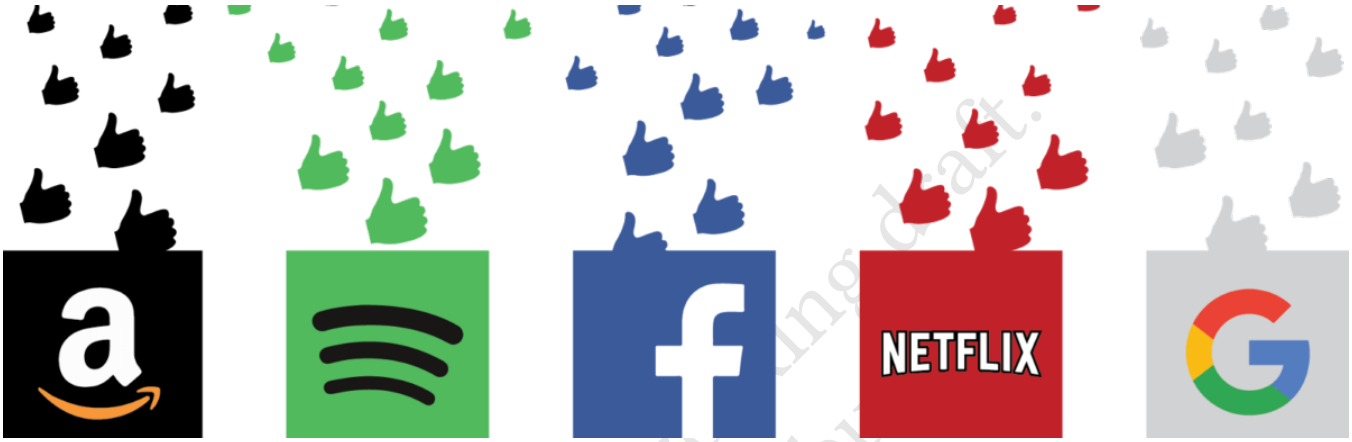
# Information Retrieval and Recommender Systems: Two Sides of the Same Coin

Archit Bansal
Department of Textile & Fibre Engineering
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India
tt1180924@iitd.ac.in

Saurav Mittal
Department of Chemical Engineering
Indian Institute of Technology Delhi
Hauz Khas, New Delhi, India
ch1180243@iitd.ac.in




## ABSTRACT

IR deals with the study of information retrieval techniques based on the inputs given by user. There is another field called IF, which also revolves around information processing and presents with information to the user which may be of interest to them. One of the widely used IF techniques is the Recommender Systems which provides personalized suggestions to users based on their interests. If we broadly look at IR and IF (RS), both are quite similar in processing the vast available information and share the most relevant ones based on various retrieval techniques. Here, in this paper, we will learn RS techniques and draw commonalities between IR and RS, and try to adapt and leverage various IR models to RS.

## KEYWORDS

information retrieval, recommender systems, ad hoc retrieval, pseudo-relevance feedback, evaluation metrics




*COL764 Information Retrieval & Web Search, Fall 2020, IIT Delhi*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn



**ACM Reference Format:**
Archit Bansal and Saurav Mittal. 2020. Information Retrieval and Recommender Systems: Two Sides of the Same Coin. In *COL764 Information Retrieval & Web Search, Fall 2020, IIT Delhi.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Information Retrieval (IR) and Information Filtering (IF) (Recommender Systems (RS) is one of the IF techniques) are closely related to each other as both deal with enormous data and present information to the users. The major difference between IR and IF is that IR acts on the queries prompted by the users and apply various techniques to rank the information retrieved to provide most relevant pieces to the users whereas in IF, the intent is to proactively search for pieces of information based on the history and past interests shown by the user. We will focus on some of the key techniques and models being used in IR and apply them to RS. We will see how we can adapt ad hoc retrieval models and pseudo relevance feedback models to be applied to RS. While this is done, we will also explore if some of the RS techniques can be used to strengthen pseudo relevance feedback model.

1. IR systems deploy many ranking techniques and metrics. We will evaluate those metrics and approaches for top-N recommendation. 2. We will see how language models from ad hoc retrieval in IR can be adapted for neighborhood computations in Recommender Systems. 3. We also explore how pseudo-relevance feedback techniques of query expansion can be adapted for recommendation tasks. 4. We look at the reverse side to see how effective recommenders can be leveraged in pseudo-relevance feedback, i.e., how

methods effectively used in recommendation can be deployed to expand queries for effective retrieval.

## 2 WHAT ARE RECOMMENDER SYSTEMS?

Today, users are demanding personalized information than giving explicit query and browsing. Recommender systems are designed to provide such user-based information. These systems are able to suggest users with the information proactively than waiting for the explicit queries. Models of RS are based on interactions between users and items, which is known as feedback. Ratings, likes, etc. constitute explicit feedback, whereas clicks, purchases, etc. constitute implicit feedback. The interactions between users and items or feedback are usually denoted as a matrix. Users are represented as rows and items as columns – user $u \in U$ and item $i \in I$ and interaction by $r(u, I)$. $r(u, I)$ is the value of interaction, i.e., rating score or the number of purchases, and $r(u, I) = 0$, if there is no interaction. $I_u$ denotes the set of items that user $u$ interacted with. $U_i$ denotes the set of users that interacted with item $i$. From here, we can find the ranked list of $n$ recommendations for user $u$.

