TRUPI: Twitter Recommendation based on Users' Personal Interests

Hicham G. Elmongui[1,2], Riham Mansour[3], Hader Morsy[4], Shaymaa Khater[5], Ahmed El-Sharkasy[4], and Rania Ibrahim[4]

[1] Alexandria University, Computer and Systems Engineering, Alexandria 21544, Egypt, elmongui@alexu.edu.eg,
[2] GIS Technology Innovation Center, Umm Al-Qura University, Makkah, Saudi Arabia elmongui@gistic.org
[3] Microsoft Research Advanced Technology Lab, Cairo, Egypt rihamma@microsoft.com
[4] Alexandria University, SmartCI Research Center, Alexandria 21544, Egypt, {hader,sharkasy,ribrahim}@mena.vt.edu
[5] Virginia Tech, Computer Science Department, Blacksburg, VA 24061, USA skhater@vt.edu

Abstract. Twitter has emerged as one of the most powerful micro-blogging services for real-time sharing of information on the web. The large volume of posts in several topics is overwhelming to twitter users who might be interested in only few topics. To this end, we propose TRUPI, a personalized recommendation system for the timelines of twitter users where tweets are ranked by the user's personal interests. The proposed system combines the user social features and interactions as well as the history of her tweets content to attain her interests. The system captures the users interests dynamically by modeling them as a time variant in different topics to accommodate the change of these interests over time. More specifically, we combine a set of machine learning and natural language processing techniques to analyze the topics of the various tweets posted on the user's timeline and rank them based on her dynamically detected interests. Our extensive performance evaluation on a publicly available dataset demonstrates the effectiveness of TRUPI and shows that it outperforms the competitive state of the art by 25% on nDCG@25, and 14% on MAP.

Keywords: Twitter, personalized recommendation, dynamic interests

1 Introduction

Twitter has emerged as one of the most powerful micro-blogging services for real-time sharing of information on the web. Twitter has more than 500 million

users [6], and the volume of tweets one receives is persistently increasing especially that 78% of Twitter users are on the ubiquitous mobile devices [34].

The high volume of tweets is getting overwhelming and reduces one's productivity to the point that more than half US companies do not allow employees to visit social networking sites for any reason while at work [28]. In addition, a large base of twitter users tend to post short messages of 140 characters reflecting a variety of topics. Individual users' interests vary over the time, which is evidenced by the dynamic *Trends* feature of Twitter, which suggests hashtags, metadata tags prefixed by the symbol #, to the users based on her followees [32].

For the above reasons, Twitter users share a need to digest the overwhelming volume. Recommending interesting content to the user is harder in the case of Twitter, and microblogs in general, as the tweet is limited in size and thus leaks the context in which it was posted. Existing systems that recommend tweets to the users either 1) provide ranking models that are not personalized to the users' interests; 2) do not capture the dynamic change in the user's interest over the time; 3) use Latent Drichlet Allocation (LDA) [5] to represent user's interests and hence is not scalable to large datasets [25] ; or, 4) assume special user marking to the tweets of interests [7, 9, 10, 13, 19, 27, 36, 39].

In this paper, we propose TRUPI, a Twitter Recommendation based on User's Personal Interests. TRUPI aims at presenting the tweets on the user timeline in an order such that tweets that are more interesting to her appears first. In order to do so, TRUPI learns the changing interests of the users over the time, and then ranks the received tweets accordingly. More specifically, TRUPI employs an ensemble of interest classifiers that indicate the most probable interest label of each tweet on the user's timeline. Tweets are then fed into a ranking model to order the tweets based on the current user's interests. The user's interests are modeled as a time variant level of interests in different topics.

The rest of the paper is organized as follows. Section 2 highlights related work. Section 3 presents an overview of TRUPI. Section 4 gives details of the interest detection and the tweet ranking. In Section 5, TRUPI's extensive performance evaluation is presented before we conclude by a summary in Section 7.

