Improving One-Class Collaborative Filtering by Incorporating Rich User Information

Yanen Li[*]
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
yanenli2@illinois.edu

Jia Hu
Department of Computer Science and Engineering
Texas A&M University
College Station, TX, 77840
vickyhujia@hotmail.com

Chengxiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
czhai@cs.illinois.edu

Ye Chen
Microsoft Corporation
1065 La Avenida, Mountain View, CA 94043
yec@microsoft.com

ABSTRACT

One-Class Collaborative Filtering (OCCF) is an emerging setup in collaborative filtering in which only positive examples or implicit feedback can be observed. Compared with the traditional collaborative filtering setting where the data has ratings, OCCF is more realistic in many scenarios when no ratings are available. In this paper, we propose to improve OCCF accuracy by exploiting the rich user information that is often naturally available in community-based interactive information systems, including a user's search query history, purchasing and browsing activities. We propose two ways to incorporate such user information into the OCCF models: one is to linearly combine scores from different sources and the other is to embed user information into collaborative filtering. Experimental results on a large-scale retail data set from a major e-commerce company show that the proposed methods are effective and can improve the performance of the One-Class Collaborative Filtering over baseline methods through leveraging rich user information.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *Information Filtering*

General Terms

Algorithms, Performance, Experimentation

[*]Work done as a summer intern at eBay Research Labs

Keywords

Recommender Systems, One-Class Collaborative Filtering, Rich User Information

1. INTRODUCTION

A Recommender System analyzes users' past behavior and predicts user's preference to improve user's satisfaction. Originally introduced by Goldberg et al. [8], Collaborative Filtering (CF) approaches are the most popular methods in recommender systems and have been extensively studied [1]. One of the most famous examples is the Netflix Prize problem, in which the most successful methods reported are CF models. However, most CF methods are focused on data sets with explicit ratings. Such explicit ratings are hard to collect in many applications because of the intensive user involvement. Recently, One-Class Collaborative Filtering (OCCF) has emerged as a very interesting problem setup where only binary data of the user's interaction can be observed through implicit feedback [16]. OCCF reflects a more realistic scenario. In fact, most of the real life recommendations with implicit user feedback can be considered as an OCCF problem. Famous examples include Amazon's product recommendation, web page bookmarking and the KDD cup 2007 "Who rated What" problem on movie recommendation.

In the OCCF problem setup, data is usually extremely sparse and unbalanced: only a small part of data is labeled as positive examples. So far, research on OCCF has focused on how to best model the missing examples [16, 15, 9, 20]. However, the information about users in these studies has been restricted to implicit judgments on items only. In many applications, we naturally have much more user information that can be leveraged. For example, in most interactive systems where users can search for items, we would naturally have a search log with users' queries and clickthroughs on search result. In a modern online market-place, there are several types of data reflecting the user-item interaction which can be utilized, such as search logs, item clickthroughs, user's transaction history and so on. To the best of our knowledge, in the setting of One-Class Collaborative Filtering, little has been studied in exploring how to exploit the rich user information to overcome the sparsity problem of the data and improve the recommendation performance.

In this paper we propose two strategies of incorporating rich user information to improve the OCCF performance. The first strategy is to use different sources of user information as independent evidence to score items, and linearly combine the scores with regular collaborative filtering scores to make the final decision. The second is to tightly embed the user information into a collaborative filtering model. Specifically, we extend the User-based CF baseline model, by replacing the user-user similarity function with a content-based similarity function or replacing the sparse transaction matrix with a denser clickthrough data matrix. We extend the Matrix Factorization baseline models by replacing the global weighting scheme with the content-based dissimilarity function. Experiments on a large-scale dataset from a major online market-place show that the proposed methods can effectively improve the recommendation performance. And the analysis on extreme cases indicates that the complementary nature of the content-based features and collaborative filtering could help to ease the *cold-start* problem in recommendation.

The paper is organized as follows: We first summarize the related work in Section 2. In section 3 and 4 we define our problem setup, and introduce types of user information we could exploit. In Section 5 we propose major strategies for incorporating rich user information into the Neighbor-based CF. In Section 6 and 7 we present experimental results on a large-scale data set, and discuss which user information performs better and when the best result can be achieved.

2. RELATED WORK

Recommendation methods can be classified as Content-based recommendation [3]; Collaborative filtering [8], and Hybrid approaches [6, 17, 18]. The Content-based approaches make the prediction based on the similarity between the item and the user's content profile. However, it needs efforts to collect and extract knowledge from the content. Collaborative Filtering (CF) approaches have been successfully applied to several real world problems, such as Amazon's product recommendation [14], Netflix's movie recommendation [13]. CF methods are popular because it does not require domain knowledge, and can discover interesting associations that the Content-based methods couldn't. However, CF suffers from the Cold Start problem, in which few ratings can be obtained when a new item enters to the system. The Hybrid approaches try to combine the Content-based and CF approaches to overcome their limitations. For example, Claypool *et al.* [6] compute CF and Content-based components separately and combine their ratings linearly for the online news recommendation. Popescul, Schein *et al.* [17, 18] propose to unify CF and Content-based evidence by probabilistic mixture of aspects. But this kind of mixture models is prone to overfitting. Most of the methods mentioned above deal with explicit user ratings, whereas the focus of this paper is on implicit user feedback, which is easier to collect, and common in modern online information systems. In addition, although methods proposed in this paper have similarity with the Hybrid methods, most of the previous Hybrid approaches do not exploit the user information as rich as ours.

In the One-Class Collaborative Filtering setup, we only have positive examples, and usually the portion of un-labeled examples is large, which leads to an extremely sparse data matrix. Instead of simply ignoring all missing examples, recent researches focus on how to take advantage of the missing examples. Pan *et al.* [16, 15] propose to employ weighted matrix factorization approximation and negative example sampling to improve the result. The basic idea in that work is to treat all unknown examples as negative examples, and assign weights to quantify the relative contribution of these examples. Sindhwani *et al.* [20] jointly learn the non-negative matrix factorization model while searching for the best classification of the labels in missing data. Our work emphasizes on improving OCCF by incorporating rich user information. Despite its importance, none of the existing literature has addressed this problem seriously, partly due to the difficulty of collecting large-scale user information.

