A Hybrid Recommender System of Tencent Microblog

Project Review

Yingzhen Li

Abstract

Precise recommendations in social networks present opportunities for novel machine learning. Our work described a recommender system of micro blog, which mined topics from messages of users by applying association rules and by combining collaborative and content-based filtering to discover their preferences. We selected and rated items according to the interests of users, as reflected by the topics in the messages. The results indicated a strong relationship between user interests and their acceptance on a given item. Several limitations of our system have led to future research. The revision of topic extraction is related to NLP studies. The analysis of sequential data, data pre-processing, and advanced item pre-selection are other probable extensions.

1 Introduction

Online social networking services like Tencent Microblog have been tremendously popular in China, with a considerable speed of user growth. Celebrities and organizations also register microblog, which leads to diversity of topics and helps attract more potential users. However, flooded information can puzzle the users and even result in the loss of them. So reducing the risk of puzzlement and recommending attractive items - celebrities and organizations - are crucial for user experience improvement and prosperity maintenance, which present opportunities for novel machine learning and data mining approaches.

Recommender systems can be categorized into content-based algorithm [7], collaborative filtering [5], and influential ranking algorithm [10]. Unfortunately, all of them consider little of user profile's fidelity, preference variance and interactions, causing difficulty of precise and stable recommendation. To overcome these weaknesses of single method, we construct a hybrid recommender system specified to Tencent Microblog, which generates ordered item list by mining the data of the platform [8].

The rest of the paper is organized as follows. Section 2 discusses the background of the problem, and Section 3 describes the design of the hybrid recommender system. Section 4 presents the training process. Section 5 shows the experimental results and discuss some improvements, and the paper is concluded in Section 6.

2 Background

Tencent launched its microblog platform - Tencent Microblog - in 2010, which then became one of the dominant microblog platforms in China based on the large user group of its instant messaging service QQ [9]. Celebrities and organizations are invited to register the platform, leading to a nice growth in the user group. Furthermore, Tencent's microblog service is embedded in its other leading platforms, hence user can write or comment a message directly on the website of Tencent Microblog or via the third-party port and related platforms.

While Tencent Microblog has the largest user group, Sina Microblog takes a commanding lead with 56.5% of China's microblog market based on active users and 86.6% based on browsing time over its competitors [6]. This fake prosperity results from the existence of the fake users, widely used spammer strategy [3] and the weird definition of active users. Tencent Microblog considers those who write(including retweet and comment) or read microblog messages - no matter on the website or other associated platforms - as active users, while Twitter and Sina Microblog define them as those who login the platform everyday.

User messages generated via from other related platforms confuse the recommender finding the real interests of users, which leads to the decrease of acceptance. Tencent Microblog users accept the recommendations in a low percentage(less than 9% according to our survey [8]), and the recommended item lists isn't updated in time, which deviate from the users' present preferences.

3 Algorithms

This section introduces a hybrid recommender system of Tencent Microblog, including the preparations - keyword analysis and user taxonomy - and the main part of it. Detailed algorithms with formulas is explained in our workshop paper.

3.1 Keyword Analysis

Mining synonyms in user's keywords helps in finding their interests. However, applying association rule algorithm [2] to find them directly in the huge keyword set is unrealistic since that involves searching all possible combinations. So we parallel this process by adopting revised FDM(Fast Distributed Mining of association rules) [4]. Moreover, we insert the ambiguous keywords into different classes simultaneously.

Let $U = \{u_i\}$ be the set of users where u_i has the value of its ID. Each user u_j has its keyword set $K_j = \{k_{jl}\}$ with weights $\mathcal{W}_j = \{w_{jl}\}$, and we denote $\mathcal{K} = \bigcup_{j=1}^{m} K_j$ as the set of all users' keywords. The database $\mathcal{DB}$ (user-keyword set) is divided into n subsets $\mathcal{DB}_i$ and these subsets are sent to the remote sites $\mathcal{RM}_i$. $\mathcal{T}^j = \{T_1^j, T_2^j, ..., T_{n_j}^j\}$ is the result generated in the j^{th} iteration where T_i^j is the i^{th} transaction of the j^{th} iteration.

