Postprint: A Parallel Collaborative Filtering Recommendation Algorithm Integrating Social Networks and Key Users

Authors: Xiao Chenglong, Wang Ning, Yonggui Wang

Date: 2018-07-09T00:00:00+00:00

Abstract

To address the problems of data sparsity, cold start, and lack of diversity in recommendation results inherent in traditional collaborative filtering recommendation algorithms, this paper proposes a collaborative filtering recommendation algorithm that integrates social network information and key users. Based on the user-item rating matrix, the algorithm incorporates user social network information to derive a social trust matrix, and integrates key user information to obtain a key user rating matrix. By leveraging these three rating matrices with different assigned weight proportions, the algorithm jointly predicts users' ratings for target items. To address the massive data challenge, the computational parallelization of the algorithm is implemented using a Spark distributed cluster. Experimental results demonstrate that the proposed algorithm can effectively alleviate the data sparsity problem, enhance processing speed, and improve recommendation accuracy.

Full Text

Parallel Collaborative Filtering Recommendation Algorithm Based on Social Networks and Key Users

Xiao Chenglong, Wang Ning†, Wang Yonggui
College of Software, Liaoning Technical University, Huludao, Liaoning 125105, China

Abstract: To address the problems of data sparsity, cold start, and lack of diversity in recommendation results inherent in traditional collaborative filtering recommendation algorithms, this paper proposes a collaborative filtering recommendation algorithm that integrates social networks and key users. Building upon the user-item rating matrix, the algorithm incorporates user social network information to derive a social trust matrix and integrates key user information

to obtain a key user rating matrix. Leveraging these three rating matrices with different weight allocations, it jointly predicts user ratings for target items. To handle massive data problems, the algorithm' s computation is parallelized using a Spark distributed cluster. Experimental results demonstrate that the algorithm can effectively alleviate data sparsity while improving processing speed and recommendation accuracy.

Keywords: social networking; parallelization; key users; collaborative filtering; big data; movie recommendation

0 Introduction

With the rapid development of the Internet industry, the explosive growth of information has brought about the "information overload"problem, and recommendation systems represent one effective solution. By analyzing user information needs and interests, recommendation systems can deliver content or products that users find appealing, preventing them from receiving large amounts of irrelevant information and enabling personalized recommendations. Unlike search engines that provide "one-to-many" information services, recommendation systems produce results that better match individual user preferences, achieving "one-to-one" information services with lower user involvement, thereby significantly reducing the cost of information seeking for users. Recommendation systems are widely used in daily life and across major web portals.

Traditional recommendation system technologies mainly include association rule-based recommendation, content-based recommendation, collaborative filtering, and hybrid approaches. Association rule-based recommendation uses association rules, taking purchased items as rule heads and rule bodies as recommendation objects. Simply put, it calculates the proportion of transactions that purchase item set Y among those that purchase item set X in a transaction database, intuitively representing users' tendency to buy certain items together with others. However, discovering association rules is the most critical and time-consuming bottleneck of the algorithm. Content-based recommendation makes suggestions based on item content information without requiring user ratings, instead using machine learning methods to derive user interest profiles from item feature descriptions. Its drawback is the requirement that item content be structured and easily extractable into meaningful features.

Collaborative filtering is one of the earliest and most successful technologies in recommendation systems. It typically employs nearest-neighbor techniques, using users'historical preference information to calculate similarities between users, then uses the weighted ratings from a target user' s nearest neighbors to predict the target user' s preference for specific items, making recommendations accordingly. However, collaborative filtering has certain limitations, such as the cold start problem and data sparsity issues. Hybrid recommendation systems combine different recommendation methods to leverage their respective strengths and compensate for weaknesses. Although theoretically numerous combination

approaches exist, they may not all be effective for specific problems. The most important principle of hybrid recommendation is to avoid or compensate for the weaknesses of individual recommendation techniques.