As per research work, the score for items can be calculated for a given user in user, item pair $(u, i)$. This has been a conventional approach and RSs have evolved over the years to provide more personalized information. In classic method, recommender systems were modeled around rating predictors. Rating predictors were used for each item to forecast the ratings a user would provide and the items with highest forecast ratings would be recommended. But this was not found to be a good practice as the user would rate items which they like to rate. To produce good ranking of items, rating predictors should predict rating of all items. In top-N recommendation system, the $n$ most relevant items for the user are shown without the predicted ratings. These are specific items which are most appealing the the user.

There are different approaches to recommendation:

- **Content-based Filtering Recommender:** Filters items similar to those that the user liked using the item descriptions. It filters and recommends based on user profile + item desriptions and use the information where user has liked the similar items earlier. Some examples here are – recommending web pages, blogs, news articles, TV programs, etc. The dependency here is on the vast data need to compute similarities between items.
- **Collaborative Filtering Recommender:** These techniques rely on user-item interactions which can be explicit (ratings, reviews, etc.) or implicit (clicks, purchases, check-ins, etc.). They look at other users' feedback for recommendations. They do not need descriptions of the items. These approaches are more popular now due to enormous information available from the crowd. The two main families of CF methods are:
  - Model-based Recommender: This technique builds a predictive model based on feedback data from the users. As any predictive model, it has a training phase. The most prominent collaborative filtering approach is matrix factorization (MF) as it usually provides quality recommendations.
  - Neighborhood-based Recommender: These are also known as Memory-based Systems. These systems directly use the user-item feedback to compute recommendations. These models build neighborhood by using similarities or distance metrics. It can be further characterized into user-based (recommends items that users with common interests liked) and item-based (recommends items similar to those you liked; similarity between items is computed using common users among items and not the content) techniques.

  Although, historically, model-based recommender systems are better than neighborhood-based recommender systems in terms of performance, model-based approaches are more complex than the neighborhood-based approach which is relatively simple and easy to audit and explain.
- **Hybrid Filtering Recommender:** These systems tend to combine both content- and collaborative-based techniques to overcome the shortcomings and gain better performance.



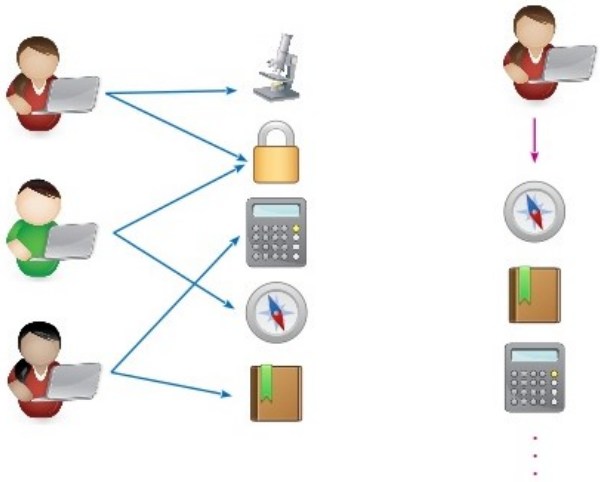
As collaborative filtering mechanisms are based on user-items interactions, new users will take time to get their preferences captured. Similarly, new items also need recommendations from many users to before getting recommended to users with similar preferences and tastes. Lack of popularity of some items may also bias the recommendation model towards popular items which may bias the evaluation and recommendation. The problem with new items may not exist in content-based filtering as recommendation is based on description and not rating. As the number of items and users grow, scalability becomes a major issue. Many large companies use parallel machines to handle millions of requests and provide recommendations.

**Matrix Factorization to build recommendation algorithm:** As per study, matrix factorization techniques are most popular in building recommendation alogorithms. Singular Value Decomposition (SVD) technique of matrix factorization is widely used in the recommender systems literature. SVD is an algebraic matrix factorization technique applied on user-item ratings matrix. It is used as Collaborative Filtering technique in a matrix structure where each row represents a user and each column represents an item. The elements of this matrix are the ratings given by users to the items in columns. The matrix is further normalized and SVD is computed. Further cosine similarity is calculated and sorted by most similar. This way top-N results are returned.

In IR, latent semantic analysis (LSA) is used to improve retrieval which is similar to SVD. Most of the search engines work by matching words in user query and documents to be searched which leads to many relevant documents getting missed as there may be different words with similar meaning in the documents. Similarly, it leads to many non-relevant documents getting retrieved as the same word many have different meanings. LSA tried to address these gaps by using dimension reduction techniques to map documents and terms into a lower dimensional semantic space. In query expansion too, matrix factorization is being used. As per research, new pseudo-relevance feedback frameworks are being proposed which are based on similar models in recommender systems.

