International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems

Axiomatic Analysis of Language Modelling of Recommender Systems

Daniel Valcarce*, Javier Parapar and Álvaro Barreiro

Information Retrieval Lab, Computer Science Deparment, University of A Coruña, Facultad de Informática, Campus de Elviña s/n, 15071, A Coruña, Spain
{*daniel.valcarce,javierparapar,barreiro*}*@udc.es*

Language Models constitute an effective framework for text retrieval tasks. Recently, it has been extended to various collaborative filtering tasks. In particular, relevance-based language models can be used for generating highly accurate recommendations using a memory-based approach. On the other hand, the query likelihood model has proven to be a successful strategy for neighbourhood computation. Since relevance-based language models rely on user neighbourhoods for producing recommendations, we propose to use the query likelihood model for computing those neighbourhoods instead of cosine similarity. The combination of both techniques results in a formal probabilistic recommender system which has not been used before in collaborative filtering. A thorough evaluation on three datasets shows that the query likelihood model provides better results than cosine similarity. To understand this improvement, we devise two properties that a good neighbourhood algorithm should satisfy. Our axiomatic analysis shows that the query likelihood model always enforces those constraints while cosine similarity does not.

Keywords: Recommender systems, language models, collaborative filtering, axiomatic analysis

1. Introduction

Recommender systems aim to provide relevant items of information to the users. Depending on which type of information the recommendation technique uses, we can distinguish different approaches.[19] Content-based methods use information about the items to suggest items which are similar to those the user liked.[6] Alternatively, collaborative filtering techniques exploit the past interactions (typically ratings) between users and items in the system.[16] Finally, there exist hybrid algorithms that combine previous approaches. In particular, we devoted this work to memory-based collaborative filtering based on explicit feedback which uses the user-item ratings directly to produce recommendations. These techniques are also called neighbourhood-based approaches because they usually compute a set of similar users (or items) called neighbourhoods.

Due to the close relationship between information retrieval (IR) and recommender systems,[2,22] different works have extended the use of Information Retrieval

*Corresponding author

methods to recommendation.[27,3,17,26] We present an axiomatic analysis of recommender systems based on language models. Language models have several applications in IR such as the query likelihood model for text retrieval and the relevance model pseudo-relevance feedback which are state-of-the-art techniques. [18,13]

One approach for formally studying an information access model is to define some desirable properties and verify if they are satisfied. This approach is called axiomatic analysis. Hazimeh and Zhai studied formally the IDF effect on several state-of-the-art pseudo-relevance feedback techniques based on the language modelling framework (including relevance models).[8] The IDF effect is a heuristic that emphasizes the selection of documents with highly specific terms. They found that the selection of the smoothing method impacts the IDF effect. In particular, they showed that Dirichlet priors, a popular smoothing method for text retrieval, is a poor choice because it demotes the IDF effect. On the other hand, Parapar et al. adapted relevance models to collaborative filtering recommendation producing highly precise recommendations.[17] In our previous work, we studied the IDF effect on relevance models for recommender systems axiomatically.[25] We found that the IDF effect is also a desirable property in recommendation related to item novelty.

Relevance models are a memory-based technique, i.e., they generate suggestions using the target user's neighbourhood. For computing those neighbourhoods, k-NN algorithm with Pearson or cosine similarity is commonly used; however, cosine similarity is the more effective.[17,25] In this paper, we want extend our previous axiomatic analysis to the computation of neighbourhoods. In addition to cosine similarity, we also employ the query likelihood model, a state-of-the-art text retrieval technique,[18] which has proven to be an effective strategy for computing user neighbourhoods.[26] Although the combination of both techniques—the query likelihood model and the relevance model—is a standard practice in information retrieval, to the best of our knowledge, this is the first time that both approaches have been jointly used in the collaborative filtering scenario. Combining both approaches results in a recommender system which is entirely based on the language modelling framework taking advance of its formal probabilistic foundations.

In this paper, we propose two desirable properties, namely User Specificity and Item Specificity, that a good neighbourhood algorithm should satisfy. We perform an axiomatic analysis showing that the query likelihood model meets these properties whereas cosine similarity does not guarantee them. Thorough experiments on several datasets confirm that relevance models produce better recommendations using the query likelihood model than cosine similarity.

In summary, the contributions of this paper are: (1) the jointly use of the query likelihood model and the relevance models for computing user neighbourhoods and producing recommendations, (2) the conception of two useful properties for a neighbourhood algorithm, (3) an axiomatic analysis of these properties on the query likelihood model and the cosine similarity and (4) an empirical evaluation of the proposed recommendation pipeline against several baselines.

2. Background and Related Work

We first introduce the top-N recommendation task and its notation. After that, we briefly describe the query likelihood model for user neighbourhood computation. Next, we outline the relevance-based language models and their adaptation to the top-N recommendation. Finally, we present a summary of different smoothing methods for the maximum likelihood estimation of language models.

