Messages Ranking in Social Network

Bo Li, Fengxian Shi, and Enhong Chen

University of Science and Technology of China, China
{libo7,sfx,cheneh}@ustc.edu.cn

Abstract. Nowadays, people engage more and more in social networks, such as Twitter, FaceBook, and Orkut, etc. In these social network sites, users create relationship with familiar or unfamiliar persons and share short messages between each other. One arising problem for these social networks is information is real-time and updates so quickly that users often feel lost in such huge information flow and struggle to find what really interest them. In this paper, we study the problem of personalized ranking in social network and use the SSRankBoost algorithm, a kind of pairwise learning to rank method to solve this problem. We evaluate our approach using a real microblog dataset for experiment and analyze the result empirically. The result shows clear improvement compared to those without ranking criteria. *abstract* environment.

Keywords: ranking, social network , microblog

1 Introduction

Nowadays, people engage more and more in social network services like Twitter, Facebook, Google+, Orkut, etc. Most of these social networks provide the short messages sharing service. People use this service heavily for staying in touch with friends and colleagues, receiving and publishing all kinds of real time information from all kinds of users with overlapping and disparate interests. On one hand, due to fast transmission rate and low cost of dissemination, the information flow updates quickly. According to recent reports, Twitter currently has over 73 million users, with more than 50,000 tweets being generated per minute. For China microblog Weibo, up to July 2010, the number of messages sent is more than 3 million per day and the average number of messages generated is close to 40 per second. On the other hand, people tend to idolize the information and have psychological dependence on it which also stimulates this information overload. For example, quite a lot of users follow hundreds or even thousands of people on microblogs.

The above two aspects bring in a problem that how users find what really interest them in this huge information flow. Imaging a Twitter lover who follows many people and these people post all kinds of messages from political to sports, from art to entertainment. To some topics the user likes, but to some other topics she/he dislikes. She/he is quite reluctant and feels frustrated if she/he has to read lots of uninteresting messages until what really catches her/his eyes and interests

her/him. Currently, most sites exhibit all the messages she/he receives in reverse chronological order when a user login in. This emphasis on time provides no guarantee that the most interesting messages appear on top, especially given that thousands of new messages are generated every minute. We think a more rational rank model is needed and thus we can better present messages to users.

In this paper, we study the message ranking problem in social network. There are some challenges for this problem. First, the text of these messages are short, usually less than 140 characters which is different from the traditional content based relevant ranking scenario. Second, due to fewer links among texts (we will use term text interchangeable to message), traditional link based ranking algorithms such as PageRank, HITS are not suitable to use. Third, the user behavior information we get is usually quite limit. For instance, in microblog, even we know user may have read a lot of messages, but we don't have explicit feedback information, like the specific extent of how user likes each message. What we have is a small portion of user behavior, such as whether she/he has replied the messages or retweeted the messages. At last, for the real-time applications, we require the time complexity of model is small. Considering all these factors, we use SSRankBoost (Semi-Supervised RankBoost) algorithm[1], a kind of learning to rank method, to solve this problem. Since user relationship is a main feature of social network, it can be mined for our ranking task. We utilize this underlying information in two perspectives: first, we analyze user authority on social network graph and exploit this as a feature for training. Second, according to the philosophy that similar users have similar interests, we measure user similarity and leverage user data to train the model. Using SSRankBoost, we modify the loss function of the model on the basis of user and his friends training sets. To evaluate the proposed approach, we use data from Sina Weibo, the biggest microblog site in China and empirically analyze the result which shows much improvement compared to those without ranking criteria.

In the remainder of the paper, first we briefly introduce some related work in section 2. Then in section 3 we apply SSRankBoost to message ranking and take microblogging as an example. At last, we present the experiment results in section 4 and conclude in section 5.

2 Related Work

Social network related problem has attracted many researchers' interests recently. Some researchers investigated the topological properties of the network [2] and user behaviors[3]. Many efforts have also been made for the influence analysis and information diffusion in social network. For example, [4] studied influence maximization on networks and proposed an algorithm to solve this problem.

Our work is different from the previous work in [5] and [6] which also tackle microblog ranking. [5] described several strategies for ranking microblogs in a real-time search engine, they proposed methods to re-rank the top-k tweets returned by an existing microblog search engine which was a non-personalized ranking while our work focus on personalized ranking. [6] studied the problem

of designing a mechanism to rank items in forums by making use of the user reviews such as "thumb" and "star" ratings. Their ranking algorithm specifically required manual user input and only utilized this feedback information, while our work and [5] take into account social network properties and properties of the message itself.