# 3 RECOMMENDER SYSTEMS EVALUATION

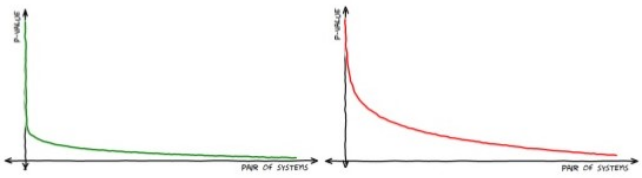
The purpose of evaluation is to choose effective models to be deployed in production. Recommender systems evaluation focuses on top-N recommendation. There are two types of evaluation. Online evaluation measures real user behavior and is expensive (e.g., A/B testing). Offline evaluation is cheap and highly reproducible. It usually constitutes the first step before deploying a recommender system. There are many metrics to evaluate RS such as error, ranking accuracy, diversity, novelty, etc. Out of these, ranking accuracy metrics are used the most.



## 3.1 Study of rank accuracy metrics for recommender systems

Many ranking accuracy metrics have been studied in IR such as precision, recall, nDCG, MAP, BPref, InfAP, and MRR. We will study their behavior in top-N recommendation using two perspectives:

- **Robustness to incompleteness:** The robustness of a metric is measured by computing the Kendall's correlation of systems rankings when changing the amount of bias.
  - **Sparcity bias:** Sparcity bias arises when users do not rate all the items and limit to a few items. Random samples from the test set are drawn to increase the bias.
  - **Popularity bias:** Popularity bias arises when popular items are recommended frequently and less popular item recommendations are ignored. The most popular items are removed to study the bias.




- **Discriminative power:** A metric is discriminative when its differences in value are statistically significant. The permutation test with difference in means is used as test statistic. The statistical test is run between all possible system pairs and the obtained $p$-values sorted by decreasing values are plotted.

## 3.2 Implications

It is found that deep cut-offs offer greater robustness and discriminative power than shallow cut-offs. Precision offers high robustness to sparsity and popularity biases and good discriminative power. nDCG provides the best discriminative power and high robustness to the sparsity bias and moderate robustness to the popularity bias.

Three recommendation dimensions are measured:

- Ranking accuracy using nDCG@100 because nDCG is robust and discriminative, and nDCG models grade relevance.
- Diversity using Gini@100 because the Gini index measures item recommendation inequality.
- Novelty using MSI@100 because MSI (mean self-information) quantifies unexpectedness of recommendations.

# 4 PSEUDO-RELEVANCE FEEDBACK MODELS FOR TOP-N RECOMMENDATION

In order to improve the search results, query expansion needs to be done by adding new terms. If this is done carefully, retrieval would yield better results. Most of the time, users are not able to indicate their queries effectively to search the information they need. In order for search engines to retrieve the relevant information as desired by users, it is vital to improve the asks by expanding the query appropriately. One of the reliable query expansion methods is relevance feedback which needs interaction with the users. The feedback given by users and the original query are combined together to expand the query which gives improved ranking of the documents. Another alternative is to use automatic query expansion that does not require inputs from the users. Pseudo-relevance feedback assumes that the top documents retrieved by the initial query is a relevant set and it extracts terms from these documents to expand the original query, and this expanded query is again fired to retrieve the relevant documents. Rocchio framework is one of the popular query expansion methods in vector space model. We will explore how pseudo-relevance feedback techniques of query expansion can be adapted for recommendation tasks.

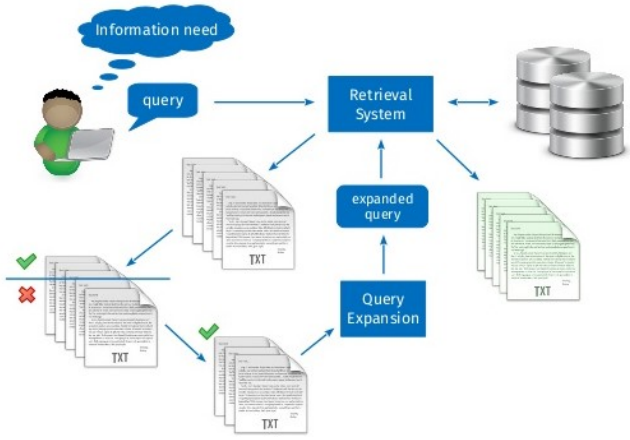


**Table 1: PRF for Recommendation**

| Pseudo-relevance feedback | Neighborhood-based recommenders |
|---|---|
| User's query | User's profile |
| Documents | Neighbors |
| Terms | Items |

Researchers have achieved great success by adapting PRF models (simply relevance models) to collaborative filtering. Lavrenko and Croft proposed two methods for estimating the relevance models in IR: RM1 and RM2.

