RE-RANKING THE TWITTER SEARCH RESULT

By

Prerna Pathak

B.Sc. in Computer Science, Maharishi Dayanand University, India, 2008

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the program of

Computer Science

Toronto, Ontario, Canada, 2014

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Re-Ranking the Twitter Search Result

Prerna Pathak
Master of Science, Computer Science, 2014
Ryerson University

ABSTRACT

Social Sharing Platforms, a great source of free and diverse information, have been center of attraction to many people. Users post their opinions, thoughts, life events, news and all other information. This data flowed into these systems has increased to such a limit making nearly impossible for a user to read all or even most of it, analyzing and utilize it. As a solution to this problem, we here are proposing an approach, which makes use of not only the tweets themselves but also their properties to re-rank the tweets given a user query. The proposed approach was implemented in a prototype system and test results were generated. A set of feedback data collected via online survey for those test results provides a good evaluation score, with an average improvement of around 10% on precision values after removing the outliers. It shows that our approach can generate improved results over the original ones.

ACKNOWLEDGEMENTS

I am indebted to the kind people around me. Without their help and support, it would not have been possible to write this thesis.

It gives me great pleasure in acknowledging the support of my supervisor Dr. Cherie Ding, for her valuable advice and great patience, whose effort made this thesis possible. I would like to thank her for the time and effort she has put in our weekly meetings and reviewing my thesis many times.

I am very thankful to Dr. Isaac Woungang, Dr. Abdolreza Abhari, and Dr. Alex Ferworn who have reviewed my thesis and given me insightful feedbacks, which enabled me to improve my thesis.

I would like to express my deep appreciations to my family, for their great patience and motivation at all times and inspiring me to follow my dreams. I owe my deepest gratitude to my parents who supported me emotionally, financially and provided comfort to me at all times.

Last, but by no means least, I share the credit of my work with my well-wishers, who encouraged me, fueled me confidence, provided me with the required resources for this work and helped me to achieve my dreams.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background on Twitter and Twitter Search

Social networking platforms, a great source of free and diverse information, have been the centre of attention for many people. Users post their opinions, thoughts, life events, news and all sorts of information on the social network.

Twitter is one of the most popular social networks people are using every day. The idea behind Twitter is that you broadcast information to anyone who chooses to follow you, and similarly, you can choose to follow people and receive their messages.

If a user wishes to find some tweets, the procedure is to search for tweets that include a particular word. For instance, if you type "neuroscience" into the search box in Twitter, you'll see all the relevant tweets that include that word. Twitter search also provides a variety of operators and filtering options using which users can limit the type of tweets displayed in the results.

1.2 Problem Statement

With the growing popularity of such social networking platforms, the amount of information has increased to a level that it is nearly impossible for a user to read all or most of it. In Twitter, one will notice that hundreds and thousands of tweets are posted on an event within a matter of minutes.

There are inbuilt search features in Twitter. However, sometimes the results may not be very helpful. Twitter considers how similar the tweet is to the query in terms of its textual content. It also considers the recency of the tweet, the popularity of the user who posts the tweet, and possibly other factors. According to Busch et al. [1], the force behind Twitter's real time search service is Earlybird. Like all other modern retrieval engines, Earlybird also builds and maintains inverted indexes but its index structure is different from those built to support the traditional web search. Because Twitter search is considered as the real-time search, EarlyBird has the capability to consume content speedily and make it searchable instantly. The detail of their searching algorithm is not available to the public. The paper on their searching platform EarlyBird also does not give the details of the ranking algorithm. Therefore in this study, we would like to explore possible ways to re-rank the tweets for a given user query, and then compare the results with the original Twitter ranking. By considering different factors, we would like to achieve goals for a better ranking result for the Twitter search, which will be tested by the experiments on a real dataset.

