameters of this model have been tuned, leading to different background conditions from the usual ones in text IR. For instance, document length seems to have no impact for rating data, since the optimal parameter b turned out to be 0. Nevertheless, the BM25 method obtains good performance results in both spaces, user- and item-based.

With respect to the inclusion of the Inverse Document Frequency (IDF) function in our framework, we have observed that its benefit depends on the actual equivalence of its representation in the space of users or items. In the item space it seems quite appropriate, while it does not improve the performance obtained by the TF method in the user space. This can be attributed to what IDF captures in each situation. In the user space, a term (user) with high IDF is a highly active user (active rater), and it is empirically observed that penalizing this degrades the performance. On the other hand, in the item space a term (item) with high IDF is an item which is similar to many other items, a concept more similar to what is captured by IDF in text retrieval –a lack of discriminative power– and hence, the use of this function seems to sensibly improve the simple TF method.

Additionally, our results show that the use of negative information about the user's preferences (i.e., low ratings) is also beneficial when IR models are employed. This is typically not the case in text retrieval, although similar conclusions have been obtained when applying these models to video retrieval (Aly et al 2010). This is consistent with results obtained when graph-based models are applied to recommendation (Clements et al 2009).

Furthermore, we have validated our proposed methods in two larger datasets, where they have proved their robustness by obtaining good performance results using the tuned parameters in a smaller dataset. Besides, the trend on recommender performance was very similar among the three used datasets.

In summary, IR models outperform state-of-the-art recommendation approaches in terms of precision (item ranking task); at the same time, when prediction accuracy is required (rating prediction task) these methods are competitive with respect to standard memory-based algorithms. This is true even when an exhaustive search for the optimal parameters or better model formulations has not been performed, neither for the IR methods (where different definitions of the TF component exists, for example) nor for CF techniques (where other different rating

normalization and similarity functions apart from Pearson's correlation have been proposed). We believe this is a sign of the robustness of our framework, and, also provides several parameters to be explored in the future.

5 Conclusion and Future Work

We have proposed a general model for memory-based CF which can fit many different algorithms, including different normalization techniques and weighting functions commonly used in IR models. An analogy between IR and memory-based CF has been found: in item-based CF, terms are seen as the items, while the term frequencies are the user ratings (in the query representation) or the item similarity (in the document representation). In user-based CF, the terms are equivalent to users, and the term frequencies are the user similarities (in the query) or the user rating for an item (in the document). The found equivalences introduce a new perspective on CF strategies, aiming to bring new insights and perhaps a better understanding of their relation to IR models, from which further IR researchers might design new and effective recommender systems drawing from the rich body of techniques, theories and principles in the IR field.

We have furthermore found that it is possible to directly apply IR models in our framework obtaining better results than classic CF algorithms. Besides that, different normalization techniques can fit into our framework and also lead to good results, providing a more formal explanation than previous approaches to the prediction of user preferences in memory-based CF.

In the experiments, we have explored two alternative recommendation tasks: item ranking and rating prediction. We have found that IR models perform better in the item ranking task, while they remain competitive with other classic CF algorithms in the rating prediction task. Moreover, our methods have proved to be more efficient than classic CF techniques in terms of average prediction time.

Moreover, the use of IDF has proved to be beneficial in the item-based representation, while its effect in the user-based situation is unclear. This is because IDF has a different meaning in each of these spaces, whereas in the latter is related with penalizing users with lots of ratings, in the former it penalizes those items that are similar to many other items.

In the future, we aim to extend the evaluation of the item ranking task with evaluation metrics that handle better the large amount of unknown relevance information prevalent in Recommender Systems, for instance, by using the infAP (Yilmaz and Aslam 2008) metric. We also aim to include more advanced text retrieval techniques into our framework, such as those proposed in (Salton et al 1983) and extend our framework to integrate model-based approaches such as SVD or LSA. Besides, as in many IR models where different frequency normalization techniques can also be used, different rating normalizations and similarity metrics could be explored in CF, such as using z-scores instead of ratings (Herlocker et al 1999) and adjusted cosine similarity instead of Pearson's correlation for item similarity (Sarwar et al 2001), which we envision as future work. Additionally, we aim to also exploit implicit user feedback in our framework, since it is the more typical input data in practical applications. Finally, we are also interested in testing these models with other publicly available datasets, such as *Netflix* or the one released for the *KDD Cup 2011*.