2 Related Work

Many recommendation systems have been proposed in the literature for Twitter. Research efforts go into many directions: from recommending hashtags [12, 21, 41] and recommending URLs [6] to providing news recommendations [26] and suggesting followees [15, 16, 20].

New approaches have been proposed to deal with recommending tweets on the user's timeline. Duan et al. use a learning-to rank algorithm using content relevance, account authority, and tweet-specific features to rank the tweets [9]. Uysal and Croft construct a tweet ranking model making use of the user's retweet behavior. They rank both the tweets and the users based on their likelihood

[6] http://www.statisticbrain.com/twitter-statistics/ visited March 2014

of getting a tweet re-tweeted [36]. Similarly, Feng and Wang propose personalized recommendation for tweets [10]. In their evaluation, the metric of measuring the interest in a tweet is whether the user would re-tweet or not. Our proposed recommender is different from these approaches as their ranking models are not personalized, and they do not capture the dynamic change in the user's interest.

Nevertheless, Pennacchiotti et al. introduce the problem of recommending tweets that match a user's interests and likes [27]. Also, Chen et al. recommend tweets based on collaborative ranking to capture personal interests [7]. These two propositions neither account for the change in the user interest over time nor work on the semantic level.

Guo et al. propose *Tweet Rank*, a personalized tweet ranking mechanism that enables the user to mark tweets as interesting by defining some interest labels [13]. Our proposed approach does not assume any special user marking.

Yan et al. propose a graph-theoretic model for tweet recommendation [39]. Their model ranks tweets and their authors simultaneously using several networks: the social network connecting the users, the network connecting the tweets, and the network that ties the two together. They represent user interest using LDA, which is not scalable for large datasets [25].

Little work has been done in the dynamic personalized tweet recommendation that accounts for the change in the user interests. Abel et al. explore the temporal dynamics of users' profiles benefiting from semantic enrichment [2]. They recommend news articles, *and not tweets*, for topic-based profiles.

Our previous work proposes an approach for dynamic personalized tweet recommendation using LDA [19]. In that work, a model is defined to classify the tweet into important or not important tweet. In TRUPI, tweets are ranked based on the user's dynamic level of interest in the tweet topic. TRUPI explores the tweet content (and semantic) along with numerous additional social features.

3 Overview of TRUPI

TRUPI recommender consists of two phases. The first phase is to create the user profiles, which contains the topics of interests to the user. The user profile is dynamic; i.e., it changes over time. The second phase occurs in an online fashion to give a ranking score to the incoming tweet. This ranking score would provide for presenting the tweet to the user by its importance.

User profiles contain the different topics of interest to the users. Each profile is characterized by a weighted set of interests or topics (e.g., sports, politics, movies, etc.) The weights represent the probability that a user is interested in a certain topic. Such probabilities are learned from the user history as follows.

First, the tweets are classified into different topics. The topics are learned from a large chunk of tweets. Those tweets are clustered so that each cluster contains tweets corresponding to the same topic of interest. Next, each cluster is labeled with the topic with the help of a series of topic classifiers. This process results into the capability of placing an incoming tweet to its matching cluster, and hence, the tweet would be labeled with the same topic as the cluster.

The dynamic user profiles are created from the history of each individual user. Her last history tweets are used to compute the probability distribution of her topics of interests. The history tweets may either be time-based or count-based. A *time-based history* of length t contains the tweets that appeared on her timeline in the last t time units. A *count-based history* of size k contains the last k tweets that appeared on her timeline. While the former might reflect the recent history, the latter might have to be used in the case of low activity on the timeline.

When the system receives an incoming tweet, it consults the user profile in order to give a ranking score to this tweet. Many features are used in this scoring technique. Those features reflect how much interest the user may have in the topic of the tweet or in the sender (e.g., a close friend or celebrity). Such features would be extracted from the history of the user. They would reflect how she interacted with the recent tweets of the same tweet topic or the past tweets coming from this sender.

