Comparing Collaborative Filtering Methods Based on User-Topic Ratings

Tieke He, Xingzhong Du, Weiqing Wang, Zhenyu Chen, Jia Liu∗
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
Software Institute, Nanjing University, Nanjing, China
∗liujia@software.nju.edu.cn

Abstract

User based collaborative filtering (CF) has been successfully applied into recommender system for years. The main idea of user based CF is to discover communities of users sharing similar interests. However, existing user based CF methods may be inaccurate due to the problem of data sparsity. One possible way to improve it is to append new data sources into user based CF. Tags which are added and generated by users is one of the new sources. In order to utilize tags effectively, user-topic based CF is proposed to extract features behind tags, assign them to topics, and measure users’ preferences on these topics. In this paper, we conduct comparisons between two user-topic based CF methods based on different tag-topic relations. Both methods calculate user-topic preferences according to ratings of items and topic weights. Experiments are conducted on the data set of MovieLens. The results show that user-topic based CF method is better than user based CF both in computational efficiency and recommendation effect. The effects are significant especially when each tag belongs to multiple topics.

Keywords: Recommender Systems, Collaborative Filtering, Topic Model, Tag

1. Introduction

Recommender systems [1] play an important role in E-commerce. Amazon1, one of the most famous online retailers, recommends products to customers. And Netflix2, one of the biggest online movie renting service providers, recommends movies to users. As one of the most important parts, recommendation algorithms have also achieved noticeable progresses.

Collaborative filtering (CF) [14] is one of the most common approaches of recommender systems. The main idea of CF is that similar users may share similar user preference patterns [6]. In CF method, user preference is represented by a vector, in which each entry indicates user’s rating on a specific item. Similarities between users could be defined by the distances between user rating vectors. Such similarities could be inaccurate due to the data sparsity problem. And the rapid growth of the amount of items may lead to low computational efficiency.

With the rapid development of Web2.0 techniques, social annotation systems are showing their effects both for users and recommender systems. Researchers have introduced many approaches on using tags in recommender systems [10, 8]. Certain tags are always related to a group of relatively limited topics, and are related to item features. Such as a movie about intelligent robots may relate to topics on science fiction or high technology. Based on these facts, we assume that a user likes some topics before he or she likes some certain items. How much a user likes an item could be measured by the related tag ratings and the correlation of the topic and the item. Based on this assumption, user-topic ratings could enhance CF method.

In user-topic approach, each item is treated as a document, and collection of items as corpus. Tags are regraded as the words for the documents. There are different ways to improve CF by user-topic ratings: one tag to one topic – hierarchical clustering and one tag to multiple topics – Latent Dirichlet Allocation (LDA) [2, 16].

LDA could discover topics of items. It generates a series of analysis results including document-topic proportions, topic-words proportions and so on. LDA could regard the document-topic proportions as the correlations of topics and movies. With these data, user-topic ratings are inferred, and then user similarity could be computed.

In this paper, we conduct experiments on two different user-topic based CF, and compare them to user-based CF. The sparsity of data set is controlled carefully, and results are evaluated by classical metrics. Results show that keeping other factors unchanged, either user-topic based CF is better than user-based CF. In these two user-topic based CF methods, LDA could improve CF better.

1http://www.amazon.com
2http://www.netflix.com
The main contributions of this article are summarized as follows.

1. Based on inferred tag ratings, we choose two different topic model – hierarchical clustering and LDA to implement two different user-topic based CF.

2. We compare user-topic based CF to user based CF, and show the effectiveness of user-topic based CF. We also illustrate that LDA is better than hierarchical clustering in user-topic based CF.

This paper is organized as follows. Section 2 gives a brief introduction to user-based collaborative filtering method. Section 3 introduces two methods to do topic extraction, tag clustering and LDA model. Section 4 shows our experiments and the analysis of experimental results. A conclusion of our work is showed in Section 5.

2. User-Based Collaborative Filtering

Collaborative filtering (CF) has been widely used in business situations. CF methods consist of User-Based CF, Item-Based CF and other variations. The main idea of User-Based CF is that similar users may share similar preferences. It calls for calculating similarities between users by their ratings on items. A higher similarity between two users means they are much more similar.