1.3 Objectives

In this thesis, we intend to perform a set of calculations considering various factors to check whether the proposed approach could achieve better ranking results. This study believes that the ranking of a tweet for a query not only depends on its content or its popularity, but also on the author who posts it. Not only should the tweet be relevant, but it should also come from a trusted source. The main factors consider in this study are categorized into two groups:

- Tweet Quality: This includes how recent the tweet is, how many times it has been retweeted, how similar its content is to the query, uniqueness of the content, and the availability and richness of the media information.

- Author Authority and Popularity: This includes how active the author is, how many friends and followers the author has, whether the author is easily reachable and whether the author is a source of information.

The main intention behind having an extensive approach and covering all aspects is to explore all possible factors that can influence the overall rank of a tweet. By covering important aspects, we have the option of trimming less influential tweets.

1.4 The Proposed Methodology

The proposed approach aims to utilize various properties of tweets to define the ranking function. Nine different ranks are calculated based on different factors that are later combined to generate the overall rank of tweets.

A tweet is as good as its content, and its author. Suppose a user searches for a keyword such as "Social Computing". In the first step, we will obtain the 200 latest tweets from Twitter, which will be used for later processing. The values of the properties such as the tweet posting time, retweet count, and presence of the media information will be collected. A set of ranking scores will then be calculated. Among them, some of the important ones include the following: the impact rank, which denotes how popular the tweet is, and how many users have retweeted it; the relevancy rank that measures the similarity between the tweet and the query; the uniqueness rank which denotes the uniqueness of content of a particular tweet compared to other tweets in the result set; the author popularity rank, which helps us to understand the popularity of the author in the network; the author connectivity rank which calculates how many exclusive followers an author has, that is, who follows the author, but is not followed back by the author; and the author authority rank which illustrates how desirable the author is within the network,

how many users follow him, whether he is a source of information, or how much he seeks information from others.

In this study we use data sources and ranking factors extensively to make sure we obtain a highly related set of high quality results. This proposed approach not only checks the content of tweets, but also has a major emphasis on who posted it, the status of its author, whether he is active enough, whether he is well connected, and whether he is a source of information.

The first few ranks focus on the content and properties of the tweet, the latter ones focus on the quality of author. The last few ranks are not calculated based on values directly from the tweet content itself. However, they are important as they ensure that the source of tweet is worthwhile.

There are two major contributions of this work: 1) to the best of our knowledge, there are very few research efforts on re-ranking of the Twitter search results, although various data properties in Twitter have been studied for different purposes such as recommender systems, our focus on Twitter search is unique; 2) the proposed method considers a combination of multiple factors including tweet related properties and user related properties, although some of the factors have been studied in other work, putting them all together is new, and among them, a few factors are novel, including content uniqueness and media richness of the tweet.

1.5 Thesis Outline

The rest of the thesis is organized as follows.

Chapter 2 provides the literature review on the background and the closely related work.

In Chapter 3, we define the Twitter related terms along with the system architecture of our proposed re-ranking algorithm. We also explain the re-ranking algorithm based on individual factors and the way to combine them.

Chapter 4, present our implementation and experimental results on Twitter data, which illustrates the effectiveness of our method.

Finally in Chapter 5, we conclude our thesis and discuss future directions we may work on.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we will first review some technologies that we have used in our approach. Then, we will review some most closely related literatures to our work.

2.1 Background

2.1.1 Vector Space Model

According to Yates and Neto in [2], Vector Space Model (VSM) is an algebraic model that represents text documents as vectors. In this model a document is represented as a vector d_j with weights associated with them for each term. A query is represented as a vector q. The presence of a term in the vector is identified by its non-zero value and the zero value identifies the absence of a term. The definition of terms is depending on the application. A term can be a single word or a phrase.

The most popular way to compute weight values is the tf-idf weighting method. We use this method in our proposed algorithm to calculate the content relevancy of tweets to a given query and to calculate the uniqueness factor of tweets. Cosine similarity formula is used to calculate the similarity between two vectors d_j and q as listed below:

$$sim(d_j, q) = \frac{d_j.q}{\|d_j\|.\|q\|} = \frac{\sum_{i=1}^{N} w_{i,j} \cdot w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}} \quad (2.1)$$

where N represents the number of terms in the vocabulary, $w_{i,j}$ is the ith word of document j, $w_{i,q}$ is the ith word of query q.