2.1. *Top-N Recommendation*

Top-N recommendation consists in generating for each user $u \in \mathcal{U}$ a ordered list of N items from $\mathcal{I}$.[5] Top-N recommendation focuses on producing a good ranking of items whereas rating prediction tries to anticipate which rating the user will give to a particular item. This work is devoted to top-N recommendation because it has been acknowledged as a more realistic and useful task than rating prediction.[9,5,19]

Collaborative filtering techniques employ the feedback between users and items to generate recommendations. In this article, we leverage explicit feedback in the form of ratings. A rating from a user u to an item i is denoted by $r_{u,i}$. Additionally, $\mathcal{I}_u$ refers to the set of items rated by the user u and $\mathcal{U}_i$ to the set of users who rated the item i. We refer to the neighbourhood of user u as V_u. We denote the sum of ratings of a user u by $|u| = \sum_{i \in \mathcal{I}_u} r_{u,i}$. Likewise, the sum of ratings over an item i is given by $|i| = \sum_{u \in \mathcal{U}_i} r_{u,i}$. Finally, we represent the sum of all ratings in the whole collection by $|\mathcal{C}|$.

2.2. *Query Likelihood Model*

The query likelihood model is a state-of-the-art technique for text retrieval[18]. It models the occurrences of terms in documents and queries as a random process following a multinomial distribution. The query likelihood model assumes a uniform document prior. Valcarce et al. adapted this approach to find user and item neighbourhoods.[26] When computing user neighbourhoods, users play the role of documents and items play the role of terms. In this way, the profile of a user is a random process of ratings. Therefore, for a given user u, we rank all the users v with respect to the following conditional probability and we keep the top k:

$$p(v|u) \propto p(u|v) = \prod_{i \in \mathcal{I}_u} p(i|v)^{r_{u,i}} \tag{1}$$

where $p(i|v)$ is the conditional probability of the item i given the user v. We estimate the conditional probability using the maximum likelihood estimate (MLE) of a multinomial distribution of items. However, since this estimator suffers from sparsity, we should employ a smoothing method.

2.3. *Relevance-Based Language Models*

Relevance-based language models or relevance models (RM) are a state-of-the-art technique for the pseudo-relevance feedback task.[13] Pseudo-relevance feedback

(PRF) provides an automatic manner for expanding queries with new terms in a retrieval engine using the top retrieved documents. PRF seeks to expand the original query prompted by the user without relevance feedback by assuming that the top documents returned by the original query are relevant. The idea is to expand the query with terms from this set of documents which is called the pseudo-relevant set. This expanded query provides the final retrieval results presented to the user.

This framework has been successfully adapted to collaborative filtering.[17] Instead of a query, we have a user whose profile (the set of ratings of that user) we want to expand with new relevant items. The key idea is to select items from the neighbourhood of the target user. Under this model, users play a two-fold role. On the one hand, they behave as queries when they are the target user of the recommendation; on the other hand, they behave as documents of the pseudo-relevant set when they are part of the neighbourhood. Conversely, items act as terms.

Lavrenko and Croft proposed two estimates for relevance models: RM1 and RM2. In contrast to information retrieval, we focus on RM2 because it works better than RM1 for recommendation.[17] The probability of an item i under the relevance model of the user u is:

$$p(i|R_u) \propto p(i) \prod_{j \in \mathcal{I}_u} \sum_{v \in V_u} \frac{p(i|v)\, p(v)}{p(i)}\, p(j|v) \tag{2}$$

We consider the prior probabilities $p(i)$ and $p(v)$ to be uniform for the sake of simplicity. We leave for future work the axiomatic analysis of non-uniform priors.[23]

2.4. *Smoothing Methods*

When using language models for recommendation, we compute the probability of an item given a user $p(i|u)$ by smoothing the maximum likelihood estimate (MLE) $p_{ml}(i|u)$ of a multinomial distribution:[17,24]

$$p_{ml}(i|u) = \frac{r_{u,i}}{|u|} \tag{3}$$

2.4.1. *Jelinek-Mercer (JM)*

This method interpolates the maximum likelihood estimator and the collection model with a parameter $\lambda \in [0, 1]$:[12]

$$p_\lambda(i|u) = (1 - \lambda)\, \frac{r_{u,i}}{|u|} + \lambda\, \frac{|i|}{|\mathcal{C}|} \tag{4}$$

2.4.2. *Dirichlet priors (DP)*

We can derive this technique using a Bayesian analysis with Dirichlet priors.[14] The parameter $\mu > 0$ controls the amount of smoothing applied:

$$p_\mu(i|u) = \frac{r_{u,i} + \mu\, \frac{|i|}{|\mathcal{C}|}}{\mu + |u|} \tag{5}$$

2.4.3. *Absolute Discounting (AD)*

This technique subtracts a constant value of $\delta > 0$ from the count of the rated items:[15]

$$p_\delta(i|u) = \frac{\max[r_{u,i} - \delta, 0] + \delta \, |\mathcal{I}_u| \, \frac{|i|}{|\mathcal{C}|}}{|u|} \tag{6}$$