RM1 (independent and identically distributed sampling):

$$p(i|R_u) \propto \sum_{v \in V_u} p(v)\, p(i|v) \prod_{j \in I_u} p(j|v)$$

RM2 (conditional sampling):

$$p(i|R_u) \propto p(i) \prod_{j \in I_u} \sum_{v \in V_u} \frac{p(i|v)\, p(v)}{p(i)} p(j|v)$$

where $I_u$ is the set of items rated by the user $u$, $V_u$ is neighborhood of the user $u$ computed with $k$NN cosine, $p(i)$ is the item prior, $p(v)$ is the user prior, and $p(i|u)$ is computed smoothing the maximum likelihood estimate with the probability in the collection.

Further smoothing is carried out for maximum likelihood estimate (MLE):

$$p_{mle}(i|u) = \frac{r(u,i)}{\sum_{j \in I_u} r(u,j)}$$

with the probability in the collection given by:

$$p(i|C) = \frac{\sum_{v \in U} r(v,i)}{\sum_{j \in I} \sum_{v \in U} r(v,j)}$$

Smoothing is used because it provides a way to deal with data sparsity, provides inverse document frequency (IDF) effect, and provides document length normalization. In RS, we have the same problems of data sparsity, item popularity/specificity, and user profiles having different sizes. Some collection-based and collection-agnostic smoothing techniques are:

**Jelinek-Mercer smoothing (JMS):** linear interpolation is controlled by $\lambda$.

$$p_\lambda(i|u) = (1-\lambda) p_{mle}(i|u) + \lambda p(i|C)$$

**Dirichlet priors smoothing (DPS):** Bayesian analysis with parameter $\mu$.

$$p_\mu(i|u) = \frac{r(u,i) + \mu p(i|C)}{\mu + \sum_{j \in I_u} r(u,j)}$$

**Absolute discounting smoothing (ADS):** a constant $\delta$ is subtracted.

$$p_\delta(i|u) = \frac{max[r(u,i) - \delta, 0] + \delta |I_u| p(i|C)}{\sum_{j \in I_u} r(u,j)}$$

**Additive smoothing (AS):** all the ratings are increased by $\gamma > 0$.

$$p_\gamma(i|u) = \frac{r(u,i) + \gamma}{\sum_{j \in I_u} r(u,j) + \gamma |I|}$$

In IR, the IDF effect is a measure of term specificity in most weighting schemes. It was born as a heuristic but was theoretically justified later. In RS, item specificity is related to item novelty. Let $u$ be a user from the set of users $U$ and $V_u$ be her/his neighborhood. Given two items $i_1$ and $i_2$ with the same ratings $r(v, i_1) = r(v, i_2)\ \forall v \in V_u$ and different popularity $p(i_1|C) < p(i_2|C)$, a recommender system that outputs $p(i_1|R_u) > p(i_2|R_u)$ is said to support the IDF effect in recommendation. The IDF effect in RM2 is analyzed axiomatically using the above smoothing methods and it is expected that additive smoothing would offer better figures of novelty.

In RM2, $p(i)$ and $p(v)$ are the item and user priors respectively. They enable to introduce a priori information into the model and provide a principled way of modeling business rules. They are similar to document priors used in IR such as linear document length prior and probabilistic document length prior.

Uniform prior estimators:

$$p_U(u) = \frac{1}{|U|} \text{ (user prior)}, \quad p_U(i) = \frac{1}{|I|} \text{ (item prior)}$$

Linear prior estimators:

$$p_L(u) = p(u|C) = \frac{\sum_{i \in I_u} r(u,i)}{\sum_{v \in U} \sum_{j \in I_v} r(v,j)} \text{ (user prior)},$$

$$p_L(i) = p(i|C) = \frac{\sum_{u \in U_i} r(u,i)}{\sum_{j \in I} \sum_{v \in U_j} r(v,j)} \text{ (item prior)}$$

Probabilistic prior using Jelinek-Mercer smoothing:

$$p_{PJMS}(u) = (1-\lambda) + \lambda \sum_{i \in I_u} p(i|C) \text{ (user prior)}$$

Probabilistic prior using Dirichlet priors smoothing:

$$p_{PDPS}(u) = \frac{\sum_{i \in I_u} r(u,i) + \mu \sum_{i \in I_u} p(i|C)}{\mu + \sum_{i \in I_u} r(u,i)} \text{ (user prior)}$$

Probabilistic prior using absolute discounting smoothing:

$$p_{PADS}(u) = \frac{\sum_{i \in I_u} max(r(u,i) - \delta, 0) + \delta |I_u| \sum_{i \in I_u} p(i|C)}{\sum_{j \in I_u} r(u,j)} \text{ (user prior)}$$

Probabilistic prior using additive smoothing:

$$p_{PAS}(u) = \frac{\sum_{i \in I_u} r(u,i) + \gamma |I_u|}{\sum_{j \in I_u} r(u,j) + \gamma |I|} \text{ (user prior)}$$

Similarly for item priors.

