LINK RECOMMENDER: Collaborative Filtering For Recommending URLs to Twitter Users

by

Nazpar Yazdanfar
B.Sc., University of Alzahra, 2010

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTERS OF SCIENCE

in the Department of Computer Science

LINK RECOMMENDER: Collaborative Filtering For Recommending URLs to Twitter Users

by

Nazpar Yazdanfar

B.Sc., University of Alzahra, 2010

Supervisory Committee

Dr. A. Thomo, Supervisor

(Department of Computer Science)

Dr. K. Wu, Departmental Member

(Department of Computer Science)

Supervisory Committee

Dr. A. Thomo, Supervisor

(Department of Computer Science)

Dr. K. Wu, Departmental Member

(Department of Computer Science)

ABSTRACT

Twitter, the popular micro-blogging service, has gained a rapid growth in recent years. Newest information is accessible in this social web service through a large volume of real-time tweets. Tweets are short and they are more informative when they are coupled with URLs, which are addresses of interesting web pages related to the tweets. Due to tweet overload in Twitter, an accurate URL recommender system is a beneficial tool for information seekers. In this thesis, we focus on a neighborhood-based recommender system that recommends URLs to Twitter users. We consider one of the major elements of tweets, *hashtags*, as the topic representatives of URLs in our approach. We propose methods for incorporating hashtags in measuring the relevancy of URLs. Our experiments show that our neighborhood-based recommender system outperforms a matrix factorization-based system significantly. We also show that the accuracy of URL recommendation in Twitter is time-dependent. A higher recommendation accuracy is obtained when more recent data is provided for recommendation.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. Alex Thomo, for mentoring, support and encouragement.

DEDICATION

I would like to dedicate my thesis to my family and friends. A special feeling of gratitude to my loving parents Vahideh and Javad Yazdanfar, and my brother Nima, whose support, encouragement, and constant love have sustained me throughout my life and without whom none of these would have happened. I also dedicate this dissertation to my best friends, especially Amir Hossein Hajizadeh who has supported me throughout the process. I will always appreciate all he has done.

Chapter 1

Introduction

Micro-blogging web sites such as Twitter offer an opportunity to investigate large-scale social systems where preferences of users are traceable from their activities. Despite Twitter's simplicity, it provides many advantages for its users in the form of social short messaging. Twitter allows users to constantly update their public time lines and follow the post history of their favorite topics and people. Due to the popularity and worldwide usage of Twitter, a growing body of research critically analyzes it from different aspects. However, from the perspective of recommender systems, there is not much work done to use Twitter data as a target for applying diverse recommendation techniques and be able to recommend various items, such as tweets, URLs, followees, or hashtags to twitterers.

In this thesis, we focus on the recommendation of URLs (web addresses mentioned in tweets) using collaborative filtering approaches. To this end, we propose a neighborhood-based (NB) approach that utilizes user-URL-hashtag connections. The advantage of an NB approach is that it provides immediate recommendations for users with newly entered records. This ability becomes important in our Twitter recommendation system because users tend to retweet URLs constantly. Intuitively,

a neighborhood approach produces recommendations based on similarities between pairs of items or pairs of users. Two items are considered to be "similar" if they have been rated (or promoted/adopted) by many users in common. Analogously, two users are considered to be "similar" if they have rated (or promoted/adopted) by many items in common. In an item-based approach, the preference prediction (of a user for an unknown item) is based on the ratings of similar items by the same user, while in a user-based approach, the preference prediction is based on the ratings of similar users for the same item.

Recommending URLs that match Twitter users' is of great importance. In recent years people benefit from social networks such as Facebook and Twitter to share their interesting URLs with their friends and followers. URL sharing gained popularity in Twitter since tweets are brief and sometimes noisy. As a result, twitterers add URLs to their tweets to expand them. Therefore, very often, published messages in Twitter contain URLs. Notifying users of URLs fitting to their interest in the "sea" of tweets assists them find recent updates more quickly.

However, discovering twitterers' favorite URLs based on collaborative filtering is a challenging problem. Every minute, hundreds of URLs are published in Twitter referring to various topics, events, and news. Twitterers' tastes in topics might vary constantly. Due to such data sparsity, naïve collaborative filtering techniques show poor performance when the similarities between pairs of items or users are small.

In order to alleviate data sparsity, a proposed solution is to organize the user preferences for items in a user-item matrix, and then reduce rank of the this matrix using matrix factorization methods such as Singular Value Decomposition (SVD) [17]. Unlike neighborhood approaches, matrix factorization methods transform both user and item vectors to a latent factor space, where similarities between items and users are generated by lower-dimensional hidden factors automatically inferred from data.

Matrix factorization methods have attracted significant attention in recent years due to their success in the Netflix Prize competition. However, applying SVD on incomplete matrices, as in the case of collaborative filtering, might produce results that are not easy to characterize. On the other hand, filling missing ratings is expensive and considerably increases the complexity. In the Twitter URL recommendation case we focus on, we show that matrix factorization methods used in the Netflix competition did not produce results of high quality.

Therefore, we turn our attention to devising a neighborhood-based approach using a three dimensional matrix connecting users, URLs, and hashtags found in the tweets. Thus, instead of using raw text-based approaches, we consider hashtags, which are words marked as important by the authors of the tweets. The hashtags of a tweet can be regarded as either topics of the URLs or summary words of the tweet. For instance, a user might post a tweet saying "I enjoy following #election news + URL". Most likely, "election" is the descriptor of the URL she is posting.

In order to utilize hashtags as approximate topic indicators of a URL, we propose an approach based on collaborative filtering to discover the user's favorite URLs. The assumption here is that different URLs presented with identical hashtags are about the same topics. Therefore, besides measuring the similarity of URLs based on the users who published them, we also exploit hashtags to boost the similarity determination of URLs. The predicted preference score of each user to each unseen item (URL) is computed by both the similarity between pairs of users and pairs of hashtags. The two scores extracted for each item are then combined to form an individual score for each user for her unobserved items. This predicted score determines whether an item should be recommended to a user or not.

Given our method, we conduct an in-depth analysis on our large tweet datasets that we collected from Twitter. Mainly, we are interested to answer three research

questions that concern the quality of our URL recommender system:

1. How does the accuracy of the recommender system vary by injecting another layer of similarity between URLs? and to what extent does the accuracy change with different similarity metrics?

2. How does the dynamics of information in Twitter influence the temporal recommendations?

3. Would refining the datasets based on specific keywords have any benefit for our recommender system?

To answer the above research questions, we implemented a CF recommender system that leverages our proposed methods to generate URL recommendations, and examined it in different cases.

More specifically, the main contributions of this thesis are:

1. We propose a representation of the extracted data as a three-dimensional matrix of user-item-hashtag to solve the problem of data sparsity when measuring the similarity between items.

2. We use weighted mean and maximum functions to combine the similarities of items over user and hashtag dimensions to predict the preference score of users to each item. We show that our methods outperform the SVD method that is known as a successful method for CF recommendation.

3. We also perform a more realistic evaluation by taking the timestamp of the postings of URLs into consideration. Using the timestamps, we only predict URLs using past URL postings, not future ones. This is a special feature not found typically in the evaluation of other recommender systems. Our experiments show that more data does not mean better results: considering URLs

posted more than seven days in the past for recommending URLs on the present day has a clear trend of higher error rate.

The thesis is organized as follows. We start from existing work in collaborative filtering recommender systems in Chapter 2. Then, we describe data collection and preparation in Chapter 3. We define our ternary relationship between users, URLs, and hashtags in Chapter 4. Chapter 5 is devoted to present our methods and we discuss how we can incorporate hashtags in measuring item similarities. In Chapter 6, we compare a matrix factorization method with our new methods. Moreover, we show how the timestamps of the postings of the URLs affect the quality of our recommendations. Results are presented in Chapter 6 and discussed in Chapter 7. We conclude our findings in Chapter 8.

A preliminary version of this thesis is published as a paper in [28].

Chapter 2

Related work

2.1 Collaborative filtering recommender systems

Collaborative filtering is regarded as a promising recommendation strategy which has attracted a great body of research from academia and industry. Google news [11] and Amazon [20] have applied this conventional recommendation approach in their systems. In recent years, there were many attempts for optimizing the performance of collaborative filtering recommendation systems. Wang et al. [26] showed that combining user-based and item-based collaborative filtering methods produces accurate recommendations. Bogers et al. [6] used the content-based filtering to improve the precision of recommendation for social bookmarking websites.

2.2 Social network recommendation systems

With the growing attention toward Twitter as a popular social network, many researchers have applied recommendation techniques in Twitter. Broadly, these techniques can be divided into two categories of content-based filtering approaches and collaborative filtering approaches. In the former technique, a so-called profile is gener-

ated for each Twitter user which represents her interests or activities. A content-based recommender system then consumes each user profile to match some items to the profile and recommends those items to the users. A great body of research ([12], [23], [24], [25]) uses this technique to recommend various items in Twitter.

In the collaborative filtering approach, on the other hand, the objective is to recommend a user the items which are evaluated by the other users who can be considered as similar the user. Collaborative filtering is put to work to recommend various items in Twitter. Chen et al. [9] focused on recommending useful tweets by adding some factors including tweet topic level, social relation and authority of the tweet's publisher to enhance Twitter-based recommendations. Benefiting from these factors, they reported a significant precision of 0.8 for their method.

Twittomender was introduced in [13] to help Twitter users find useful followees, who produce relevant content, and prevent information overload. This work also employs proven collaborative filtering approach.

Another recommendation system for Twitter, called TWITOBI was introduced in [14]. TWITOBI uses probabilistic modeling for collaborative filtering to recommend both tweets and followees to users. This work utilizes not only tweet messages but also the relationships between users to produce better recommendations.

Another interesting line of work is [10] which proposes a system to recommend emergency news to Twitterers by analyzing patterns of information diffusion via tweets.

Regarding to the effectiveness of collaborative filtering, we chose a neighborhood-based collaborative filtering method as our target recommendation method. Inspired by [6] that constitutes an extra layer of connections between user and items, we also connect users and URLs via an extra level. [6] exploits that layer to recommend bookmarks to users in social bookmarking websites. The authors of this work also

employ a content-based approach for their study. In this thesis, we investigate the effect of adding hashtags as the summary or the topic indicator of the URLs without employing content-based filtering in Twitter. Yet, like them, we leverage hashtags that Twitterers post in their tweets as metadata to boost the quality of our recommendations. Our work differs from their work in a sense that when we try to find the similarity between the items, we utilize both user and hashtag layers.

Chapter 3

Data

This chapter describes the data acquisition from Twitter. The collected data is used to evaluate the performance of our URL recommender system. Since, our goal is to recommend Twitter users the URLs that might be interesting to them, we are interested to find all the tweets that contain at least one URL. To this end, we leverage Twitter APIs that make some the tweets publicly available at no cost. These APIs can be put into two main classes based on their design and access method. The first class is REST APIs that are based on the REST architecture. The pull strategy for data retrieval is used in this type of APIs and a user needs to explicitly request it to be able to collect data.

The second class is streaming APIs which provide a continuous stream of tweets with no further input from the user. Unlike REST APIs, streaming APIs employ a push strategy for data retrieval [18]. Here, we use streaming APIs for our data collection phase. Depending on the type of information being retrieved, streaming APIs are divided into three categories of public, user (single-user) and site streams (multi-user). As our purpose is to collect data to implement a URL recommendation system for a great number of users with various interests, we found the public

streaming API to be the most useful one.