On the other hand, rich user information, such as search logs and clickthrough data has been reported to be able to improve the performance of personalized search in the field of information retrieval [10, 2, 19, 22]. Joachims [10] proposes to improve the retrieval quality of search engines by learning from the clickthrough data. Shen *et al.* [19] propose a decision-theoretic framework for optimizing search performance via user feedback. Tan *et al.* [22] integrate user search history into the query language model to improve the performance of language modeling approach. Agichtein *et al.* [2] examine the effectiveness of user behavior using a popular search engine, and demonstrate that web search ranking can be improved by incorporating such behavior. Personalized search and recommendation are similar but different in that search focuses on satisfying users' momentary information needs with regard to specific queries, while recommendation systems try to provide users with interesting contents based on their preferences.

3. ONE CLASS COLLABORATIVE FILTERING AND BASELINE MODELS

In this section, we first define the problem of OCCF and then introduce two representative approaches to it, which we will extend later by incorporating user information.

The goal of One-Class Collaborative Filtering is to predict the preference of a user on available items given that there are only positive implicit feedback examples in the data set. In this paper, we are interested in predicting users' future purchase preference based on the historical behavior of the users. Formally, let

- $U = \{u_1, u_2, ..., u_m\}$ be a group of users,
- $I = \{i_1, i_2, ..., i_n\}$ be a set of items,
- $R = (R_{ij})_{m \times n}$ be the user-item preference matrix, where $R_{ij} \in \{0, 1\}$.

The value of an element in R is 1 in a positive purchase record between some u and i, otherwise its corresponding element value in R is either 0 or 1. We aim at predicting a sorted list of top-k items $L = (L_1, L_2, ...L_k), L_i \in I$ which match the user's actual purchased items. We measure the prediction performance by Mean Average Precision and Mean Percentage Ranking, which will be described in detail in Section 6.2.

3.1 Neighbor-based Collaborative Filtering

Neighbor-based collaborative filtering systems have been introduced one decade ago and still are the most popular models for recommender systems. Neighbor-based CF systems can be classified in user-based and item-based systems depending on the way past preference judgments are used. A user- based system makes new predictions by first finding users with similar ratings to an active user and then takes a weighted combination of their ratings. More formally let u be the active user and i an item which is not rated by u. Then the predicted rating of u to i, $R(u,i)$ is obtained by

$$R(u,i) = \bar{r}_u + \frac{\sum_{a \in U}(r_{a,i} - \bar{r}_a)w_{u,a}}{\sum_{a \in U} w_{u,a}} \tag{1}$$

where $r_{a,i}$ is the rating of user a for item i, $\bar{r}_u$ and $\bar{r}_a$ are the mean ratings of users u and a and $w_{u,a}$ is the similarity weight between users u and a. On the other hand, in an item-based system predictions are made by finding similarly rated items and then calculating a weighted combination of their ratings:

$$R(u,i) = \bar{r}_i + \frac{\sum_{k \in I}(r_{u,k} - \bar{r}_k)w_{i,k}}{\sum_{k \in I} w_{i,k}} \tag{2}$$

where now $\bar{r}_i$ is the mean rating of item i and $w_{i,k}$ is the similarity weight between item i and k. Finding neighbors by using item-based similarity has computational advantage since in a typical CF setting, item space is more stable and bounded than user space. However, in a long-tail and highly dynamic market-place, this difference becomes less obvious. For example, the company from which we collect the data has 200M active users and 20M new listings added every month. Moreover, meta-data usually is available for users. Therefore, we decided to choose the User-based CF as our baseline model, which we call the UCF model.

3.2 Matrix Factorization for Collaborative Filtering

Matrix Factorization Models have been successfully applied to the Netflix movie recommendation [13]. It is superior to the neighbor-based models in reducing the Root Mean Squared Error (RMSE) where the explicit ratings of items are available. The basic idea of Matrix Factorization for explicit ratings is that the rating matrix R can be approximated by decomposing R into two low-rank matrices X and Y. For each R_{ij}:

$$R_{ij} \approx X_i Y_j^T, \tag{3}$$

Where $X = \{X_1, X_2, ..., X_m\}^T$ is a m by k matrix in which the i-th row X_i is a k-dimension vector representing a user's preference for latent factors. And $Y = \{Y_1, Y_2, ..., Y_n\}^T$ is a n by k matrix where the j-th row Y_j is a k-dimension vector representing an item's affiliation with latent factors. To avoid overfitting, we usually find X and Y with smaller values by Tikhonov-regularization [23], that is equivalent to solving the following unconstrained optimization problem:

$$\underset{X,Y}{\text{Argmin}} \left\{ \|R - XY^T\|_F^2 + \lambda(\|X\|_F^2 + \|Y\|_F^2) \right\} \tag{4}$$

where $||\cdot||_F$ is the Frobenius Norm of a matrix (Euclidean Norm) and λ is a coefficient controlling how much regularization is needed. The user-item matrix R is usually very sparse with a lot of missing value. In this case, there is no exact solution for X and Y. Alternating Least Squares (ALS)[7] is an effective iterative algorithm to solve the optimization problem. The ALS algorithm works as follows: it first assigns random values to matrix Y, and updates X by minimizing the Loss function defined by *Eq.* (4):

$$X = RY(Y^TY + \lambda I)^{-1}, \tag{5}$$

then fix X, update Y according to

$$Y = R^TX(X^TX + \lambda I)^{-1}, \tag{6}$$

the algorithm iterates these steps until X and Y converge to a local optimum.