Apriori algorithm is applied to generate the candidate transaction set $\mathcal{C}_i^j = Apriori\_gen(T_i^{j-1})$ at $\mathcal{RM}_i$ in the beginning of the j^{th} iteration where $\mathcal{C}_i^j = \{C_{i1}^j, C_{i2}^j, ..., C_{ih_j}^j\}$ are candidate transactions in this iteration. Then the remote site $\mathcal{RM}_i$ computes the local support and confidence of C_{il}^j and eliminates those which fail to satisfy local minimums $supp\_local$ or $conf\_local$. Then it sends the remaining candidate transactions C_{il}^j to the polling site $\mathcal{PL}_K$, where $K = polling(C_{il}^j)$ is a hash function.

$\mathcal{PL}_K$ gathers the candidate transactions C_{il}^j, computes global support $supp\_global(C_{il}^j)$ and confidence $conf\_global(C_{il}^j)$ by sending request to remote sites for local values' return. Then $\mathcal{PL}_K$ filters out the candidates which fail to satisfy the constraint of $supp\_global$ and $conf\_global$. Generally the local minimums coincide with the globals. For convenience we still denote the updated candidate sets as $\mathcal{C}_i^j$.

Then home site gathers $\mathcal{C}_i^j$ from the polling sites to generate the result of transactions(keyword classes) in the j^{th} iteration:

$$\mathcal{T}^j = \bigcup_i \mathcal{C}_i^j.$$

The process is terminated if no new transaction is generated, else the home site broadcasts the transactions T_i^j to the remote site $\mathcal{RM}_K$ where $\mathcal{RM}_K$ is the original remote site of $T_i^j = C_{Km}^j$, and starts the next iteration. The final result of keyword class is

$$keyword\_class = \{class_1, class_2, ..., class_N\},$$

where $class_i = \{k_{i1}, k_{i2}, ..., k_{im}\}$ is the set of synonyms.

The choice of minimums affects the precision and computational complexity tremendously. We sampled 1000 users' keywords and found out that these users have their keyword weights average in 0.14, so we assign $supp\_local = supp\_global = 0.2$, a little higher than the average weight. $conf\_local/conf\_global$ is affected by $supp\_local/supp\_global$, in this case $conf\_local = conf\_global = \frac{0.14}{0.2} = 0.7$.

3.2 User Taxonomy

We classify some users of Tencent Microblog as fake users (see Figure 1), who seldom login directly but still have records of tweets generated from other related platforms. These users' messages could hardly reflect their interests, and they rarely interact with other users and have few favorites for the same reason. In addition, we also consider the spammers as fake since they seldom use microblog even indirectly.

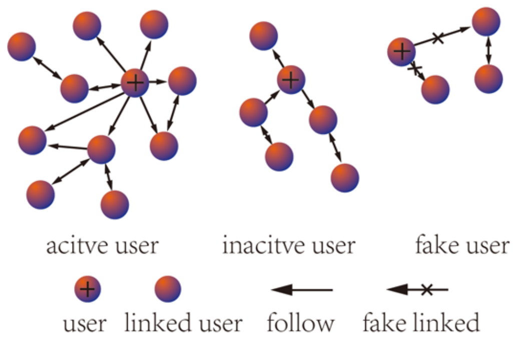

Figure 1: User taxonomy. Active users have more followees than inactive users. Links related to fake users are eliminated since they're not real users in the user network.

