TeRec: A Temporal Recommender System Over Tweet Stream

Chen Chen Hongzhi Yin Junjie Yao Bin Cui
Department of Computer Science and Technology
Key Laboratory of High Confidence Software Technologies, Peking University
{chenchen628, bestzhi, junjie.yao, bin.cui}@pku.edu.cn

ABSTRACT

As social media further integrates into our daily lives, people are increasingly immersed in real-time social streams via services such as Twitter and Weibo. One important observation in these online social platforms is that users' interests and the popularity of topics shift very fast, which poses great challenges on existing recommender systems to provide the right topics at the right time. In this paper, we extend the online ranking technique and propose a temporal recommender system – *TeRec*. In TeRec, when posting tweets, users can get recommendations of topics (hashtags) according to their real-time interests, they can also generate fast feedbacks according to the recommendations. TeRec provides the browser-based client interface which enables the users to access the real time topic recommendations, and the server side processes and stores the real-time stream data. The experimental study demonstrates the superiority of TeRec in terms of temporal recommendation accuracy.

1. INTRODUCTION

Online social streams such as Facebook News Feed and Google Buzz have emerged as important channels of online information. Real-time microblogging services, such as Twitter (twitter.com) and Weibo (weibo.com), have experienced an explosion in global user adoption over the past years. Millions of people are reading statuses, tweets and learning breaking news, useful tips and funny stories to keep up with their friends' daily lives.

Users of these online social services not only encounter the problem of information overload, but also have mutable interests which change fast along with the social information streams. These features pose great challenges to recommender systems, since the purpose of such systems is to provide suitable recommendations that match users' real-time interests, which is quite difficult among massive candidates and fast changing user preferences.

Most of state-of-the-art recommender systems are based on the popular method known as collaborative filtering (CF) which achieves good results in modeling user preferences and item features offline, but they usually suffer from at least one of the following drawbacks when dealing with the situation mentioned above: 1. Delay on model updates caused by the expensive time cost of re-running the offline CF model. 2. Loss in recommendation accuracy due to the sacrifice of user modeling, in order to pursue a faster model updating speed. 3. Disability to capture users' latest interest due to the fact that latest entries used for updating CF models are often overwhelmed by the large data of the past.

For example, a Weibo user Tim has been reading and tagging posts with hashtag *#JapanEarthquake* for a long time, but recently he started to focus on posts with hashtag *#AmericanElection*, indicating the shift of his recent interest. A real temporal recommender system should have the ability to capture this signal and make fast responses to the recommendation list accordingly. Most existing systems, however, fail to capture the shifting signal in real time, resulting in a long delay to update user interests.

In this demonstration, we propose *TeRec*, a recommender system which can overcome the above drawbacks. TeRec works in a stream data environment (Weibo), and provide real-time recommendations according to users' preference at any specific moment. Instead of generating recommendations using only item similarities and without user modeling [1], TeRec models users and items using competitive matrix factorization which can achieve more accurate results.

The basic idea of TeRec is to use the *hashtags* as the surrogates for interesting topics. When a user is about to post a tweet, our system predicts the user's current interests, and recommends several topics (hashtags) that he might be interested to use in this tweet. After a tweet is posted, TeRec updates the matrix of factors instantly according to the actual hashtag user used, so that it can prepare with a new recommended list the next time the user wants to post. In the following, we derive three key features in our system TeRec to facilitate temporal recommendation and real-time updating.

First, on the contrary of most existing recommender systems, TeRec doesn't wait until a certain amount of inputs have been accumulated before updating the recommender model and results. On each of the new entries, TeRec performs instantaneous updates on the latent factors of related user and items. To speed up the updating process, we extend the technique used in [5] by keeping a representative sampled set of the dataset in a reservoir and update the latent factors using only this reservoir and the newly observed input.