Tweets are preprocessed before entering our system. The aim of this preprocessing is to prepare the tweet for the subsequent algorithms. The text of the tweet is normalized as follows:

- Ignore tweets that do not convey much information. This includes tweets with number of tokens less than 2.
- Replace slang words with their lexical meaning using a slang dictionary. We use the Internet Slang Dictionary & Translator [1].
- Lexically normalize extended words into their canonical form. We replace any successive repetition of more than two characters with only two occurrences. For instance, *coooool* will be transformed into *cool*.
- Normalize Out-of-Vocabulary (OOV) words. We detect OOV words in a tweet using GNU Aspell [11]. For each OOV word, we get candidate replacements based on lexical and phonemic distance. We then replace the OOV word with the correct candidate based on edit distance and context.
- Identify named entities in the tweet and associate them with it for later usage. We use the Twitter NLP tools [3].
- Extract hashtags from the tweet and associate them with it.
- Run a Twitter spam filter [30] on any tweet containing a URL. If a tweet turns out to be spam, it is ignored. Otherwise, extract the URLs from the tweet and associate them with it.
- Represent the tweet as a feature vector using TF-IDF representation [29]. To emphasize on the importance of the tweet, we doubled the weights for the hashtags and named entities. This is in line with the fact that tweets with hashtags get two times more engagement as stated in [17].

4 Interest Detection and Tweet Ranking

This section provides details for how the user's interests are detected and how his tweets would be ranked in TRUPI, the proposed Twitter recommendation system.

4.1 Tweet Topic Classification

Since our main target is to capture the user's dynamic interests on the semantic level, we will need first to understand the tweet by classifying the tweets into topics (i.e., sports, music, politics, etc.)

Tweet Clustering and Classification It is worth mentioning that since the tweet is short in nature, its context is leaked and it is not easy to capture it. Hence, we enrich the tweets' text by grouping tweets that talk about the same topic in one cluster using the Online Incremental Clustering approach [4].

Online incremental clustering is applied by finding the most similar cluster to a given tweet, which is the one whose centroid has the maximum cosine similarity with the tweet [24]. If the cosine similarity is above a certain threshold, the tweet is inserted into that cluster. Otherwise, the tweet forms a new cluster by itself.

Next, topic-based binary SVM classifiers are applied to classify the cluster into one of the topics [8]. The cluster is classified into a topic if the confidence score exceed a certain threshold. The binary SVM classifiers were trained by labeling the tweets using a predefined list of keyword-topic pairs that is obtained by crawling DMOZ – the Open Directory Project [31]. The pairs are constructed as follows. From dmoz.org/Recreation/Food/, for instance, we get <drink, cheese, meat, ... >. We create the list of *keyword-topic* pairs as <drink, Food>, <cheese, Food>, <meat, Food> ...

Two white lists are automatically constructed for each topic; one for hashtags, and one for named entities. The white lists consist of hashtags and named entities that belong to a certain topic. The construction is described in Section 4.1.

Upon the arrival of a tweet to the user, it is clustered using the used online incremental clustering algorithm. The tweet will be labeled with the same label of the cluster it belongs to. If the tweet does not belong to any cluster, or if the tweet belongs to an unlabeled cluster, we check whether the tweet contains hashtags or named entities that belong to our white lists. In this case, the tweet is labeled with the corresponding topic. Otherwise, we try URL labeling, which labels the tweet using the content and the slugs of the URL.

White Lists Construction Two white lists are automatically constructed for each topic. One list for named entities and one for hashtags contained in the tweets. These white lists would be looked up for incoming tweets. The rational behind these white lists is that some named entities or hashtags are associated with specific topics. For instance, *Ronaldo* and *#WorldCup* would be associated with *sports*, whereas *Madonna* and *#Beatles* are associated with the topic *music*.

Constructing Named Entities White Lists: We used DBpedia [22], a large-scale multilingual knowledge base extracted from the well-known online encyclopedia, Wikipedia [37]. First, we retrieved the different resources; i.e., named entities, along with their types. Then we grouped the types that belong to the same topic together. This is done by projecting the types on the formed clusters

Table 1. Examples of Wikipedia's Named Entities Retrieved from DBpedia.