Compared with these major recommendation techniques, collaborative filtering-based recommendation systems offer better accuracy and effectiveness, along with excellent robustness, making them widely used across major recommendation platforms. Consequently, the cold start and data sparsity problems in collaborative filtering methods have attracted attention from scholars worldwide. Leng Yajun et al. argue that data sparsity adversely affects collaborative filtering from two aspects: insufficient accuracy in neighbor searching and excessively few neighbor ratings. Weng Xiaolan et al. also identify data sparsity, cold start, and scalability as the main challenges facing collaborative filtering recommendation algorithms in their survey. To address data sparsity, various solutions have been proposed: some use singular value decomposition to reduce rating matrix dimensionality and thus decrease sparsity, while others propose tag-weighted rating methods to minimize the impact of objective factors on user ratings and effectively alleviate rating bias. Many scholars have also highlighted the necessity of trust relationships in collaborative filtering algorithms, with Jøsang et al. surveying systems that can obtain trust and reputation, arguing that these should be applied as security mechanisms in collaborative filtering systems. Some research integrates user social trust and rating similarity to propose a new matrix completion recommendation method that significantly improves prediction accuracy and alleviates sparsity issues. Yang et al. propose an improved collaborative filtering algorithm that combines user trust relationships, enhancing recommendation performance by integrating sparse user evaluation data with sparse social trust network data. While these methods effectively alleviate data sparsity and improve recommendation accuracy to varying degrees, they have limitations, such as high computational complexity of singular value matrix decomposition, overfitting, lack of precision, and data source singularity issues when using only user-item ratings or only user trust relationships, leading to distorted recommendation results.

Therefore, this paper proposes a parallel collaborative filtering recommendation algorithm that integrates social networks and key users. On one hand, it uses Spark distributed clusters to improve processing speed for massive data; on the other hand, it simultaneously considers user social network information, key user data from user groups, and user-item rating information, effectively avoiding data source singularity and thereby improving prediction accuracy, solving the cold start problem, and providing diverse recommendation results. The algorithm mainly includes the following steps: (a) data preprocessing to obtain user-item rating information, user trust social network information, and information about several key users with high trust levels in the user group through analysis of the Epinions public dataset; (b) similarity calculation, which assigns different weight ratios to multiple data sources and uses Pearson correlation coefficient to calculate similarities between users, obtaining the target user' s nearest neighbor set; (c) rating prediction, which calculates recommendation

scores for all items to the target user and selects top-N items for recommendation; and (d) accuracy measurement, which evaluates recommendation quality through statistical accuracy and decision support accuracy.

1 Traditional Collaborative Filtering Recommendation Algorithm

Traditional collaborative filtering recommendation algorithms mainly include user-based collaborative filtering, item-based collaborative filtering, and model-based collaborative filtering. These three methods have different focuses and applicable environments, but for recommendation systems generally include three steps: building a rating matrix, calculating similarity, and predicting ratings to generate recommendations. Each step involves a series of calculations and data processing.

1.1 Building the Rating Matrix

Establish a user-item rating matrix by preprocessing raw data to obtain a matrix R as shown in , where rows represent users (U) and columns represent items (I). Assuming there are 3 users and 5 items, values in the matrix represent user ratings for items on a scale of 1-5, with higher scores indicating greater interest. Empty values indicate that the user has not rated the item.

1.2 Calculating Similarity

The accuracy of similarity calculation is a crucial factor affecting recommendation quality. Widely used similarity measures include Euclidean distance similarity, Jaccard similarity, cosine similarity, adjusted cosine similarity, and Pearson similarity. Pearson similarity is suitable for situations with low data sparsity and many co-rated items. Since our algorithm uses multiple data sources to compensate for data sparsity, we select Pearson similarity, calculated as shown in equation (1):

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

where r_{ui} and r_{vi} represent ratings of user u and user v for item i, $\bar{r}_u$ and $\bar{r}_v$ represent average ratings of user u and user v, and I_{uv} represents the intersection of items rated by both users u and v.

1.3 Predicting Ratings and Generating Recommendations

After similarity calculation, several users with high similarity are selected as the target user' s nearest neighbor set. The target user' s rating for an item is predicted using the neighbor users' rating data, and the top-N items with the

best predicted ratings are recommended. To predict user u' s rating for item i, we consider the target user' s neighbor set and their ratings for the item. Let $N(u)$ be user u' s nearest neighbor set, then the prediction formula is shown in equation (2):

$$P_{u,i} = \bar{r}_u + \frac{\sum_{v \in N(u)} sim(u,v) \times (r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u)} |sim(u,v)|}$$

After calculating predicted ratings, items with higher ratings (top-N) are selected and recommended to the user.

1.4 General Process of Collaborative Filtering

This section introduces the general process of collaborative filtering recommendation systems, from raw data to rating matrix, similarity calculation, rating prediction, and final recommendations. The process is illustrated in [Figure 1: see original paper].