2.1.2 PageRank

PageRank [3, 4] is a measure of reachability of a web page. It indicates what are the chances of a user will end up on that page while surfing the web. That is, how well the page is connected by other pages or how well other pages points or links to it.

PageRank of a page is dependent on the PageRank of pages linking to it. For a page *E*, its PageRank is defined as *PR(E)*. If a page has links from pages with high PageRank, its own PageRank will be high too [3, 4].

The original PageRank algorithm was proposed to deal with a linked web graph consisting of web pages. It can be used to process any kind of graphs. In our proposed algorithm, users are linked through their following relationships. By calculating PageRank of users in this graph, their importance levels can be ranked, which will affect the importance levels of their tweets. The formula for calculating PageRank is listed below,

$$PR(a) = \frac{q}{t} + (1 - q) \sum_{i=1}^{n} \frac{PR(p_i)}{L(p_i)} \quad (2.2)$$

where, *a is* a page for which PageRank is calculating,

q is a damping factor,

t represents the total number of pages,

n represents the number of pages pointing to *a*,

p_i represents a page which is pointing to *a*,

L represents the number of links from page p_i.

2.1.3 Hub and Authority Ranking

A link analysis algorithm was developed by Jon Kleinberg [5] to rate web pages; it was known as Hyperlink-Induced Topic Search (HITS) (also known as hubs and authorities). It was proposed at almost the same time as PageRank.

Hubs are large web directories that do not own the information but lead users to other valuable information authorities. To be a good hub the page should lead you to good authoritative pages and to be a good authority the page should be linked to by several hubs. Two scores are calculated for each page, one based on the quality of content known as its authority, and the other based on the number of links to other pages known as its hub value. The HITS algorithm aims at delivering the most relevant page to what is asked for in the search query.

We consider two types of updates: Authority Update Rule and Hub Update Rule. In order to calculate the hub/authority scores of each node, the Authority Update Rule and the Hub Update Rule are applied iteratively. A k-step application of the Hub-Authority algorithm entails applying for *k* times first the Authority Update Rule and then the Hub Update Rule.

Based on Authority Update Rule,

$$A(p) = \sum_{q \in B, q \to p} H(q) \tag{2.3}$$

Based on Hub Update Rule,

$$H(p) = \sum_{q \in B, p \to q} A(q) \tag{2.4}$$

where, *B* is the base set,

q and *p* are web pages in *B*,

A(p) is the authority score for *p*,

H(p) is the hub score for *p*.

The final hub-authority scores of nodes are determined after infinite repetitions of the algorithm. As directly and iteratively applying the Hub Update Rule and Authority Update Rule leads to diverging values, it is necessary to normalize the matrix after every iteration. Thus the values obtained from this process will eventually converge to final non-fluctuating values. [5].

2.2 Related Work

2.2.1 Measuring Various User or Tweet Attributes in Twitter

In this section we are reviewing the papers which talk about various attributes in Twitter such as analyzing the microblog postings, various users and posts.

Pal and Count [6] have proposed the features and methods which can be used to make an ordered list of top authors for finding topical authorities in microblogging environments. They suggested a number of features of authors and observe that topical signal and mention impact are marginally more important than other features. They also presented that the cluster probability is an effective method to screen a large mass of outliners in the feature space and select best specialized users on which the ordering can be applied more strongly. Finally, they showed that Gaussian based ranking is more efficient method to order users. Results revealed that their method is more efficient in near real time scenarios and is better than the baseline models.