Sports	Music	Politics	Food
Jim Hutto	Nelson Bragg	Dante Fascell	Pisto
Tiger Jones	Maurice Purtill	James Kennedy	Teacake
Stephon Heyer	Michael Card	Riffith Evans	Apple pie
Allen Patrick	Faizal Tahir	Barack Obama	Pancake
Fujie Eguchi	Claude King	Daniel Gault	Potato bread

and assigning the topic of the cluster to the named entity type. For instance, the types *Musical Artist* and *Music Recording* would fall, among others, in clusters that are labeled with the topic *music*. This grouping would results into having white lists for the topics of interests (e.g., *music*). Each white list contains the named entities associated with the corresponding topic. Table 1 contains examples of the named entities associated with some topics of interests.

Since we get the canonical form of the named entities from DBpedia, we extend the white lists with the synonyms of the named entities as well. For instance, for *Albert Einstein*, we also add *Einstein, Albert Eienstein, Albert LaFache Einstein, Alber Einstein*, etc. We use WikiSynonyms to retrieve the synonyms of the named entities given the canonical form [38].

Constructing Hashtags White Lists: The construction of the white lists for the hashtags is more involved than that for the named entities. Different hashtags are to be associated with their corresponding topics. This association needs to be learned from the historical tweets as follows.

The procedure starts by looping on each tweet w in the labeled clusters. Each tweet is assigned a confidence score, denoted by $C(w)$, that refers to how confident we are with respect to the learned topic of this tweet. This confidence score is the same as the classification confidence of the binary SVM topic classifier used to label the cluster in the aforementioned topic classification step.

Each hashtag, denoted by h, is assigned a score, that is a measure of its relatedness to each topic p. The score function $score(h|p)$ is defined as

$$score(h|p) = \frac{\sum\limits_{\substack{w \sim p \\ h \in w}} C(w)}{|P| + \sum\limits_{h \in w} C(w)} \tag{1}$$

where P is the set of the adopted topics of interests and $w \sim p$ means that the tweet w is labeled as topic p. The rational behind this scoring function is: the more tweets belonging to a certain topic, tagged with a certain hashtag, the closer relation is between this hashtag and the topic. Also, $|P|$ is added in the denominator to prevent the score to be 1 and to discriminate between the heavily-used and lightly-used hashtags in the cases where they are not used in more than one topic.

Hashtags with score above 0.7 were chosen to be added to our hashtags white lists. Table 2 gives the top 5 related hashtags to each topic obtained using our approach.

Table 2. Top-5 Topic Related Hashtags.

Sports	Music	Politics	Food
#running	#dance	#radio	#organic
#baseball	#jazz	#liberty	#wine
#cycling	#opera	#libertarian	#beer
#lpga	#christian	#environmental	#coffee
#basketball	#rock	#politicalhumor	#chocolate

URL Labeling We found that 56% of the extracted URLs have slugs in their path. A slug is the part of a URL which identifies a page using human-readable keywords (e.g., heavy metal in http:www.example.org/category/heavy-metal). Slug words are extracted from the URL using regex and are added to the tweet text. The expanded tweet is labeled again using the binary SVM classifier.

In addition, 6% of the extracted URLs are for videos from [40]. For such URLs, the *category* field on the video page is used as the label of the tweet.

4.2 Dynamic Interests Capturing

In this subsection, we describe how TRUPI captures the dynamic level of interest of the user in a certain topic.

For a certain user u, the dynamic level of interest is computed for each topic from the recent history of the user interaction on the microblog. This interaction includes her tweets, re-tweets, replies, and favorites. As described before, the tweets of the user u are either unlabeled or are successfully labeled as described in Section 4.1. For each tweet that is labeled when it fell in one of the clusters, it gets assigned the same confidence score of the binary SVM topic classifier. For tweets that are labeled using a white list, their confidence score is treated as 1. For unlabeled tweets, the confidence score is 0.