Given user list \( U = \{u_1, u_2, \ldots, u_n\} \) and item list \( \{i_1, i_2, \ldots, i_m\} \), a user \( u \) can be represented by his rating vector \( r_u = (r_{u,1}, r_{u,2}, \ldots, r_{u,m}) \). \( r_{u,i} (i \in \{1, m\}) \) stands for user \( u \)'s rating on the \( i \)th item. The similarity between user \( u \) and \( v \) can be measured by the distance between \( r_u \) and \( r_v \), known as the Pearson's Correlation Coefficient. Equation 1, which states the similarity between user \( u \) and user \( v \), denotes the computation of Pearson’s Correlation Coefficient.

\[
\text{sim}_{u,v} = \frac{\sum_{g \in G} (r_{u,g} - \bar{r}_{u,G})(r_{v,g} - \bar{r}_{v,G})}{\sqrt{\sum_{g \in G} (r_{u,g} - \bar{r}_{u,G})^2} \sqrt{\sum_{g \in G} (r_{v,g} - \bar{r}_{v,G})^2}} \tag{1}
\]

To note the items rated by both of \( u \) and \( v \) (known as the co-rated items), \( G \) is used to stand for the set of co-rated items of \( u \) and \( v \). \( \bar{r}_{u,G} \) and \( \bar{r}_{v,G} \) represents \( u \)'s and \( v \)'s average rating of \( G \). A user may have personal bias on rating, which means a user may always tend to give high or low ratings. To alleviate this situation, each user’s average rating is subtracted from his or her ratings.

\[
\text{pred}(u, i) = \bar{r}_{u,G} + \frac{\sum_{v \in N_{sim}} \text{sim}_{u,v} \cdot (r_{v,i} - \bar{r}_{v,G})}{\sum_{v \in N_{sim}} |\text{sim}_{u,v}|} \tag{2}
\]

Equation 2 demonstrates how to predict a user \( u \)'s rating on an item \( i \). When user \( u \)'s rating \( r_{u,i} \) on an item \( i \) is predicted, only a set of similar users to user \( u \) which are denoted by \( N_{sim} \) are taken into prediction. Items can be recommended to user \( u \), if their predicting ratings are higher than user \( u \)'s average rating.

However, there always exist a lot of items which user \( u \) has rated but \( v \) has not, or vice versa. This could lead to inaccuracy due to too few co-rated items. A new user has no co-rated items with other users which makes it even worse.

3. User-Topic Based Collaborative Filtering

User-topic based CF is an improvement to the user-based CF method. Instead of using the sparse user-item ratings, we use inferred user-topic ratings to compute user similarity. Traditional prediction methods are used to generate recommendations for users. This section presents a detailed explanation of user-topic based CF.

3.1. Topic Extraction

The basic motivation behind user-topic based CF is to measure to what extent a user likes a specific topic. The most important is to extract the abstract topics, also known as latent semantics inside the items. Social annotation systems provide a chance. In social annotation systems, users use tags to express their personal viewpoints on items. This makes it possible for us to extract topics from these tags.

There are two methods to carry out this work – tag clustering and topic models. More specifically, hierarchy clustering [13] and the Late Dirichlet Allocation (LDA) model [2]. Both methods are state-of-art algorithms. This section gives the process and formulation of these two methods, including mathematical notations.

3.2. Hierarchical Clustering on Tags

Hierarchical Clustering is a simple while useful clustering algorithm [13, 15]. In detail, there are two main types of Hierarchical Clustering which are top-down approach and bottom-up approach. We adopt the bottom-up approach in this paper. Given a set of tags \( T = \{t_1, t_2, \ldots, t_n\} \), \( t_i \) denotes a certain tag. At first, each tag is placed in a single topic, so the initial set of clusters is

\[
C = \{c_1 = \{t_1\}, c_2 = \{t_2\}, \ldots, c_n = \{t_n\}\}
\]

In each iteration, two nearest clusters are picked out and aggregated together. So the distance between two clusters should be defined. In this paper, the distance between clusters is computed by tags. [3] proposed a method named co-occurrence probability. A tag \( z \) is denoted as an vector \( p_z \),
which stands for its co-occurrence probabilities distribution. Co-occurrence probability \( p_z(t) \) means the probability that tag \( t \) is tagged if tag \( z \) is tagged on one item. It is defined in Equation 3.

