A Scalable Music Recommendation System

Mingruo Shi[1,a,*]

[1]Beijing Wuzi University, 101149, China

[a] shimingruo@163.com

*Mingruo Shi

Keywords: Recommendation, Content based filtering, Ranking, Hadoop

Abstract: A content match based on filtering solution is proposed to handle large number of users and songs for search engine as a recommendation system is increasing important in online video application, news recommendation and electronic commerce especially in search engine. Our method could be applied to search engine like Google, Bing and Baidu to highly enhance user experience.

1. Introduction

Recommender system has become extremely used nowadays in a variety of areas: some popular applications include movies, music, news, books, research articles, search queries, social tags, and online shopping in general. It's also extended to recommendation for experts [1], collaborators [2], jokes, restaurants, garments, financial services [3], life insurance, online dating, and Twitter pages [4].

A list of recommendations will be finally produced by recommender system typically in one of a few ways. The ways include collaborative filtering, content-based filtering or personality-based approach [5,11,12]. Collaborative filtering approaches building a model from a user's past behavior (items previously purchased or selected, and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in [6]. Content-based filtering approaches utilize a series of discrete characteristics of an item to recommend additional items with similar properties [7]. These approaches are often combined (called Hybrid Recommender Systems).

A recommendation system usually includes two major parts. The first part is recommendation which will create a recommendation item candidates based on user's visit/click history. The other part is rank which will select top n recommendation results from the recommended candidates by a ranking algorithm. There are some papers discussing some scalable recommendation to users including Google news recommendation [8,10]. To our best knowledge, no literature gives the detailed introduction how a large-scale recommendation system is designed and implemented scaled both to number of users and items on the music recommendation vertical.

In this paper, a Content-based filtering recommendation algorithm is proposed scalable to both users and items which can be applied to music recommendation esp. on search engine.

The paper is organized as follows: section 2 describes in detail our algorithm to support large number of users and songs. Section 3 introduces our experiments. Our ongoing work is summarized in section 4.

2. Algorithm

Problem Statement Problem statement is for recommendation as follows:

Input:

A list of *n* users $\{U_1, U_2, \dots, U_n\}$ with user profile

A list of *m* songs $\{S_1, S_2, \dots, S_m\}$ with song content and status information

The visit history of all users for the fixed period (e.g. 3-month log)

Output:

A list of preferred songs for each user for today.

Our solution is to develop a content-based recommendation system. We firstly extract the side information for user to build user profile and song information for match. Then use the visiting history and side information including user profile and song information in both training phase and rank phase of the algorithm.

2.1. Basic Idea

At high level, the idea to recommend songs to a user, say A, is to find the set of users, say C_A, that are like A, and recommend to A items viewed by users in C_A. Pseudocode:

Algorithm A:

```
Choose a threshold $\theta$
For each user A and possible interested songs M
  • $C_A \leftarrow \Phi$  // the set of songs user A may be interested in
  • If $s(P_A, m) \geq Q$ // $s(P_A, m)$ is the score how A is possibly interested in song $m$
      Add $(m, s(P_A, m))$ to $C_A$
  • Recommend to A at most k songs in $C_A$ by rank score $s(P_A, m)$
```

2.2. Extract Music Features

Our target is to recommend songs to users. we therefore only consider active listing songs. And extract the same features as user profile. For example, we could extract features such as country, genre, release date, singer name, singer age, singer gender and #reviews.

2.3. Rank Recommendation Result

We could calculate match score between users and songs with search history. For matched user and song pair, we could get match score by the following way:

$$s_{u,s} = \sum_{i=1}^{n} c_i f_i \ . \qquad (1)$$

$s_{u,s}$ is the match score. n is the number of matched features for user u and song s. f_i is the feature value in user's profile for a give user u, we get top k songs ranked by match score satisfy $s_{u,s_1} \geq s_{u,s_2} \geq \cdots \geq s_{u,s_k}$ where k is a predefined number.

2.4. Decide Coefficients

There are a few ways to decide the coefficients c_i during user and song matching process. We use logistic regression machine learning method on user search history to figure out the optimized coefficients c_i.

Finally search engine could make a recommendation to user in some ways. For example, a user could be recommended by search engine's home page or recommendation email.

3. Experiments

3.1. Evaluation Methodology

We take three-month song entity search log in our test search engines and split the visit log into the training set and the test set. The test set is one-day log, i.e., the visit log of the latest day. We use the training set to get recommendation for every user and compare the recommendation list with the list of songs that users visited in the test set. When we measure, users/entities in the test file that is NOT in the training file will be removed. Note that the offline evaluation does not truly reflect the performance of the online system (the performance should be much better when we launch the system online) since we haven't posed the recommendation to users yet. But it helps us determine what ideas can potentially improve the performance of the online system.

Running time: In log, the number of distinct users is up to 14749347 (~14 million) and songs is 2436148 (about 2 million). The training can be done in 3 hours, using about 65 computers in a Hadoop cluster. So, in the following, we mainly discuss about the prediction accuracy.