Despite the adaptability of streaming APIs, there are some constraints and limitations for using them. Since Twitter uses Open Authentication, each request for data retrieval must be signed with valid Twitter user credentials [1]. Also, only 1% of all public tweets flowing through Twitter [2] will be returned to the user. Another limitation is that there is no filter to request only tweets with URL.

To be able to use the Twitter streaming API, an application was developed to keep a persistent HTTP connection open and collect the tweets and store them into big files. The application was run from May 1 to May 22, 2012. In this period, we obtained accessible public tweets comprising of about 8 million tweets submitted by 4 million users. On average, 362,717 tweets were collected daily. Tweets were distributed almost the same on each day of data collection. Each tweet, also known as "status", includes: (1) a tweet identifier, (2) a user identifier, (3) date and time of creation, (4) textual content, (5) URLs, and (6) hashtags.

As the streaming sample API delivers a feed of tweets without any constraints on the language of the tweet, our tweet corpus contains tweets published in different languages and countries. Some of them were only pure text, meaning that they had no URL or hashtag. Since we aim at recommending URLs to users, those tweets were not useful for us. We only focused on the tweets that contain at least one URL in their text. Next, we narrowed down our filtering to *"active users"* and *"active URLs"* as defined in the following.

3.1 User and URL Selection

We observed that there were users who had only one tweet in our collection, or users who had posted the same tweet many times. In order to reduce the data sparsity

Table 3.1: General statistics about our tweet collection

Number of tweets	7,979,777
Number of users	4,285,186
Number of URLs	6,666,457
Number of hashtags	5,998
Number of active users	22,543
Number of active URLs	326

and remove spam, we constructed a profile for each user. If a user has more tweets with URLs than a specific threshold, then it is labeled as an *"active user"*. Also, if a URL is shared by more active users than a specific threshold, it is called an *"active URL"*. Here, we choose 20 and 10 as thresholds for active users and active URLs, respectively. Restricting our data to active users and active URLs, we finally obtained 22,543 unique users and 326 unique URLs.

3.2 Hashtag Selection

Twitter allows users to emphasize on the keywords of their tweets by creating hashtags. Twitterers use the hashtag symbol # before a relevant keyword or a phrase to categorize tweets. Intuitively, hashtags can be assumed as topic indicators of both tweets and URLs. We extracted the hashtags of all tweets regardless of the tweet language.

Chapter 4

Data representation

Unlike other works (cf. [22, 5, 21]) that model a topic for each item from the textual content of posts, we directly benefit from hashtags by considering them as the topics of items (URL). Our major goal here is to improve the quality of collaborative filtering recommendations by creating another layer of correlation between pairs of items that employs hashtags. In this way, each URL not only has a vector of users who have tweeted it, but also has another vector of hashtags that are assigned to it.

Assuming a ternary relationship between items, users, and hashtags similarly as in T. Boger and A. van den Bosch [6], we define our relationship as a 3D array

$$\mathbf{R}(i_k, u_l, h_m),$$

where:

- an item (URL) is referred as i_k, where $k \in [1, K]$, and K is the number of items,
- a user is referred as u_l, where $l \in [1, L]$ and L is the number of users,
- a hashtag is referred as h_m, where $m \in [1, M]$, and M is the total number of distinct hashtags.

Typically, in item-based collaborative filtering recommender systems, there is an item-user matrix that is used to keep the records about how users and items are connected using users' ratings for items. However, in the context of Twitter, the ratings should be inferred from users' behavior as Twitter does not support explicit ratings. Therefore, we form our $K \times L \times M$ item-user-hashtag array with binary values. If URL i_k is posted by user u_l who has also defined hashtag h_m, we fill cell $\mathbf{R}[i_k, u_l, h_m]$ with 1, otherwise 0.

From the initial 3D array, we create the following two 2D matrices. The first is the $K \times L$ item-user matrix IU, where each item i_k is represented as a row vector of users who submitted the item, and each user u_l is represented as a column vector of items the user posted. The second matrix is the $K \times M$ item-hashtag matrix IH that we obtain by aggregating R over users as follows:

$$IH(i_k, h_m) \quad = \quad \sum_{l=1}^{L} R(i_k, u_l, h_m)$$

This matrix incorporates hashtags in the collaborative filtering approach describing the connection between items and hashtags. In this matrix, each item i_k is represented as a row vector of hashtags assigned to the tweets containing the item, and each hashtag is represented as a column vector of items posted in tweets having the hashtag. The cells of matrix *IH* contain the number of times a certain hashtag is assigned via a tweet to an item. In our method, each item i_k is described by two vectors: a vector of users who have posted i_k in their tweets, and a vector of hashtags that were assigned to the tweets containing i_k.

The main problem we focus is to predict whether a user u_l will like an item i_k. For this, our method computes a score $\hat{r}_{k,l} \in [0, 1]$, and based on this score we produce the recommendations.

Chapter 5

Our method

Unlike [6] that uses only an item-tag matrix for calculating similarities between items, we improve item-item similarity computations by benefiting from both item-hashtag and item-user matrices. For instance, assume we have two item vectors $\vec{I_k} = \langle u_1, u_2, u_3 \rangle$ and $\vec{I_j} = \langle u_4, u_5, u_6 \rangle$ in matrix IU, with corresponding hashtag vectors $\vec{I_k} = \langle h_1, h_2, h_3 \rangle$ and $\vec{I_j} = \langle h_2, h_3, h_4 \rangle$ in matrix IH, respectively. Considering IU alone, as in the conventional collaborative filtering, we get 0 as the correlation of $\vec{I_k}$ and $\vec{I_j}$, since they have no user overlap. On the other hand, following the approach of [6] that measures item similarities based on only an item-hashtag matrix we suffer information loss for the items that have overlaps in both user and hashtag dimensions. Consequently, the relevance score $\hat{r}_{k,l}$ of such items should be boosted when measuring item-item similarity based on hashtag and user correlations simultaneously.

In order to calculate the correlation between two items, i_k and i_j, a similarity measure is required. We use four similarity measures:

- *Euclidean*
- *Cosine similarity*
- *Jaccard similarity*

- *Dice Coefficient*

Formally, the similarities between two items i_k to i_j for the above measures are as follows:

$$\begin{aligned} sim_{Eucliean}(i_k, i_j) &= -||i_k - i_j|| \\ sim_{Cosine}(i_k, i_j) &= -\frac{i_k \cdot i_j}{||i_k|| \cdot ||i_j||} \\ sim_{Jaccard}(i_k, i_j) &= \frac{|i_k \cap i_j|}{|i_k \cup i_j|} \\ sim_{Dice}(i_k, i_j) &= \frac{2 \cdot |i_k \cap i_j|}{|i_k| + |i_j|} \end{aligned}$$

where $||.||$ denotes the length of a vector (square root of the sum of squares of the components), whereas $|.|$ denotes the cardinality of a vector considered as a set of elements.

Since obtaining item-item similarity scores by focusing only on the hashtag dimension or only on the user dimension causes loss of information, once the similarities between the items in item-user and item-tag matrices are measured via a similarity function, we need a technique to combine the two scores so that the similarity between items i_k and i_j is generated as a single score. To this end, we use two methods: (a) weighted mean, and (b) maximum, which are described in the following. Formally, the weighted mean method is defined as:

$$WMS(i_k, i_j) = \frac{\alpha \cdot sim_u(i_k, i_j) + \beta \cdot sim_h(i_k, i_j)}{(\alpha + \beta)} \tag{5.1}$$

where $sim_u(i_k, i_j)$ is the similarity between two items in item-user matrix IU, $sim_h(i_k, i_j)$ is the similarity between two items in item-hashtag matrix IH, and α and β are the weights we give to the importance of each one of these two similarities that are determined via testing.

We also use the maximum function $MAXS(i_k, i_j)$ that outputs the maximum value of $sim_u(i_k, i_j)$ and $sim_h(i_k, i_j)$ as:

$$MAXS(i_k, i_j) = max\{sim_u(i_k, i_j), sim_h(i_k, i_j)\} \tag{5.2}$$

After combining the similarity scores generated from two matrices of item-user and item-hashtag, we find the prediction scores $\hat{r}_{k,l}$ (to recommend item i_k to user u_l) following an item-item collaborative filtering approach.

Specifically, the top N similar items (to i_k) that user u_l has posted are ranked in a descending order and inserted into a list, called $L_{k,l}$. For predicting score $\hat{r}_{k,l}$, one way is to consider the *mean* or the *maximum* similarity to i_k of the items in list $L_{k,l}$. If the calculated score $\hat{r}_{k,l}$ passes a threshold, we recommend i_k to u_l. In summary, our algorithm for predicting the score $\hat{r}_{k,l}$ of i_k for user u_l is described in three major steps:

- Compute the list of the top N similar items to i_k which are posted by u_l, called $L_{k,l}$. The item-item similarities are computed using the *WMS* or *MAXS* described earlier.
- Then, compute the mean or the maximum similarity to i_k of the items on $L_{k,l}$. This is score $\hat{r}_{k,l}$.
- If $\hat{r}_{k,l}$ is greater than a threshold value, we recommend item i_k to u_l.

Chapter 6

Evaluation

In this chapter, we consider three experiments on our URL recommender system to find the answers for our three research questions. In the first experiment, we examine the impact of using hashtags as metadata to improve the performance of the URL recommender system. To show how the accuracy of collaborative filtering changes, we consider hashtag similarities as well as user similarities when computing the correlations between pairs of items (URLs).

In the second experiment, we investigate whether our recommender is time-sensitive. As Twitterers' topics of interest constantly change over time, we expect that recommendation quality varies by time. To check this, we run a second set of experiments to observe temporal variability in recommendation accuracy.

In the last experiment, we try to explore the effect of keyword-based dataset enrichment on our collaborative filtering recommendation system. Due to the data sparsity in data collection phase, we speculate that data filtering based on specific keyword filters will result in better recommendation performance.

For measuring the performance, we follow the hide-one out approach in which we hide one URL posting and then try to predict it using our method.

In addition to these experiments, other experiments are also conducted for investigating the effect of potential parameters such as α, β, top N neighbors and recommendation threshold in recommendation performance. By knowing which factors play key roles in the URL recommender performance, we can better improve the performance of our URL recommender.

6.1 Evaluation metrics

The first step towards selecting an appropriate method is to decide which properties of the collaborative recommender system to focus on. The performance of a recommendation system revolves around a variety of properties such as accuracy, scalability, and so forth. In all of our experiments we targeted the accuracy of our URL recommendation system. Therefore, we elected three metrics for quantifying the accuracy as following:

- *Root Mean Squared Error (RMSE)*

$$RMSE = \sqrt{\frac{\sum_{(k,l)\in TestURLs}(r_{k,l} - \hat{r}_{k,l})^2}{|TestURLs|}} \tag{6.1}$$

  where $r_{k,l}$ is 1 if user u_l has posted item (URL) i_k and $\hat{r}_{k,l}$ is the predicted score for item i_k by user u_l, and $|Testset|$ is the number of test URL.