2 Collaborative Filtering Recommendation Algorithm Integrating Social Networks and Key Users

Unlike traditional collaborative filtering, this paper proposes a parallel collaborative filtering recommendation algorithm that integrates user social network information and key user data, expanding data sources from a single source to three, effectively alleviating dataset sparsity. Different weights are assigned to the three data sources to improve prediction accuracy and recommendation diversity. The algorithm also proposes a strategy for identifying key users in user groups—users with high trust levels and influence who are more universal. This approach addresses the cold start problem in collaborative filtering systems: for new users lacking rating data and social trust relationships, key user information can be used for system recommendations.

2.1 Algorithm Description and Model Design

The algorithm mainly includes the following components: data preprocessing to build the user-item rating matrix, user social rating matrix, and key user rating matrix; using the mathematical optimization software 1stOpt for regression analysis to determine weight parameters for the three data sources; normalizing the data; calculating similarity to obtain the target user' s nearest neighbor set; and finally predicting target item ratings from the neighbor set' s item ratings to generate top-N recommendations.

2.1.1 Building the Social Network Trust Matrix Different datasets require different methods for extracting the social network trust matrix. This paper uses the publicly available Epinions dataset where trust relationships are

explicitly expressed with trust data in the range [0,1] and empty values for no rating. For datasets without explicit trust data, trust can be constructed from implicit trust data such as co-rated items—the more consistent the ratings on common items, the higher the trust between users, and vice versa. Methods such as User Position Similarity (UPS) and user indirect credibility can be used to construct trust degrees. The user trust relationship is shown in , where trust relationships are bidirectional: user a may trust user b without b trusting a. For non-explicit trust data, trust degrees vary and require normalization.

2.1.3 Correlation Similarity Calculation In addition to calculating similarity from the user-item rating matrix, our algorithm also requires calculating similarities corresponding to the social trust relationship matrix and the key user rating matrix. The similarity calculation method is the same as in equation (1). After obtaining similarity results for different matrices, the final similarity calculation formula with different weight allocations is shown in equation (3):

$$sim(u,v) = \alpha \times sim_{user}(u,v) + \beta \times sim_{sns}(u,v) + \gamma \times sim_{key}(u,v)$$

where $\alpha, \beta, \gamma \in (0,1)$ and $\alpha+\beta+\gamma = 1$. These are weight adjustment parameters for different rating matrices. $sim_{user}(u,v)$ is user similarity calculated from the user-item rating matrix, $sim_{sns}(u,v)$ is user similarity from the social network trust matrix, and $sim_{key}(u,v)$ is user similarity based on the key user rating matrix, yielding the final user similarity.

2.1.4 Algorithm Flow The proposed algorithm includes: (a) data preprocessing to obtain three major matrices; (b) calculating item similarity using Pearson correlation coefficient as in equation (3); (c) predicting target item ratings from neighbor set ratings as in equation (2); and finally generating top-N recommendations. The algorithm flow is illustrated in [Figure 3: see original paper].

2.2 Parallel Algorithm Flow Design

The collaborative filtering recommendation algorithm integrating social networks and key users is implemented on a Spark distributed cluster. After data preprocessing to obtain the three major matrices, Spark provides various distributed storage options and matrix computation operations. The Spark programming model offers operators such as map(), reduce(), join(), and filter(). Three similarity RDDs are calculated from the major matrices, then merged with different weight allocations to obtain final user similarity, compute predicted ratings, and generate recommendations. During parallelization in Spark, data is stored in RDD format distributed across Spark memory, accessing the HDFS file system only once throughout the process, reducing data access frequency and communication overhead compared to Hadoop clusters. The pseudocode for parallel implementation is as follows:

```
# Read rating data, trust relationships, and key user ratings from HDFS
Sc1 = SparkContext(sys.argv[1], "Userbased")
Sc2 = SparkContext(sys.argv[2], "SNSbased")
Sc3 = SparkContext(sys.argv[3], "Keyuserbased")

# Parallelization: obtain user-item inverted index table (using Lines1 as example)
Item_{user} = Lines1.parallelize(0 until P).map(parseVectorOnItem).groupByKey().map(x[0])

# Calculate similarities: user similarity (user_{sim}(u,v)), trust user similarity (sns_{si
user_{sim}(u,v) = Pairwise_{users}.map(lambda x: calcSim(x[0], x[1])).map(lambda x: keyOnFi

# Combine similarities with different weights to obtain final similarity
sim(u,v) = user_{sim}(u,v).join(sns_{sim}(u,v)).join(key_{sim}(u,v)).map(x => [1][2][3]xxx,

# Recommend top-N items of interest to users
Recommend_{item} = sim(u,v).map(lambda x: topNRecommendations(x[0], x[1], uib.value, 50)).co
```