Two main metrics below are considered for the prediction accuracy:

(1) *TrueRate*: We assign score 1 to a user if he/she visited a recommended item in the test set and 0 otherwise.

$$TrueRate = Total\ Score/(Total\ \#\ Users). \tag{2}$$

(2) NDCG (Normalized Discounted Cumulative Gain): we define NDCG @ k as:

$$NDCG_k = DCG_k/DCG_k^*. \tag{3}$$

Where $DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{\log_2(i+1)}$ and DCG_k^* is the ideal DCG_k. We assign rel_i to equal 1 if there was a click on item ranked in position i. Here, k is the recommendation length, i.e, select top k entities/songs to present to users. Basically, NDCG assign higher score to visited items that in the top of the recommendation list than the visited items in the bottom of the list.

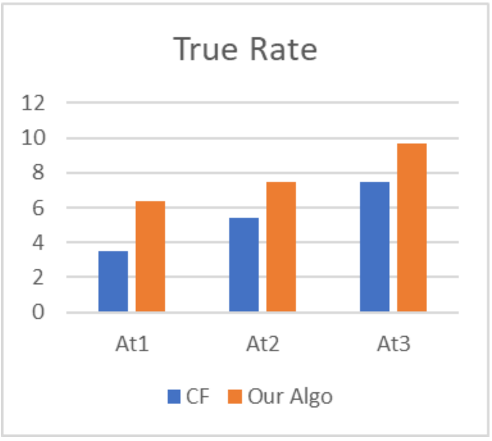

Figure 1 True Rate.

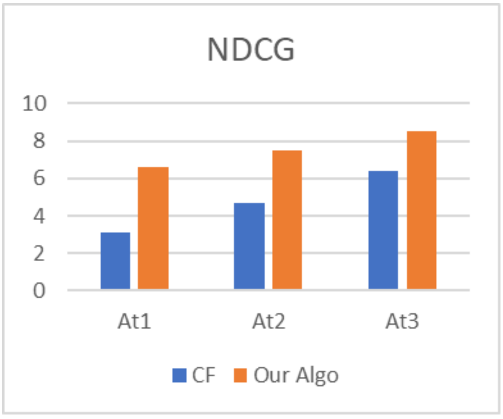

Figure 2 NDCG.

Evaluation Result Besides our algorithm, we also developed a Collaborative Filtering algorithm which doesn't use any song entity information at all. However, our algorithm takes important signals from users' search history and search graph. Note that we only take top (at most) 3 songs to recommend to users. The evaluation results are shown in Fig. 1 and Fig. 2.

As Fig. 1 and Fig. 2 suggest, our algorithm clearly outperforms collaborative filtering algorithm.

4. Future Work

Firstly, personalized ranking methods from Information Retrieval could be leveraged. Researchers came up with various ranking strategies to present relevant information to users. There is a line of research that incorporate user's interests into the ranking, called personalized ranking [11]. The same idea can be applied in ranking the recommendation result. Finally, probabilistic recommendation algorithm might be helpful. pLSI (probabilistic latent semantic indexing)-based algorithm may outperform our algorithm [12].

Acknowledgements

This research was financially sponsored by the key lab of intelligent logistic system BZ0211 PXM2015_014214_00000.

References

[1] H. Chen, A. G. Ororbia II, C. L. Giles ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries, in arXiv preprint 2015.

[2] H. Chen, L. Gou, X. Zhang, C. Giles Collabseer: a search engine for collaboration discovery, in ACM/IEEE Joint Conference on Digital Libraries (JCDL) 2011

[3] Alexander Felfernig, Klaus Isak, Kalman Szabo, Peter Zachar, The VITA Financial Services Sales Support Environment, in AAAI/IAAI 2007, pp. 1692-1699, Vancouver, Canada, 2007.

[4] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF:The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web.

[5] Hosein Jafarkarimi; A.T.H. Sim and R. Saadatdoost A Naïve Recommendation Model for Large Databases, International Journal of Information and Education Technology, June 2012

[6] Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010.

[7] R. J. Mooney & L. Roy (1999). Content-based book recommendation using learning for text categorization. In Workshop Recom. Sys.: Algo. and Evaluation.

[8] Das, Abhinandan S., et al. "Google news personalization: scalable online collaborative filtering." WWW'07.

[9] Indyk, Piotr. "A small approximately min-wise independent family of hash functions." Journal of Algorithms 38.1 (2001): 84-90.

[10] Jure Leskovec, Anand Rajaraman, Jeff Ullman, "Mining of Massive Datasets", Chapter 3.

[11] Bouadjenek, Mohamed Reda, et al. "Evaluation of personalized social ranking functions of information retrieval." ICWE'13

[12] Liu, Jiahui, et al. "Personalized news recommendation based on click behavior." ICIUI'10.