Due to the absence of login records, the activeness function $act(u_j)$ counts the number of tweets and interactions and computes u_j's activeness by applying the thresholds $min\_activeness$ and min_action:

$$act(u_j) = tweet \times is\_fake(u_j),$$

where

$$is\_fake(u_j) = \frac{1 + sgn(at + retweet + comment - min\_action)}{2},$$

$$user\_class(u_j) = \begin{cases} \text{active}, & act(u_j) \geq min\_activeness \\ \text{inactive}, & 0 < act(u_j) < min\_activeness \\ \text{fake}, & act(u_i) = 0 \end{cases}.$$

We assign $min\_activeness = 100$ and $min\_action = 20$ since only 33.2% of the users have written more than 100 tweets, and apply the algorithm to divide the user group into 3 classes.

An appropriate user taxonomy helps in improving the precision of recommendation. Users with similar favourites often accept similar items, hence dividing users into smaller groups by their interests can balance the precision and computational complexity. However, we haven't done this due to the sparsity of successful recommendation records, which reflect the user's interests directly.

3.3 Generating Recommendations

After keyword analysis and user taxonomy which are preparations of the recommendation, it comes the main part of our hybrid recommender system, consisting of item popularity ranking, (potential)interests discovery and the grading function, to generate recommended items and evaluate the possibility of acceptance or rejection. The system maps the users' (potential)interests to their corresponding item categories and grades selected candidates in these categories with indicators of similarity and popularity. It also contains special algorithms with respect to fake users in order to reach a precise recommendation.

3.3.1 Item Popularity Ranking

An item is a specific user, which can be a famous person, an organization, or a group. Items are organized in different categories of professional domains by Tencent to form a hierarchy (see Figure 2). For example, an item, Dr. Kaifu LEE, is represented as *science-and-technology.internet.mobile* [1].

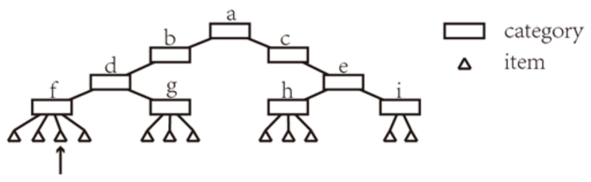

Figure 2: Item categories organized in hierarchy. The pointed item belongs to the category a.b.d.f.

The number of an item's followers indicates its popularity directly. Recommending hot items in a user's interested field promotes the possibility of acceptance effectively. Let $I = \{i_1, i_2, ..., i_n\}$ be the item set and h_k be the category (hierarchy) of an item i_k(h_k may coincide in some h_j, and the set of h_k is denoted as H). Then the rank of i_k in h_k is computed by $hot_k = get\_hot\_rank(i_k, h_k)$ where the function counts and normalizes the number of i_k's followers and return its ranking in h_k.

For users who show little of their preferences(especially the fake users) we recommend the most popular items in the whole itemset. Similarly the hot rank of i_k in the whole item set I is $HOT_k = GET\_HOT\_RANK(i_k)$.

3.3.2 Mining Interests from Keywords

Users are inclined to accept items of their interests. Active users have more keywords which reflect their favourites, and we map these interests to the hierarchy H to obtain candidate items.

Consider the keyword classes set $keyword\_class = \{class_i\}$ generated by the keyword analysis. A mapping $\mathcal{KH}$ is defined to construct the keyword class of a given category h_k (see section 3.3.4 for details). Suppose a given user u_j(or a given item i_k) has keywords K_j with weights. A function $key\_class(u_j) = \{class_{ji}\}$ computes the keyword class of a given user u_j (or item i_k), and the corresponding weight of $class_{ji}$ is

$$\overline{W}_{ji} = \sum_{k_l \in K_j \cap class_{ji}} w_l.$$

A vector function $class\_weight()$ is defined on $U \cup I$ to compute the weight of keyword classes in the user or item's keyword set.

After keyword analysis of individuals the target categories $\{h_k\}$ is generated where h_k satisfies $\mathcal{KH}(h_k) \cap key\_class(u_j) \neq \varnothing$, hence the candidate items are the items in each h_k(suppose i_k included). The similarity between candidate i_k and u_j is the normalized Euclid distance of these 2 vectors.