2.4.4. *Additive Smoothing (A)*

In contrast to the previous methods, additive smoothing (also known as Laplacian smoothing) is a collection-agnostic method. It increases all the ratings by a parameter $\gamma > 0$:

$$p_\gamma(i|u) = \frac{r_{u,i} + \gamma}{|u| + \gamma|\mathcal{I}|} \tag{7}$$

2.4.5. *Discussion*

Smoothing methods play a crucial role in the effectiveness of different language modelling approaches. For example, the preeminent smoothing methods for the query likelihood model in text retrieval are Jelinek-Mercer and, particularly, Dirichlet priors.[28] For computing neighbourhoods, these smoothing methods are still the best ones, but Jelinek-Mercer is slightly better than Dirichlet priors. [26]

In contrast to the query likelihood model, the optimal smoothing method for relevance models is neither Jelinek-Mercer nor Dirichlet priors. When performing pseudo-relevance feedback in retrieval, an axiomatic analysis of RM1 showed that additive smoothing is a better smoothing method than the others because it does not demote the IDF effect.[8] For collaborative filtering, relevance models work better with Absolute Discounting than with Dirichlet priors or Jelinek-Mercer. However, a posterior axiomatic analysis of RM2 for collaborative filtering showed that the IDF effect is related to item novelty in recommendation advocating for the use of additive smoothing in this setting.[24]

3. Axiomatic Analysis of Language Modelling of Recommender Systems

Axiomatic analysis has proven to be a useful tool for studying language models formally [7,8,25]. In the context of pseudo-relevance feedback, Hazimeh and Zhai thoroughly analysed the IDF effect on different state-of-the-art techniques.[8] Thanks to the axiomatic analysis, they were able to conclude that collection-based smoothing techniques demote this effect.

3.1. *Previous Work on the IDF Effect for Recommendation*

In our previous work,[25] we have shown that the IDF effect is closely connected to the concept of novelty in the recommendation task. In information retrieval, novelty

is usually defined as the proportion of relevant documents in the ranking that are unknown to the user.[1] For top-N recommendation, it is common to measure novelty as how unusual the recommended items are.[10] The inverse document frequency (IDF) estimates term specificity and is calculated as the inverse of the number of documents in the collection that contains the target term.[21,20]

Although IDF was not based on a formal analysis, it has shown to be a useful and robust heuristic.[21] Additionally, Robertson later provided a theoretical motivation of this term weighting function.[20] When using relevance models for generating recommendations, items play the role of words. Since the IDF effect favours specific terms over popular ones, this effect is a beneficial property for improving novelty.

We adapted the definition of the IDF effect to the collaborative filtering domain, and we analysed relevance models with different smoothing methods axiomatically.[25] We found that collection-based smoothing methods penalise the IDF effect on RM2. Conversely, additive smoothing, as a collection-agnostic method, neither supports nor violates the IDF effect. The experiments showed that additive smoothing produces more diverse and novel recommendations. Moreover, it also provides an improvement regarding ranking accuracy. [25]

3.2. *Neighbourhood Methods*

Previous work on using relevance models for recommendation has computed neighbourhoods using k-NN algorithm with Pearson or cosine similarities.[17,24,25] In general, cosine similarity provides better results than Pearson in terms of accuracy metrics for top-N recommendation.[5] Additionally, for relevance models, this also holds.[17,25] For this reason, in this article, we focus on studying why replacing cosine similarity as the neighbourhood technique with the query likelihood model can lead to improvements in the quality of the recommendations.

In particular, we used the query likelihood model with Jelinek-Mercer smoothing for computing neighbourhoods.[26] The combination of the query likelihood model with RM2 provides better results than using k-NN algorithm with cosine similarity. To analyse this phenomenon, we enunciate two desirable heuristics that a good neighbourhood technique should satisfy, and we study if the query likelihood model with Jelinek-Mercer or cosine similarity supports these heuristics.

For computing neighbourhoods, we use the k-NN algorithm which consists in computing the top k neighbours to each target user u according to a pairwise user similarity metric $f(u, v)$.

Cosine similarity measures the cosine of the angle between two vectors. If we represent user profiles as vectors of ratings, cosine similarity is given by:

$$f_{cos}(u, v) = cos(u, v) = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i}\, r_{v,i}}{\sqrt{\sum_{i \in \mathcal{I}_u} r_{u,i}^2} \sqrt{\sum_{i \in \mathcal{I}_v} r_{v,i}^2}} \tag{8}$$

On the other hand, for the query likelihood model, we take logarithms in the query likelihood model to ease the computation. Applying logarithms to Eq. 1 and

using Jelinek-Mercer smoothing from Eq. 4, the log-likelihood is given by:

$$f_{ql}(u,v) = \log p(u|v) = \sum_{i \in \mathcal{I}_u} r_{u,i} \log \left((1-\lambda) \frac{r_{v,i}}{|v|} + \lambda \frac{|i|}{|\mathcal{C}|} \right) \tag{9}$$

The above formula needs to be computed over all the items rated by the target user u. However, we can use a more convenient rank equivalent expression that only needs to be computed over the items rated by both users u and v:[28]

$$\begin{aligned} \log p(u|v) &= \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \log \left(1 + \frac{(1-\lambda) \frac{r_{v,i}}{|v|}}{\lambda \frac{|i|}{|\mathcal{C}|}} \right) \\ &= \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \log \left(1 + \frac{(1-\lambda) \, r_{v,i} \, |\mathcal{C}|}{\lambda \, |i| \, |v|} \right) \end{aligned} \tag{10}$$

Next, we present the two proposed heuristics for neighbourhood computation: User Specificity and Item Specificity.