3 Experiments and Results Analysis

3.1 Experimental Data

The Epinions public dataset is used for experiments, collected by Massa from http://www.epinions.com. It includes 49,290 users rating 139,738 different items with integer scores in [1,5], where higher scores indicate greater preference. It also contains 664,824 social relationships, of which 487,181 represent positive relationships considered as trust data with trust value 1, and the remainder 0 indicating no trust relationship.

3.2 Experimental Environment

The experimental cluster uses a professional Spark cluster provided by QingCloud Console (https://www.qingcloud.com/), eliminating tedious cluster setup while ensuring complete consistency across machines and offering various specifications including different processor models and memory configurations. The experiment configures one master node and five worker nodes with Spark version 2.2.0, all machines having dual-core processors and 4GB RAM.

3.3 Evaluation Metrics

Common collaborative filtering evaluation metrics include two categories: (1) Rating accuracy metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Normalized MAE, suitable for systems focusing on precise rating prediction; (2) Classification accuracy metrics including precision, recall, F1-measure, and ROC AUC, suitable for systems with explicit binary preferences. This paper uses MAE and RMSE from rating accuracy, and precision and recall from classification accuracy.

MAE calculates the average error between predicted and actual ratings across all test user-item pairs:

$$MAE = \frac{1}{N} \sum_{u,i \in T} |P_{u,i} - R_{u,i}|$$

where T is the test set user data collection, N is the number of test items, $P_{u,i}$ is the predicted rating, and $R_{u,i}$ is the actual rating.

RMSE calculates the root mean square error:

$$RMSE = \sqrt{\frac{1}{N} \sum_{u,i \in T} (P_{u,i} - R_{u,i})^2}$$

Since the dataset uses a 5-point scale, to convert it to binary classification, ratings above 3 are considered liked and others disliked. Let $R(u)$ be the recommended item list and $T(u)$ be items in the test set with ratings above 3. Precision and recall are calculated as:

$$Precision = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|}$$

$$Recall = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|}$$

3.4 Experimental Results Analysis

To evaluate comprehensive performance, our algorithm is compared with: (a) traditional user-based collaborative filtering (UserCF) using only the user-item rating matrix; (b) traditional item-based collaborative filtering (ItemCF) using the same matrix but calculating item similarities; and (c) TDSRec, a collaborative filtering algorithm integrating social network features and rating preferences.

Using the first 500 data entries from Epinions, we tested 42 weight allocation combinations for the three similarities with $\alpha + \beta + \gamma = 1$ and $\alpha, \beta, \gamma \in (0, 1)$. The best performance was achieved with $\alpha = 0.4, \beta = 0.2, \gamma = 0.4$. With these weights, we tested MAE and RMSE against other algorithms and examined precision and recall across different dataset sizes (first 100, 500, 1000, and 2000 entries). Results are shown in [Figure 4: see original paper] and [Figure 5: see original paper].

As dataset size increases, prediction accuracy gradually improves, showing positive correlation. Recall also increases but shows a declining trend at 2000 entries. While other algorithms typically show an inverse relationship between precision

and recall, our algorithm follows this trend but with less pronounced variation. The integration of user-item ratings and social network trust ratings improves accuracy, while key user ratings enhance recommendation diversity since key users have more varied interests.

Parallel performance was tested on a 6-machine Spark cluster, comparing runtime with single-machine execution and calculating speedup:

$$S = \frac{T_1}{T_P}$$

where S is speedup, T_1 is single-processor runtime, and T_P is runtime on P processors. Results are shown in .

Table 4: Parallelization Performance Validation

Dataset Size	Single Machine (min)	Spark Cluster (min)	Speedup
40000 entries	28.5	6.2	4.6

Distributed cluster computing demonstrates effective improvement in problem-solving speed, with cluster size and time efficiency showing essentially linear negative correlation. While data distribution inevitably incurs communication costs, the speedup trend shows that communication overhead proportion gradually decreases as data scale increases.