Proceedings of the VLDB Endowment, Vol. 6, No. 12

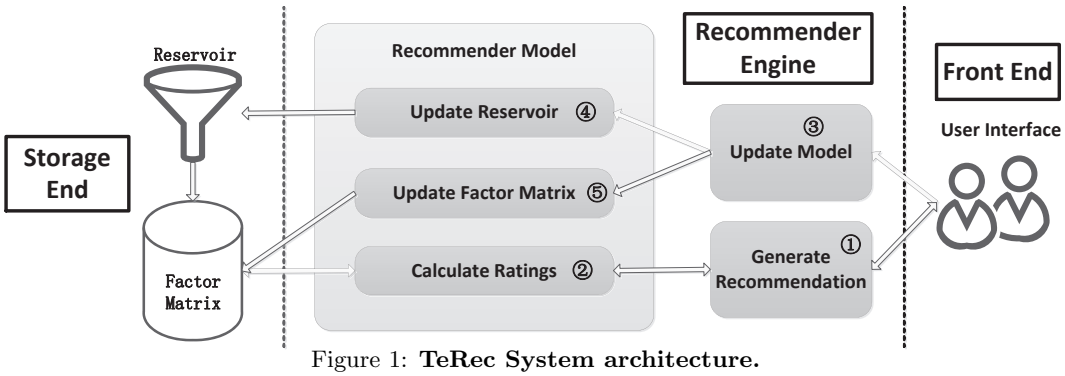


Figure 1: **TeRec System architecture.**

Second, we propose a new sampling strategy of the reservoir, in order to keep a sample of instances representing users' real-time interests, rather than overall preferences.

Third, we exploit two selective sampling strategies for the negative instances used in competitive collaborative filtering, to ensure that the most informative instances are used for model updates.

On top of that, we construct our temporal recommender system over tweet stream, named $TeRec$, which enhances the collaborative competitive approach for matrix factorization [6] with the proposed online updating technique. To competitively update the model in real time, we sample the most representative positive and informative negative instances instead of using all instances buffered in the reservoir, to perform stochastic gradient descent updates based on active learning principles [3][7].

We construct the $TeRec$ system for hashtag recommendation over tweet stream, and also conduct extensive experiments by comparing with state-of-the-art recommendation approaches on the real-world dataset crawled from Weibo.

The rest of this paper is organized as follows. In section 2 we details the architecture of our system and the real-time recommender model. We show the user interface of our system and the experimental comparison with existing works in section 3, and conclude in section 4.

2. SYSTEM ARCHITECTURE AND ALGORITHMS

In this section, we introduce the system framework and explain the details of the two kernel factors of our algorithm.

2.1 System Architecture

TeRec mainly provides a browser-based service which helps users to choose proper hashtags when they post tweets. The server side of the system models and stores user preferences, calculate predicted ratings between users and hashtags, and recommends a list of hashtags as users post tweets at the browser side of the system.

As shown in Figure 1, our system works in three layers. The front end, mainly consisting of User Interface, deals with the interactions between user and data, such as displaying the recommendation results and collecting user feedbacks. In the storage end, we keep a matrix of factors to represent user preferences and item characteristics, we also maintain a reservoir of input entries instead of saving all past information in the system.

The key component of the recommender system is between UI and storage layers, which performs two tasks, i.e., generate recommendation and update recommender model. The running process of TeRec is also shown in Figure 1. As data and requests come into the system in a stream, TeRec works as follows: 1. User Interface receives requests from users and asks recommender for recommendations. 2. Recommender Model calculates ratings and returns recommended lists to User Interface. 3. User Interface receive user feedback and update the recommender model. 4. Recommender Model updates the reservoir of past entries. 5. Recommender Model updates the matrix of factors using updated reservoir.

The recommender model behind TeRec is based on the competitive Matrix Factorization [4][6], and our major contribution in this work is the novel updating strategy to facilitate real time processing in streaming scenario. Specifically, the updating process consists of two parts: 1. Maintain a reservoir which contains the most informative entries. 2. Select useful positive and negative instances to perform competitive model updates incrementally. We present our designed algorithms of these two processes in the following.