- *Recall*

  Recall is defined as the proportion of test URLs that are recommended.

$$Recall = \frac{|\hat{r}_{k,l} > prediction\ threshold|}{|TestURLs\ |} \tag{6.2}$$

- *Mean Average Precision at three (MAP@3)*

MAP@3 is the proportion of top three posted and recommended URLs to the recommended ones for each user. Therefore, for each user u_l by considering her top three items, MAP@3 is computed as:

$$MAP@3_l = \frac{|(k,l) : r_{k,l} = 1 \; and \; k \in \; top \; three \; recommended \; items \; to \; user \; l|}{3} \tag{6.3}$$

and the overall MAP@3 for a method is defined as:

$$MAP@\mathcal{3} = \frac{1}{|L|} \sum_{l=1}^{L} MAP@3_l \tag{6.4}$$

6.2 Results of our method

In order to quantify how a recommendation system based on our method performs, we started by developing a neighborhood-based URL recommender that builds an item-item similarity matrix based on both user and hashtag dimensions. As explained in Chapter 5, four similarity functions, Euclidean, Cosine similarity, Jaccard and Dice, have been exploited to find the similarity score between each two pairs of items. Next, two methods of *WMS* and *MAXS* were consumed to integrate hashtag similarities with user similarities. For this preliminary study both α and β were equally set to 0.5.

After building up the $K \times K$ item-item similarity matrix, we filtered the matrix by only considering the top N most similar neighbors to each item. Since our main focus is to show the effectiveness of our method, in the early phases we define some of our algorithm parameters as constants. We chose 10 for top N, 20 for active user threshold and 10 for active URL threshold, respectively. Also, an unbiased threshold of 0.5 was considered as prediction threshold.

In addition to four similarity functions and two functions for user and hashtag similarity integration, we also have two approaches to decide whether an item should

Table 6.1: Results of Recall

	MAXS		WMS	
	Maximum	Average	Maximum	Average
Euclidean	0.7929	0.7907	0.7921	0.7259
Cosine	0.6127	0.5536	0.3664	0.3577
Jaccard	0.5089	0.4766	0.3611	0.3505
Dice	0.6023	0.5432	0.3664	0.3562

Table 6.2: Results of RMSE

	MAXS		WMS	
	Maximum	Average	Maximum	Average
Euclidean	0.0791	0.1051	0.3155	0.3324
Cosine	0.3144	0.3473	0.4617	0.4874
Jaccard	0.3982	0.4313	0.5149	0.5418
Dice	0.3249	0.3575	0.4666	0.4924

be recommended to a user based on its top N most similar neighbors. As the result, sixteen different cases have been tested using our designed neighborhood-based recommendation system.

The initial results are summarized in tables 6.1, 6.2 and 6.3. Considering recall as the recommender accuracy metric, we can observe a significant recall of 79% when we used Euclidean similarity with MAXS. The best recall achieved for MAXS with maximum function. However, the recall for WMS method is not remarkable. Table 6.2 demonstrates the same pattern for RMSE. The least error of 0.0791 was obtained for Euclidean in MAXS method with maximum function. On the other hand, based on the precision results, we can notice a good accuracy when WMS is used for incorporating hashtags and users similarities with a MAP@3 of 0.5802 as shown in table 6.3.

We compared our preliminary results of MAP@3 with the work in [6] in which a content-based folksonomic recommender was proposed. Bogers and Bosch implemented a baseline collaborative filtering recommender where they used only the item-tag matrix to calculate the similarities between items. According to our results,

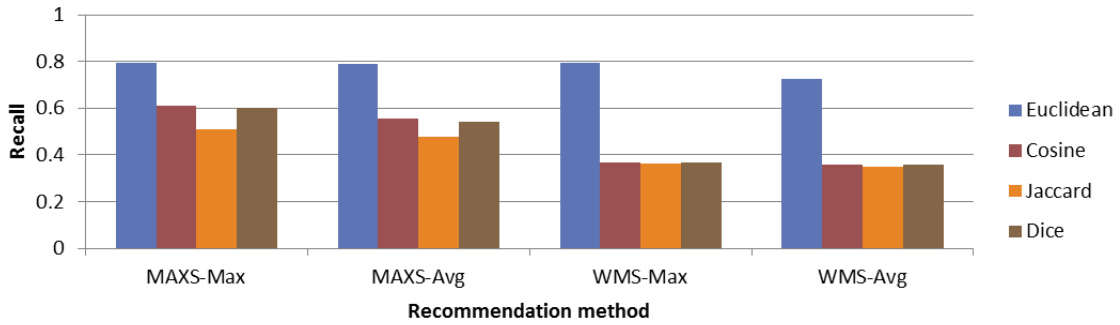


Figure 6.1: Recall of our method

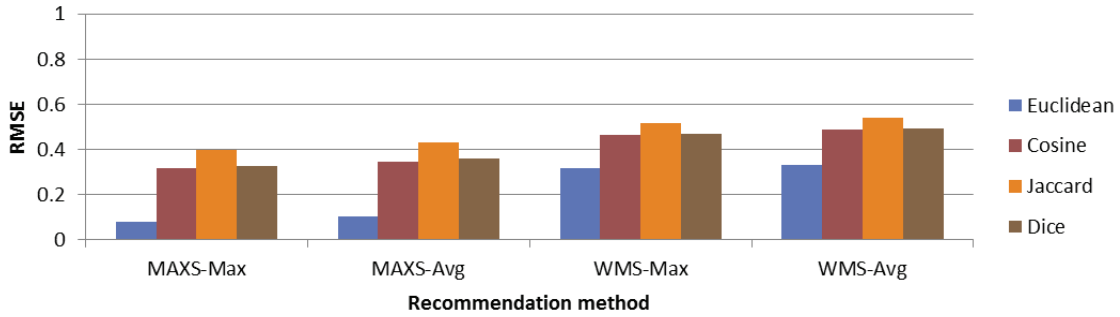


Figure 6.2: RMSE of our method

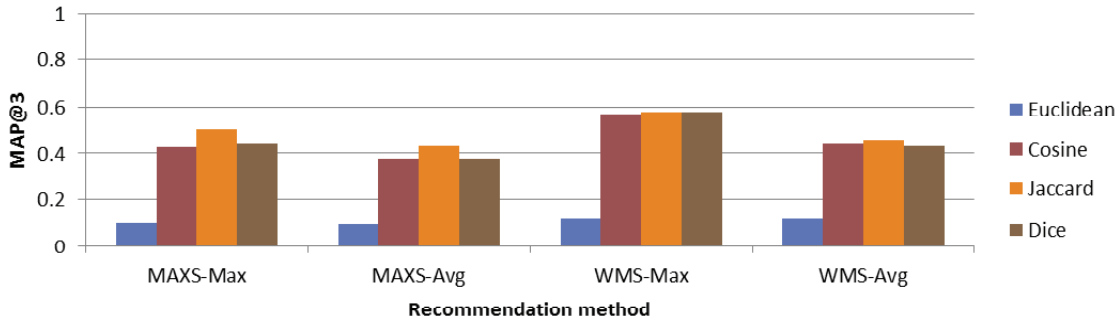


Figure 6.3: MAP@3 of our method

Table 6.3: Results of MAP@3

	MAXS		WMS	
	Maximum	Average	Maximum	Average
Euclidean	0.1010	0.0967	0.1213	0.1210
Cosine	0.4293	0.3750	0.5702	0.4393
Jaccard	0.4984	0.4315	0.5770	0.4536
Dice	0.4400	0.3765	0.5802	0.4304

fusing item-tag and item-user matrices significantly improve the precision of recommendation comparing to Bogers and Bosch's work for both collaborative filtering and content-based recommender systems. A recommendation system with high precision also was reported in [8]. But, their evaluation is not comparable with ours since they did not consider collaborative filtering in their design.

The performance with each similarity functions are also compared against each other in figures 6.1, 6.2, and 6.3.

6.3 Comparison with the Matrix Factorization method

The effect of incorporating hashtags as metadata can be shown when we compare our methods against the best-in-class collaborative filtering approach based on matrix factorization from the Netflix competition [16]. Hence, to verify the effect of employment of both hashtag and user similarities in computing the prediction scores, a second recommender system was implemented based on a matrix factorization method.

6.3.1 Matrix factorization-based recommendation

One of the collaborative filtering approaches is matrix factorization which models the connection between users and items in a latent factors space where factors are derived automatically from data. In latent factor space of dimensionality f, each item i_k in item-user matrix is represented as a vector of $q_i \in R^f$ and each user u_l is represented

as a vector of $p_u \in R^f$, respectively. The preference prediction scores are generated by inner products of two vectors q_i and p_u as:

$$r_{u,i} = \mu + b_u + b_i + {q_i}^T p_u. \tag{6.5}$$

where μ is the mean of all ratings, b_u is user bias and b_i is item bias. In our case, since we are working with binary ratings, we set b_u and b_i to 0.

6.3.2 Results of comparison with matrix factorization

Our matrix-factorization based recommender employs the SVD algorithm implemented in LingPipe tool kit [3] to generate recommendations. LingPipe tool kit is mainly used for processing text using computational linguistics. Aside from text analysis applications, the SVD algorithm that is included in this tool kit can be used for collaborative filtering as well [4].

We used the results of this method as the basis of comparison to our collaborative filtering method. In this experiment, we worked with the same data as in our target recommender with active user threshold of 20 and URL threshold of 10. We set the maximum number of factors in SVD algorithm to 2 and the initial value for singular vector dimensions to 0.1. Also, we chose 50 and 5000 for the minimum and maximum number of training per factor, respectively.

The output of matrix factorization-based recommender shows a huge RMSE of 0.859, a low recall of 0.153 and a poor MAP@3 of 0.008. We compared these results with the best results of our method in figures 6.4, 6.5 and 6.6. In our method, we obtained the least RMSE and the best recall when we used Euclidean as the similarity function (figure 6.5 and 6.4), and the most significant MAP@3 achieved with Dice Coefficient function (figure 6.6).

Figure 6.4: Recall of our method vs. matrix factorization method.

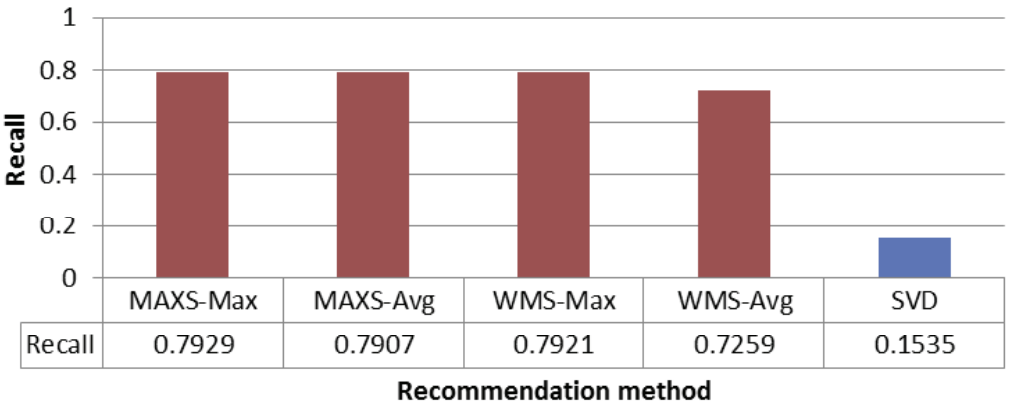


Figure 6.5: RMSE of our method vs. matrix factorization method.