Castillo et al. [7] analyzed the microblog postings related to the trending issues and categorized them as reliable and not reliable, based on the features collected from them. They used the features from users' postings, repostings and from the content to external sources. They assessed their methods using a good number of human assessments about the reliability of content in the given sample of Twitter postings. The emphasis of their experiments was the reliability of content spread through social media network. They have demonstrated that the

time-sensitive issues can be differentiated spontaneously as newsworthy or informal. Also they observed that the newsworthy topics often include URLs and have propagation trees. The trustworthy news is generally broadcasted through authors who already posted many messages and have many re-tweets. Finally they found that there are calculable variances in the path the messages are broadcasted and can be classified as newsworthy or informal with accuracy in the range of 70%-80%.

Bakshy et al. [8] dealt with the aspects and effects of 1.6M Twitter users on the follower graph during a two month period in the year 2009. The investigation was done by tracing the 74 million diffusion events over the follower graph during this duration. The research reveals that the largest flows are made by the users who were leading in the past. Also they found that URLs which were rated more by users were more likely to flow. In spite of such results, they found the flows that particular user or URL makes are unpredictable. Thus they saw that word of mouth diffusion can only be captured reliably by focusing on the great number of influencers who are the users more influenced by other users' posts. This provided the average effects of the influencers. It is considered that sometimes the most influential users are cost effective and their performance can be felt by using "ordinary influencers", who are the users who exerting less or average influences.

Yang et al. [9] dealt with the issue of automatically knowing the most appreciated posts to a large audience. Stress is laid over the automatic ways to know the tweets which not only are of great concern to the writers of the post on Twitter and their friends but a large audience too. The social networking site, Twitter is modelled as a user's graph and tweet nodes linked by the retweet edges. The model presents different HITS algorithm which relates to the retweet graph for generating a stable order of tweets. Basis of this method is that the retweet relationship in the

graph can be used to find the tweet which not only draws the attention of the writers' network but beyond that and impels others to retweet.

Meeyoung et al. [10] showed that many influential users can affect on various topics. The power law distribution indicated that the top users have unequal type of influence. The temporal analysis indicated the various influential networks with users. Top news firms have a high level of retweets over various topics whereas celebrities are good at getting mentions on different topics. Finally, they found concentrated efforts of users are required to gain the influence. In order to achieve and keep influence, users need to possess great personal involvement.

2.2.2 Twitter Ranking or Re-Ranking

In this section we are reviewing papers that deal with different procedures and methods which help in ranking or re-ranking of tweets.

Duan et al. [11] proposed three types of tweet features that have been studied and a proposition to rank tweets by using learning to rank algorithms has been discussed. The most effective tweet features are first identified. Research suggests that the system could use some tweet features like number of follower, presence of URL, list of friends etc., to see which features performs the best for providing good ranks. Among all of the above, whether a tweet contains a URL is the most important feature. According to their research, the number of times an account is mentioned by other users is far more effective to determine the account authority than the number of followers of a specific account.

Zhang et al. [12] proposed a combined learning to rank framework that uses both general and query specific proof of relevance for real time Twitter search. In other words the characteristics of queries are better tapped with the help of query biased ranking model which is

learned by a semi-supervised transductive learning algorithm. Finally this approach is integrated with the traditional approach of ranking in order to display tweets according to user preferences. A wide variety of studies on the standard Tweets11 dataset [13] suggest that the query biased approach is far better than the traditional ranking system. There is also a probability of duplicating training data without using human labels on given new queries. Studies reveal that the number of interactions in the process in transductive learning algorithm is the main reason of its efficiency.

Sarma et al. [14] proposed a solution for developing ranking models for forums. It mainly dealt with thumb and star ratings based and comparison based review of items in the forum. The advantages of comparison based ranking model are its correctness and ability to rapid convergence with minimal user's feedback. An online forum called Shout Velocity [15] has been discussed in detail based on comparison based ranking model. According to experiments based on artificially extracted data and real data from Shout Velocity, it is clear that the comparison based ranking model is far superior than the thumb based ranking on certain selected properties like ranking accuracy, review feedback bandwidth, low latency and fairness.