3.3. *User Specificity*

When computing the similarity between users to produce a user neighbourhood, we often find users with very broad tastes and a lot of ratings. This kind of users are very similar to many other users concerning standard similarity metrics because they have several items rated in common to almost every user. However, these users are not very informative—they like almost everything—and their contribution to the final recommendation is noisy. In other words, we prefer candidate neighbours that have in common with the target user a significant part of the their profile. Formally, we can state this axiom as follows:

Definition 1. *User Specificity effect.* Given three users u, v and w from the set of users $\mathcal{U}$ such that $\mathcal{I}_u \cap \mathcal{I}_v = \mathcal{I}_u \cap \mathcal{I}_w$, $r_{u,i} = r_{v,i} = r_{w,i}$ for each $i \in \mathcal{I}_u \cap \mathcal{I}_v$ and $|v| < |w|$, the User Specificity effect enforces $f(u,v) > f(u,w)$.

3.3.1. *User Specificity effect in Cosine*

Assuming the users u, v and w as in the definition of the User Specificity effect, if the sign of $cos(u,v) - cos(u,w)$ is positive, then cosine enforces this heuristic. For simplicity, we refer to the root of the squared sum of the ratings of a user u by $\|u\| = \sqrt{\sum_{i \in \mathcal{I}_u} r_{u,i}^2}$.

$$\begin{aligned} cos(u,v) - cos(u,w) &= \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \, r_{v,i}}{\|u\| \|v\|} - \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_w} r_{u,i} \, r_{w,i}}{\|u\| \|w\|} \\ &= \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \, r_{v,i} \, (\|w\| - \|v\|)}{\|u\| \|v\| \|w\|} \\ &> 0 \quad \text{if} \;\; \|w\| > \|v\| \end{aligned} \tag{11}$$

We can see that cosine similarity only enforces the User Specificity effect when $\|w\| > \|v\|$, but this is not guaranteed. We know that $|w| > |v|$, that is, $\sum_{i \in \mathcal{I}_w} r_{w,i} > \sum_{i \in \mathcal{I}_v} r_{v,i}$, but this does not guarantee that $cos(u,v) - cos(u,w) > 0$ because $\sum_{i \in \mathcal{I}_w} r_{w,i} > \sum_{i \in \mathcal{I}_v} r_{v,i}$ does not imply $\sqrt{\sum_{i \in \mathcal{I}_w} r_{w,i}^2} > \sqrt{\sum_{i \in \mathcal{I}_v} r_{v,i}^2}$. Therefore, in general, we cannot say that cosine enforces this heuristic.

3.3.2. *User Specificity effect in the Query Likelihood Model*

Assuming the users u, v and w as in the definition of the User Specificity effect, if the sign of $\log p(u|v) - \log p(u|w)$ is positive, then the query likelihood model (using Jelinek-Mercer smoothing) supports this heuristic.

$$\begin{aligned}
&\log p(u|v) - \log p(u|w) = \\
&\quad \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \log \left(1 + \frac{(1-\lambda)\, r_{v,i}\, |\mathcal{C}|}{\lambda\, |i|\, |v|}\right) - \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_w} r_{u,i} \log \left(1 + \frac{(1-\lambda)\, r_{w,i}\, |\mathcal{C}|}{\lambda\, |i|\, |w|}\right) \\
&\qquad = \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \left[\log \left(1 + \frac{(1-\lambda)\, r_{v,i}\, |\mathcal{C}|}{\lambda\, |i|\, |v|}\right) - \log \left(1 + \frac{(1-\lambda)\, r_{w,i}\, |\mathcal{C}|}{\lambda\, |i|\, |w|}\right)\right] \\
&\qquad > 0
\end{aligned} \tag{12}$$

We can see that $\log p(u|v) - \log p(u|w) > 0$ because $|v| < |w|$ and $r_{v,i} = r_{w,i}$ for each $i \in \mathcal{I}_v \cap \mathcal{I}_w$. Therefore, the query likelihood model using Jelinek-Mercer smoothing supports the User Specificity effect.