3 Personalized Messages Ranking in Social Network

Before discusssing the detail of our approach, we first describe our setting and some notations here. Generally, for each target user, the system is given a set of some labeled message set M where each sample m_i in it is associated with a label $y_i \in \{0, 1\}$, representing different level of relevance or preference. In our experiment, we set $y_1 = 1$ or $y_0 = 0$ which means user prefers a sample m_i with $y_i = 1$ more compared with a sample m_j with label $y_j = 0$. As we have only two labels and there is no order between samples with the same label, the feedback information we used is called bipartite and thus our ranking is bipartite ranking [1]. We denote M_+, M_- to represent these two disjoint sample sets, the samples with label 1 and samples with label 0. For $\forall m_+ \in M_+, m_- \in M_-$, the function $\Phi(m_+, m_-) = 1, \Phi(m_-, m_+) = -1$, while for m_1, m_2 which are in the same set, $\Phi(m_1, m_2) = 0$. Finally, the goal of learning can be defined as the search of a scoring function $H : m \rightarrow R$ which assigns higher scores to more relevant instances than to less relevant or irrelevant ones.

3.1 SSRankBoost

Assume the target user is u_0, and his friend set is $F^{'(0)} = \{u_{k_1}, u_{k_2}, \ldots, u_{k_n}\}$. The similarity between u_0 and $u_{k_i} (i = 1, \ldots, n)$ is calculated through some kind of criteria (to be discussed in following section), and we select the top K most similar users and form set $F^{(0)} = \{u_{k_1}, u_{k_2}, \ldots, u_{k_K}\}$ for u_0, their corresponding similarities are $\{s_{k_1}, s_{k_2}, \ldots, s_{k_K}\}$. For convenience, we define $s_0 = 1$ and get the user set for u_0 as $U^{(0)} = \{u_0\} \bigcup F^{(0)}$. The training samples that we clean from the source data for each user in $U^{(0)}$ is $M_i^{'(0)}$, e.g. the training samples for u_0 in $U^{(0)}$ is $M_0^{'(0)}$. To utilize friends' information for the target user u_0, beside $M_0^{'(0)}$, we also use some of the data in each $M_i^{'(0)} (i = 1, \ldots, K)$, which is represented as $M_i^{(0)}, M_i^{(0)} \subset M_i^{'(0)}$ to train the model for u_0. We denote $M_0^{(0)} = M_0^{'(0)}$, then the data used for training for u_0 is $M^{(0)} = \bigcup_{i=0}^{K} M_i^{(0)}$. Furthermore, M_i^+, M_i^- as the corresponding sample set with label 1 and label 0 for each M_i which means messages in M_i^+ are all labeled with 1 and messages in M_i^- are all labeled with 0.

In this paper, we use SSRankBoost algorithm to model user interests. The final ranking function $H(m)$ that we need in SSRankBoost is in the form of $H = \sum_{t=1}^{T} \alpha_t h_t$. Let r_i be the respectively ranking loss function for set M_i which defines in equation (2), r_i minimizes the average numbers of irrelevant examples scored better than relevant ones in each M_i separately. Then by summing all the

r_i, we have the ranking loss $r(H)$ of H defined in equation (1). In the equation, s_i is the disfactor weight for each M_i, $D_i(m_-, m_+) = 1/(\mid M_i^+ \mid\mid M_i^- \mid)$ (we will denote $P_i = \mid M_i^+ \mid\mid M_i^- \mid)$ is the weight for pair (m_-, m_+) (at the t-th iteration, we use $D_{t,i}$ to denote weight D_i), $[\pi]$ is equal to 1 if the predicate π holds and 0 otherwise in equation(2). As r_i is the respectively ranking loss function for set M_i, using the upper bound $[x \geq 0] \leq e^x$, we have its bound shown in equation (3), so the total cost $r(H)$ also have a bound.