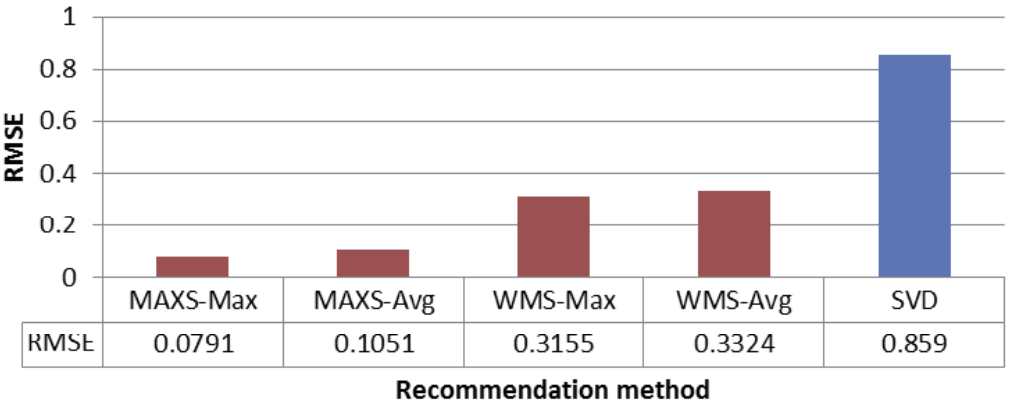
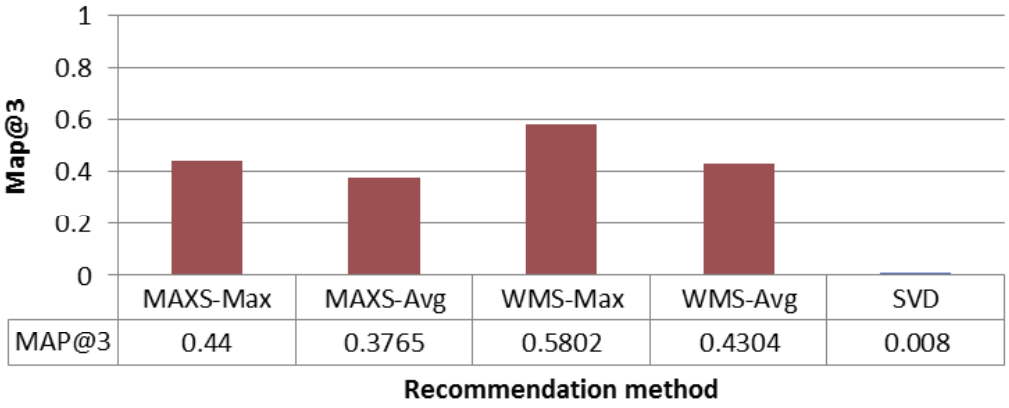


Figure 6.6: MAP@3 of our method vs. matrix factorization method.



We can clearly observe a significant difference of 0.7999 in RMSE, 0.6399 in recall and 0.5722 in MAP@3 comparing the SVD method to our best results. Even if we compare the accuracy of the SVD-based recommender with the accuracy of our target recommender not only based on the best results, we can still observe noticeable gaps between the accuracy of the two recommendation systems. By this comparison, we can prove the positive effect of incorporating hashtags in our URL recommender. Figures 6.4, 6.5, and 6.6 demonstrate the accuracy of two recommendation systems employing different methods in more detail.

6.3.3 Temporal variation in recommendation system accuracy

According to previous studies, taking the time variance into account when implementing a collaborative filtering recommendation system may generate more precise recommendations. Based on [15, 19], time-incorporated recommender systems produce more accurate recommendations than the pure collaborative filtering systems. Here, we run a experiment to show that our Twitter-based recommender is time-sensitive, meaning that recall, RMSE, and MAP@3 differ as tweets are aging from the time of first collected tweet.

To this end, we define a time window indicating the number of days we move backward in time from the time of post of the URL which we hide and try to predict. The initial window size is set to 7 days since the number of users and URLs were small in previous days which causes incorrect calculation of the performance.

Figures 6.7, 6.8, and 6.9 show the effect of increasing the time window. The temporal recall in figure 6.7 illustrates that as window size increases, recall follows a downward trend. The best recall happens somewhere between day 8 to day 10. In other words, when the time difference between URL posts becomes larger, the distance

between URLs increases and as the consequence less relevant recommendations are obtained. Also, all four similarity functions and both WMS and MAXS methods show similar temporal behavior.

The outcome of RMSE validates our findings for recall. Similar to recall, with larger time lag between the URL post timestamps, the correlation between URLs declines and RMSE rises. This finding matches the result of [27] that explains sharing URLs increases to reach a peak and then starts to fall after a period. As a result, in our case it can be inferred that item similarities based on their common users and hashtags start to decline as old URLs will eventually have fewer users or hashtags in common with more recent ones. This conforms to an explanation that Twitterers typically care more for fresh and recent URLs.

Unlike temporal recall and RMSE that followed the same patterns for all similarity functions and both methods of MAXS and WMS, MAP@3 revealed a different pattern for Euclidean similarity. According to the time dependent precision for Euclidean, as the window size grows, MAP@3 drops marginally for both MAXS and WMS. However, for Cosine, Jaccard, and Dice functions, MAXS and WMS displayed different trends. For MAXS method, temporal precision drops with the increase of window size while for WMS, it rises.

The peak in precision was obtained before day 11 for MAXS and day 22 (the last day) for WMS. It can be concluded from this result that WMS is appropriate when the time difference between the posts are large. However, the best accuracy is obtained for MAXS if the recommendation is based on the URLs in a time window of 8 to 10 days. In other words, to achieve the optimal accuracy in recommending URLs at the day d, we should consider the URLs posted from days $d-10$ or $d-8$ for MAXS and from the first day to day d for WMS.

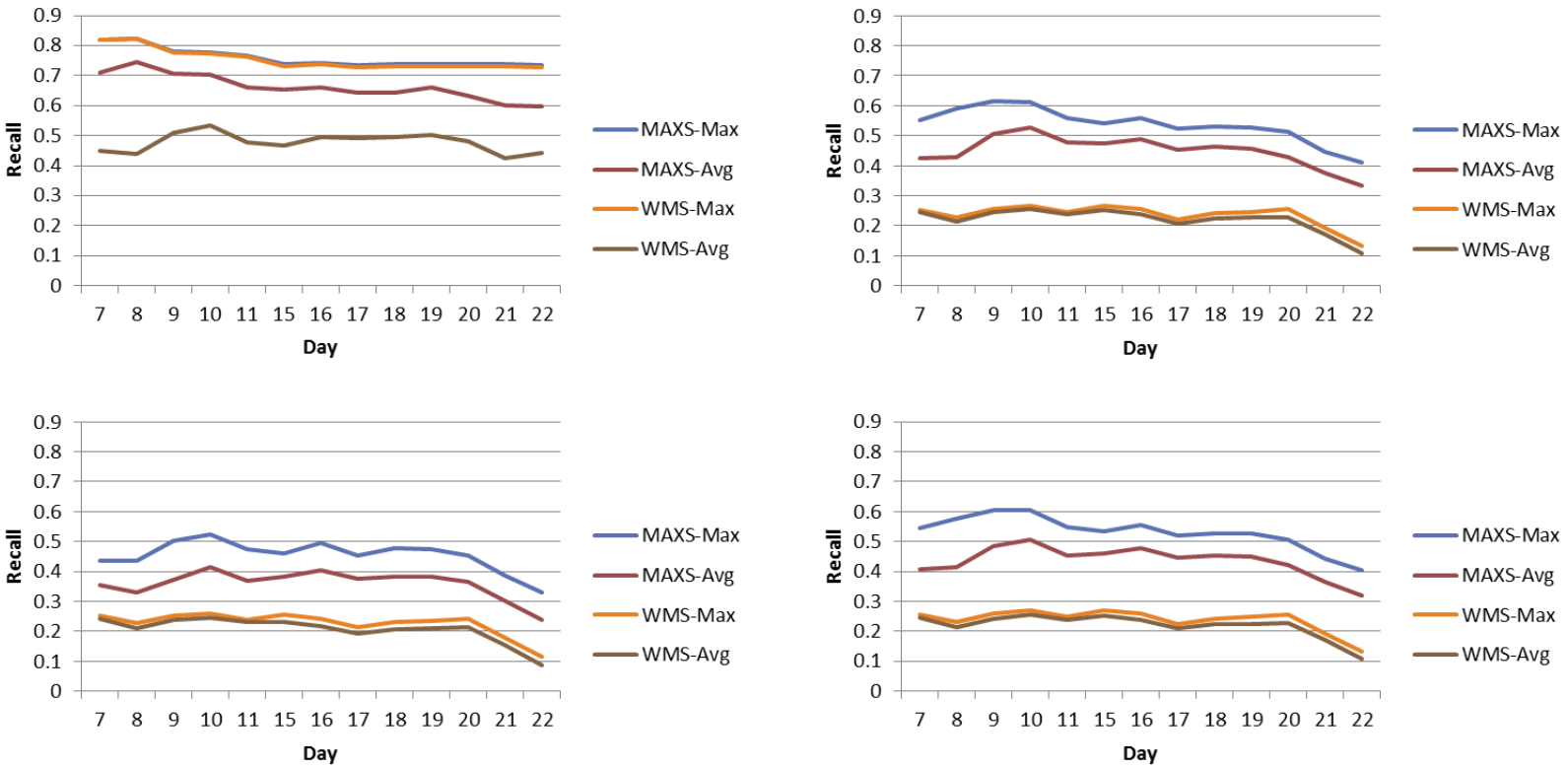


Figure 6.7: Recall of time sensitive analysis: Euclidean similarity [**Top-Left**], Cosine similarity [**Top-Right**], Jaccard similarity [**Bottom-Left**], Dice similarity [**Bottom-Right**].

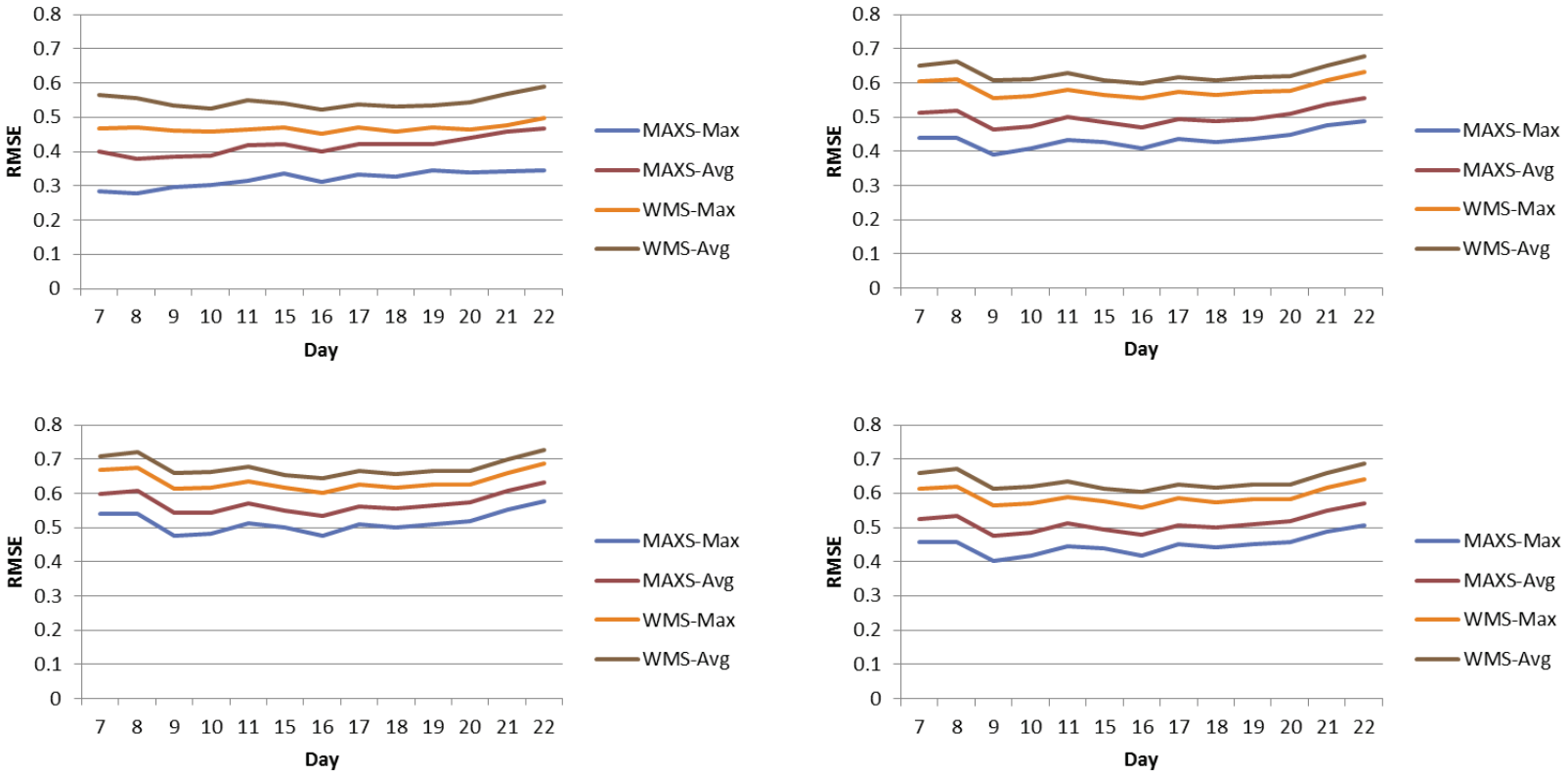


Figure 6.8: RMSE of time sensitive analysis: Euclidean similarity [**Top-Left**], Cosine similarity [**Top-Right**], Jaccard similarity [**Bottom-Left**], Dice similarity [**Bottom-Right**].