3.4. *Item Specificity*

The previous heuristic gives the most importance to those candidate users whose tastes mainly agree with the target user and avoids those users that have rated many other non-relevant items with respect to the target user. Now we address a different property related to the items that users have in common. If we have two candidate users v and w with the same common ratings with the target user u except for two items j and k which are rated by only one of these users, we prefer the user who has the most specific item. With this heuristic, we seek to give more importance to highly specific items rather than general and popular items. We believe that the former kind of items is more informative than the latter. Additionally, we believe that this heuristic may help to provide more diverse neighbourhoods improving the novelty and diversity of recommendations.[4] Formally, we can define our heuristic as follows:

Definition 2. *Item Specificity.* Let u be the target user and v and w be two candidate users from the set of users $\mathcal{U}$ such that $|v| = |w|$ and let j and k be two items from the set of items $\mathcal{I}$ such that $j \in \mathcal{I}_u \cap \mathcal{I}_v$ and $k \in \mathcal{I}_u \cap \mathcal{I}_w$. Given that

$(\mathcal{I}_u \cap \mathcal{I}_v) \setminus \{j\} = (\mathcal{I}_u \cap \mathcal{I}_w) \setminus \{k\}$, $r_{u,j} = r_{v,j} = r_{u,k} = r_{w,k}$ and $r_{u,i} = r_{v,i} = r_{w,i}$ for each $i \in \mathcal{I}_u \cap \mathcal{I}_v \cap \mathcal{I}_w$, if $|j| < |k|$, then the Item Specificity effect enforces $f(u,v) > f(u,w)$.

3.4.1. *Item Specificity effect in Cosine*

Assuming the users u, v and w as in the definition of the Item Specificity effect, if the sign of $cos(u,v) - cos(u,w)$ is positive, then cosine enforces this heuristic.

$$\begin{aligned} cos(u,v) - cos(u,w) &= \\ &= \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i}\, r_{v,i}}{\|u\|\|v\|} - \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_w} r_{u,i}\, r_{w,i}}{\|u\|\|w\|} \\ &= \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v \cap \mathcal{I}_w} r_{u,i}\, r_{v,i}\, (\|w\| - \|v\|) + r_{u,j} r_{v,j} \|w\| - r_{u,k} r_{w,k} \|v\|}{\|u\|\|v\|\|w\|} \\ &> 0 \quad \text{if} \;\; \|w\| > \|v\| \end{aligned} \tag{13}$$

Taking into account that $r_{u,j} = r_{v,j} = r_{u,k} = r_{w,k}$, we can see that cosine similarity only enforces the Item Specificity effect when $\|w\| > \|v\|$ which is a condition based on the users' ratings and not on the specificity of the items. Therefore, in general, cosine similarity does not enforce this heuristic.

3.4.2. *Item Specificity effect in the Query Likelihood Model*

Assuming the users u, v and w as in the definition of the Item Specificity effect, if the sign of $\log p(u|v) - \log p(u|w)$ is positive, then the query likelihood model (using Jelinek-Mercer smoothing) supports this heuristic.

$$\begin{aligned} &\log p(u|v) - \log p(u|w) = \\ &\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{u,i} \log \left(1 + \frac{(1-\lambda)\, r_{v,i}\, |\mathcal{C}|}{\lambda\, |i|\, |v|}\right) - \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_w} r_{u,i} \log \left(1 + \frac{(1-\lambda)\, r_{w,i}\, |\mathcal{C}|}{\lambda\, |i|\, |w|}\right) \\ &\qquad = r_{u,j} \log \left(1 + \frac{(1-\lambda)\, r_{v,j}\, |\mathcal{C}|}{\lambda\, |j|\, |v|}\right) - r_{u,k} \log \left(1 + \frac{(1-\lambda)\, r_{w,k}\, |\mathcal{C}|}{\lambda\, |k|\, |w|}\right) \\ &\qquad > 0 \end{aligned} \tag{14}$$

First, we get rid off the sums using $(\mathcal{I}_u \cap \mathcal{I}_v) \setminus \{j\} = (\mathcal{I}_u \cap \mathcal{I}_w) \setminus \{k\}$ and $r_{u,i} = r_{v,i} = r_{w,i}$ for each $i \in \mathcal{I}_u \cap \mathcal{I}_v \cap \mathcal{I}_w$. Since $r_{u,j} = r_{u,k} = r_{v,k} = r_{w,k}$ and $|v| = |w|$, we can see that the difference is positive when $|j| < |k|$. Therefore, we conclude that the query likelihood model with Jelinek-Mercer smoothing enforces the Item Specificity heuristic.

With these formal results, we can argue that cosine similarity does not enforce the User and Item Specificity effects in contrast to the query likelihood model

Table 1: Datasets statistics

Dataset	Users	Items	Ratings	Density
MovieLens 100k	943	1,682	100,000	6.305%
MovieLens 1M	6040	3,706	1,000,209	4.468%
R3-Yahoo!	15,400	1,000	365,703	2.375%
LibraryThing	7,279	37,232	749,401	0.277%

smoothed with Jelinek-Mercer. Next, we present an empirical evaluation of both approaches.

4. Experiments

In this section, we assess the performance of relevance models for collaborative filtering with two different neighbourhood approaches. We used Additive smoothing for RM2 because of the previous findings of its relationship with the IDF effect.[25] For computing the user neighbourhoods, we used the k-NN algorithm with cosine similarity and the query likelihood model with Jelinek-Mercer smoothing (QL-JM).

We used multiple collections for the evaluation: the MovieLens 100k and MovieLens 1M film datasets[a], the R3-Yahoo! music collection[b] and the LibraryThing book dataset[c]. Table 1 details the statistics of these collections.