In the One-Class Collaborative Filtering setting, there are several strategies to adopt the Matrix Factorization framework. One way is to simply treat all missing values in R as negative examples (AMAN). This method transforms the OCCF to the regular Matrix Factorization setting where the response variables only take 0 or 1 in value. In this case, the above algorithm can be directly applied to the OCCF problem. However, this strategy doesn't fully utilize the missing examples that would be positive ones. A better method proposed in [16] is to treat all missing values as negative, but with weights controlling their relative contribution to the loss function (wAMAN):

$$L_w(X,Y) = \sum_{i,j} W_{ij} \left((R_{ij} - X_i Y_j^T)^2 + \lambda(\|X_i\|_F^2 + \|Y_j\|_F^2) \right) \tag{7}$$

By minimizing $L_x(X,Y)$, X and Y can be solved via weighted low-rank approximation with ALS [21]. In this paper we include the AMAN method and wAMAN method as our two baselines in the Matrix Factorization methods. In wAMAN, we adopt the global weighting method for assigning the weights. We will elaborate two strategies of incorporating rich user information into the AMAN and wAMAN methods in Section 5.

4. REPRESENTATION OF USER INFORMATION

The online market-place records a large amount of user information in the interaction of its users. Such information, usually as a kind of implicit feedback, can be exploited to infer the user's purchase preference. This information includes search query logs, item clickthroughs, and transaction history. In this paper we sought to explore different ways to utilize these kinds of user information to improve the One-Class Collaborative Filtering problem.

4.1 Representing a User's Search Profile

Search keywords strongly imply a user's purchase preference. However, the search queries are usually short and compact, and vary by time period as the user's purchase interest shifts by time. To construct the search profile for user

u, let S_u be the set of keywords issued by u over a fixed period of time. It is very natural to represent the user's search profile as a vector space model as follows:

$$S^s(u) = ((w_{u1}, c_{u1}), (w_{u2}, c_{u2}), ..., (w_{ui}, c_{ui}), ...) \quad (8)$$

Where w_{ui} are keywords, c_{ui} are $TFIDF$ weightings, and $c_{u1} \geq c_{u2} \geq c_{u3}$.... By truncating $S^s(u)$ after the first n words, we create a signature of the n most popular words in the vocabulary of u. A limitation of this signature is that common words (e.g. NEW and NWT in the cellphone category) tend to appear in many profiles without contributing to the descriptive power of any of them. We use a standard weighting strategy to promote words that are descriptive of a particular user. For a given word w_{ui} we let

$$idf(w_{ui}) = \log\left(\frac{|U|}{|w_{ui} \in V_j|}\right), \quad (9)$$

where U is the set of all users and $|w_{ui} \in V_j|$ is the number of users whose vocabularies V_j contain w_{ui}. We then let

$$c_{ui} = tf(w_{ui}) \times idf(w_{ui}) \quad (10)$$

where $tf(w_{ui}) = |w_{ui} \in S_u|/|S_u|$ is the term frequency of word w_{ui} in the concatenated search queries of S_u.

4.2 Representing a User's Purchase and Browsing Profile

Besides the search query history, we can also assign profiles to a user according to his purchasing and item browsing history. The purchase history is recorded in the transaction logs and browing history is captured in the web behavioral logs. We argue that the items purchased by a user provide evidence for the user's preference for these items. In the context of our purchasing and clickthrough data, the most reliable and content-rich descriptors of an item lie in its title written by the seller. The search engine indexes items solely using these attributes, and since the majority of the purchased items are found using content search, the title of the item provides a significant source of data about the users' preference. Similar to the search keywords, we use word vector to represent a user, with an important difference that we just use the word count in computing $\bar{c}_{ui}$ instead of using term frequency:

$$S^p(u), S^t(u) = ((w_{u1}, \bar{c}_{u1}), (w_{u2}, \bar{c}_{u2}), ..., (w_{ui}, \bar{c}_{ui}), ...) \quad (11)$$

where $S^p(u)$ is calculated by the user's purchase history while $S^t(u)$ by the browsing history, and

$$\bar{c}_{ui} = |w_{ui} \in S_u| \times idf(w_{ui}) \quad (12)$$

The reasons for this treatment are as follows:

1. We want the signature to reflect the number of times an item was purchased by user u. Thus, a user who bought 10 items with identical titles, each containing one word i-pod should have a different signature from a user who bought only one item with the same title, although word frequencies are the same in these cases.

2. Since the titles are written by sellers rather than buyers, and the titles are rather compact (title length is limited to 55 characters), there is little danger of giving bias to longer titles. On the other hand, our weight will give a bias to words which appear in many titles, which is precisely our goal.

4.3 Similarity Measures

Our main idea of leveraging rich user information is to use the profiles of users as *extra* evidence for computing similarity of users as well as similarity between users and items. We can then combine these similarity values with a collaborative filtering algorithm to improve prediction accuracy. Here we define two similarity metrics: similarity between two users, and similarity between a user and an item.

- **Similarity between users**: Once we formulate the vector representation of user's profile, we can measure the similarity between users x and y using the *cosine* similarity:

$$sim(x, y) = cos(S_x, S_y) = \frac{S_x \cdot S_y}{||S_x||_2 \times ||S_y||_2} \quad (13)$$

- **Similarity between user and item**: Similarly, we can define a content profile for an item i as:

$$S(i) = ((w_{i1}, \hat{c}_{i1}), (w_{i2}, \hat{c}_{i2}), ..., (w_{ik}, \hat{c}_{ik}), ...) .$$

where $\hat{c}_{ik}$ is the number of times a word w_{ik} appears in the item description. Then we define the similarity between user u and item i by

$$sim(u, i) = cos(S_u, S_i) = \frac{S_u \cdot S_i}{||S_u||_2 \times ||S_i||_2} \quad (14)$$

where $S_u \in \{S^s_u, S^p_u, S^t_u\}$.

5. INCORPORATING RICH USER INFORMATION

Our goal is to produce a ranked list of items with high to low user preference. There are many strategies for combining all sources of information and output a ranking. Here we propose two methods.