\[
p_z(t) = \sum_{m \in I} q(t|m)Q(m|z) \quad (3)
\]

\[
q(t|m) = \frac{\text{number of times tag } t \text{ on item } m}{\text{overall number of tags on item } m} \quad (4)
\]

\[
Q(m|z) = \frac{\text{number of times tag } z \text{ on item } m}{\text{number of times tag } z \text{ on all items}} \quad (5)
\]

\[
z' \text{'s co-occurrence probabilities can be expressed in a vector } p_z \text{ and its feature is described in Equation 6.}
\]

\[
\sum_{i=1}^{n} p_z(t) = 1 \quad (6)
\]

Since tags could be represented as co-occurrence probability distribution, the distance between tags could be computed by Jensen-Shannon divergence (JSD). A correct JSD result is a finite value ranging from zero to one. Equation 7 is the computation of JSD.

\[
JSD(A||B) = \frac{1}{2} D(A||M) + \frac{1}{2} D(B||M) \quad (7)
\]

\[
M = \frac{1}{2} (A + B) \quad (8)
\]

\[
D(A||B) = \sum_{i=1}^{n} A(i) \ln \frac{A(i)}{B(i)} \quad (9)
\]

\[
A \text{ and } B \text{ are the co-occurrence distributions of two tags. } n \text{ is the dimension of } A \text{ or } B. \text{ In Equation 9, if } A(i) \text{ and } B(i) \text{ are both 0 then we define } \frac{A(i)}{B(i)} \text{ as 1. We now can define the distance between clusters based on the distance between tags. Given cluster } c_i \text{ and } c_j \text{ with } N_i \text{ and } N_j \text{ tags respectively. The distance between } c_i \text{ and } c_j \text{ is defined in Equation 10.}
\]

\[
\text{Dis}(c_i, c_j) = \frac{\sum_{t_1 \in c_i, t_2 \in c_j} JSD(t_1||t_2)}{N_i \times N_j} \quad (10)
\]

A set of clusters are generated after all the iterations accomplished. Next step is to compute each topic’s weight on each item. Equation 11 describes the computation of tag \( z \)’s weight on item \( m \). Equation 12 describes the computation of topic \( c \)’s weight on item \( m \).

\[
w_m(t) = \frac{\sum_{z \in T_m} n(m, z) \times p_z(t)}{\sum_{z \in T_m} n(m, z)} \quad (11)
\]

In Equation 11, \( T_m \) stands for all tags applied to item \( m \), and \( n(m, z) \) denotes the times that tag \( z \) has been tagged on item \( m \). In Equation 12, topic weight is the sum of all included tag weights.

### 3.3. Latent Dirichlet Allocation

This section introduces how to get the item-topic weights by topic model. Topic models have been widely used in many areas, especially in the area of NLP (Natural Language Processing). [4] proposed the Latent Semantic Index (LSI). They used the SVD method to do dimension reduction. Actually, LSI is not a style of topic model but the basis of probabilistic latent semantic analysis (pLSI). pLSI [7] does similar work with LSI, but pLSI is a generative probability model.

Latent Dirichlet Allocation (LDA) is a popular topic model. It performs well in many research works. In LDA, each document is drawn from a distribution over a specific group of topics, and these topics are shared by all the documents in the corpus. Each topic is a distribution over a vocabulary which contains all the unique words in the corpus. Given a set of items as a corpus, each item in the set as a document and tags of the items as words, it is natural to use the LDA model in our scenario. A basic assumption for the LDA model is “bag-of-words”, which means the order of a certain document’s words can be neglected. So the tags are disordered in our scenario.

Original LDA is proposed by Blei based on the EM algorithm. [5, 9] proposed a simple parameters estimation method called the Gibbs Sampling. And [11] proposed an implementation of the above method. It is applied in our experiment for its usability. Although the LDA model can infer both document-topic distribution and topic-word distribution at the same time, the former is essential for the following computation. The LDA model assumes that document-topic distribution is drawn from a Dirichlet Distribution. The sum of all topic probabilities in a certain document is 1 according to the definition of the Dirichlet Distribution. With this definition, it is unnecessary to normalize the item-topic probabilities, they can be directly used as item-topic weights.