Teevan et al. [16] showed how users search Twitter content and how their Twitter search is different from web search. A detailed scrutiny of large query logs was undertaken to find out and distinguish the search behavior of users who issue queries to both Twitter and web search engines. The main reason for searching Twitter stems from the fact that users need timely information (related to news or events), and they also need social information (related to other users and popular trends). There is a great difference between the queries submitted to Twitter and web search engines and this can be understood by studying the queries. Queries submitted to Twitter are shorter with longer words, and more references to people. The frequency of queries

was also quite different for both (Twitter and web) search engines. Twitter is used mainly for viewing new content while web is used for studying and learning a topic. Queries in Twitter are more popular, repeated frequently and are less dynamic than web queries. Queries in Twitter are more social and event based while web queries are factual and navigational.

Shen et al. [17] presented a supervised learning method that will rank tweets based on their efficiency on the user's interest. It implies that good quality tweets will appear on the top while low quality at the bottom. A system known as topic model is created to assess the topics in a tweet structure and determine the user's interest in each topic. The system is built on the basis of the aggregate tweet generated or consumed by the user. These aggregate tweets are identified for a particular user as a single document. All the tweets are aggregated because the word repetition makes the topic model more accurate than using a single tweet as a document.

According to Feng and Wang [18], a new problem was identified called personalized tweet ranking so that users could access valuable information amidst enormous amount of tweets. A general graph model was created to interpret retweet behaviour. Feature vectors of nodes and edges were used to depict various sources of information. Based on the graph, they proposed a feature-aware factorization model to re-rank the tweets, which combines the linear discriminative model and the low-rank factorization model effortlessly.

2.2.3 Recommender Systems for Twitter

In this section we are reviewing papers which build the recommendation systems for Twitter.

Chen et al. [19] suggested a collective tweet ranking model for suggesting useful tweets to users on Twitter. Their method benefits by the collective filtering based recommendation by

gathering important information from many users. Extra related content helpful for knowing personal interests is combined in their collective ranking model by cautious plan of features. Their final method used three major elements: tweet content, user social relations and explicit features such as authority of the publisher and quality of the tweet. The experiments on the real-world data showed all the data can help in improving the recommendation performance for the users.

Chaoji et al. [20] presented the difficulty of recommending connections in Twitter with the clear objective of increasing data spread in the network. They suggested novel algorithms for recommending connections that acknowledges the submodular criteria taking to computationally possible approximation procedure in the presence of the above limits. Results on the real graphs showed the dominance of their method in comparison with the common heuristics.

Phelan et al. [21] proposed that large amount of data from the real-time networks like Twitter can be collected as a useful source of recommending knowledge. Buzzer is a news recommendation system that is able to collect the conversations that are posted on Twitter. Buzzer is capable to order RSS news by extracting trending data from both the public Twitter timeline and from the timeline of tweets generated by a user's network. The method Buzzer used is the content based approach.

Chen et al. [22] said that all of the social media platforms have one thing in common that is interaction between various users. Twitter and Facebook is known for being mediums of communication across the world. Due to this, finding an interesting conversation to read poses a user with a challenge mainly because there is enormous amount of data and what is interesting differs from one person to another. In this paper they have detailed 5 algorithms that will suggest influential conversations to the Twitter users based on the chain of tweets, issue being discussed

and the tie-strength. An online user study was conducted to compare the algorithms and feedback was sought from the real users of Twitter. This study helped in analysing how the different purposes of using Twitter affected the user's choice of conversations and performance of the algorithms.

2.2.4 Twitter's Own Search Function

This category of papers deals with different Twitter's own search options like Earlybird [1] as describe in chapter one.

Lin and Efron [23] presented the notion of a temporal relevance profile, that user can openly include with keyword search query. They introduce temporal relevance profiles and show that how their advantage can be taken by present retrieval models. Data tracked from TREC 2011 and 2012 by oracle experiments [24] on microblog empirically proves that their approach has the possibility to meaningfully increase the quality of retrieved results.