5.1 User Information as Independent Evidence

The first strategy we employ is to treat all kinds of user information as independent evidence, and linearly combine their scores to produce a final ranking score. Given a user u and an item i, let F^{ucf}_{ui} be the ranking score obtained by the User-based Collaborative Filtering (UCF) method defined by *Eq.* (1). And let $X^1_{ui}, X^2_{ui}, ..., X^n_{ui}$ be the score from different information source (e.g. $X_{ui} = sim(u, i)$, the content similarity between S_u and i, where $S_u \in \{S^s_u, S^p_u, S^t_u\}$). Then the final score for item i is:

$$Y_{ui} = w^0_{ui} \cdot F^{ucf}_{ui} + w^1_{ui} \cdot X^1_{ui} + w^2_{ui} \cdot X^2_{ui} + ... + w^n_{ui} \cdot X^n_{ui}, \quad (15)$$

the wights $w^0_{ui}, w^1_{ui}, ...$ is trained using Gradient Descent under least square error to the ground truth in training dataset. We name this type of methods as *UCF+Features*

The same strategy can be applied to the Matrix Factorization baselines (AMAN and wAMAN). Once we obtain the X and Y matrix (please refer to Section 3.2), we can assign a score of user u to item i according to *Eq.* (3):

$$F^{AMAN}_{ui} = X^{AMAN}_u \times (Y^{AMAN}_i)^T, \quad (16)$$

and

$$F^{wAMAN}_{ui} = X^{wAMAN}_u \times (Y^{wAMAN}_i)^T, \quad (17)$$

then use *Eq.* (15) to output a final score. We refer this type of methods to MF+Features

5.2 Embedding User Information into Collaborative Filtering

Besides treating the user information as independent evidence, we can also directly embed this information into the User-based Collaborative Filtering or the Matrix Factorization framework. For the UCF framework, remember in *Eq.* (1), the user-user similarity can be an arbitrary function. So we can replace it with the similarity between users' profiles defined in *Eq.* (13). In addition, the user-item matrix itself can be replaced by a denser matrix in the clickthrough data. We call this kind of embedding methods UCFWithFeatures. For the MF framework, AMAN and wAMAN are two baseline methods representing the state of the art of the MF approach. Similar to wAMAN, we also treat all missing examples as negative examples. But instead of using a global weighting scheme, a better way for assigning the weight for each negative example is to look at the similarity between the user and the item: the more similar they are, the less weight we should assign to that negative example. And this similarity is measured by the content features. Hence we use

$$D_{ij} = 1 - sim(i,j) \tag{18}$$

for assigning the weights to the negative examples, and remaining the weights for positive examples to be 1. Where $sim(i,j)$ is defined by *Eq.* (13). With this important substitution, we then aim to find a solution by minimizing the following loss function:

$$L_d(X,Y) = \sum_{i,j} D_{ij} \left((R_{ij} - X_i Y_j^T)^2 + \lambda(\|X_i\|_F^2 + \|Y_j\|_F^2) \right) \tag{19}$$

We refer to these methods as MFWithFeatures. The low-rank matrices X, Y can be solved by weighted ALS as follows: In order to solve X, we first fix Y, and take derivatives of $L_d(X,Y)$ with respect to each entry of X,

$$\frac{1}{2}\frac{\partial L_d(X,Y)}{\partial X_{ir}} = \sum_j D_{ij}(X_i Y_j^T - R_{ij})Y_{jr} + \lambda(\sum_j D_{ij})X_{ir}, \forall 1 \le i \le m, 1 \le r \le k$$

Then for the i-th row in X, we have

$$\begin{aligned}\frac{1}{2}\frac{\partial L_d(X,Y)}{\partial X_i} &= \frac{1}{2}\left(\frac{\partial L_d(X,Y)}{\partial X_{i1}}, ..., \frac{\partial L_d(X,Y)}{\partial X_{ik}}\right) \\ &= X_i\left(Y^T \tilde{D}_i Y + \lambda\left(\sum_j D_{ij} I\right)\right) - R_i \tilde{D}_i Y\end{aligned} \tag{20}$$

where $\tilde{D}_i \in \mathbb{R}^{n\times n}$ is a diagonal matrix with entries of i-th row in D on the diagonal, I is a $k \times k$ identity matrix. Let the partial derivative $\frac{1}{2}\frac{\partial L_d(X,Y)}{\partial X_i} = 0$, we get

$$X_i = R_i \tilde{D}_i Y (Y^T \tilde{D}_i Y + \lambda(\sum_j D_{ij})I)^{-1}, \forall 1 \le i \le m \tag{21}$$

Similarly, by fixing X and taking derivative of $L_d(X,Y)$ with respect to each entry of Y, we have

$$Y_j = R_j^T \vec{D}_j X (X^T \vec{D}_j X + \lambda(\sum_i D_{ij})I)^{-1}, \forall 1 \le j \le n \tag{22}$$

where $\vec{D}_j \in \mathbb{R}^{m\times m}$ is a diagonal matrix with entries of j-th column in D on the diagonal. The algorithm for estimating the low-rank matrices X, Y is described in ***Algorithm*** 1. For the runtime performance, since in each iteration it takes time $O(k^2mn)$ to update the X (or Y), the computational complexity of running ***Algorithm*** 1 is $O(N_{itr}k^2mn)$. Where m, n is the number of users and items, repectively, k is the rank of matrix X and Y, and N_{itr} is the number of iterations.