The evaluation shows that the uniform estimator gives the best estimate of the user prior probability. And Jelinek-Mercer smoothing gives the best results regarding the item prior which uses probabilistic item prior estimate.

## 4.1 Rocchio framework

Relevance models are very effective recommenders but they have high computational cost, several hyperparameters need to be tuned, and different smoothing and prior choices need to be made. We study four term scoring functions used within the Rocchio framework:

Rocchio weights (RW):

$$p_{RW}(i|u) = \sum_{v \in V_u} \frac{r(v,i)}{|V_u|}$$

Robertson selection value (RSV):

$$p_{RSV}(i|u) = p(i|V_u) \sum_{v \in V_u} \frac{r(v,i)}{|V_u|}$$

Chi-square (CHI2):

$$p_{CHI2}(i|u) = \frac{[p(i|V_u) - p(i|C)]^2}{p(i|C)}$$

Kullback-Leibler divergence (KLD):

$$p_{KLD}(i|u) = p(i|V_u) \log \frac{p(i|V_u)}{p(i|C)}$$

For neighborhood size normalization, neighborhoods are computed using clustering algorithms:

- **Hard clustering:** every user appears in only one cluster and clusters may have different sizes. e.g., $k$-means.
- **Soft clustering:** each user has its own neighbors and when we set $k$ to a high value, we may find different amounts of neighbors. e.g., $k$NN algorithm.

The idea proposed was to consider the variability of neighborhood sizes. Large neighborhoods are equivalent to a query with a lot of results, i.e., the collection model is closer to the target user. Small neighborhoods imply that neighbors are highly specific, i.e., the collection is very different from the target user. The MLE is biased to perform neighborhood size normalization:

$$p_{nmle}(i|V_u) \overset{rank}{=} \frac{1}{|V_u|} \frac{\sum_{v \in V_u} r(v,i)}{\sum_{v \in V_u} \sum_{j \in I} r(v,j)}$$

$$p_{nmle}(i|C) \overset{rank}{=} \frac{1}{|U|} \frac{\sum_{u \in U} r(u,i)}{\sum_{u \in U} \sum_{j \in I} r(u,j)}$$

The studies showed that recommendation approaches presented low numbers of diversity and novelty with high values of nDCG. It was found that RW, RSV, CHI2, and KLD algorithms are efficient compared to RM2 for recommendation. But quality of the neighborhoods has a role to play in their effectiveness.

## 4.2 Improving neighborhoods

Neighborhood-based methods are usually simple, efficient, and explainable, but their effectiveness relies largely on the quality of the neighbors. The most common approach is to compute the $k$ nearest neighbors ($k$NN algorithm) using a pairwise similarity.

Weighted sum recommender (WSR) stems from NNCosNgbr. The MLE is biased to perform neighborhood size normalization:

NNCosNgbr:

$$\hat{r}_{u,i} = b_{u,i} + \sum_{j \in J_i} \text{shrunk\_cosine}\ (i,j)\ (r(u,j) - b_{u,i})$$

Item-based weighted sum recommender (WSR-IB):

$$\hat{r}_{u,i} = \sum_{j \in J_i} \cos\ (i,j)\ r(u,j)$$

User-based weighted sum recommender (WSR-UB):

$$\hat{r}_{u,i} = \sum_{v \in V_u} \cos\ (u,v)\ r(v,i)$$

Comparing the effectiveness of WSR and NNCosNgbr algorithms, it was found that both user- and item-based versions of WSR significantly outperformed NNCosNgbr on all datasets in ranking accuracy.

## 4.3 Improving cosine with an oracle

Weighted sum recommender with $k$NN cosine works well in top-N recommendation. However, there is a room for improvement of this similarity measure. Finding the best neighborhood is an NP-hard problem. An approximate oracle was built using a greedy approach that generates ideal neighborhoods.

The neighborhoods produced by the greedy oracle may be impossible to achieve with similarities based on co-occurrence. Hence, a simpler oracle was developed based on cosine similarity by finding the best neighborhoods that cosine similarity can provide by tuning the value $k$ for each user. This oracle is seen as an adaptive $k$NN algorithm that uses the optimal $k$ for each user. On comparing the quality of the recommendations produced using WSR with cosine similarity, the cosine-based oracle provided better results than the baselines. However, its effectiveness seems much more achievable compared to the greedy oracle.

By studying the properties of the neighborhoods provided by the oracles, cosine similarity was modified by penalizing the cosine similarity to add user profile size normalization (similar to the pivoted document length normalization in IR). The IDF effect was added to cosine similarity to increase the user profile overlap of the neighbors. The studies suggest that penalized cosine with IDF outperformed all baselines in terms of nDCG@100. Adding the IDF heuristic to the cosine formula provides a boost in ranking accuracy, diversity, and novelty without adding any new parameter.