$$r(H) = r_0 + \sum_{i=1}^{k} s_i r_i = \sum_{i=0}^{k} s_i r_i \tag{1}$$

$$r_i = \sum_{m_-, m_+ \in M_i^- \times M_i^+} D_i(m_-, m_+)[H(m_-) - H(m_+)] \tag{2}$$

$$r_i \leq \sum_{m_-, m_+ \in M_i^- \times M_i^+} D_i(m_-, m_+) exp(H(m_-) - H(m_+)) \tag{3}$$

$$r(H) = \sum_{i=0}^{k} s_i r_i \leq \varepsilon = \sum_{i=0}^{k} \sum_{m_-, m_+ \in M_i^- \times M_i^+} s_i D_i(m_-, m_+) exp(H(m_-) - H(m_+)) \tag{4}$$

In the implement of SSRankBoost, the algorithm get a weak learner h_t according to pair weights at each bound. At the beginning of the algorithm, all sample pairs are considered to be uniformly distributed, that is, the pair weight $D_{0,i}(m_-, m_+) = 1/P_i, \forall (m_-, m_+) \in M_i$. In each iteration of SSRankBoost, the weight for each pair is therefore increased or decreased depending on whether h_t orders that pair correctly, leading to the following update rules on the t-th iteration:

$$D_{t+1,i}(m_-, m_+) = \frac{D_{t,i}(m_-, m_+) exp(\alpha_t(h_t(m_-) - h_t(m_+)))}{Z_{t,i}} \tag{5}$$

Where $Z_{t,i}$ is normalization factor such that $D_{t,i}$ remains probability distributions. For a fast implement, SSRankBoost replaces the $D_{t,i}(m_-, m_+)$ with $v_{t,i}(m_-)$ and $v_{t,i}(m_+)$ and the relationship between them is following:

$$D_{t,i}(m_-, m_+) = v_{t,i}(m_-) v_{t,i}(m_+) \tag{6}$$

The corresponding framework of SSRankBoost applied to messages ranking is given in algorithm 1.

In the above framework, the algorithm needs to train a weak learner at each iteration. We also adopt {0,1}-valued weak learner and as for how to find the weak learner $h_t(m)$ and choose the weight α_t, reader can refer to SSRankBoost algorithm[1].

Algorithm 1: SSRankBoost algorithm

Input: $M_i, i = 0, 1, ..., K$,M_i is the data set from u_i
Output: ranking function $H = \sum_{i=1}^{T} \alpha_t h_t$
for $i \leftarrow 0$ **to** K **do**
  $\forall m \in M_i^+, v_i(m) = \frac{1}{|M_i^+|}$;
  $\forall m \in M_i^-, v_i(m) = \frac{1}{|M_i^-|}$;
for $i \leftarrow 0$ **to** K **do**
  $Q_{0,i} = 1$;
for $t \leftarrow 1$ **to** T **do**
  train the weak learner using $v_{t,i}(m), i = 0, 1, ..., K$;
  choose the weight α_t ; **for** $i \leftarrow 0$ **to** K **do**
    $\forall m \in M_i$,update;
    $$v_{t+1,i}(m) = \begin{cases} \frac{v_{t,i} exp(-\alpha_t h_t(m))}{Z_{t,i}^+} & \text{if} \quad m \in M_i^+ \\ \frac{v_{t,i} exp(\alpha_t h_t(m))}{Z_{t,i}^-} & \text{if} \quad m \in M_i^- \end{cases} ;$$
    where $Z_{t,i}^+, Z_{t,i}^-$ normalized $v_{t,i}$ over M_i^+ and M_i^-;
    update $Q_{t,i}$: $\quad Q_{t,i} = Q_{t-1,i} Z_{t,i}^+ Z_{t,i}^-$;

3.2 Features

In this article, we use the Sina Weibo(http://weibo.com) data for experiment and evaluate the proposed model. In this section, we introduce all the features we used in this data set. Sina Weibo is the most popular microblog service in China which is much like the Twitter. We use the open API that it provides and crawle the data from December 2011 to February 2012. In this dataset, there are 224977 users and 196499 messages.

To learn a ranking function, it is necessary to design a suitable set of features which will impact performances significantly. In this dataset, we implement the following features for target user u_x and the specific message m:
1) Message aspect features: the tf-idf value, length of text, whether the text contains a URL, topic vector extracted by the LDA model, the retweet number of the message, the replied number of message.
2) Author aspect features: the author authority, the author's message number.
3) Combined features: the interactions ratio of u_x and u_y (u_x is the target user, u_y is the author of specific message m), the common friend ratio of u_x and u_y, the similarity of them. Here interaction means the behavior of user u_x retwweted/replied messages from u_y.