2.2 Temporal Reservoir Update

Considering that the data could be very huge and most of them are useless, we need to sample a subset of informative inputs (i.e. the reservoir) to speed up model updates.

Specifically, we employ and extend the technique of random sampling with a reservoir [5], which is widely used in data streaming, and recently has been proposed for binary classification [8] and recommendation [2].

For simplicity, we represent the reservoir as a list $R_t := \{s_1, s_2, \cdots, s_c\}$ denoting the reservoir maintained at time t including c instances from input stream S. The traditional reservoir technique, widely used in recent literatures [8, 2], aims to capture an accurate "sketch" of all history data under the constraint of fixed space c, while we are more interested in users' recent as well as current behaviors, than their overall preferences, since our task is to produce temporal recommendations, which requires our model to capture user's interests precisely at the moment recommendations being provided. So, we propose a novel mechanism to create and update the reservoir, which is specially designed for our temporal recommendation task.

When the size of input data reaches c, the fixed size of the reservoir, we need to replace some of the instances stored in the reservoir with the newly observed inputs to maintain the fixed size of the reservoir.

Different from Vitter's algorithm in [5], which propose to accept the t-th data with probability $\frac{c}{t}$ and replace an random instance from the reservoir in the purpose of creating

an accurate "sketch" of current dataset, we present our new reservoir sampling mechanism, which better fits the context of temporal recommendation in social streams. The t-th data instance is added into the reservoir with probability $1 - \frac{c}{t}$ which encodes our intuition that the update frequency of the reservoir increases with the increasing time t, since it is not necessary to update the reservoir soon after the model is initialized using all available training data. If the new data is decided to be put into the reservoir, the probability of data instance s_i, which is already in the reservoir, being replaced is $1 - P(s_i \in R_{t-1})$, in which $P(s_i \in R_{t-1})$ is defined as follows:

$$P(s_i \in R_{t-1}) \propto \exp \frac{1}{t-i} \tag{1}$$

where $P(s_i \in R_{t-1})$ is an exponential decay function, widely used in the time series analysis; $t - i$ denotes the difference between the current time order t and the time order i of the data instance s_i arriving in our system. The above function performs quite well in our TeRec system and has the distinct ability to capture users' representative recent behaviors and interests in real-time.

2.3 Incremental Model Update

On top of the reservoir introduced above, which reserves instances indicating user's recent preferences, our model updates dynamically with every newly arriving data, and produces its recommendations always according to the latest updated model. Figure 2 presents the process of online incremental model update. In order to update the model in real time, we only update the latent factors of current u, the representative positive items $p \in SP_u$ sampled by **SamplePositiveInput** function, and the informative negative items $n \in SN_u$ sampled by **SampleNegativeInput**. The sampled items are used to represent an sketch of user's latest positive and negative preferences.

Input: Continually coming tuple of (u, i, t), indicating the event at t that user u used item i
Output: (W, H) that represent the latent factors of users and items

//select positive inputs from user history
$SP_u = \textbf{SamplePositiveInput}(u, t-1) \cup i$;
//select negative inputs to represent user's negative preferences
$SN_u = \textbf{SampleNegativeInput}(u, t)$;
for $round = 1$ ***to*** T **do**
  for ***each*** p ***in*** SP_u **do**
    $\hat{R}_{up} = \textbf{Rate}(u, p)$;
    $\hat{R}_{u\overline{SN_u}} = \textbf{Rate}(u, \overline{SN_u})$;
    $\eta = \textbf{hinge}(\hat{R}_{up} - \hat{R}_{u\overline{SN_u}})$;
    //update the model of related user and items
    $W_u = W_u + \alpha\eta(H_p - \frac{\sum_{j \in SN_u} H_j}{|SN_u|}) - \alpha\beta W_u$;
    $H_p = H_p + \alpha\eta W_u - \alpha\beta H_p$;
    for ***each*** j ***in*** SN_u **do**
      $H_j = H_j - \alpha\eta W_u - \alpha\beta H_j$;
    end
  end
end
return (W, H);

Figure 2: **Algorithm of Incremental Model Update.**

Since in the reservoir, we only store user and item pairs observed in the stream which can reflect users' recent behaviors and interests, selecting positive inputs can be a quite simple process of uniformly sampling a subset from R_{t-1} at random. But how to sample a few negative items from N_u to represent u's negative preference is still a problem. In order to acquire as much information as possible from the sampled SN_u for model updating, we design two selecting strategies as follows.