3.3.3 Discovering Potential Interests

Few inactive users have enough keywords, hence we design indirect collaborative filter to mine their potential interests from their followees and even their followees' followees (see Figure 3).

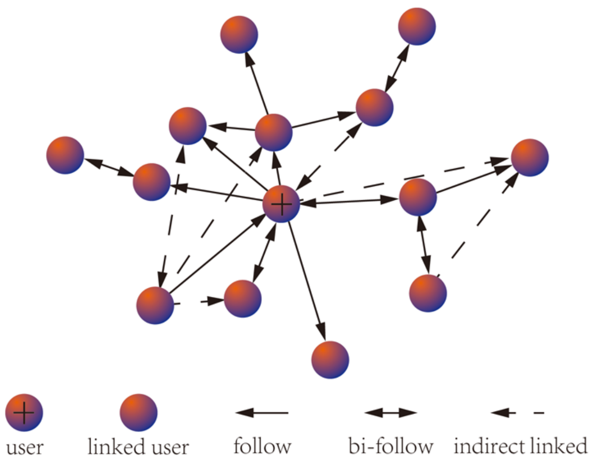

Figure 3: User network with indirect links ($depth = 2$). Indirectly linked users interact with each other even without followships. The length of the arrows represents the familiarity between two users(a user may be more familiar with some indirectly linked users than its followees).

Let $depth$ be the maximal levels amount of the searching process, in fact $depth \leq 3$ is enough for the process to mine a user's potential interests. The related users of u_j is $related\_users(u_j) = search\_followee(u_j, depth)$. Then for every u_k in $related\_users(u_j)$ we compute the keyword classes as mentioned above and merge them into the set

$$potential\_key(u_j) = \bigcup_{u_k \in related\_users(u_j)} key\_class(u_k)$$

where the i^{th} keyword class $class_{ji}$ has the weight

$$\widetilde{W}_{ji} = \sum_{\substack{u_k \in related\_users(u_j), \\ class_{kl_k} = class_{ji}}} \overline{W}_{kl_k} fami(u_j, u_k).$$

$fami(u_j, u_k)$ computes the familiarity of u_j and u_k by adopting indicators of interactions(at(@), retweet and comment) which could only happen in linked users.

Finally we merge $key\_class(u_j)$ and $potential\_key(u_j)$ into the set $interests(u_j) = \{class_{j1}\}$ with weight of $class_{jl}$

$$W_{jl} = \begin{cases} \overline{W}_{jl_1}, & class_{jl} = class_{jl_1} \in key\_class(u_j) \\ \widetilde{W}_{jl_2}, & class_{jl} = classjl_2 \in potential\_key(u_j) \\ \frac{1}{2}(\overline{W}_{jl_1} + \widetilde{W}_{jl_2}), & class_{jl} \text{ in both sets} \end{cases}.$$

Correspondingly the target category $\{h_k\}$ satisfies $\mathcal{KH}(h_k) \cap interests(u_j) \neq \varnothing$, and the candidate items are in each h_k as mentioned before (suppose i_k included). We modify the vector function $class\_weight(u_j)$ which is defined in section 3.3.2 by using $interests(u_j)$ to substitute $key\_class(u_j)$, i.e. W_{jl} instead of $\overline{W}_{jl}$, and compute the similarity of u_j and i_k as before. In this way, we get the similarity between items and inactive users. The algorithm can also be applied to the recommendation for the active users with smaller value of *depth*.

3.3.4 Grading Function

The grading function $grade(u_j, i_k)$ computes the possibility of acceptance (positive grade) or rejection (negative grade) with indicators of i_k's popularity and $sim(u_j, i_k)$ computed as above (see Figure 4). Then we pick out the first k candidates and sort them in descending order to generate final recommendation, where in our case $k = 3$.