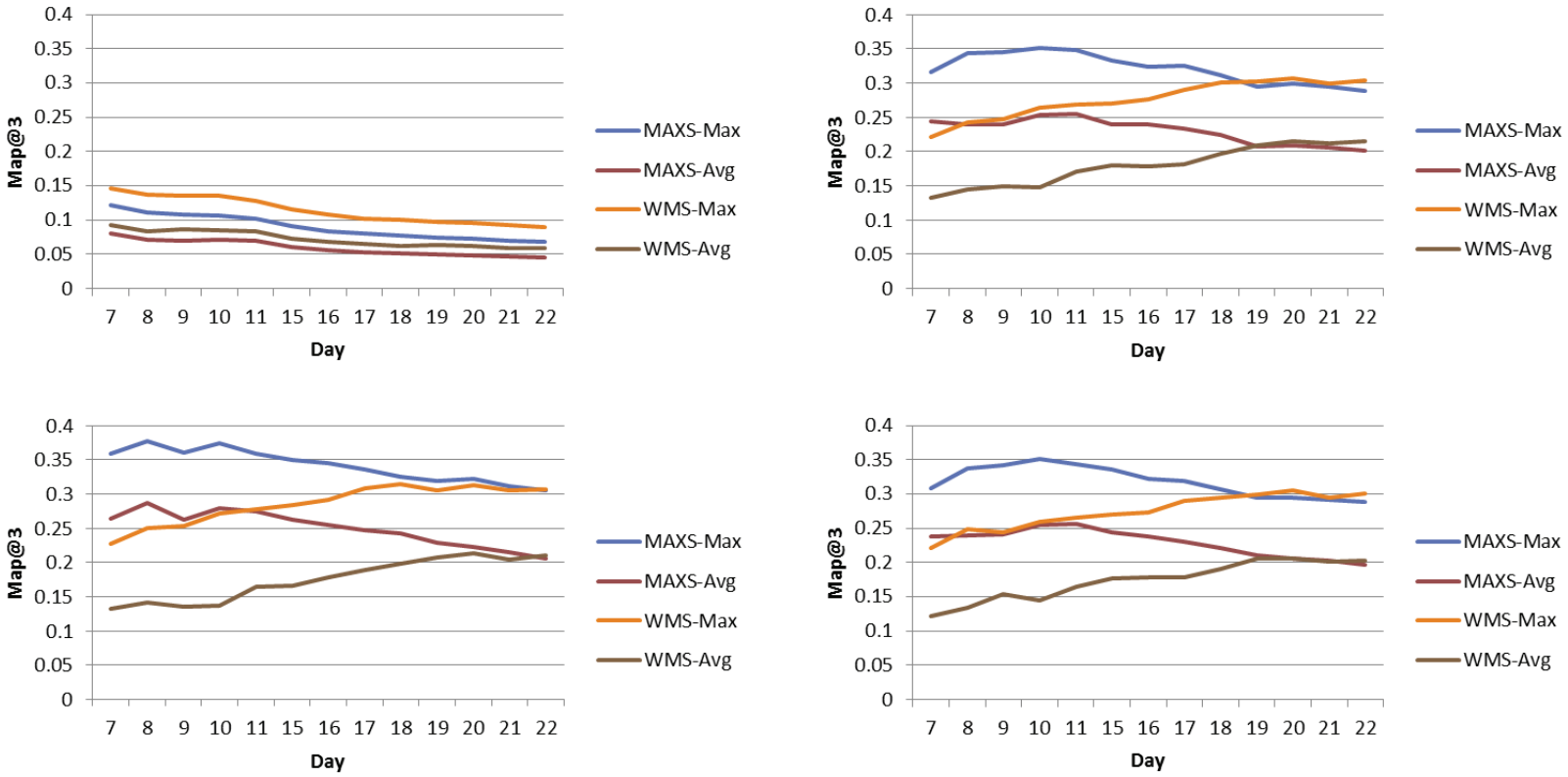


Figure 6.9: MAP@3 of time sensitive analysis: Euclidean similarity [**Top-Left**], Cosine similarity [**Top-Right**], Jaccard similarity [**Bottom-Left**], Dice similarity [**Bottom-Right**].

6.4 Impact of enriching the dataset

In this set of experiments, we explore to what extent the dataset enrichment affects the accuracy of the recommendations. In our Twitter-based recommendation system, we focus on the similarity of the URLs based on users and hashtags. We showed how incorporating hashtags in measuring the similarity of URLs changes the accuracy. We tested our method on the sample data that we have collected for almost a month using Twitter Streaming API. As mentioned before, this API only returns 1% of the tweets randomly. This randomness causes data sparsity and consequently less accurate recommendations. Therefore, enriching the data which is used to evaluate the performance of a recommender is important. This matter is even more important in our recommendation system where one of our similarity factors is hashtags. When tweets are sparse and about diverse topics, hashtags become sparse as well. Hence, the similarities between the URLs based on hashtag dimension decrease. To address this issue, we collected three other datasets focusing on a special keyword each.

For sampling, we used Twitter Streaming API with a filter this time on the tweets that should be delivered. By determining what tweets should be delivered on the stream we can capture most of the tweets around a specific keyword.

According to Twitter Streaming API documents [2], each filter is a phrase containing one or more terms separated by space. A phrase will match if all the terms in the phrase are present in each tweet. The text of the tweet, URLs and text of hashtags are checked for matches by default in the API. Also, if the filter keyword is set to "Twitter" for instance, the tweets containing TWITTER, twitter, "Twitter", #twitter, @twitter, and http://twitter.com will be returned.

We used three phrases of "London Olympics", "American Idol" and "US Election" as our filters on the stream. We found these keywords popular at the time of data collection for each category of sports, entertainment, and politics. From July 12 to

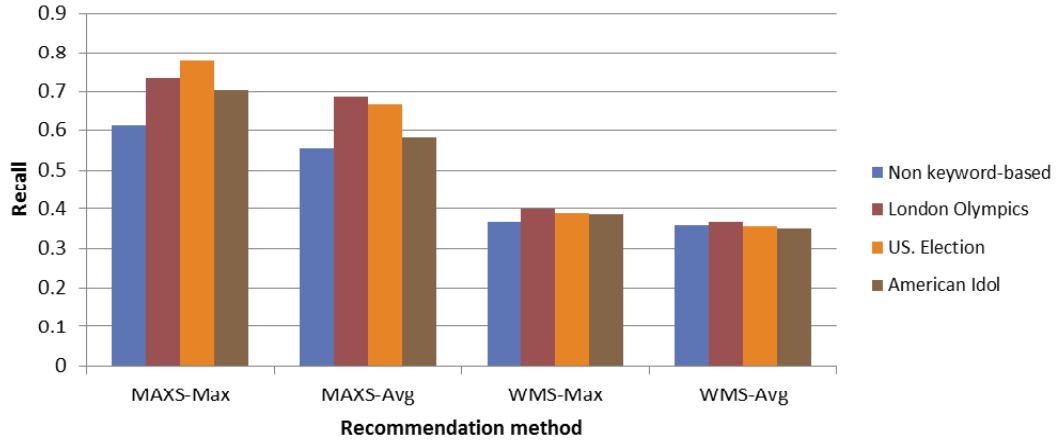


Figure 6.10: Recall of the recommender system using different datasets

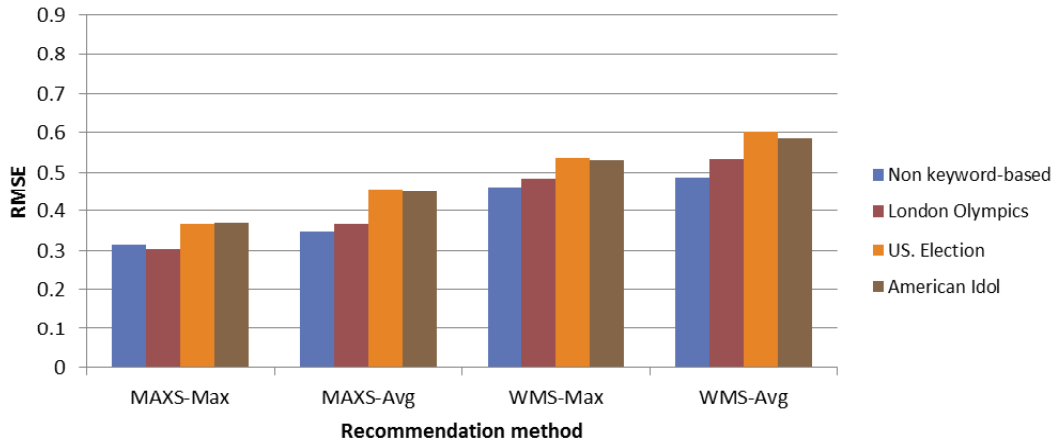


Figure 6.11: RMSE of the recommender system using different datasets

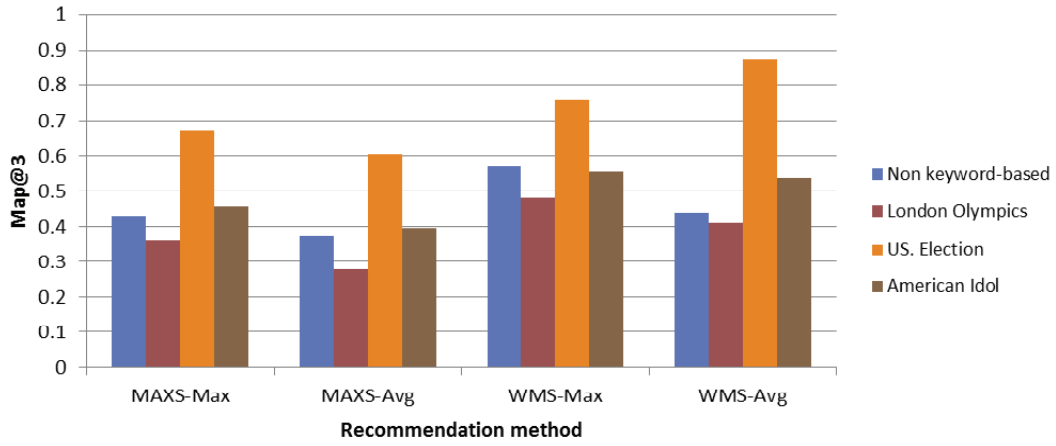


Figure 6.12: MAP@3 of the recommender system using different datasets

Table 6.4: General statistics about keyword-based datasets

	London Olympics	American Idol	US. Election
Number of tweets	763025	132741	18770
Number of users	385749	64856	12801
Number of URLs	363230	61291	8436
Number of hashtags	34309	4566	2082
Number of active users	6149	1167	214
Number of active URLs	6324	942	191

August 15 in 2012, we captured three separate datasets. Table 6.4 reports the details of each keyword-based dataset.