## 4.4 Language models for computing neighborhoods

With data growing at an exponential rate, it is important to meet the information needs of the users retrieving relevant pieces of information. Retrieval systems are the core of search engines. Ad hoc retrieval model is one such mechanism to filter the relevant documents (web pages, news articles, images, etc.) and then rank them. For ranking the documents, various mathematical models are being used – vector space model, boolean model, BM25, Sparck-Jones, probabilistic models, language models, neural models, etc. Vector space model and language models are the most important ad hoc retrieval approaches. We will try to adapt language models in IR for neighborhood computations in recommender systems.

So far, we looked at improving cosine similarity with ideas from IR. But better can be done than cosine similarity. Let's study cosine similarity from IR perspective.

Computing neighborhoods using cosine similarity is equivalent to search in the vector space model. If users and items are swapped, an analogous item-based approach can be derived. Sophisticated search techniques can be used for finding neighbors!

Statistical language models are a state-of-the-art ad hoc retrieval framework. Documents are ranked according to their posterior probability given the query:

$$p(d|q) = \frac{p(q|d)\ p(d)}{p(q)} \overset{rank}{=} p(q|d)\ p(d)$$

The query likelihood, $p(q|d)$, is based on a unigram model:

$$p(q|d) = \prod_{t \in q} p(t|d)^{c(t,d)}$$

The document prior, $p(d)$, is usually considered uniform.

Ad hoc retrieval:

$$p(d|q) \stackrel{rank}{=} p(d) \prod_{t \in q} p(t|d)^{c(t,d)}$$

Language models for finding neighborhoods are user-based and item-based collaborative filtering:

$$p(v|u) \stackrel{rank}{=} p(v) \prod_{i \in I_u} p(i|v)^{r(v,i)} \text{ (user-based)}$$

$$p(j|i) \stackrel{rank}{=} p(j) \prod_{u \in U_i} p(u|j)^{r(u,j)} \text{ (item-based)}$$

For the user-based collaborative filtering, a multinomial distribution is assumed over the count of ratings:

$$p_{mle}(i|v) = \frac{r(v,i)}{\sum_{j \in I_v} r(v,j)}$$

However, it suffers from sparsity and smoothing needs to be done.

On comparing the user- and item-based WSR and RM2 (with cosine similarity) and the query likelihood model (with Jelinek-Mercer smoothing), it is found that the language modeling approach improves the ranking accuracy, diversity, and novelty of all the neighborhood-based recommenders compared to cosine similarity.

To explain why language models with Jelinek-Mercer smoothing work better than cosine similarity, an axiomatic analysis is performed by defining user specificity and item specificity properties.

- **User specificity:** Given the target user $u$ and the candidate neighbors v and w such that $I_u \cap I_v = I_u \cap I_w$, $r(u,i) = r(v,i) = r(w,i) \forall i \in I_u \cap I_v$, $and |v| < |w|$, the user specificity property enforces $\text{sim}(u,v) > \text{sim}(u,w)$.
- **Item specificity:** Let $u$ be the target user and $v$ and $w$ be two candidate users such that $|v| = |w|$. Let $j$ and $k$ be two items from the set of items $I$ such that $j \in I_u \cap I_v$, $k \in I_u \cap I_w$. Given $(I_u \cap I_v) \backslash \{j\} = (I_u \cap I_w) \backslash \{k\}$, $r(u,j) = r(v,j) = r(u,k) = r(w,k)$, $r(u,i) = r(v,i) = r(w,i) \forall i \in I_u \cap I_v \cap I_w$. If $|j| < |k|$, then the item specificity property enforces $\text{sim}(u,v) > \text{sim}(u,w)$.

The differences in effectiveness may be thought to be related to these properties.

# 5 OTHER RECOMMENDATION TASKS

Top-N recommendation is the most prominent task in recommender systems. However, recommendation technologies are used in many industrial scenarios. Here, we focus on two less popular recommendation problems – long tail liquidation and user-item group formation.

## 5.1 Long tail liquidation

Item popularity follows a long tail distribution. If we take the example of e-commerce, the excess stock or inventory causes revenue loss to business. A recommendation task is formulated centered on the liquidation of long tail items. An item-based adaptation of relevance models is also proposed to deal with this novel task.

The long tail liquidation problem: Let $I' \subset I$ be the items to be liquidated, the aim is to find a scoring function $s' : I' \times U \rightarrow \mathbb{R}$ such that for each item $i \in I'$, a ranked list of $n$ users $L_i^n \in U^n$ can be built, that are most likely interested in such item $i$. Three strategies to discriminate long tail products:

Least rated products:

$$I' = \{i \in I \big| |U_i| < c_1\}$$

Lowest rated products:

$$I' = \left\{ i \in I \middle| \frac{\sum_{u \in U_i} r_{u,i}}{|I_i|} < c_2 \right\}$$

Least recommended products:

$$I' = \{i \in I \big| i \notin L_u^{c_3}, \forall u \in U\}$$

An item-based relevance probabilistic model:

IRM2:

$$p(u|R_i) \propto p(u) \prod_{v \in U_i} \sum_{j \in J_i} p(v|j) \frac{p(u|j)\, p(j)}{p(u)}$$

MLE with additive smoothing:

$$p_\gamma(u|i) = \frac{r(u,i) + \gamma}{\sum_{v \in U_i} r(v,i) + \gamma |U|}$$

Item neighborhoods: $\mathcal{J}_i$ is computed using $k$NN algorithm with cosine similarity.