Acknowledgements This work is supported by the Spanish Government (TIN2011-28538-C02-01) and the Regional Government of Madrid (S2009TIC-1542). We also thank the anonymous reviewers for their valuable comments.

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans on Knowl and Data Eng 17:734–749

Agichtein E, Brill E, Dumais S (2006) Improving web search ranking by incorporating user behavior information. In: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '06, pp 19–26

Aizawa A (2003) An information-theoretic perspective of tf-idf measures. Inf Process Manage 39:45–65

Aly R, Doherty AR, Hiemstra D, Smeaton AF (2010) Beyond shot retrieval: Searching for broadcast news items using language models of concepts. In: ECIR, pp 241–252

Amati G, Van Rijsbergen CJ (2002) Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans Inf Syst 20:357–389

Baeza-Yates RA, Ribeiro-Neto B (1999) Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA

Belkin NJ, Croft WB (1992) Information filtering and information retrieval: two sides of the same coin? Commun ACM 35:29–38

Bellogín A, Castells P, Cantador I (2011) Precision-oriented evaluation of recommender systems: an algorithmic comparison. In: RecSys, pp 333–336

Breese JS, Heckerman D, Kadie CM (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: UAI, pp 43–52

Brown EW, Callan JP, Croft WB (1994) Fast incremental indexing for full-text information retrieval. In: Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB '94, pp 192–202

Buckley C, Voorhees EM (2004) Retrieval evaluation with incomplete information. In: Sanderson M, Järvelin K, Allan J, Bruza P (eds) SIGIR, ACM, pp 25–32

Clements M, de Vries AP, Reinders MJT (2009) Exploiting positive and negative graded relevance assessments for content recommendation. In: WAW, pp 155–166

Cohen E, Lewis DD (1999) Approximating matrix multiplication for pattern recognition tasks. J Algorithms 30(2):211–252

Cöster R, Svensson M (2002) Inverted file search algorithms for collaborative filtering. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '02, pp 246–252

Croft WB, Metzler D, Strohman T (2009) Search Engines - Information Retrieval in Practice. Pearson Education

Demartini G, Gaugaz J, Nejdl W (2009) A vector space model for ranking entities and its application to expert search. In: ECIR, pp 189–201

Deshpande M, Karypis G (2004) Item-based top-n recommendation algorithms. ACM Trans Inf Syst 22(1):143–177

Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender Systems Handbook, Springer, pp 107–144

Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. Commun ACM 35:61–70

Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '99, pp 230–237

Hofmann T (2004) Latent semantic models for collaborative filtering. ACM Trans Inf Syst 22:89–115

Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, IEEE Computer Society, Washington, DC, USA, pp 263–272

Jin R, Chai JY, Si L (2004) An automatic weighting scheme for collaborative filtering. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '04, pp 337–344

Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD, pp 426–434

Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42:30–37

Losada DE, Azzopardi L (2008) An analysis on document length retrieval trends in language modeling smoothing. Inf Retr 11(2):109–138

Manning CD, Raghavan P, Schtze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA

McLaughlin MR, Herlocker JL (2004) A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '04, pp 329–336

Metzler D, Zaragoza H (2009) Semi-parametric and non-parametric term weighting for information retrieval. In: Proceedings of the 2nd International Conference on Theory of Information Retrieval: Advances in Information Retrieval Theory, Springer-Verlag, Berlin, Heidelberg, ICTIR '09, pp 42–53

Pavlov D, Manavoglu E, Pennock DM, Giles CL (2004) Collaborative filtering with maximum entropy. IEEE Intelligent Systems 19:40–48

Pazzani M, Billsus D (1997) Learning and revising user profiles: The identification of interesting web sites. Mach Learn 27:313–331