In the above features, we use the topic of the messages which is extracted by using unsupervised learning method LDA[7]. For features in terms of author, we use the author authority. Here, author authority means user's PageRank value in social network graph. The graph vertices represent users and edges representing the follower relationship between them, in another word, $w_(i, j)$ is

1 in the adjacency matrix of the graph if u_i follow u_j. Then we apply PageRank on this graph and get each user's authority.

3.3 User Similarity

To caculate the user similarity, We think there are some rules that can be obeyed: 1) the more common friends u_a and u_b have, the more similar they are; 2) the more interactions they have, the more similar they are; 3) the more similar their texts are, the more similar they are.
Let the common friends number of u_x and u_y be s, and their friends number be L_{u_x} and L_{u_y} respectively, the total interaction number that they have are I_{u_x} and I_{u_y} respectively, then we define $cf\_sim(u_x, u_y) = \frac{s}{L_{u_x}}, in\_sim(u_x, u_y) = \frac{I_{u_x u_y}}{I_{u_x}}$, where $I_{u_x u_y}$ denotes the number of interactions between u_x and u_y. At last, we also consider text similarity between users. We treat all the messages one user writes to be an article. Then we extract the topic by using LDA, whose each element represents the weight of the topic. We can also caculate the TF-IDF vector of each article. Assume the topic vector for u_x and u_y are $\overrightarrow{v_1}, \overrightarrow{v_2}$, while TF-IDF vectors are $\overrightarrow{t_1}, \overrightarrow{t_2}$, we define $text\_sim(u_x, u_y) = 0.5sim(\overrightarrow{v_1}, \overrightarrow{v_2}) + 0.5sim(\overrightarrow{t_1}, \overrightarrow{t_2})$ where $sim(\overrightarrow{x}, \overrightarrow{y})$ is two consine correlation between $\overrightarrow{x}, \overrightarrow{y}$. Combining the three similiarities, we finally get the similarity between u_x, u_y with following equation:

$$sim(u_x, u_y) = \alpha cf\_sim(u_x, u_y) + \beta in\_sim(u_x, u_y) + \gamma text\_sim(u_x, u_y) \quad (7)$$

Where $\alpha > 0, \beta > 0, \gamma > 0, \alpha + \beta + \gamma = 1$ and those three constants are the importance weights of the three kinds of similarities respectively.

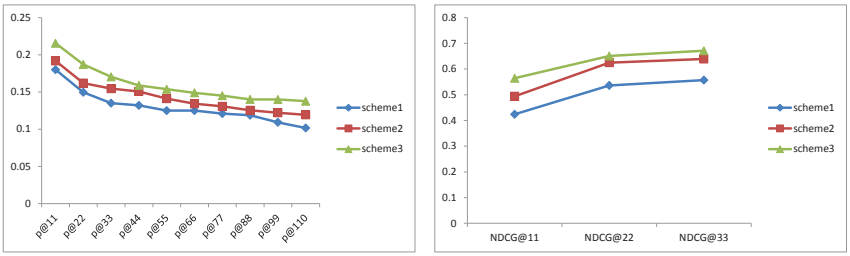
3.4 Choose Microblog Samples

For a target user u, the behavior information we have is which messages she/he retweeted/replied, and which messages she/he didn't retweet/reply. It's obvious that if user retweets/replies a message, it indicates that user likes it. Assume that the messages user received from friends form a list $L = \{m_0, m_1, \ldots, m_t\}$ in which messages are arranged in reverse chronological order. If user retweetes/replies m_k, then we set y_k to be 1 and for messages in set $N_5(m_k) = \{m_i, m_i \notin I(u) \wedge m_i \in L, i = k-5, \ldots, k-1, k+1, \ldots, k+5\}$, we set their lables to be 0. Here $I(u)$ is set of messages user have interacted with, $N_5(m_k)$ is the neighbor of m_k whose distance to m_k is no more than 5 in L. We didn't choose all the messages which u didn't have interactions because we don't know whether user likes it or not. But it's strongly believed that if two messages m_x and m_y appear at the successive positions and if u retweeted/replied m_x while didn't do it for m_y, then we can say that m_x is more preferred by u, and value $\Phi(m_x, m_y)$ can bet set to 1.