Here, to tackle the problem of how to liquidate long tail items, we designed an item-based adaptation of relevance model to recommendation. The results show that some techniques are good approaches to the task of recommending users to long tail items. However, in general, the proposed IRM2 method is the best option for the long tail liquidation task.

## 5.2 User-item group formation

The user-item group formation (UIGF) problem aims at finding the best companions for a given item and a target user. IRM2 estimates the relevance of a user given an item and deals with long tail item liquidation with uniform priors. The user relationships can be modeled with different priors estimators.

Given the target user $u \in U$, the recommended item $i \in I$, and an integer k, the UIGF problem seeks to find the set $F_{u,i}^G \subseteq U$ such that:

$$F_{u,i}^G = \arg\max_{F_u^*} \sum_{v \in F^*} p(v|R_i) \text{ s.t. } F^* \subseteq U, |F^*| = k$$

The information about group dynamics is encoded in priors:

Uniform prior (U):

$$p_U(v) = \frac{1}{|F_u|}$$

Common Friends (CF):

$$p_{CF}(v) \propto \frac{1}{|F_u \cap F_v|}$$

Common group friends (CGF):

$$p_{CGF}(v) \propto \frac{1}{|(\bigcup_{w \in F_{u,i}^G} F_w) \cap F_v|}$$

Group closeness (GC):

$$p_{GC}(v) \propto \frac{1}{|F^G_{u,i} \cap F_v|}$$

Ground truth groups were used to evaluate UIGF approaches, i.e., users who checked in the same place within 4 hours, groups of at least 4 members, and each user must be friends with at least another group member. For each group, a random member is selected as the target user and the place where the group registered as the target item. The UIGF model is asked to form a group of $k$ friends for this specific user and item. The precision of the recommended group is evaluated against the ground truth groups.

## 6 RECOMMENDER SYSTEM MODELS FOR PSEUDO-RELEVANCE FEEDBACK

Till now, we explored adapting IR models to recommendation models. Here, we look at the opposite direction of adapting recommendation models to retrieval models, especially the linear methods used in RS to be adapted to pseudo-relevance feedback. There are two linear methods – document based and term based, evaluated for pseudo-relevance feedback. With these methods, tasks of pseudo-relevance feedback are modeled as a matrix decomposition. Original query is expanded using these factorizations of similarities matrix with inter-document or inter-term. The framework is known as LiMe (Linear Methods) and it is not based on language models but on linear methods.

- **DLiMe:** learns inter-document similarities.
- **TLiMe:** learns inter-term similarities.

SLIM is a state-of-the-art recommendation model that learns an item-item similarity matrix using linear methods.

Matrix formulation: Let $X \in \mathbb{R}^{m \times n}$ be the extended pseudo-relevant set matrix, the aim is to find an inter-term similarity matrix $W \in \mathbb{R}_+^{n \times n}$ such that:

$$X = X \times W$$

$$\begin{pmatrix} Q \\ D_1 \\ \cdots \\ D_{m-1} \end{pmatrix}_{m \times n} = \begin{pmatrix} Q \\ D_1 \\ \cdots \\ D_{m-1} \end{pmatrix}_{m \times n} \times \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{pmatrix}_{n \times n}$$

$$\text{s.t. } \text{diag}(W) = 0,\ W \geq 0$$

To fill matrix X, we consider:

$$x_{ij} = \begin{cases} s(t_j, Q) & \text{if } i = 1 \text{ and } f(t_j, Q) > 0, \\ s(t_j, D_{i-1}) & \text{if } i > 1 \text{ and } f(t_j, D_{i-1}) > 0, \\ 0 & \text{otherwise} \end{cases}$$

$$s_{tf-idf}(t, D) = (1 + \log_2 f(t, D)) \times \log_2 \frac{|C|}{df(t)}$$

where $f(t, D)$ is the number of occurrences of term $t$ in $D$ (or $Q$).

Matrix optimization problem:

$$W^* = \arg\min_W \frac{1}{2}||X - XW||_F^2 + \beta_1||W||_{1,1} + \frac{\beta_2}{2}||W||_F^2 \quad (1)$$

$$\text{s.t. } \text{diag}(W) = 0,\ W \geq 0$$

Bound constrained least squares optimization problem with elastic net ($l_1$ and $l_2$ regularization) penalty:

$$\vec{w}^*_{.j} = \arg\min_{\vec{w}_{.j}} \frac{1}{2}||\vec{x}_{.j} - X\vec{w}_{.j}||_2^2 + \beta_1||\vec{w}_{.j}||_1 + \frac{\beta_2}{2}||\vec{w}_{.j}||_2^2 \quad (2)$$