Pickens J, Cooper M, Golovchinsky G (2010) Reverted indexing for feedback and expansion. In: CIKM, pp 1049–1058

Robertson S (2002) Threshold setting and performance optimization in adaptive filtering. Inf Retr 5:239–256

Robertson SE, Sparck Jones K (1988) Relevance weighting of search terms, Taylor Graham Publishing, London, UK, UK, pp 143–160

Rölleke T, Wang J (2008) Tf-idf uncovered: a study of theories and probabilities. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '08, pp 435–442

Rölleke T, Tsikrika T, Kazai G (2006) A general matrix framework for modelling information retrieval. Inf Process Manage 42(1):4–30

Salakhutdinov R, Mnih A (2008) Bayesian probabilistic matrix factorization using markov chain monte carlo. In: ICML, pp 880–887

Salton G, Wong A, Yang CS (1975) A vector space model for automatic indexing. Commun ACM 18:613–620

Salton G, Fox EA, Wu H (1983) Extended boolean information retrieval. Commun ACM 26:1022–1036

Sarwar BM, Karypis G, Konstan JA, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: WWW, pp 285–295

Shani G, Gunawardana A (2011) Evaluating recommendation systems. In: Recommender Systems Handbook, pp 257–297

Singhal A (2001) Modern information retrieval: A brief overview. IEEE Data Eng Bull 24(4):35–43

Sivic J, Zisserman A (2003) Video google: A text retrieval approach to object matching in videos. In: Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2, IEEE Computer Society, Washington, DC, USA, ICCV '03, pp 1470–1477

Soboroff I, Nicholas C (2000) Collaborative filtering and the generalized vector space model (poster session). In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '00, pp 351–353

Spärck Jones K (1972) A statistical interpretation of term specificity and its application in retrieval. J of Documentation 28(1):11–21

Spärck Jones K, Walker S, Robertson SE (2000) A probabilistic model of information retrieval: development and comparative experiments part 2. Inf Process Manage 36:809–840

Takács G, Pilászy I, Németh B, Tikk D (2008) Matrix factorization and neighbor based algorithms for the netflix prize problem. In: RecSys, pp 267–274

Tomasic A, García-Molina H, Shoens K (1994) Incremental updates of inverted lists for text document retrieval. In: Proceedings of the 1994 ACM SIGMOD international conference on Management of data, ACM, New York, NY, USA, SIGMOD '94, pp 289–300

Vargas S, Castells P (2011) Rank and relevance in novelty and diversity metrics for recommender systems. In: RecSys, pp 109–116

Voorhees EM (1999) The trec-8 question answering track report. In: TREC

Wang J (2009) Language models of collaborative filtering. In: Proceedings of the 5th Asia Information Retrieval Symposium on Information Retrieval Technology, Springer-Verlag, Berlin, Heidelberg, AIRS '09, pp 218–229

Wang J, de Vries AP, Reinders MJT (2006) A user-item relevance model for log-based collaborative filtering. In: ECIR, pp 37–48

Wang J, Robertson S, Vries AP, Reinders MJ (2008) Probabilistic relevance ranking for collaborative filtering. Inf Retr 11:477–497

Wong SKM, Ziarko W, Wong PCN (1985) Generalized vector space model in information retrieval. In: SIGIR, pp 18–25

Xu S, Bao S, Fei B, Su Z, Yu Y (2008) Exploring folksonomy for personalized search. In: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '08, pp 155–162

Yilmaz E, Aslam JA (2008) Estimating average precision when judgments are incomplete. Knowl Inf Syst 16(2):173–211

Yu K, Schwaighofer A, Tresp V, Xu X, Kriegel HP (2004) Probabilistic memory-based collaborative filtering. IEEE Trans on Knowl and Data Eng 16:56–69

Zaragoza H, Hiemstra D, Tipping ME (2003) Bayesian extension to the language model for ad hoc information retrieval. In: SIGIR, pp 4–9

Zhai C, Lafferty J (2001) A study of smoothing methods for language models applied to ad hoc information retrieval. In: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New York, NY, USA, SIGIR '01, pp 334–342