Algorithm 1 Weighted ALS for AMAN

```
Require: user-item matrix $R$, weight matrix $D$,
    latent factors rank $k$
Ensure: low rank matrices $X$ and $Y$.

 1: Initialize $Y_{jr}$:
 2: $Y_{jr} := RandNum, \forall 1 \le j \le n, 1 \le r \le k$
 3: Initialize $D_{ij}, \forall 1 \le i \le m, 1 \le j \le n$:
 4: if $R_{ij} = 0$ then
 5:     $D_{ij} := 1 - sim(i,j)$
 6: else
 7:     $D_{ij} = 1$
 8: end if
 9: repeat
10:     $X_i := R_i \tilde{D}_i Y (Y^T \tilde{D}_i Y + \lambda(\sum_j D_{ij})I)^{-1}, \forall 1 \le i \le m$
11:     $Y_j := R_j^T \vec{D}_j X (X^T \vec{D}_j X + \lambda(\sum_i D_{ij})I)^{-1}, \forall 1 \le j \le n$
12: until $X$, $Y$ converge
13: return $X$, $Y$
```

5.3 Baseline Methods and Models with Rich User Infomation

Here we fully describe the baseline models and the advanced models incorporating rich user information. We first introduce 6 baseline models, which generate recommendations from different modeling aspects. 4 of these models are considered as weak baselines (PopRank, SchKW, CT, AMAN), and the remaining 2 are considered as strong baselines (UCF and wAMAN).

- **PopRank**: The first method is sorting all items based on their popularity, so that the top recommended items are the most popular one in terms of the number of times bought by users. This simple measure is supposed to have reasonable performance, as people tend to concentrate on few popular items such as IPhone. We use it as a weak baseline model.

- **SchKW**: In order to investigate whether users' search keyword history helps improving the prediction performance, we need to know the performance of using search keyword history alone. This is a Content-based model which computes ranked list of items based on how similar of the item title description to the user's search keyword profile, as defined in *Eq.* (14)

- **CT**: Similar to search keyword history, a typical e-commerce system would have recorded the Clickthroughs of its users. The CT model is to use the content, i.e. title of the browsing items, to compute the ranked list of items based on the similarity between the title description of a predicted item and the user's browsing profile, as defined in *Eq.* (14).

- **AMAN**: AMAN stands for All Missing as Negative example. It is a weak baseline model for the OCCF problem setting. In AMAN, all non-positive examples are assigned to 0, and the Alternating Least Squares optimization procedure is adopted to solve the factorization. An ordered list of items can be obtained by sorting the scores calculated by multiplying two low-rank matrices.

- **UCF**: Neighbor-based Collaborative Filtering is one of the most popular and successful models for recommendation. Here we choose the User-based Collaborative Filter described in Sec. 2 as our strong baseline model. We explored several variants of this scheme, such as the number of neighbors, and found that the performance is not sensitive to the large number of neighbors. So we decide using the most common form of the User-based Collaborative Filtering (UCF), with 30 nearest neighbors. More neighbors are also tested, but there is no significant difference than using 30 neighbors; so we use 30 for the sake of efficiency. The vector cosine similarity is used as the similarity measurement.

- **wAMAN**: Weighted version of AMAN, as described in Section 3.2

The following models with rich user information can be classified by two categories: one is called UCF+Features, in which the models are associated with the User-based Collaborative Filtering. The other one is called MF+ Features, in which the models are related to Matrix Factorization. We first list the UCF+Features models.

- **UCF+SchKW**: One of the major goals of this paper to assess the extent to which users' search keyword history helps the prediction of users' future purchase behavior. As described in Sec. 3, this model is a linear combination of the scores from the User-based Collaborative Filtering and the Content-based method. The Content-based method computes the similarity between the predicted item's title description and the Keywords in user's Search History. The optimal value of the weight w is trained on the training (7 weeks). Parameter setting of the UCF component is the same as the baseline UCF model.

- **UCF+SchKW+CT**: The clickthrough behavior is a very valuable source of information for probing user's preference. This model produces a ranked list of items by linearly combining scores from clickthrough and the other two components listed above. We use linear regression (least square error) method to obtain the optimal weights among these components from the training set.

- **UCFWithSchKW**: Besides using search keyword history as an independent source of information, users' search keyword history can be directly embedded into the User-based Collaborative Filtering framework. Remember that the similarity function between users can be an arbitrary function; we replace the similarity function by a Content-based similarity metric between users based on their search keyword history.

- **UCFWithCT**: Similarly, clickthrough data be directly embedded into the User-based Collaborative Filtering framework by replacing the sparse transactional user-item binary rating matrix with a denser user-item clickthrough data matrix.

- **UCFWithCT+SchKW+CT**: This method is similar to the UCF+SchKW+CT method, but different in the user-item rating matrix used. Here we use the clickthrough data matrix instead of the transactional user-item matrix. The clickthrough data matrix is less sparse and it would include many records in the transactional user-item matrix: a user usually browses the item before actually purchasing it. But the confidence of the user's preference on an item is lower in the clickthrough data.

Table 1: Advanced Methods Summary

Methods	Baseline	Combination Method	User Info
UCF+SchKW	UCF	Linear	Query
UCF+CT	UCF	Linear	CT
UCF+SchKW+CT	UCF	Linear	Query, CT
UCFWithSchKW	UCF	Embed	Query
UCFWithCT	UCF	Embed	CT
UCFWithCT+SchKW+CT	UCF	Embed and Linear	Query,CT
wAMAN+SchKW	wAMAN	Linear	Query
wAMAN+CT	wAMAN	Linear	CT
wAMAN+SchKW+CT	wAMAN	Linear	Query,CT
wAMANWithSchKW	wAMAN	Embed	Query
wAMANWithCT	wAMAN	Embed	CT

On the other hand, the MF+ Features category has 5 related models as listed below. Since they are defined in a similar way as the UCF+Features, we only briefly list them and point out some important differences.

- **wAMAN+SchKW**: Linearly combining scores from wAMAN and SchKW. Please refer to UCF+SchKW.

- **wAMAN+CT**: Linearly combining scores from wAMAN and CT. Please refer to UCF+CT.

- **wAMAN+SchKW+CT**: Linearly combining scores from wAMAN and SchKW and CT. Please refer to UCF+SchKW+CT

- **wAMANWithSchKW**: wAMAN with weighting scheme replaced by SchKW. A specific weighting from user u to item i is defined by *Eq.* (14). Solution can be obtained by *Algorithm* 1.