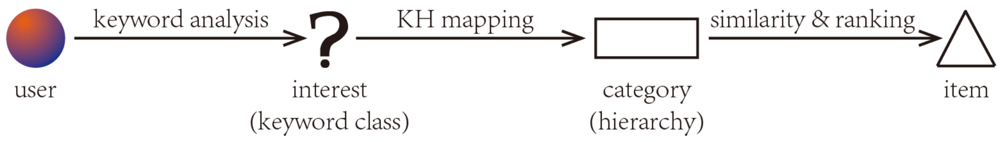

Figure 4: Recommendation process. Keywords analysis extracts user's interests, $\mathcal{KH}$ mapping match these interests to corresponding item categories thus obtains the set of candidates, and the grading function assign grades of the recommendation by computing the similarity and popularity.

Let I and H be the item set and the category (hierarchy) set as previously defined and suppose we have extracted keyword classes of each user and item. $\mathcal{KH}(h_k)$ computes the keyword classes of a given category h_k with corresponding weight of $class_{kp}$

$$\hat{W}_{kp} = average(\overline{W}_{jl_j}), i_j \in h_k, key\_class(i_j)(l_j) = class_{jl_j} = class_{kp}.$$

We revise the definition of $class\_weight()$ (see Section 3.3.3) by extending its domain to H. Then the the ratio of u_j's fondness for category h_k $fond()$ is defined on $U \times H$ by

$$fond(u_j, h_k) = g(class\_weight(u_j) \cdot class\_weight(h_k), 100),$$

where $g(x, y)$ is a normalization function.

Finally the grading function of active/inactive users is

$$grade(u_j, i_k) = 2fond(u_j, h_k)(\alpha_1 hot_k + \alpha_2 sim(u_j, i_k)) - 1, \alpha_1 + \alpha_2 = 1, \alpha_i \geq 0.$$

Valued in $[-1,1]$, the grading shows the possibility of acceptance (positive grade) or rejection (negative grade). Considering the variance of user preferences in a certain period, a revised grading function is defined as

$$revised\_grade(u_j, i_k) = \frac{1}{\lambda} time(u_j, h_k) grade(u_j, i_k),$$

where $time(u_j, h_k) = 1 + (\lambda - 1)e^t, t \in (-\infty, 0]$. t is the time of u_j's latest acceptance of items in h_k (the current time is $t = 0$ and the default time is $t = -\infty$ if no acceptance). λ indicates the proportion of recent interest (in our experiment $\lambda = 2$).

Fake users receive different treatment. Observing the difficulty to apply similarity and familiarity function, $potential\_key(u_j)$ is not generated, i.e. $interests(u_j) = key\_class(u_j)$, $W_{jl} = \overline{W}_{jl}$. The grading function of their recommendations only encompasses the hot rank and the preference:

$$grade(u_j, i_k) = (1 + fond(u_j, h_k))HOT_k - 1.$$

This definition emphasizes the popularity of the item in I and increases the grading if i_k and u_j's interests coincide.

4 Training

We perform the stochastic gradient training to obtain the optimal parameters, which affects the performance of our system. However, we omit this training of fake users' grading algorithm since its parameters are initialized already. To accelerate the training we only search the related users of u_j to compute the gradient of potential interests' weights, but in fact we should include all the nodes (users) in u_j's social network for computation.

5 Experiment and Improvement

5.1 Training Result

In our experiment we sampled 5,938 users' recommendation records from the dataset [8] stochastically and divided them into 2 subsets for training and testing. We assigned the same proportion of at(@), retweet and comment when computing $fami(u_j, u_k)$. Table 1 presents the results of the training process. The result shows an evident discrepancy of α_1, which reflects the inclination of accepting popular items. Inactive users prefer items with similar interests while active users prefer items with high popularity.

user class	user	followee	interaction	keyword	α_1
active	3919	46	87	10	0.33
inactive	1194	27	42	8	0.18
fake	825	18	2	5	/

Table 1: Training Sets and Optimal Parameters. Fake users' grading function has no parameters to update so we omit the training process of it.