Without loss of generality, we assume a time-based history. On any day d, a user u is active if he interacts with a tweet. The set of tweets with which she interacted is denoted by W_d. Her level of interest at a topic p is computed as

$$L_{u,d}(p) = \sum_{\substack{w \sim p \\ w \in W_d}} C(w) \tag{2}$$

where $C(w)$ is the confidence score in the topic label of a tweet w and $w \sim p$ means that the tweet w is labeled as topic p.

Upon the arrival of a new tweet w', the user's level of interest in this tweet would be a function of two things: 1) the topic of the tweet, which is labeled as p' with the same method as in Section 4.1, and 2) the dynamic level of interest in the different topics of the tweets earlier this day and in the past week as in [19]. The user's level of interest in the tweet w', which arrived on day d', is computed as

$$I_u(w') = C(w') \sum_{d=d'-7}^{d'} L_{u,d}(p') \tag{3}$$

where $C(w')$ is the confidence score in the topic label of a newly arrived tweet w'.

The user's profile contains the topics of interests to the user. Specifically, for each topic p', it contains the dynamic level of interest of the user in it as

$$\mathrm{DynLOI}_u(p') = \sum_{d=d'-7}^{d'} L_{u,d}(p') \tag{4}$$

The dynamic level of interest is computed over a sliding window. At the beginning of each day, the oldest day in the window is flushed away and incoming tweets enters the window. This sliding window allows for the incremental evaluation of the dynamic level of interest. In other words, the dynamic level of interest does not have to be recomputed with every incoming tweet, which allows for TRUPI to work in a timely manner.

4.3 User Tweets Ranking

A machine-learned ranking model [18] is used to assign a ranking score to the incoming tweets to a certain user. The tweets are posted in a descending order of the assigned ranking scores.

RankSVM is the data-driven support vector machine used to rank the tweets [9]. It learns both the ranking function and the weights of the input features. For a user u_1 receiving a tweet w authored by user u_2, TRUPI uses five ranking feature categories, which are:

- The dynamic level of interest in the topic of w, as computed in Section 4.2.
- The popularity features of w. They include the number of favorites and retweets of w.
- The authoritative features of u_2. They include the number of u_2's followers, followees, and tweets.
- The importance features of w to u_1. They are divided into two groups: 1) globally importance features, which include whether w contains a URL or a hashtag, and 2) locally importance features, which include whether w mentions u_1 or whether it contains a hashtag that was mentioned by u_1 during last week.
- The interaction features. They include the number of times u_1 mentioned, favorited, or replied to a tweet authored by u_2. They also include the similarity between the friendship of u_1 and u_2 (the number of common users both of them follow). They also include the number of days since the last time they interacted together.

5 Experimental Evaluation

We perform extensive experiments to evaluate the quality performance of the proposed recommender. In our experiments, we compare with the state-of-the-art techniques. All used machine-learning algorithms were executed from the WEKA suite [14].

Table 3. 5-fold Cross Validation for Topic Classification

Topic	Precision	Recall	F1
Sports	94.76%	94.68%	94.72%
Music	94.29%	94.29%	94.29%
Politics	98.16%	98.19%	98.17%
Food	98.11%	98.09%	98.10%
Games	97.65%	97.62%	97.64%

We adopted the dataset used in [23], which is publicly available at [35]. This dataset contains 284 million following relationships, 3 million user profiles and 50 million tweets. We sampled this dataset to retrieve 20,000 users, 9.1 million following relationships and 10 million tweets. The sampling consisted of the first 20K users from a breadth first search that started from a random user (id = 25582718). We complemented the dataset by crawling Twitter using the Twitter REST API [33] to retrieve all the actions which have been done on the sampled tweets including Favorite, Re-tweet and Reply-to.

We consider the positive examples during periods which we call Active Periods. An active period is a time frame during which the user have performed an action. Only tweets within this active period are considered in our system in order to avoid the negative examples caused by users being inactive during long periods of time. We scan the tweets in a window of size 10 tweets before and after the re-tweeted, replied to, or favorited tweet.