- **wAMANWithCT**: Similar to wAMANWithSchKW, but instead using CT for deriving the weighting scheme.

We summarize important aspects of all advanced models in Table 1.

6. EXPERIMENTAL SETUP

6.1 Datasets

We test our proposed method on a large-scale dataset from the market-place of a major e-commerce company. The data set includes transactions, user search logs and clickthrough records in the CELLPHONE & ACCESSORIES catalog in

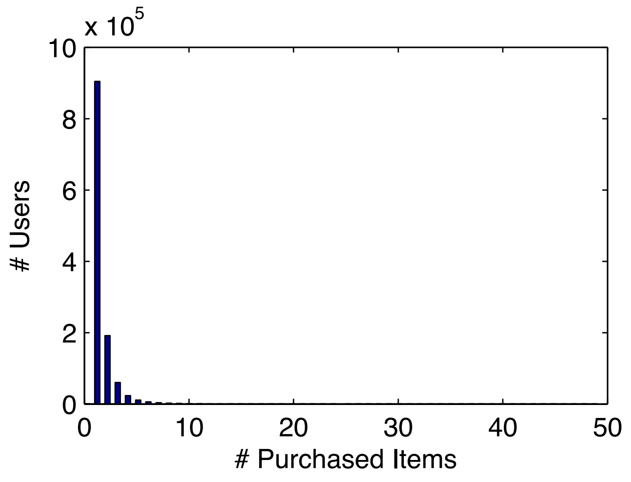

Figure 1: Histogram of the training set

a period of 9 weeks. We split the data into three parts, the first 7 weeks of data is used for training, the 8-th week's data a validation set, and the 9-th week's data is for testing. We first remove the users who have no transactions in either of the data sets, and remove the users who have no search queries in the training set. Finally there are 1.29 million users and 1.85 million items in the training and validation set, and 17,135 users and 274,830 items in the test set. Please note that the data sets are very sparse, with about 49.3% of the users only bought one item in the training set. The items in the datasets are transformed to unique items by a generative clustering method, resulting in 20,409 unique items (we use the term item briefly in the rest of the paper). Figure 1 shows some statistics of the dataset. The reasons and process of doing the unique item mapping are as follows: for an online marketplace featuring long-tail items, the life span of the item is ephemeral, so is the user-item relationship. This makes the user-item matrix of transactional counts extremely sparse (order of magnitude sparser than Netflix data), thus precluding the application of some traditional CF approaches such as SVD-based low-rank projection. To address this issue, Chen and Canny [5] have proposed a generative clustering algorithm mapping ephemeral items to more persistent latent product concepts. Specifically, the clustering algorithm uses product-to-item likelihood as the distance metric, where an item title is assumed to be generated from a latent product, word-by-word or property- by-property, following appropriate parametric distributions.

6.2 Evaluation Metrics

We use two standard measures to evaluate the prediction accuracy: Mean Average Precision, and Mean Percentage Ranking.

1. **Mean Average Precision (MAP)**: Mean Average Precision accesses the overall precision performance based on precision at different recall levels. It calculates the mean of the average precision (AP) over all users in the test set. AP for user u is computed at the point of each of the preferred items in the ranked list:

$$AP_u = \frac{\sum_{i=1}^{N} prec(i) \times pref(i)}{\#\text{ of purchased items}} \quad (23)$$

where $prec(i)$ is the precision at ranked position i, $pref(i)$ is a binary preference function at position i. The MAP is the mean of AP_u over all users.

2. **Mean Percentage Ranking (MPR)**: Because of the nature of the One Class Recommendation, we don't have reliable feedback for user's preference for items. A user who hasn't bought an item doesn't necessary mean he doesn't like it. On the other hand, due to the money commitment, the purchase behavior is a strong indication of user preference over the purchased item. Here we use another recall-oriented metric called Mean Percentage Ranking, which is used in [11] and [4], to measure the user satisfaction of items in an ordered list. Let $rank_{ui}$ be the percentile-ranking of item i within the ordered list of all items for user u. $rank_{ui} = 0\%$ means that item i is most preferred by user u. The higher ranking (until $rank_{ui} = 100\%$ is reached) indicates that i is predicted to be less desirable for user u. The way of calculating the MPR in our experiment setup is as the following: for each actual pair of a user and the purchased item, we randomly select 1000 other items, and produce an ordered list of these items. Then, we keep track of where the actual purchased item is ranked, and calculate the expected percentage ranking for all users and items:

$$MPR = \frac{\sum_{u,i} r_{ui} \times rank_{ui}}{\sum_{u,i} r_{ui}} \quad (24)$$

Where r_{ui} is a binary variable indicating whether user u purchases item i. It is expected that a randomly produced list would have a MPR of around 50%.

7. RESULTS AND DISCUSSION

7.1 Results On the UCF Based Models

We first analyze the effect of treating user information as independent evidence to the User-based CF model. Table 2 indicates that Search Keywords as independent information (UCF+SchKW) improves the result significantly, the absolute MAP gain over the UCF baseline reaches 0.0243, and the MPR gain over UCF is 6.5%. Compared to the Search Keywords, the Clickthrough history (UCF+CT) only outperforms the UCF slightly, because the content in the item title might not be as informative as the Search Keywords, or people tend to not purchase duplicated items. In addition, combining three sources of information (UCF+SchKW+CT) gets the best result, suggesting the effectiveness of the inclusion of rich user information.