4.1. *Evaluation Methodology*

Traditionally, recommender systems were evaluated in terms of error metrics for the rating prediction task. Acknowledging that the top-N recommendation task is a better modelling of the recommendation process, evaluation is usually performed with ranking-oriented metrics as well as diversity and novelty metrics.[9,5,4]

We used the same metrics as in our previous work.[25] We measured the ranking accuracy with normalised cumulative discounted gain (nDCG) which employs graded relevance to measure the gain of a recommendation based on its position in the recommendation list.[11] To study diversity, we used the complement of Gini index which assesses the inequality of the distribution of item recommendations.[4] Finally, we employed mean self-information (MSI) for measuring the ability of the recommender system to generate unexpected recommendations (item novelty).[29] Both nDCG and Gini range from 0 to 1. In contrast, MSI ranges from 0 to $+\infty$. In all metrics, the higher its value, the better.

We apply a cut-off of 10 to all metrics for taking into account only the top 10 recommendations for each user. Since users usually only consider the first sugges-

[a] http://grouplens.org/datasets/movielens
[b] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[c] http://www.macle.nl/tud/LT/

Table 2: Values of nDCG@10, Gini@10 and MSI@10 on the MovieLens 100k (ML-100k), MovieLens 1M (ML-1M) and LibraryThing (LT) datasets. All the nDCG@10 values with an asterisk indicate a statistically significant improvement over the other neighbourhood approach based on a permutation test ($p < 0.05$).

Method	**Metric**	**ML-100k**	**ML-1M**	**R3-Yahoo!**	**LibraryThing**
Cosine	nDCG@10	0.3958	0.3562	0.0262	**0.1835***
	Gini@10	0.0610	0.0362	0.0481	0.0171
	MSI@10	11.175	12.109	18.605	27.997
QL-JM	nDCG@10	**0.4031***	**0.3680***	**0.0270***	0.1739
	Gini@10	**0.0669**	**0.0469**	**0.0623**	**0.0221**
	MSI@10	**11.425**	**12.907**	**19.709**	**29.405**

Table 3: Optimal parameters for the neighbourhood approaches on the MovieLens 100k (ML-100k), MovieLens 1M (ML-1M) and LibraryThing (LT) datasets.

Parameter	**ML-100k**	**ML-1M**	**R3-Yahoo!**	**LibraryThing**
k	50	100	125	25
λ	0.70	0.85	0.70	0.80

tions, we seek to obtain high-quality top recommendations. We also explored smaller cut-offs obtaining the same results in terms of relative differences in performance among the recommender systems. We use the training-test splits provided by the MovieLens 100k collection. For the rest of the datasets, we employed 80% of ratings of each user for the training set and the rest for the test.

For RM2, we set the Additive smoothing parameter γ to 0.01 as recommended in our previous work.[25] We tuned the Jelinek-Mercer smoothing parameter λ from 0 to 1 in steps of 0.05 and the number of neighbours k from 25 to 200 in steps of 25.

4.2. *Results and Discussion*

We present a summary of the results of RM2 with both neighbourhood strategies in Table 2. Using grid search, we set the number k of nearest neighbours to 50 for MovieLens 100k, 100 for MovieLens 1M, 125 for R3-Yahoo! and 25 for LibraryThing. The optimal values of the smoothing parameters λ were between 0.7 and 0.85 for the different datasets. Table 3 summarizes the optimal parameters for each collection.

We found that the query likelihood model outperformed cosine similarity regarding nDCG@10 in three out of four collections. All the improvements between systems were significant according to the permutations test with a significance value

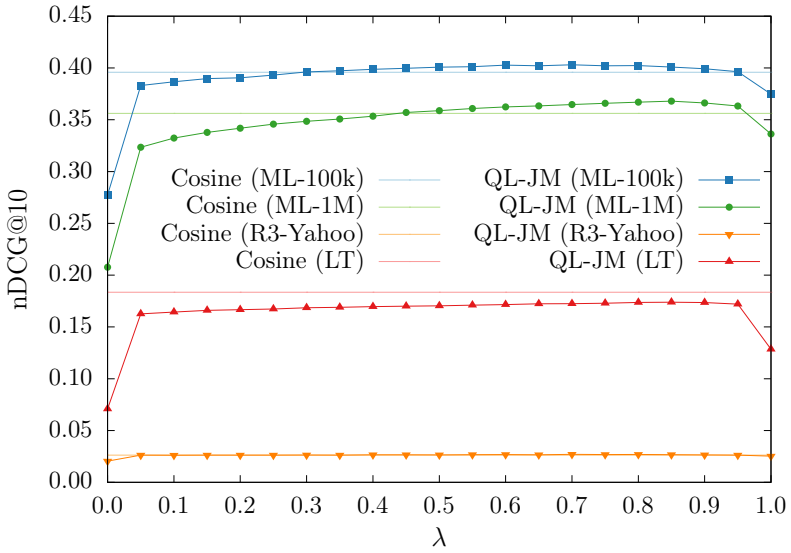

Fig. 1: Values of nDCG@10 for RM2 using Additive smoothing with $\gamma = 0.01$ with different neighbourhood similarities on the four collections.