For the topic classification, we use the micro-averaged F-measure, which considers predictions from all instances [24]. For the ranking module, we use the normalized discounted cumulative gain (nDCG), which measures the performance based on a graded relevance of the recommended entities [24]. We also use the Mean Average Precision (MAP), which has been shown to have especially good discrimination and stability [24].

5.1 Performance of Topic Extraction

In our experiments, we adopted 5 topics of interests, namely, sports, music, politics, food, and games. A binary SVM classifier is created for each topic. The tweets were labeled using the list of keyword-topic pairs obtained from DMOZ - the Open Directory Project [31]. The classifiers were evaluated by measuring the 5-fold cross validation accuracy on our labeled 10M tweets dataset. The results are shown in Table 3.

5.2 Performance of Personalized Binary Recommendation

The personalized binary recommendation model does not rank the tweets. It just tells whether an incoming tweet is important to the user or not if she acts upon it [19]. The ground truth adopted was that a tweet is important to a user if she replied to, re-tweeted, or favorited it. The feature used in TRUPI were the

Table 4. 10-fold Cross Validation for Binary Recommendation Classifiers

Approach	Precision	Recall	F1
DynLDALOI(J48)	74.22%	88.61%	80.78%
TRUPI	85.70%	82.76%	84.20%

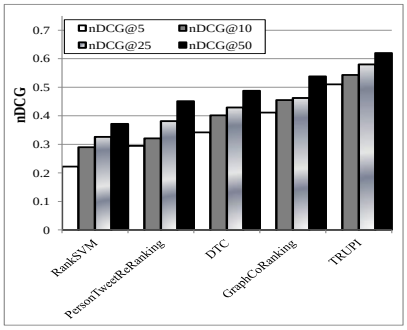

Fig. 1. Personalized Ranking Recommendation using (nDCG)

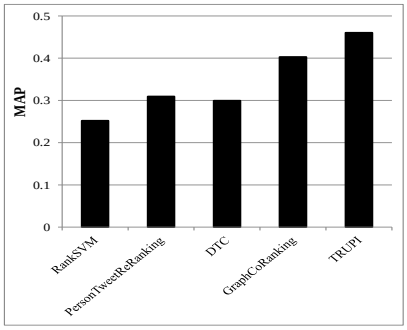

Fig. 2. Personalized Ranking Recommendation (MAP)

same features used in the ranking model in Section 4.3. We compared TRUPI with the state-of-the-art binary recommendation technique, DynLDALOI, which introduced the notion of dynamic level of interest in the topics using LDA [19]. Table 4 shows the 10-fold cross validation of the binary recommendation. The F1 measure of TRUPI outperforms DynLDALOI by 4.23%.

5.3 Performance of Personalized Ranking Recommendation

We evaluated the TRUPI personalized ranking recommendation model through extensive experiments. TRUPI was compared with four state-of-the-art techniques: 1) RankSVM [9], which learns the ranking function and weight of the input features; 2) DTC [36], where a decision tree based classifier is build and a tweet ranking model is constructed to make use of the user's re-tweet behavior; 3) PersonTweetReRanking [10], which consider whether a tweet is important if the user re-tweets it; and 4) GraphCoRanking [39], where the tweets are ranked based on the intuition that there is a mutually reinforcing relation between tweets and their authors. Our ground truth was whether the user was interested, i.e., acted upon a top ranked tweet.