In this paper, an implementation of LDA – GibbsLDA++ is applied. The input of the GibbsLDA++ is the corpus in a text format, including each tag’s count in each item. It also needs several input parameters: the number of topics, hyper-parameter \( \alpha \) and \( \beta \). We find that 100 is an appropriate number of topics according to the experimental results, and \( \alpha \) and \( \beta \) are set as default value of 50 and 0.1 respectively. 2000 Gibbs Sampling iterations were conducted. After that, GibbsLDA++ gives
the document-topic proportions and they could be transformed into item-topic weights.

3.4. Inferred user-topic Ratings

Item-topic weights could be generated from the above methods. Next step is to infer the user-topic ratings, which is the foundation of user similarity computation. Each item relates to the universal set of topics, but it has some outstanding topics or known as high-weighted topics. If a user rates high for an item means that he or she likes that item, he would prefer higher-weighted topics to lower-weighted topics of that item, which indicates that the user is attracted by the item’s outstanding topics with a high probability. Therefore, inferring user u’s rating on topic t could be calculated by equation 13.

\[
r_u(t) = \frac{\sum_{m \in I_u} w(m,t) \cdot r_{u,m}}{\sum_{m \in I_u} w(m,t)}
\]

(I_u stands for all the items that user u has rated. w(m,t) denotes the weight of tag t on item m. r_{u,m} represents u’s rating on m. The user-topic ratings are denser than user-item ratings, and applying user-topic ratings can well improve recommendation accuracy. Finally, a vector r_u is generated for each user to express his ratings on topics, which will be used in the following steps.

3.5. Generate Recommendation

After the previous steps, each user is denoted by a rating vector r_u consisting of his user-topic ratings. In our experiments we use the Pearson Correlation Coefficient to compute user similarity. A modified edition is denoted in Equation 14. Then Equation 15 is used to predict user ratings on movies.

\[
sim_{u,v} = \frac{\sum_{t \in T} (r_{u,t} - \bar{r}_u)(r_{v,t} - \bar{r}_v)}{\sqrt{\sum_{t \in T} (r_{u,t} - \bar{r}_u)^2 \sum_{t \in T} (r_{v,t} - \bar{r}_v)^2}}
\]

\[
pred_{u,m} = \bar{r}_u + \frac{\sum_{v \in U_m} sim_{u,v} \cdot (r_{v,m} - \bar{r}_v)}{\sum_{v \in U_m} |sim_{u,v}|}
\]

Each user u has a vector v which denotes his ratings on topics. In practical, user u has ratings on all topics. In Equation 14, T stands for the set of all the topics. r_{u,t} is u’s rating on topic t, while r_{v,t} is for user v. The average ratings of u and v on topics are denoted by \bar{r}_u and \bar{r}_v. In Equation 15, U_m denotes the set of all the users that have rated item m.

After getting the predicted ratings for u, it is easy to generate a recommendation list for u. By sorting the predicted ratings and choosing k highest ratings, we can recommend top k items to u.

4. Experimental Evaluation

4.1. Evaluation Metric

In this paper, we choose MAE(Mean Absolute Error) and RMSE(Root Mean Squared Error) to evaluate the experiments. These two metrics are often used to evaluate recommendation accuracy, especially for those recommender systems which produce predicted ratings. They reveal the average errors between real ratings and predicted ratings. Equation 16 describes MAE’s computation, and Equation 17 describes RMSE’s.

\[
MAE = \frac{\sum_{i=1}^{N_{pr}} |r_i - \hat{r}_i|}{N_{pr}}
\]

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N_{pr}} (r_i - \hat{r}_i)^2}{N_{pr} - 1}}
\]

In Equation 16 and 17, N_{pr} stands for the count of predicted ratings. r_i and \hat{r}_i are real rating and predicted rating. Given the same recommendation results, RMSE will always be larger than or equal to the MAE. And the difference between them reflects the variance of the individual errors in the recommendation results. In both metrics, a low value means a high accuracy.