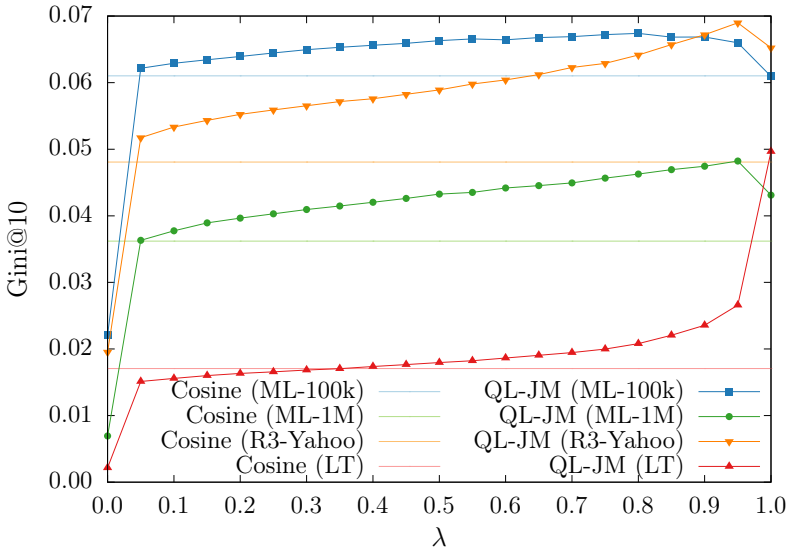

Fig. 2: Values of Gini@10 for RM2 using Additive smoothing with $\gamma = 0.01$ with different neighbourhood similarities on the four collections.

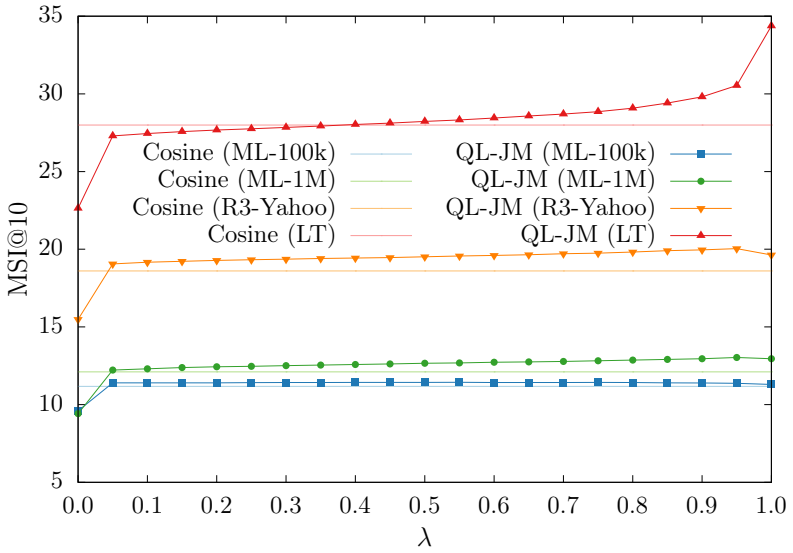

Fig. 3: Values of MSI@10 for RM2 using Additive smoothing with $\gamma = 0.01$ with different neighbourhood similarities on the four collections.

of 0.05. Additionally, QL-JM provided better diversity and novelty figures than cosine similarity in all the datasets.

It is worth noting that improving accuracy as well as diversity and novelty numbers is a notable result in the recommendation task. There exist a well-known trade-off between accuracy and diversity and novelty. [29] To promote the novelty and diversity of the results, a recommender system may try to suggest less common items increasing the risk of producing inadequate recommendations since popular items tend to be a safer option to recommend. In contrast, the query likelihood model is capable of selecting more diverse neighbours without hurting the accuracy performance in three out of four datasets. We believe that the Item Specificity effect of QL-JM helps in reach this achievement.

To study the sensibility of the parameter λ, we plot the values of nDCG@10, Gini@10 and MSI@10 of QL-JM compared to cosine similarity in Figs. 1, 2 and 3, respectively. Without smoothing ($\lambda = 0$), the performance was very poor. When we increased the value of λ, all the metrics improved surpassing cosine similarity (except for nDCG@10 in the LibraryThing collection). Overall, we found that values around 0.7 and 0.9 provided the best results in terms of the three metrics. When we used only the information in the collection ($\lambda = 1$), the quality of the recommendations worsened except for the diversity and novelty figures of QL-LM in the LibraryThing collection. When $\lambda = 1$, all the neighbours have the same similarity to the target user because similarities are solely based on collection statistics. Under this setting, we compute random neighbourhoods which may yield unexpected results.

5. Conclusions and Future Work

We proposed to combine two methods based on language models to build a highly effective memory-based recommender system with a sound probabilistic foundation. On the one hand, we used the query likelihood model to compute user neighbourhoods. On the other hand, we employed the relevance modelling approach to producing recommendations. Our evaluation found that computing neighbourhoods using the query likelihood model provided better results than cosine similarity (which was the best method used in previous literature). For analysing this phenomenon, we proposed two heuristics (User Specificity and Item Specificity) that may be useful for calculating neighbourhoods. Our axiomatic analysis of cosine similarity and the query likelihood model showed that the former does not enforce these heuristics while the latter does. This work complements our axiomatic analysis presented in our previous work that was applied to relevance models for collaborative filtering.[25]

As future work, we envision to adapt more models from the field of information retrieval to collaborative filtering tasks and analyse them axiomatically. We also plan to study additional smoothing approaches. Moreover, it would be interesting to explore other useful heuristics for the top-N recommendation task.