4 Experiment Result

We select 1023 users for experiments and use the metric precison and NDCG[8] to evaluate our results. We implement three schemes to analyze the impact of

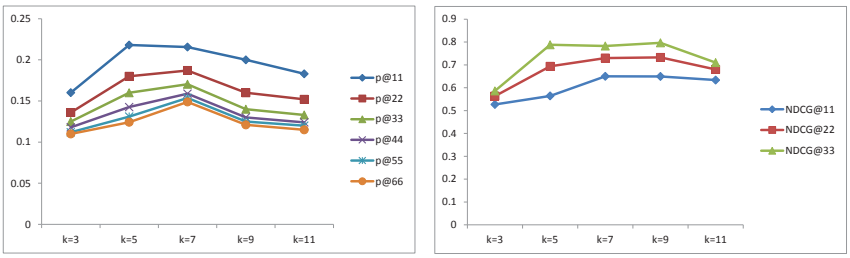
friends' data for users. The first scheme is to use all user's data for all users, which means we don't distinguish each user's data and it's not a personalized ranking. For instance, if there is 10 users, each has 100 samples, then we train the general model once with the 1000 samples and use this model for every user. This considers that every user has similar taste. The second scheme is to use user's own data for each user which suffers from the data spareness problem. The third scheme is to use k user's friend combined with user's own data to train the model, which is the method we discuss in this paper. Obviously, the second and third schemes are personalized ranking.



(a) precision of three schemes (b) NDCG of three schemes

Fig. 1. results of three schemes

Figure 1(a) and 1(b) show the result of precision, NDCG of the three schemes. Comparing the result of scheme 1,2 to 3, we can see the personalized ranking outperforms non-personalized ranking. It shows scheme 3 performs better than scheme 1,2 clearly. By comparing scheme 2 and scheme 3 in each figure, we can see the impact of friends data and the effects of our algorithm. From Figure 1(a) and 1(b), we see incorporating friend data using our algorithm really improves the result.



(a) precision of three schemes (b) NDCG of three schemes

Fig. 2. results of different K

We also study the impact of different K in the algorithm. For a target user and each of his K friends, we select C messages for training. Figure 2(a) and Figure 2(b) show the result of different K. From the figures, we can see when $K \geq 5$ and $K \leq 9$, the performance is best. When K is small, the performance improves as K grows. However, when $K \geq 9$, the benefit of friend data is decreasing, and the results become worse. To explain the results, we analyze the data and find that in the top K($K \geq 9$) friends lists, there are some users who have many followers and generate messages very frequently. These users mostly are celebrities or organizations and most of the common users have followed them for receiving news. In some sense, the data from these users could not characterize the target user's interest very well. For this reason, when too much data is used, the benefit of friend data is decreased.

5 conclusion

Generally, this paper studies how to filter the real time web to find the most interesting messages for each user in social network. We apply SSRankBoost algorithm to rank the messages that user subscribed in social network. We utilize all kinds of features to evaluate the ranking method in the real data set. We think there is still a lot of room for improvement. One way is to incorporate more features into the ranking model, such as using the replied/retweeted author information of the message, the retweeted speed of the messages, the novelty of the message, etc.

References

1. Amini, M.R., Truong, T.V., Goutte, C.: A boosting algorithm for learning bipartite ranking functions with partially labeled data. In: SIGIR. (2008) 99–106
2. Mislove, A., Marcon, M., Gummadi, P.K., Druschel, P., Bhattacharjee, B.: Measurement and analysis of online social networks. In: Internet Measurement Comference. (2007) 29–42
3. Schneider, F., Feldmann, A., Krishnamurthy, B., Willinger, W.: Understanding online social network usage from a network perspective. In: Internet Measurement Conference. (2009) 35–48
4. M.Richardson, P.Domingos: Mining knowledge-sharing sites for viral marketing. In KDD'02 (2002) 61–70
5. Nagmoti, R., Teredesai, A., Cock, M.D.: Ranking approaches for microblog search. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology **1** (2010)
6. Sarma, A.D., Sarma, A.D., Gollapudi, S., Panigrahy, R.: Ranking mechanisms in twitter-like forums. In: WSDM. (2010) 21–30
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3** (2003) 993–1022
8. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. **20**(4) (2002) 422–446