In order to study how the performance of our link recommender alters by injecting topic-related tweets, we ran our method from the first experiment on each dataset with less sparse tweets. Our assumption is that the tweets about a specific topic have more common hashtags. Therefore, we expect the URL similarities to boost based on the hashtag dimension.

Similar to the first experiment, the similarity between items (URLs) is measured by employing both users and hashtags similarities. Then by employing WMS and $MAXS$, we find a score indicating the correlation of pairs of items. Next, we predict if an item should be recommended to a user by taking maximum or average of the predicted scores for the items labeled as the top similars to the item.

Figures 6.10, 6.11 and 6.12 present the results of the performance of the recommender that utilized keyword-based datasets. Since the results for all four similarity functions showed the same pattern, we only reported the results of Cosine similarity function using four methods for generating recommendations. The results of this experiment are compared against our preliminary results in section 6.2.

A remarkable improvement in recall was observed for recommendation system with all three datasets in MAXS. By applying WMS on these datasets, the increase was marginal in recall.

In terms of error, using London Olympics dataset helped to lessen the RMSE slightly in MAXS with maximum function while no improvement was observed using other datasets.

The results of MAP@3 in figure 6.12 imply that enriching datasets mostly impacted the precision. When we used MAXS method and WMS method with average function, two out of three datasets showed a significant growth in precision. Using WMS method with maximum function, only US. Election dataset had better precision than the non-keyword-based.

By analyzing the overall effect of dataset enrichment, we found that filtering tweets based on a keyword has different impacts on the performance of the recommender system. For recall, the performance rises in MAXS while for MAP@3, both MAXS and WMS methods are affected. The highest precision was achieved with WMS and average function. Therefore, when we try to see the effect of dataset enrichment on the quality of recommendations, it is important to know which method or evaluation metric we are considering.

6.5 Effect of other parameters in recommendations

In our method for recommending URLs, there are various parameters that might influence the accuracy of recommendations. In this experiment, we are interested to see the effect of individual parameters in the performance of our target recommendation system. When investigating the effect of one parameter, other parameters are constant.

6.5.1 Effect of threshold for active users and active URLs

To examine the performance, we collected sample tweets from using Twitter Streaming API. Because of the limitations of this API, our tweet collection is sparse. To cope with data sparsity, data pre-processing is necessary before feeding it into the URL recommender system.

As part of the data cleansing process, two thresholds were defined as active user and active URL thresholds (section 3.1). For all the previous experiments, the user and URL thresholds were set to 20 and 10, respectively. The active user threshold is chosen to be greater than the URL threshold since in our tweet dataset, the number of users are more than the number of posted URLs.

To be able to analyze active user and active URL thresholds solely, we set α, β to 0.5, N (the number of similar items to each item) to 10 and the recommendation threshold to 0.5. Next, we applied various user and URL thresholds on our main tweet collection.

The performance results are reported in figures 6.13, 6.14 and 6.15. This experiment was repeated for each of similarity functions (four similarity functions) and each of the methods (four methods) for calculating the preference scores.

The results presented in figures 6.13, 6.14 and 6.15 are only for Cosine similarity where we used the average of top similar items. The same pattern was achieved for other cases which we did not report here. As we can see in figure 6.13 when active user and active URL thresholds increase, the performance in term of recall increases as well.

The experiment with thresholds of 10 and 5 for active user and active URL has the lowest recall, while user threshold of 30 and URL threshold of 10 return the best results amongst the selected variations. It's due to the fact that increasing the threshold for active user and active URL decreases the sparsity of the collection which

in turn results in better recommendations.

RMSE follows the same trend (figure 6.14) and gets better as we increase the thresholds. However, this pattern is less obvious for precision (figure 6.15). Yet, a URL threshold of 5 has caused a decrease in precision of the recommendation. The best result is achieved by the thresholds of 20 for active user and 15 for active URL, followed by 30 and 20 (for active user) having 10 as the threshold of an active URL.

On the other hand, this experiment has other promising results. As we can see in figures 6.13 and 6.15, $MAXS$ results in a better recall in recommendation, while WMS is good at precision. This result is relevant regardless of the selected thresholds. Therefore, we can choose our method (between $MAXS$ and WMS) based on our requirement of having a better recall versus a good precision.

Having a good recall means that each two relevant URLs are similar at least in one dimension. Therefore in $MAXS$, one of hashtag or user similarity will ensure the selection of the correct URL, which gets a better recall. However, it will decrease the precision as it may also recommend irrelevant URLs. The WMS method, though, will normalize both similarities and converging to a "generally" good similarity method, which will be more precise, but may have a less effective recall. Further experiments can validate this hypothesis, and find a reasonable trade-off between them.

6.5.2 Effect of α and β in WMS method

One of our solutions for incorporating hashtags as metadata in an item-based URL recommender is to consider a weight for hashtag dimension (α) and a weight for user dimension (β). These weights show the importance of each dimension when computing URL similarities. To this end, we introduced WMS method and generated the recommendations based on this method. In the preliminary experiment where we tried to show the significance of our work, both weights were set to 0.5 to avoid any

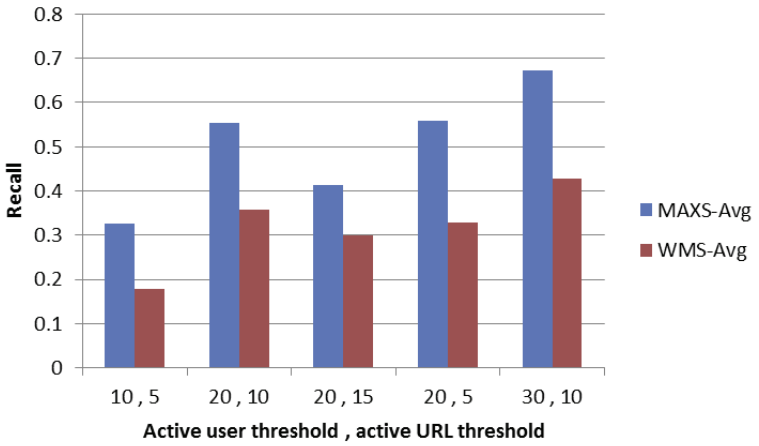


Figure 6.13: Recall based on active user and active URL thresholds

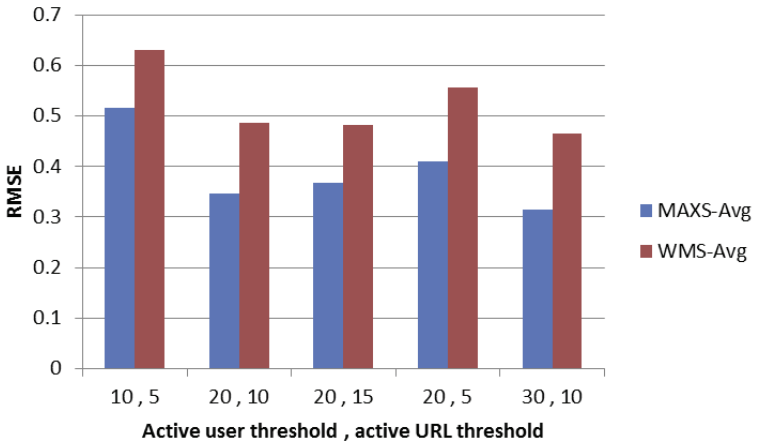


Figure 6.14: RMSE based on active user and active URL thresholds

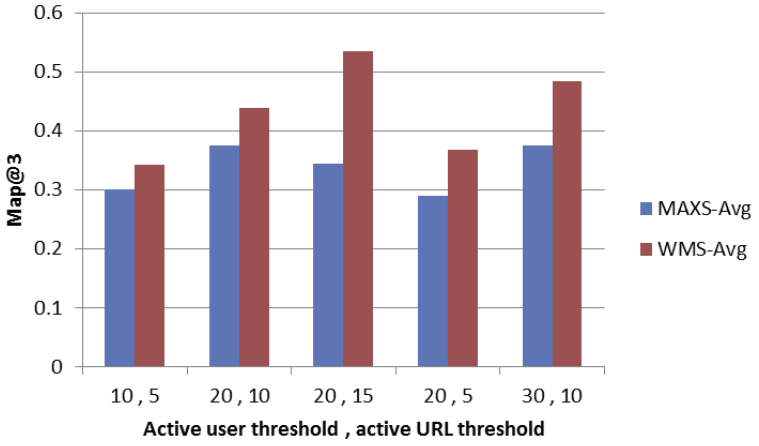


Figure 6.15: MAP@3 based on active user and active URL thresholds

bias.

To investigate the effect of each dimension in the performance of our recommender, we change the assigned weights and recalculate recall and precision of the recommendations for each situation. Results are presented in figures 6.16, 6.17 and 6.18. Note that the experiment was run using WMS-based methods as α and β are not used in $MAXS$ approach.

Based on our findings, a balance between the weight of hashtag similarity (α) and the weight of user similarity (β) does not lead to a good performance in terms of recall. Figure 6.16 illustrates that the best recall was achieved for a high α and a low β. Also, if we increase α and decrease β, the recall stays the same to the point that both are equal to 0.5. After this point, the recall rises notably.

For RMSE (figure 6.17), the more we raised α and lowered β, the performance got better. Like recall, the best performance happened when the weight for user dimension was highest.

Opposite from recall and RMSE, the best MAP@3 occurred at α 0.8 and β 0.2. Also WMS with maximum function behaved differently compared to WMS with average function. When we used average function, there was not any considerable changes in MAP@3 after increasing α to 0.5 and decreasing β to 0.5. However, with maximum function, there is a notable decrease from α 0.1 to 0.4 and a increase from α 0.4 to *alpha* 0.8.