Table 3 summarizes the results for embedding user information into the UCF model. Interestingly, by only replacing the user-item matrix with a denser Clickthrough data matrix, the MAP improves by 0.0343 and the MPR by 10.4% over the UCF model. It suggests that the extremely sparse dataset is the major obstacle for Collaborative Filtering. On the other hand, the way for handling other sources of information will significantly affect the result. For example, while

Table 2: Performance of UCF+Features methods

Methods	MAP	Gain in UCF	MPR %	Gain in UCF %
PopRank	0.0120	-	35.8	-
UCF	0.0513	-	28.5	-
SchKW	0.0486	-	31.6	-
CT	0.0302	-	30.9	-
UCF+SchKW	0.0756	**0.0243**	22.0	**6.5**
UCF+CT	0.0577	0.0064	27.0	1.5
UCF+SchKW+CT	0.0794	**0.0281**	20.3	**8.2**

Table 3: Performance of UCFWithFeatures methods

Methods	MAP	Gain in UCF	MPR %	Gain in UCF %
PopRank	0.0120	-	35.8	-
UCF	0.0513	-	28.5	-
UCFWithSchKW	0.0501	-0.0012	31.5	-3.0
UCFWithCT	0.0856	**0.0343**	18.1	**10.4**

the Search Keywords effectively improve the result when using as independent information, it drops by 0.0012 in MAP and 3.0% in MPR over the baseline when acting as an embedded component of the UCF model: it does not alleviate the sparsity problem of the data.

We can also see in Table 4 that adding all user information achieves the best result over UCF baseline. But the best MAP (0.1037) is still very low compared to other datasets. It indicates that recommending items in a large-scale online market-place is a very difficult problem.

7.2 Results On the MF-based Models

Table 5 and Table 6 show the MAP and MPR results on the Matrix Factorization-based models. We try different number of factors k from 40, 60 to 100, and find that the performance increases as the k increases. $k = 100$ is chosen in reporting the results. Firstly, the un-weighted version of MF model AMAN performs badly in this case, suggesting simply treating all missing examples as negative examples without weighting does not fit for highly sparse OCCF problem. Secondly, previous studies from [16] [9] show that the weighted MF methods perform well in the OCCF problem; however, results in this study show that the wAMAN model with global weighting scheme performs not as well as the UCF model in the same test set (0.0314 vs 0.0513 in MAP and 33.4% vs 28.5% in MPR). This result suggests that in the MF model, it is not guaranteed to capture the relevant information between users and items by simply assigning a uniform weight in the negative examples; it might vary from dataset to dataset. Whereas the UCF model could perform more robust across datasets since the relevant information from neighborhoods is easier to explain. On the other hand, in Table 5, Search Keywords and Clickthrough again demonstrate the effectiveness as additive evidence to the baseline model. Still, Search Keyword is more effective than Clickthrough, delivering significantly higher gain over the baseline (0.0269 vs 0.0036 in MAP gain). In addition, adding up all information (wAMAN+SchKW+CT) still gets the best result.

Table 4: Comparison of the well-performing models

Methods	MAP	Gain in UCF	MPR %	Gain in UCF %
UCF	0.0513	-	28.5	-
UCFWithCT	0.0856	0.0343	18.1	10.4
UCFWithCT +SchKW	0.1022	0.0509	15.4	13.1
UCFWithCT +SchKW+CT	0.1037	**0.0524**	15.0	**13.5**

Table 5: Performance of MF+Features methods

Methods	MAP	Gain in wAMAN	MPR %	Gain in wAMAN %
AMAN	0.0085	-	38.2	-
wAMAN	0.0314	-	33.4	-
SchKW	0.0486	-	31.6	-
CT	0.0302	-	30.9	-
wAMAN+SchKW	0.0583	**0.0269**	27.3	**6.1**
wAMAN+CT	0.0350	0.0036	29.0	4.4
wAMAN +SchKW+CT	0.0621	**0.0307**	23.3	**10.1**

Table 6: Performance of MFWithFeatures methods

Methods	MAP	Gain in wAMAN	MPR %	Gain in wAMAN %
AMAN	0.0085	-	38.2	-
wAMAN	0.0314	-	33.4	-
wAMANWithSchKW	0.0607	**0.0293**	25.5	**7.9**
wAMANWithCT	0.0577	0.0263	28.1	5.3

7.3 When The Models Perform Well

The analysis of results above supports the idea that rich user information is effective in solving the OCCF problem. We want to further investigate in what scenarios the rich user information is most effective. As the results show, MF-based models are not as good as UCF-based models in this case, therefore we put our focus on the UCF-based models. The typical transactional user-items matrix in e-commerce is extremely sparse. There are 1.3 million users and 17 thousand unique items, but only 1.8 million items purchased. The histogram in Figure 1 shows the statistics in the training dataset: it is highly skewed, with 49.3% of people only buying one item in 7 weeks. It is known that the performance of the Neighbor-based CF drops drastically when there is no neighborhood supports on the items. In that scenario, the content similarity between the item and the user search profile might promote the ranking score from 0 to a reasonable level.

To test this hypothesis, we stratify the users in the test set into three sub-groups according to the scores which the UCF model outputs. These scores indicate the support from neighbors. We observe that the range of that score is from 0 to 1, and we group them into [0, 0.3), [0.3, 0.5) and [0.5, 1] empirically, representing low, middle and high supports.

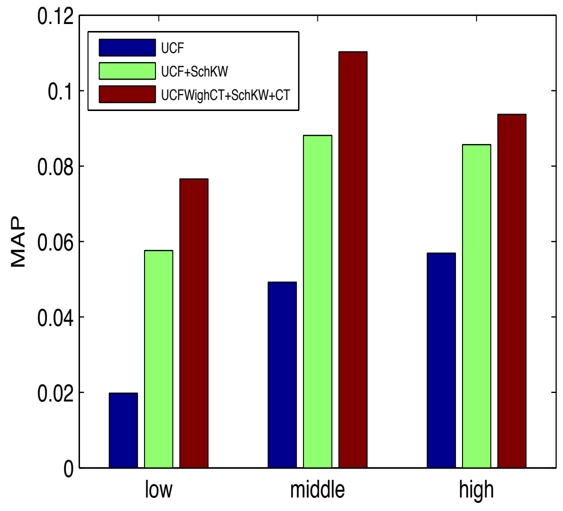

Figure 2: MAP in different UCF groups

Then, we re-run two well-performing models: UCF+SchKW and UCFWithCT+SchKw+CT, as well as the UCF model. Figure 2 summarizes the experimental results. We can see clearly in Figure 2 that while the two additive models gain over UCF in both groups of middle and high neighbor support. They gain significantly in the group with low neighbor support. This result indicates the complementary nature of the neighbor-based CF model and the information in Search Keywords and Clickthroughs. As an extreme case, if the score from UCF model is 0, other sources of information of the user-item association would help overcoming the problem.