Figures 1 and 2 show the performance of the ranking recommendation techniques using nDCG@5, nDCG@10, nDCG@25, nDCG@50, and MAP. The figures show that TRUPI outperforms all the other competitors. TRUPI outperforms RankSVM by 130%, 87%, 78%, 67%, and 83% on nDCG@5, nDCG@10, nDCG@25, nDCG@50, and MAP respectively. It also outperforms PersonTweetReRanking by 73%, 69%, 52%, 37% and 49% on them respectively. Similarly, it outperforms DTC by 49%, 35%, 35%, 27%, and 54% on the same metrics

Table 5. TRUPI's versions

Version	Features
TRUPI v1	A base version (without all the added features)
TRUPI v2	Adding the hashtag white lists to v1
TRUPI v3	Adding the named entities white lists to v2
TRUPI v4	Adding the URL labeling to v3

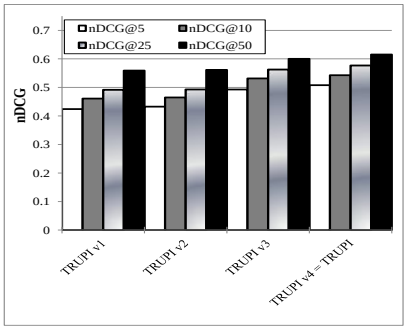

Fig. 3. Effect of TRUPI's Components (nDCG)

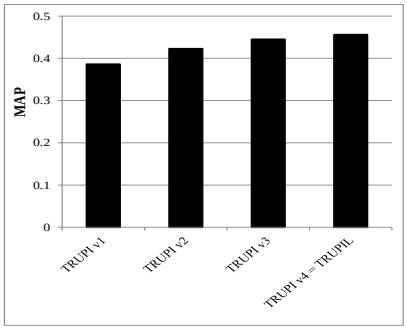

Fig. 4. Effect of TRUPI's Components (MAP)

respecitively. Finally, TRUPI outperforms GraphCoRanking by 24%, 19%, 25%, 15%, and 14% on them respectively.

5.4 Analyzing TRUPI's Components

We scrutinized the different components of TRUPI. We illustrate the effect of the different components by looking at 4 versions of TRUPI. The versions are shown in Table 5. Note that TRUPI v4 is the full fledge proposed TRUPI.

Figures 3 and 4 show the performance of the different versions of TRUPI using nDCG@5, nDCG@10, nDCG@25, nDCG@50, and MAP. Adding the hashtags white lists gave a relative gain for nDCG@10 by 2% on the base case. The named entities white lists gave 14% on the same metric, whereas the URL labeling added 3%. On MAP, they added 10%, 5%, 2% respectively.

6 Acknowledgement

This material is based on work supported in part by Research Sponsorship from Microsoft Research.

7 Conclusion

In this paper, we proposed TRUPI, a personalized recommendation system based on user's personal interests. The proposed system combines the user social features and interactions as well as the history of her tweets content to attain her

interests. It also captures the dynamic level of users' interests in different topics to accommodate the change of interests over time. We thoroughly evaluated the performance of TRUPI on a publicly available dataset and have found that TRUPI outperforms the competitive state-of-the-art Twitter recommendation systems by 25% on nDCG@25, and 14% on MAP.

References

1. http://www.noslang.com/. Internet Slang Dictionary & Translator (Last accessed 2014/06/01).
2. Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing User Modeling on Twitter for Personalized News Recommendations. In *UMAP'11*, 2011.
3. Alan Ritter and Sam Clark. https://github.com/aritter/twitter_nlp, 2011. Twitter NLP Tools (Last accessed 2014/06/01).
4. Hila Becker, Mor Naaman, and Luis Gravano. Beyond Trending Topics: Real-World Event Identification on Twitter. In *Procs. ICWSM'2011*, 2011.
5. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learnnig Research*, 3:993–1022, 2003.
6. Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed Chi. Short and Tweet: Experiments on Recommending Content from Information Streams. In *CHI*, 2010.
7. Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao, and Yong Yu. Collaborative Personalized Tweet Recommendation. In *Procs. of SIGIR'12*, 2012.
8. Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.
9. Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. An empirical study on learning to rank of tweets. In *COLING'10*, 2010.
10. Wei Feng and Jianyong Wang. Retweet or Not?: Personalized Tweet Re-ranking. In *Procs. of WSDM'13*, pages 577–586, 2013.
11. GNU Aspell. http://aspell.net/, 2011. (Last accessed 2014/06/01).
12. Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using Topic Models for Twitter Hashtag Recommendation. In *Procs. of WWW'13 Companion*, 2013.
13. Yuhong Guo, Li Kang, and Tie Shi. Personalized Tweet Ranking Based on AHP: A Case Study of Micro-blogging Message Ranking in T.Sina. In *WI-IAT'12*, 2012.
14. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.
15. John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *RecSys'10*, 2010.
16. John Hannon, Kevin McCarthy, and Barry Smyth. Finding Useful Users on Twitter: Twittomender the Followee Recommender. In *ECIR'11*, pages 784–787, 2011.
17. Huffington Post's Twitter Statistics. http://www.huffingtonpost.com/belle-beth-cooper/10-surprising-new-twitter_b_4387476.html. (Last accessed 2014/06/01).
18. Thorsten Joachims. Optimizing Search Engines Using Clickthrough Data. In *Procs. of KDD'02*, pages 133–142, 2002.
19. Shaymaa Khater, Hicham G. Elmongui, and Denis Gracanin. Tweets You Like: Personalized Tweets Recommendation based on Dynamic Users Interests. In *SocialInformatics'14*, 2014.

20. Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is Twitter, a Social Network or a News Media? In *Procs. of WWW'10*, pages 591–600, 2010.
21. Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On Recommending Hashtags in Twitter Networks. In *Procs. of SocInfo'12*, pages 337–350, 2012.
22. Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 2014.
23. Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. Towards Social User Profiling: Unified and Discriminative Influence Model for Inferring Home Locations. In *Procs. of KDD'12*, pages 1023–1031, 2012.
24. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
25. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In *ICLR'2013 Workshops*, 2013.
26. Gianmarco De Francisci Morales, Aristides Gionis, and Claudio Lucchese. From Chatter to Headlines: Harnessing the Real-time Web for Personalized News Recommendation. In *Procs. of WSDM'12*, pages 153–162, 2012.
27. Marco Pennacchiotti, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Making Your Interests Follow You on Twitter. In *Procs. of CIKM'12*, 2012.
28. Robert Half Technology. http://rht.mediaroom.com/index.php?s=131&item=790 , 2009. Whistle - But Don't tweet - While You Work (Last accessed 2014/06/01).
29. Gerard Salton and Christopher Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
30. Igor Santos, Igor Miñambres-Marcos, Carlos Laorden, Patxi Galán-García, Aitor Santamaría-Ibirika, and Pablo Garcia Bringas. Twitter Content-Based Spam Filtering. In *Procs. of CISIS'13*, pages 449–458, 2013.
31. The Open Directory Project. http://www.dmoz.org/. (Last accessed 2014/06/01).
32. Twitter. http://www.twitter.com/, 2006. (Last accessed 2014/06/01).
33. Twitter REST API. https://dev.twitter.com/docs. (Last accessed 2014/06/01).
34. Twitter Usage. http://about.twitter.com/company. (Last accessed 2014/06/01).
35. UDI-TwitterCrawl-Aug2012. https://wiki.cites.illinois.edu/wiki/display/forward/Dataset-UDI-TwitterCrawl-Aug2012, 2012. (Last accessed 2014/06/01).
36. Ibrahim Uysal and W. Bruce Croft. User oriented tweet ranking: a filtering approach to microblogs. In *Procs. of CIKM'11*, pages 2261–2264, 2011.
37. Wikipedia. http://www.wikipedia.org/, 2001. (Last accessed 2014/06/01).
38. WikiSynonyms. http://wikisynonyms.ipeirotis.com/. (Last accessed 2014/06/01).
39. Rui Yan, Mirella Lapata, and Xiaoming Li. Tweet Recommendation with Graph Co-ranking. In *Procs. of ACL'12*, pages 516–525, 2012.
40. YouTube. http://www.youtube.com/, 2005. (Last accessed 2014/06/01).
41. Eva Zangerle, Wolfgang Gassler, and Günther Specht. Recommending#-Tags in Twitter. In *Procs. of SASWeb 2011*, pages 67–78, 2011.