6.5.3 Effect of top N for selecting similar items

Our method for generating recommendations is a neighborhood-based method. In a neighborhood-based method the preference score for each item is computed based on its neighbors, which means the most similar items to it. In our work, by aggregating the similarities of neighbor URLs for each URL, we predict a score (i.e. the

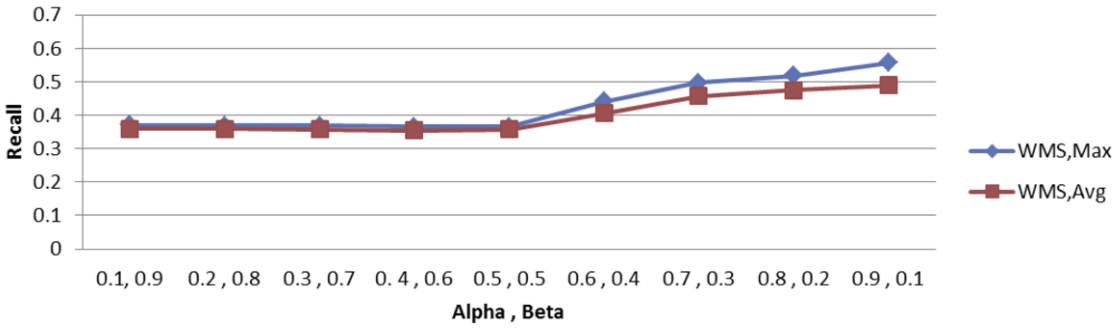


Figure 6.16: Recall based on the variation in α and β

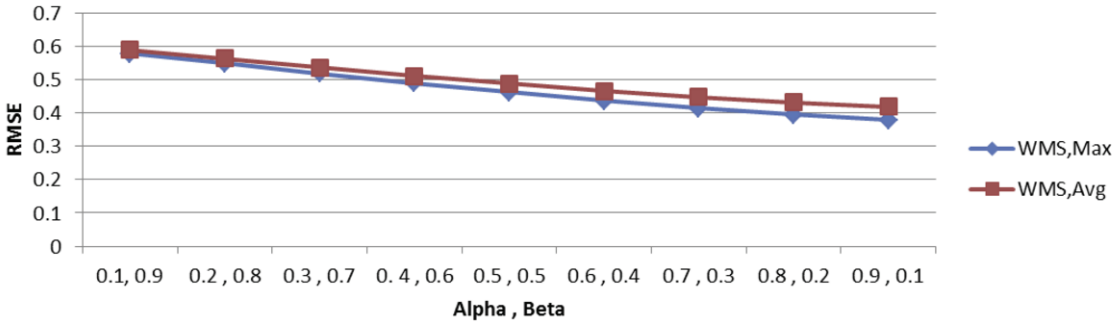


Figure 6.17: RMSE based on the variation in α and β

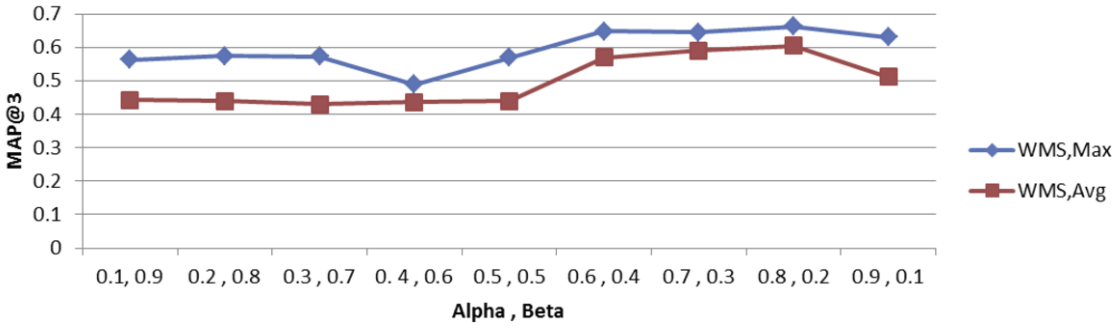


Figure 6.18: MAP@3 based on the variation in α and β

preference score) which shows how likely it is for a user to post a URL. Hence, the number of candidate neighbors might change the preference scores which will affect the recommender performance.

To find how performance fluctuates, we changed the number of top similar neighbors from 5 to 20. Based on figures 6.19, 6.20 and 6.21, using 5 similar neighbors results in a stronger accuracy, both in terms of recall and precision. A lower number of N reduces the chance of an irrelevant or "noisy" URL to interfere with the quality of recommendations. Again in this set of results, we can see that generally MAXS is better in recall measure, while WMS has more precise results. This is in line with the observations in section 6.5.1.

6.5.4 Effect of recommendation threshold

Another parameter in the quality of recommendations is the threshold for predicting whether a user would post a URL or not. After aggregating the similarities of the top N similar items using maximum or average functions, a preference score is obtained for each item. If the calculated score passes a recommendation threshold, we recommend that item. If the recommendation threshold is set to a high value, we may loose some recommendations. On the other hand, if we choose a low threshold, the recommendation system might end up recommending the URLs that users are not interested in them. Hence, selecting a correct recommendation threshold is an important matter and affects the accuracy of a recommendation system.

To be consistent with the previous experiments, we only report the results for Cosine similarity function as other similarity functions have the same effect. We chose three low, medium and high thresholds and compared the recommnder performance in figures 6.22 and 6.22.

Based on recall, when threshold is low (0.3), more items are recommended to users

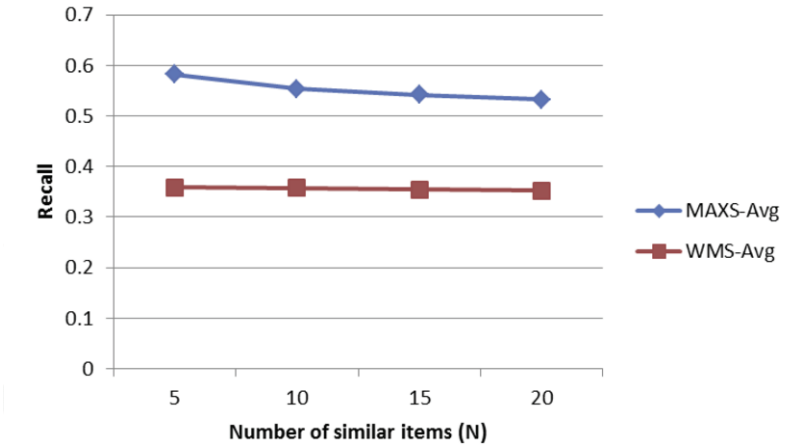


Figure 6.19: Effect of top N similar items on recall

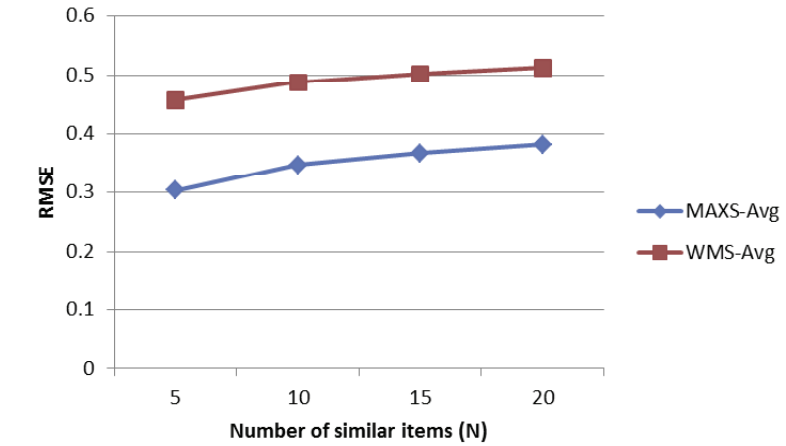


Figure 6.20: Effect of top N similar items on RMSE

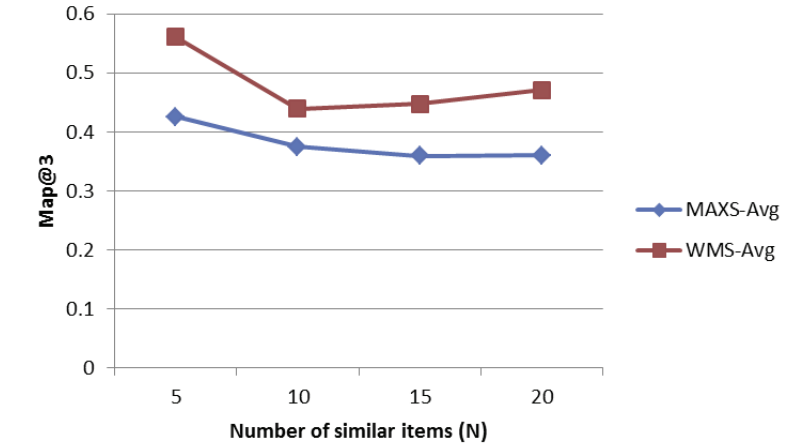


Figure 6.21: Effect of top N similar items on MAP@3

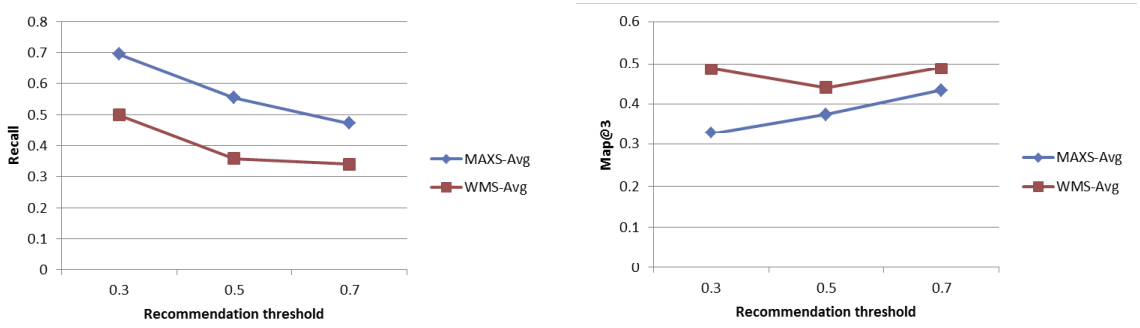


Figure 6.22: Recall [**Left**] and MAP@3 [**Right**] change with recommendation threshold

and as the result, recall is higher. As the threshold increases, we observe less recall for both MASX and WMS. Evaluating the URL recommender in terms of MAP@3 shows that for WMS both 0.3 and 0.7 reached the same precision of 0.5. However, for MAXS method, the more we increase the threshold the better result we get. Since changing recommendation threshold has no effect on RMSE, only recall and MAP@3 results are presented.

Chapter 7

Discussion

7.1 Analysis of the results

We ran several experiments using different parameters and datasets and gathered the results in order to prove the effectiveness of our method. In this chapter, we revisit the research questions and validate them using the results of those experiments.

In each of the experiments, while testing one parameter, other ones were kept constant in order to remove the bias introduced by non-relevant parameters. For each of the experiments, we evaluated our method using all three metrics of recall, RMSE and MAP@3. We also employed four similarity functions of Euclidean, Cosine, Jaccard and Dice in order to examine their behavior in different situations. Again except one situation, i.e. testing α and β, both WMS and MAXS methods were used for evaluation.

Having all sixteen conditions (four similarity metrics x four types of algorithm), we could see if the results are consistent between all three performance metrics of recall, RMSE and MAP@3. Some of the parameters showed a potential for a trade-off between recall and precision. For instance, Euclidean showed to be a good candidate

for the applications where recall has an importance, while other similarity measures showed to be good in MAP@3. Other cases are discussed in the following sections.

7.1.1 Incorporating hashtags

The first research question was about integrating hashtags in the collaborative filtering method and examining how the performance of the URL recommender system changes based on different evaluation measures. Even though the performance was shown to be dependent on the selected parameters, e.g. its associated weight, in general, our recommender proved to have a better performance comparing to a matrix factorization-based recommender. This was in terms of recall, RMSE, and precision (MAP@3), as illustrated in figures 6.4, 6.5 and 6.6.

In terms of the recall, our recommender showed a performance between 0.7929 (using Euclidean and with MAXS-max method) and 0.3503 (using Jaccard and with WMS-average), while matrix factorization-based recommender had a performance of 0.1535. Considering MAP@3 as a measure of precision, while SVD showed a precision of 0.008, our recommender showed a precision of more than 0.0967 in the worst case (Euclidean MAXS-average) and achieved 0.5802 (Dice WMS-max).