7.4 How Much Information is Needed

If the rich user information is effective, an interesting question is how much user information is needed? This question is difficult to answer if all kinds of user information are involved. So we control all other variables, and only select the Search Keywords, which is an important type of user information, to access how the performance changes with the amount of user information varies. In this experiment, we divide the users to different groups according to the number of Search Keywords they have in the training set. A step of 50 keywords is used, which results in 30 user groups. And we use the *UCF+SchKW* to compute the average MAP of each group. Results are plotted in Figure 3. This experiment shows very interesting results: the model performs well on people who have small amount of Search Keywords, and performance drops in the groups of people who have relatively more Search history; and interestingly, performance grows again in the groups who have a lot of search history. The last groups of people represent the frequent buyers; they tend to buy similar things repeatedly so that the accumulated Search history would help in recommending items to them.

7.5 Time Effects of the Search History

Research work in [12] shows the importance of time dynamics in Collaborative Filtering. The above results show

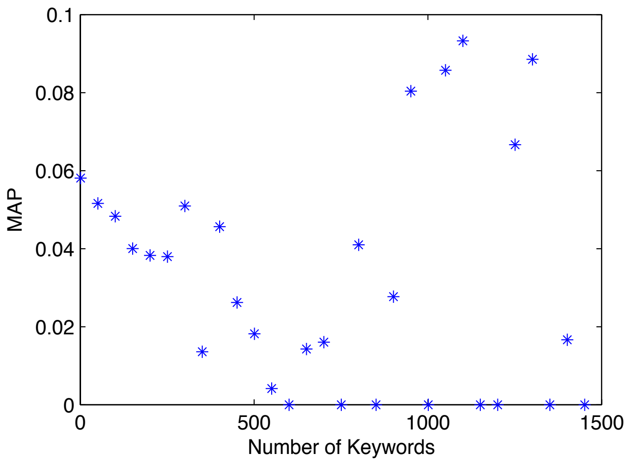

Figure 3: Number of Keywords VS Performance

Table 7: UCF+SchKW model on 4 Periods

	7 weeks	4 weeks	2 weeks	1 week
MAP	0.0782	0.0788	0.0795	0.0798
MPR%	21.8	21.5	21.0	21.2

the importance of the search keywords in overcoming the limitation of the User-based CF. We further want to know the time effects on the search keyword information. We select search keywords from 4 periods of time (7 weeks, 4 weeks, 2 weeks and one week), and randomly select 2000 users who have at least one search keyword in his search history. Table 7 shows the performance on the *UCR+SchKW* model. Interestingly, long search history doesn't produce better results. This result suggests that we can only keep the most recent search logs of the users and can still obtain reasonable performance in recommending items to users.

8. CONCLUSIONS AND FUTURE WORKS

In this paper we propose two major strategies of incorporating rich user information to improve the OCCF performance: one is treating the information as independent evidence and combining the scores from all sources of user information to produce the final recommendation. The other is to tightly include the rich user information into the models. After careful analysis of the results, we have found the following conclusions: (1) Rich user information, such as Search Keywords and Clickthrough data, is very effective to overcome the sparsity in One-Class Collaborative Filtering. (2) Rich user information helps the most when the neighbor-based methods have very low support from its neighbors. (3) The short term search query history tends to perform as well as, if not better than, the long term history; so we can leverage users' recent search query history to make recommendations.

In the future, we plan to investigate more sophisticated probabilistic models for unifying all available resources. Additional user information is shown to be effective, and more sophisticated models have the potential to further improve the recommendation performance. We are also interested in designing efficient and scalable algorithms for combining rich user information, since modern large-scale information systems desire scalable recommender solutions.

9. ACKNOWLEDGMENTS

This paper is based upon work supported in part by the National Science Foundation under grant CNS-0834709. We also thank Huizhong Duan from the University of Illinois at Urbana-Champaign and Maks Ovsjanikov from Stanford University for their help in the data analysis and discussion.

10. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

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

[3] M. Balabanovíc and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[4] W. Chen, J.-C. Chu, J. Luan, H. Bai, Y. Wang, and E. Y. Chang. Collaborative filtering for orkut communities: discovery of user latent behavior. In *WWW '09: Proceeding of the 18th International World Wide Web Conference*, pages 681–690. ACM, 2009.

[5] Y. Chen and J. F. Canny. Probabilistic clustering of an item. *U.S. Patent Application 12/694,885*, filed: Jan, 2010.

[6] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*, August 1999.

[7] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(4):489–498, 1979.

[8] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[9] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 263–272, 2008.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, NY, USA, 2002. ACM Press.

[11] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, NY, USA, 2008. ACM.

[12] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, New York, NY, USA, 2009. ACM.

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[14] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan/Feb 2003.

[15] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 667–676. ACM, 2009.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 502–511, 2008.

[17] A. Popescul, L. Ungar, D. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 437–444, 2001.

[18] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002. ACM Press.

[19] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 824–831, New York, NY, USA, 2005. ACM.

[20] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. A family of non-negative matrix factorizations for one-class collaborative filtering. In *ACM RecSys 2009*, 2009.

[21] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, pages 720–727. AAAI Press, 2003.

[22] B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 718–723, New York, NY, USA, 2006. ACM.

[23] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems.* John Wiley & Sons, 1977.