5.2 Prediction and Precision Evaluation

We computed $grade(u_j, i_k)$ of all $result(u_j, i_k)$ in testing subset and generated ordered item list of u_j(see section 3.3.4) to test the trained system. The evaluation metric is the mean average precision $AP@3$ [?] which KDD Cup's organizers adopted. Table 2 presents the $MAP@3$ (mean value of $AP@3(u_j)$) results and Table 3 presents the recommended item lists and the average precision of some users. The precision of fake users' prediction is much lower than others' in our experiment due to the difficulty of their interests' extractions. Adjusting *min_action* or recommending their linkers on other related platforms like QQ might help improve the results.

active	inactive	fake	total
0.41066	0.46879	0.33606	0.41198

Table 2: Prediction Evaluation. Mining potential interests from inactive users' followees improves the performance of recommendation. Fake users' result is not good as the others.

u_j	user class	item	accepted item	$AP(u_j)$
2071402	active	1606902	1606902	0.83
		1760350	1774452	
		1774452		
942226	inactive	1606902	1606902	1.00
		1606609		
		1774452		
193889	fake	1760642	1774862	0.33
		1774684		
		1774862		

Table 3: Examples of Prediction. User 2071402 accepts the 1^{st} and 3^{rd} items, then $AP@3 = (\frac{1}{1} + \frac{2}{3})/2 = \frac{5}{6}$; User 942226 only accepts the 1^{st} item, then $AP@3 = \frac{1}{1} = 1$; User 193889 only accepts the 3^{rd} item, then $AP@3 = \frac{1}{3}$.

5.3 Improvements of the System

There are approaches to enhance the performance and overcome the limitations of our system. Recommendation based on demographic methods [?] can help in enhancing the percentage of acceptance. Refined keyword analysis and user taxonomy can improve the recommendation. Users who follow items in the same category or interact with users who have explicit preferences can be grouped in identical user class. They share synonyms in their keywords and accept similar items in a high possibility based on the similarity of preferences. Adaptation to the frequently updated microblog platform's database can get user's present interests. Fortunately user's interests and behaviours are stable in a short period, so the system only needs retrains stochastically and gradually, which is fast and accurate.

6 Conclusion and Future Work

This review presented a hybrid recommender system for microblog to solve Track 1 task, KDD Cup 2012. The system analysed the synonyms and behaviours of different users, extracted their (potential) interests, found the target categories, graded the candidate items in those categories with indicators of popularity and similarity, and finally generates ordered item lists respect to each user. Experimental result showed high performance of our algorithm. Future works includes the initialization of grading function's parameters needs improvement. Dynamic algorithms which reduce the risk of inaccuracy by searching the best algorithm through competition also deserves further study.

References

[1] K. C. 2012. Predict which users (or information sources) one user might follow in tencent weibo. http://www.kddcup2012.org/c/kddcup2012-track1, 2012.

[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, 1993.

[3] Baidu. Zombie fans on weibo. http://baike.baidu.com/view/4047998.htm, 2010.

[4] D. Cheung, J. Han, V. Ng, A. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *Parallel and Distributed Information Systems, 1996., Fourth International Conference on*, dec 1996.

[5] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 1997.

[6] Kyle. Sina commands 56% of china's microblog market. http://www.resonancechina.com/2011/03/30/sina-commands-56-of-chinas-microblog-market/, March 2011.

[7] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 2001.

[8] Y. Niu, Y. Wang, G. Sun, A. Y. B. Dalessandro, C. Perlich, and B. Hamner. *The Tencent Dataset and KDD-Cup'12.* KDD-Cup Workshop, 2012.

[9] Tencent. About tencent. http://www.tencent.com/en-us/at/abouttencent.shtml, 2012.

[10] Z. Wang, Y. Tan, and M. Zhang. Graph-based recommendation on social networks. In *Web Conference (APWEB), 2010 12th International Asia-Pacific*, 2010.