- The sampled negative items should have high rating scores predicted by the current model, so that they can provide enough information for correcting the model to the right direction.
- The sampled negative items should be overall highly rated by other users recently, but not rated by the given user so that they can distinguish u's unique preferences from others.

The intuitions behind the proposed two selective sampling strategies are as follows. (1) Assuming that u is not interested in item i, however the predicted $\hat{R_{ui}}$ under current model is quite high. Thus, it's obvious that the latent factors of u and i need to be updated to reduce $\hat{R_{ui}}$. So i should be added into SN_u with priority. On the contrary, an item with low predicted rating score is less informative to be sampled into SN_u, because updating it will make little difference to correct the model. (2) As for the second strategy, the items which are overall highly rated by other users, but not rated by the given user, are more discriminative and informative to better capture the user's unique interests.

3. DEMONSTRATION

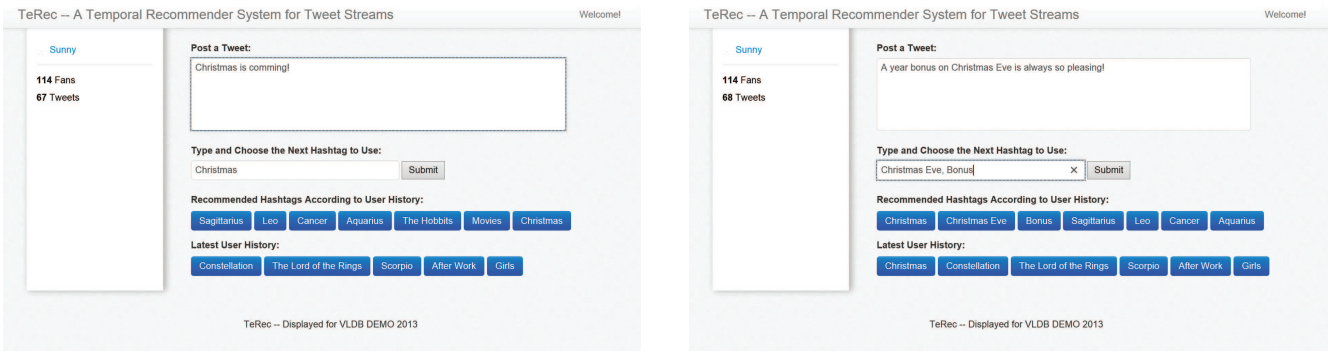
Based on the aforementioned system architecture, we construct our TeRec system which can recommend hashtags for users who post tweets on Weibo. In the following subsections, we introduce the user interface and functions of our system, and show some experimental results comparing with some existing methods based on the real-world dataset crawled from Weibo.

3.1 User Interface and Case Study

The TeRec system consists of a web-based service. The primary UI is shown in Figure 3. In the system, when users post tweets, the system automatically suggest a list of hashtags for them to use. These hashtags are selected for a user according to their rating score predicted by our temporal recommender model. The users can choose to click on some of the recommended hashtags to take them in use, or use none of them. Either way, TeRec will record the feedback and update recommender model immediately. When the user wants to post a tweet again, he will be presented a different recommended list of hashtags.

The two parts in Figure 3 shows a real case from the usage of our system.

The presented user is a young girl who has just graduated. From the history of her tweets and tags, we can learn that she's quite fond of constellation and frequently posts and tags about it. So at the time point of subfigure 3a, the recommended list of hashtags mainly consists of constellation names. The system also recommended some movies and a tag about Christmas to her. After the user choose Christmas as tag, TeRec receives the feedback and updates recommender model instantly. As we can see from subfigure 3b, the list of recommended hashtags has been updated. It now contains Christmas Eve and Bonus that may interest her more than some old movies.