This experiment has another promising result too. As shown in tables 6.1, 6.2 and 6.3, Euclidean and Cosine conveyed to be the best choices having the recall as the accuracy metric, while Jaccard and Dice are good candidates if precision is important in the application. This is also the case while comparing MAXS and WMS methods. While MAXS showed an overall advantage in recall, WMS can be used to achieve a good precision. This means that when implementing a recommender tool, we can choose between these possible methods based on the requirements we are addressing. This conclusion can be examined further having more datasets and using statistical methods, and to find a good trade-off between different methods.

7.1.2 Time-sensitivity

Our results also supported our assumption that our URL recommender is time-sensitive. Conventionally, in a collaborative filtering recommender system the more data is used for training a recommender, the accuracy increases as users' preferences are captured better. However, the outcome of the temporal performance experiment contradicts this matter. For Twitter data, the accuracy of recommendations differs over time. Therefore, we can argue that considering all users' tweets will not necessary result in producing better recommendations.

We also noticed that there is a trade-off between the temporal recall and temporal precision. If the performance is measured based on the recall or RMSE, regardless of the similarity function, the optimum performance is obtained when more recent tweets are consumed. Accordingly, when we want to make recommendations to a user, based on her new posts, it is better to consider the submitted URLs from only 8 to 10 past days. In terms of the similarity function, Euclidean showed to be a good candidate function for finding the distance between URLs for temporal recommendations.

On the contrary, if the performance is calculated based on MAP@3, the results showed to be dependent on the method employed. In order to produce recommendations based on a user's posts, we should consider all of her URL posts in the dataset while using WMS method and only the ones from 8 to 10 previous days when MAXS method is used. Furthermore, Euclidean similarity is not working well for temporal recommendations if precision is important to us. However, Jaccard results in a better performance comparing to other three methods. This is in line with the conclusion presented in section 7.1.1.

7.1.3 Keyword-based recommender

We investigated how filtering datasets adds value to our URL recommender systems where we calculate the similarity between URLs based on users and hashtags. In terms of recall all datasets showed better performance for MAXS method compared to our non-keyword based dataset. However, for the precision, one dataset (i.e. "US. election"), showed a consistently better results than the non-keyword one in all the methods. Our recommender system showed a better precision on the dataset of "American Idol", comparing to the non-keyword dataset except for WMS method and maximum function. The last dataset (related to "London Olympics") did not have any improvement over the base dataset.

A better result in keyword-based datasets might be based on either of the following reasons. First, tweets in a keyword-based dataset are more relevant to each other and they share more common URLs and hashtags. This fact helps to have higher item similarities, which in turn causes the recommendations to be more relevant. On the other hand, the data for non-keyword based dataset is more sparse, as it is targeting a wider range of users and tweets. This is due to the nature of Twitter Streaming API, which returns a random sample of the tweets from all of the tweets posted by Twitter users. These tweets are related to different topics and are posted by people all around the world, and therefore are not necessarily useful for the generation of recommendations for a specific user.

This implies that in a real-world recommendation system, even having all the data and not sampled ones, we can still filter the dataset and use a subset of it while recommending a URL to the user. This will decrease the amount of computation required by the recommendation system, and the result of our experiments showed that this can happen without losing the recommendation performance.

7.1.4 Other parameters

Our recommendation system has many configurable parameters. While we used a default value for those parameters in our experiments, each of them can be modify to tune the performance of the recommender system based on the usage scenario.

As we followed a neighborhood-based method, a limited number of neighbors, N, for each URL were considered while computing the rating score of the URL. The results showed a consistent pattern within recall and precision, i.e. a lower number of neighbors resulted in a better recommendation. This result is in line with the findings in section 7.1.3, which means having less, but more relevant data helps improving the performance of a recommendation system.

We also used different thresholds to determine if a URL should be recommended to a user or not. Figure 6.22 showed that a higher threshold results in a better precision, while decreasing the recall of the recommendation. This result is expected and matches the normal behavior of these two factors in information retrieval systems. This also shows that the trade-off between these two metrics is unavoidable ([7]).

Also, in another experiment, the thresholds used for cleansing the dataset were examined. The general pattern, both in terms of recall and precision, was that higher thresholds resulted in better recommendations. This was an expected result and indicates that a lower level of sparsity caused enhancement in the quality of the recommender.

7.2 Limitation and future work

Our URL recommeder system was designed to generate useful URL recommendations for Twitter users. To be able to evaluate our neighborbood-based method, we captured sample tweet datasets. As mentioned before, we benefited from Twitter

Streaming API which only returns 1% of tweets at the time of the request. Therefore, our data samples became sparse. For this reason, one of the challenges that our collaborative filtering recommender system had to cope with was data sparsity. To lessen the impact of this issue for recommendation performance, we used active user and active URL thresholds. Despite the fact that we ran several experiments to analyze different thresholds, yet further investigation can be done to find different ways to alleviate tweet sparsity even more.

Another challenge that we faced in evaluation phase was that our collaborative filtering recommender was memory-based. As a result, we could only consider a part of our collected data (only 22 days) in measuring the accuracy of our recommender. We used the same dataset for proving the time-sensitivity of our recommender as well. For future work, we can consider a larger periods of time, for example a few months, to analyze after what period Twitteres switch to new topics and are not interested to be recommended the old URLs.

Moreover, due to the time and space complexity when we compared our target recommender with a matrix factorization-based recommender, we configured SVD algorithm with some constant parameters. For instance, we set the initial value to 0.1 and the number of factors to 2. In future work, more variation of SVD algorithm can be tested and our proposed method can be compared against the best run.

Also, we can explore deeper to see what happens if we only filter hashtags around a topic rather than around a specific keyword. This can be interesting as we can make sure that URL similarities will boost with hashtags around a specific topic.

Chapter 8

Conclusions

In this thesis, we analyzed the performance of a collaborative filtering recommender system that utilized our proposed neighborhood-based methods for recommending URLs to Twitter users. The core contribution of this thesis was two-fold. First, we presented two main methods to be able to integrate hashtag similarity with the user similarity for URLs (which is used in conventional recommender systems), and come up with a single preference score for users for their unseen URLs.

We evaluated our system with the most commonly used metrics for prediction accuracy including recall, RMSE and MAP@3. The analysis of the results suggest that the accuracy of our neighborhood-based recommender system improved significantly in comparison with SVD as a successful method in matrix factorization. Our experiments showed that our solution can achieve a better accuracy of 0.79 in terms of recall, 0.08 in terms of RMSE, and 0.58 in term of MAP@3. With these results, combining item similarities based on various dimensions turned out to be an advantageous solution to in collaborative filtering.

As the second main contribution, we showed that our Twitter-based recommender system is time-sensitive. Meaning that providing more posting history of users will

not necessarily result in more accurate performance and it will damage recommender accuracy for Twitter. We found that there is a trade-off between recall and precision in temporal performance. To improve recall of our recommendations, MAXS seems to be a more useful method with considering the URLs not older that 10 previous days. Oppositely, for boosting MAP@3, WMS with considering all the URLs seemed to be a better approach.

Overall, the experiment for temporal accuracy variation demonstrated that taking the timestamps of the postings of the URLs into account is quite beneficial as it affects the recommendation performance.

Our solution to alleviate data sparsity by enriching dataset also turned out to be capable of delivering good recommendations. We used datasets around certain keywords, and we found that their results were improved over the non-keyword based dataset in terms of at least one of recall or precision.

We noticed somewhat a trade-off between recommendation recall and precision. We examined the parameters in our recommender system to verify the effect of each of them. In most cases recall and RMSE conveyed the same behavior. However, when recall increases precision of the system declines. The only exception was in top N neighbor analysis that a better performance is achieved by a smaller value of N, both for recall and precision.

Recommendation threshold, as might be expected, showed a clear potential for a trade-off between recall and precision. Changing the threshold from low to high, URLs move from being recommended to not-recommended. Therefore recall drops and MAP@3 enhances. For WMS method, the weights assigned to user-based and hashtag-based similarities showed a trade-off between recall and precision, having hashtag-based similarity weighted at 0.2 and 0.4 resulting in the best precision, while moving toward user-based similarity increases recall.

Bibliography

[1] https://dev.twitter.com/docs/auth/obtaining-access-tokens.

[2] https://dev.twitter.com/docs/streaming-apis.

[3] http://alias-i.com/lingpipe.

[4] http://alias-i.com/lingpipe/demos/tutorial/svd/read-me.html.

[5] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997.

[6] T. Bogers and A. van den Bosch. Collaborative and Content-based Filtering for Item Recommendation on Social Bookmarking Websites. In *Proceedings of the ACM RecSys'09 Workshop on Recommender Systems & the Social Web*, pages 9–16, New-York, NY, USA, 2009.

[7] Michael K. Buckland and Fredric C. Gey. The relationship between recall and precision. *JASIS*, 45(1):12–19, 1994.

[8] Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed Chi. Short and tweet: experiments on recommending content from information streams. CHI '10, pages 1185–1194, New York, NY, USA, 2010. ACM.

[9] Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yoa, and Yong Yu. Collaborative personalized tweet recommendation. SIGIR '12, Portland, Oregan,USA, 2012. ACM.

[10] Jiesi Cheng, Aaron Sun, Daning Hu, and Daniel Zeng. An information diffusion-based recommendation framework for micro-blogging. *J. AIS*, 12(7), 2011.

[11] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[12] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. RecSys '10, pages 199–206, New York, NY, USA, 2010. ACM.

[13] John Hannon, Kevin McCarthy, and Barry Smyth. Finding useful users on twitter: Twittomender the followee recommender. In *ECIR*, pages 784–787, 2011.

[14] Younghoon Kim and Kyuseok Shim. Twitobi: A recommendation system for twitter using probabilistic modeling. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 340–349, 2011.

[15] Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, April 2010.

[16] Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[17] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[18] Shamanth Kumar, Fred Morstatter, and Huan Liu. *Twitter Data Analytics*. Springer, New York, NY, USA, 2013.

[19] Tong Queue Lee, Young Park, and Yong-Tae Park. A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications*, 34(4):3055 – 3062, 2008.

[20] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76 – 80, jan/feb 2003.

[21] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.

[22] Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill & webert: Identifying interesting web sites. In *AAAI/IAAI, Vol. 1*, pages 54–61, 1996.

[23] Marco Pennacchiotti, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. Making your interests follow you on twitter. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 165–174, New York, NY, USA, 2012. ACM.

[24] Owen Phelan, Kevin McCarthy, Mike Bennett, and Barry Smyth. Terms of a feather: Content-based news recommendation and discovery using twitter. In Paul Clough, Colum Foley, Cathal Gurrin, GarethJ.F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Mudoch, editors, *Advances in Information Retrieval*, volume 6611 of *Lecture Notes in Computer Science*, pages 448–459. Springer Berlin Heidelberg, 2011.

[25] Taketoshi Ushiama and Tomoya Eguchi. An information recommendation agent on microblogging service. In James OShea, NgocThanh Nguyen, Keeley Crockett, RobertJ. Howlett, and LakhmiC. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, volume 6682 of *Lecture Notes in Computer Science*, pages 573–582. Springer Berlin Heidelberg, 2011.

[26] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. SIGIR '06, pages 501–508, New York, NY, USA, 2006. ACM.

[27] Shaomei Wu, Chenhao Tan, Jon M. Kleinberg, and Michael W. Macy. Does bad news go away faster? In *ICWSM*, 2011.

[28] Nazpar Yazdanfar and Alex Thomo. Link recommender: Collaborative-filtering for recommending urls to twitter users. In *ANT/SEIT*, pages 412–419, 2013.