4 Conclusion

To address rating data sparsity, cold start problems, and diversity issues in recommendation systems, along with big data processing challenges, this paper proposes and implements a parallel collaborative filtering recommendation algorithm integrating social networks and key users. Experimental results show the algorithm effectively alleviates data sparsity, improves recommendation accuracy, and reduces system processing time.

Future work will further investigate evaluation metrics for recommendation systems to propose comprehensive evaluation methods, including quantitative assessment of recommendation diversity and novelty.

References

[1] Liu Hongxia. A Survey of Collaborative Filtering Technique in Recommendation System [J]. Information Security and Technology, 2016, 7(3).

[2] Xia Peiyong. Research on Collaborative Filtering Algorithm of Personalized Recommendation Technology [D]. QingDao: Ocean University of China, 2011.

[3] Su Xingning, Yang Jianlin, Deng Sanhong, et al. Data Mining Theory and Technology [M]. Beijing: Science and Technology Literature Publishing House, 2003.

[4] BaezaYates, Ricardo A, RibeiroNeto, et al. Modern Information Retrieval [J]. 1999, 43(1): 26-28.

[5] Xu Hailing, Wu Xiao, Li Xiaodong, et al. Comparison study of internet recommendation system [J]. Journal of Software, 2009, 20(2): 350-362.

[6] Leng Yajun, Lu Qing, Liang Changyong. Survey of Recommendation Based on Collaborative Filtering [J]. PR&AI, 2014, 27(8): 720-734.

[7] Weng Xiaolan, Wang Zhijian. Research process of collaborative filtering recommendation algorithm [J]. Computer Engineering and Applications, 2018, 54(1): 25-31.

[8] Kumar R, K. Verma B, Sunder Rastogi S. Social popularity based SVD+recommender system [J]. International Journal of Computer Applications, 2014, 87(14): 33-37.

[9] Kong Xinxin, Su Benchang, Wang Hongzhi, et al. Research on the modeling and related algorithms of label-weight rating based recommendation system [J]. Chinese Journal Of Computers, 2017, 40(6): 1440-1452.

[10] Jøsang A, Ismail R, Boyd C. A survey of trust and reputation systems for online service provision [J]. Decision Support Systems, 2006, 43(2).

[11] Wang Meiling, Ma Jun. A novel recommendation approach based on users weighted trust relations and the rating similarities [J]. Soft Computing, 2015, 20(10): 3981-3990.

[12] Yang B, Lei Y, Liu J, et al. Social collaborative filtering by trust [J]. IEEE Trans on Pattern Analysis & Machine Intelligence, 2017, 39(8).

[13] Wang Shengsheng, Zhao Haiyan, Chen Qingkui, et al. Latent Factor Model for Personalized Recommendation [J]. Journal of Chinese Computer Systems, 2016, 37(5): 881-889.

[14] Liu Qingwn. Research on Recommender Systems based on Collaborative Filtering [D]. Hefei: University of Science and Technology of China, 2013.

[15] Li Bin. A Survey of Recommender System [J]. Modern computer, 2014(2): 7-10.

[16] Du Yongping, Du Xiaoyan, Huang Liang. Improve the collaborative filtering recommender system performance by trust network construction [J]. Chinese Journal of Electronics, 2016, 25(3): 418-423.

[17] Sun Hong, Zuo Teng. Research on algorithm of micro-blog user influence based on PageRank [J]. Application Research of Computers, 2018, 35(4).

[18] Massa P, Bhattacharjee B. Using Trust in Recommender Systems: An Experimental Analysis [C]// Proc of the 2nd International Conference on Trust Management. 2004: 221-235.

[19] Rennie J D M, Srebro N. Fast maximum margin matrix factorization for collaborative prediction [C]// Proc of the 22nd International Conference on Machine Learning. New York: ACM, 2005: 713-719.

[20] Zhu Yuxiao, Lv Linyuan. Evaluation metrics for recommender systems [J]. Journal of University of Electronic Science and Technology of China, 2012, 41(2): 163-175.

[21] Guo Ningning, Wang Baoliang, Hou Yonghong, et al. Collaborative filtering recommendation algorithm based on characteristics of social network [J]. Journal of Frontiers of Computer Science and Technology, 2018, 12(2): 208-217.

Note: Figure translations are in progress. See original paper for figures.