6. Acknowledgments

This work was supported by i) "Ministerio de Economía y Competitividad" of the Government of Spain and the ERDF (project TIN2015-64282-R), ii) Xunta de Galicia – "Consellería de Cultura, Educación e Ordenación Universitaria"(project GPC ED431B 2016/035), and iii) Xunta de Galicia – "Consellería de Cultura, Educación e Ordenación Universitaria" and the ERDF ("Centro Singular de Investigación de Galicia" accreditation 2016-2019 ED431G/01). The first author also acknowledges the support of the "Ministerio de Educación, Cultura y Deporte" of the Government of Spain (grant FPU014/01724).

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison Wesley, 2011.
2. N. J. Belkin and W. B. Croft. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Commun. ACM*, 35(12):29–38, 1992.
3. A. Bellogín, J. Wang, and P. Castells. Bridging Memory-Based Collaborative Filtering and Text Retrieval. *Inf. Retr.*, 16(6):697–724, 2013.
4. P. Castells, N. J. Hurley, and S. Vargas. Novelty and Diversity in Recommender Systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 881–918. Springer, 2nd edition, 2015.
5. P. Cremonesi, Y. Koren, and R. Turrin. Performance of Recommender Algorithms on Top-N Recommendation Tasks. In *RecSys '10*, pages 39–46, 2010.
6. M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Semantics-Aware Content-Based Recommender Systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 119–159. Springer, 2nd edition, 2015.

7. H. Fang, T. Tao, and C. Zhai. A Formal Study of Information Retrieval Heuristics. In *SIGIR '04*, page 49. ACM Press, 2004.
8. H. Hazimeh and C. Zhai. Axiomatic Analysis of Smoothing Methods in Language Models for Pseudo-Relevance Feedback. In *ICTIR '15*, pages 141–150, 2015.
9. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
10. N. Hurley and M. Zhang. Novelty and Diversity in Top-N Recommendation – Analysis and Evaluation. *ACM Trans. Internet. Technol.*, 10(4):1–30, 2011.
11. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.
12. F. Jelinek and R. L. Mercer. Interpolated Estimation of Markov Source Parameters from Sparse Data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, 1980.
13. V. Lavrenko and W. B. Croft. Relevance-Based Language Models. In *SIGIR '01*, pages 120–127, 2001.
14. D. J. C. MacKay and L. C. B. Peto. A hierarchical Dirichlet language model. *Nat. Lang. Eng.*, 1(03):289–308, 1995.
15. H. Ney, U. Essen, and R. Kneser. On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Comput. Speech Lang.*, 8(1):1–38, 1994.
16. X. Ning, C. Desrosiers, and G. Karypis. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 37–76. Springer, 2nd edition, 2015.
17. J. Parapar, A. Bellogín, P. Castells, and Á. Barreiro. Relevance-Based Language Modelling for Recommender Systems. *Inf. Process. Manage.*, 49(4):966–980, 2013.
18. J. M. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR '98*, pages 275–281. ACM, 1998.
19. F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, 2nd edition, 2015.
20. S. Robertson. Understanding inverse document frequency: on theoretical arguments for IDF. *J. Doc.*, 60(5):503–520, 2004.
21. K. Spärck Jones. A Statistical Interpretation of Term Specificity and its Retrieval. *J. Doc.*, 28(1):11–21, 1972.
22. D. Valcarce. Exploring Statistical Language Models for Recommender Systems. In *RecSys '15*, pages 375–378, 2015.
23. D. Valcarce, J. Parapar, and Á. Barreiro. A Study of Priors for Relevance-Based Language Modelling of Recommender Systems. In *RecSys '15*, pages 237–240, 2015.
24. D. Valcarce, J. Parapar, and Á. Barreiro. A Study of Smoothing Methods for Relevance-Based Language Modelling of Recommender Systems. In *ECIR '15*, pages 346–351. Springer, 2015.
25. D. Valcarce, J. Parapar, and Á. Barreiro. Additive Smoothing for Relevance-Based Language Modelling of Recommender Systems. In *CERI '16*, pages 1–8. ACM, 2016.
26. D. Valcarce, J. Parapar, and Á. Barreiro. Language Models for Collaborative Filtering Neighbourhoods. In *ECIR '16*, pages 614–625. Springer, 2016.
27. J. Wang, A. P. de Vries, and M. J. T. Reinders. A User-Item Relevance Model for Log-based Collaborative Filtering. In *ECIR '06*, pages 37–48. Springer, 2006.
28. C. Zhai and J. Lafferty. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.
29. T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the Apparent Diversity-Accuracy Dilemma of Recommender Systems. *PNAS*, 107(10):4511–5, 2010.