(a) Recommended hashtags before user feedback (b) Updated recommend list after user feedback

Figure 3: **User Interface and Example Case.**

3.2 Experiments

The experiments are conducted to stimulate a real-world circumstance under which our system is used by hundreds of thousands of people. We used Weibo dataset as the input stream, and tested *TeRec* recommender system against several other algorithms.

3.2.1 Dataset

The Weibo dataset consists of nearly 500 million tweets posted within about 7 months. We extract $\langle uid, tid, time \rangle$ features from each tweet and list them according to *time*. To better verify the performance of our model, we set up a 15-10 limitation to users and tags in our dataset. That is, each user in the dataset has used at least 15 tags and each tag is used at least for 10 times. After applying the 15-10 limitation, our dataset consists of 87287 users, 29334 tags and nearly 20 million tweets.

3.2.2 Experimental Results

We use Top-N Recall as our evaluation metric to measure the performance of our model. That means, at each time we suggest a Top-N list to a user, and check whether the actually used item is among the Top-N list.

The baseline approaches include: 1. the naive solution of Topic Popularity, which ranks the hashtags simply by their current popularity of usage. 2. WRMF, one of the state-or-the-art offline matrix factorization model for item predictions [4]. 3. RMFX, a model proposed in Ernesto's [2] recent work, which can achieve partly online and much quicker updates of matrix factorization for item prediction.

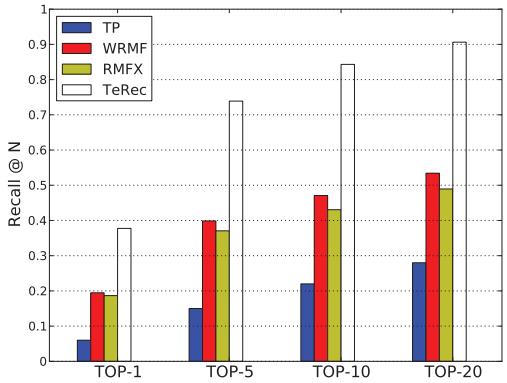


Figure 4: **Experiment results.**

Figure 4 shows the experimental results of different recommender models on this dataset. We can tell from the result that, our model not only outperforms the naive solution, but also have significant improvement on the well-performed WRMF and RMFX models.

In general, our demo system can perform well under both practical observation and experimental stimulation.

4. CONCLUSIONS

We proposed TeRec, a real-time recommender system, which can model user interest instantly and provide recommendations according to their latest preferences. TeRec provides a web-based service which allows users to post tweets, receive recommended hashtags and give feedbacks. We designed several strategies to ensure the accuracy of the recommender model, as well as the fast updating speed. We tested our system in both practical use and experimental studies, and the results showed the superiority of our system for real time hashtag recommendation for tweet streams.

5. ACKNOWLEDGEMENT

This research was supported by the National Natural Science Foundation of China under Grant No. 60933004, 61073019 and 61272155.

6. REFERENCES

[1] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. Streamrec: a real-time recommender system. In *SIGMOD*, pages 1243–1246, 2011.

[2] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *ACM Recsys*, pages 59–66. ACM, 2012.

[3] S. Ertekin, J. Huang, L. Bottou, and L. Giles. Learning on the border: active learning in imbalanced data classification. In *CIKM*, pages 127–136. ACM, 2007.

[4] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. IEEE, 2008.

[5] J. Vitter. Random sampling with a reservoir. *TOMS*, 11(1):37–57, 1985.

[6] S. Yang, B. Long, A. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *SIGIR*, volume 11, pages 295–304, 2011.

[7] H. Yu. Svm selective sampling for ranking with application to data retrieval. In *KDD*, pages 354–363. ACM, 2005.

[8] P. Zhao, S. Hoi, R. Jin, and T. Yang. Online auc maximization. In *ICML*, 2011.