$$\text{s.t. } w_{jj} = 0, \vec{w}_{.j} \geq 0$$

To expand the original query, the first row of $X$ is reconstructed:

$$(Q')_{1 \times n} = (Q)_{1 \times n} \times \begin{pmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{pmatrix}_{n \times n} \quad (3)$$

$$\hat{x}_{1.} = \vec{x}_{1.} \times W^*$$

A probabilistic estimate of a term $t_j$ is computed given the feedback model $\theta_F$:

$$p(t_j|\theta_F) = \begin{cases} \frac{\hat{x}_{1j}}{\sum_{t_v \in V_{F'}} \hat{x}_{1v}} & \text{if } t_j \in V_{F'}, \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The second retrieval is performed interpolating the original query model with the feedback model:

$$p(t|\theta'_Q) = (1 - \alpha)\ p(t|\theta_Q) + \alpha p(t|\theta_F) \quad (5)$$

The hyperparameter $\alpha$ controls the interpolation. This is a standard procedure in state-of-the-art PRF techniques.

The results of the experiments suggest that TLiMe is the only method that offered significant improvements over LM in MAP and nDCG. Moreover, either DLiMe or TLiMe achieved the highest figures in robustness index on most datasets.

## 7 CONCLUSIONS

In this paper, we explored recommender system approaches and cross-pollination of ideas between IR and RS:

- We studied the robustness and discriminative power of ranking accuracy metrics.
- We studied the adaptation of different pseudo-relevance feedback models to top-N recommendation as memory-based recommenders and found that relevance models offer highly accurate recommendations and techniques from the Rocchio framework are a very cost-effective alternative.
- We studied the usage of ad hoc retrieval models to compute better neighborhoods in collaborative filtering. We studied how language models outperform cosine similarity.
- We studied the adaptation of relevance models to novel recommendation tasks and how these models can tackle the issue of long tail items.
- We studied the proposal of a novel pseudo-relevance feedback framework adapted from recommendations method.

### 7.1 Future directions

1. The robustness and discriminative analysis can be explored to different types of metrics such as diversity or novelty metrics. 2. The study can be extended to the adaptation of various other pseudo-relevance feedback models to top-N recommendation. 3. Other neighborhood computation techniques can be analyzed using the methodology based on oracles. 4. Other ad hoc retrieval models to

compute neighborhoods can be examined. 5. LiMe can be extended with richer features (based on Wikipedia, query logs, etc.).

## REFERENCES


[1] Nicholas J. Belkin and W. Bruce Crof. 1992. Information filtering and information retrieval: two sides of the same coin? *Commun. ACM* 35, 12 (1992), 29–38. https://doi.org/10.1145/138859.138861

[2] Alejandro Bellogín, Pablo Castells, and Iván Cantador. 2017. Statistical biases in Information Retrieval metrics for recommender systems. *Information Retrieval Journal* 20, 6 (2017), 606–634. https://doi.org/10.1007/s10791-017-9312-z

[3] Rocío Cañamares and Pablo Castells. 2018. Should I Follow the Crowd? *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval - SIGIR '18* (2018). https://doi.org/10.1145/3209978.3210014

[4] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. *Proceedings of the Fourth ACM Conference on Recommender Systems - RecSys '10* (2010). https://doi.org/10.1145/1864708.1864721

[5] Alfonso Landin, Daniel Valcarce, Javier Parapar, and Alvaro Barreiro. 2019. PRIN: A Probabilistic Recommender with Item Priors and Neural Models. *Advances in Information Retrieval* (2019), 133–147. https://doi.org/10.1007/978-3-030-15712-8_9

[6] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. *IEEE 11th International Conference on Data Mining* (2011). https://doi.org/10.1109/icdm.2011.134

[7] Eva Suárez-García, Alfonso Landin, Daniel Valcarce, and Álvaro Barreiro. 2018. Term Association Measures for Memory-based Recommender Systems. *Proceedings of the 5th Spanish Conference on Information Retrieval - CERI '18* 6 (2018), 1–8. https://doi.org/10.1145/3230599.3230606

[8] Daniel Valcarce. 2019. *Information Retrieval Models for Recommender Systems*. Ph.D. Dissertation. Universidade da Coruña, Spain.

[9] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. 2018. On the robustness and discriminative power of information retrieval metrics for top-N recommendation. *Proceedings of the 12th ACM Conference on Recommender Systems - RecSys '18* (2018). https://doi.org/10.1145/3240323.3240347

[10] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. 2020. Assessing ranking metrics in top-N recommendation. *Information Retrieval Journal* 23 (2020), 411–448. https://doi.org/10.1007/s10791-020-09377-x

[11] Hai-Tao Yu, Adam Jatowt, Roi Blanco, Hideo Joho, and Joemon M. Jose. 2017. An in-depth study on diversity evaluation: The importance of intrinsic diversity. *Information Processing Management* 53, 4 (2017), 799–813. https://doi.org/10.1016/j.ipm.2017.03.001