4.2. Data Set and Preprocessing

We choose the MovieLens data set which contains both the user-movie ratings and tags. This data set contains 10,000,054 ratings and 95,580 tags applied to 10,681 movies by 71,567 users of the online movie recommender service MovieLens.

The original data set contains too many noisy records which will interfere the experimental results. In that case,
Table 2. Experimental Results

<table>
<thead>
<tr>
<th>Sparsity</th>
<th>Metrics</th>
<th>MAE</th>
<th>RMSE</th>
<th>MAE</th>
<th>RMSE</th>
<th>MAE</th>
<th>RMSE</th>
<th>MAE</th>
<th>RMSE</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CF</td>
<td>0.6672</td>
<td>0.7512</td>
<td>0.6402</td>
<td>0.7010</td>
<td>0.5947</td>
<td>0.6227</td>
<td>0.5809</td>
<td>0.5933</td>
<td>0.5734</td>
<td>0.5734</td>
</tr>
<tr>
<td></td>
<td>TC</td>
<td>0.6485</td>
<td>0.7153</td>
<td>0.6344</td>
<td>0.6883</td>
<td>0.5967</td>
<td>0.6235</td>
<td>0.5834</td>
<td>0.5989</td>
<td>0.5756</td>
<td>0.5769</td>
</tr>
<tr>
<td></td>
<td>LDA</td>
<td>0.6441</td>
<td>0.7068</td>
<td>0.6284</td>
<td>0.6762</td>
<td>0.5923</td>
<td>0.6134</td>
<td>0.5776</td>
<td>0.5877</td>
<td>0.5701</td>
<td>0.5667</td>
</tr>
</tbody>
</table>

We adopt certain rules to filter the noises. We will discuss it in the following paragraphs. After preprocessing, we divide the data set into smaller parts by sparsity. This allows us to compare these state-of-art algorithms in different sparsity.

4.3. Experimental Results and Analysis

After preprocessing, we finally get ten data sets. Each data set will be divided into two parts, the training set and the test set. For each user, we randomly select 30 ratings to construct the test set. And the remaining part is the training set. For each data set we have tried all the three methods – CF method, tag clustering and LDA. We also compare the experimental results in MAE and RMSE.

In Table 2, the computation of Sparsity $S$ of the training set is denoted by Equation 18.

$$S = 1 - \frac{\text{Count(Ratings)}}{\text{Count(Users)} \cdot \text{Count(Movies)}} \quad (18)$$

Figure 1 and 2 reveal that the LDA method always performs better than CF method and tag clustering method. But when the ratings become denser, its advantage over CF method won’t be obvious. Actually this is a phenomenon that could be explained – CF method suffers from data sparsity problem, but CF method could produce more accurate recommendation results with denser ratings. The tag clustering method only performs better than CF method with sparse ratings.
Overall these two figures suggest that user-topic based CF method are better than user based CF method. In user-topic based CF, LDA is better than tag clustering. It is because a tag is assigned to multiple topics in LDA, but is assigned to only one topic in tag clustering. In practice, a tag could carry different meanings in different perspectives. LDA extracts each topic from the distributions over the set of tags, so its implementation is closer to reality.

Nevertheless, when ratings become denser, the dimension of user ratings vector increases. CF method becomes inefficient due to computation complexity of user similarity. But the LDA method will still be effective if the amount of topics is defined. From the perspective of recommendation effect and computation efficiency, LDA outperforms the other two methods.

5. Conclusions

This paper presents comparisons between user-based and user-topic based CF along with different sparsity. We use tag clustering and the LDA model to do topic extraction from items, then we compute user preferences on the latent topics. As we did in user-based collaborative filtering, we predict user ratings by user-topic preferences.

The results show that, when ratings are relatively sparse, LDA model performs much better than user-based CF method. When ratings become relatively denser, LDA model still outperforms CF method and achieves better efficiency. We believe that each tag belonging to multiple topics makes the user-topic based CF more accurate.

There still exist some insufficiency in our experiments. Such as tags which represent users’ preference are directly removed from the data set or the LDA model can also be used to model users. We will try to relieve these issues in the future.

6. Acknowledgments

The work described in this article was partially supported by the National Natural Science Foundation of China (11171148, 61003024, 61170067).

References