Asadi and Lin [25] used machine-learned modules considering a multi-stage retrieval design which contains three stages. First stage is of a fast, "cheap" candidate generation stage; Second stage is extraction of feature stage and the last is "expensive" re-ranking stage. The second stage can be achieved by document vector index that is a mapping from document ids to document representations. For better feature extraction they deliberate other organizations of data structure which includes design choice, document organization, complex term proximity features and how compression of structures can be achieved. In brief, for efficiently encoding term ids they introduced a new document-adaptive hashing scheme. By experiments on memory footprint and feature extraction speed they assessed the effect of their design. Overall, their

results show that their system design has extra advantages in terms of elasticity and also needs less memory overall as compared in speed to using an old positional inverted index.

Because Twitter search is considered as the real-time search, EarlyBird has the capability to consume content speedily and make it searchable instantly. Much of the focus was put on the indexing component of EarlyBird in [1], and the detail of their searching algorithm is not available to the public. Therefore in this study, we would like to explore possible ways to re-rank the tweets for a given user query, and then compare the results with the original Twitter ranking. There are other approaches in this area too, but their approach is different. For instance, Chen et al. [19] proposed a collective filtering based tweet ranking model by gathering important information from many users. Asadi and Lin [25] used machine-learned modules considering a multi-stage retrieval design. Duan et al. [11] used three types of tweet features and learning to rank algorithm to learn the ranking model. This work is different in that we divide ranking features into two types – tweet-related and user-related, and each type include multiple features, and the combination of all these features is used to generate the final ranking order.

2.3 Summary

In this chapter we first described the methods such as VSM, PageRank and Hub/Authority calculation, which we are going to use in our proposed algorithm. Then we reviewed work on various Twitter related topics such as Measuring Twitter user's influence, Twitter ranking or re-ranking, Twitter recommender and Twitter's own search. In our proposed algorithm we are re-ranking tweets based on different aspects. This is going to be described in the next chapter.

CHAPTER 3

METHODOLOGY

Based on the discussions of previous chapters, we propose an algorithm for the re-ranking of tweets based on two major concerns: tweets and users. We re-ranked the tweet based on individual factors related to the tweet itself and the user who posted the tweet. In the following sections, first we will define some Twitter related terms that we use in our discussion, then we will describe the current search process in Twitter, and after that, we will explain our system architecture and our proposed re-ranking algorithm in details.

3.1 Definition of Twitter-Related Terms

Below we give definitions and explanations on some Twitter related terms.

- *User* can be anyone or anything. They tweet, follow others, create lists, have a home timeline, can be mentioned, and can be looked up. There are various properties [26] associated with the *User* object that uniquely define and identify a user. Among these numerous properties associated with the *User* object, we have used the following in our algorithm:
    - *ID* is an integer representation of the unique identifier for a user. This number is greater than 53 bits, and some programming languages may have difficulty in interpreting it. Using a signed 64 bit integer for storing this identifier is safe. An example could be: "id":6253282.
    - *Screen_name* is the handle, or alias, that this user is identified with. *Screen_name* is unique but subject to change. Typically, the screen name is a maximum of 15 characters long, but some older accounts may exist with longer names. An example could be: "screen_name":"twitterapi".

- *Name* refers to the name of the user, as they have defined it. And it is not necessarily a person's real name. This name is typically capped 20 characters, but subject to change. An example could be: "name":"twitterapi".
- *Friends_count* refers to the number of users this account is following (that is, their "followings"). An example could be: "friends_count": 32.
- *Followers_count* is the number of followers this account currently has. An example could be: "followers_count": 21.

- *Tweet* is the basic atomic building block of all things in Twitter. Users post tweets, also known more generically as "status updates." Tweets can be embedded, replied to, liked, unliked and deleted. Users can amplify the broadcast of tweets authored by other users by retweeting. Retweets can be distinguished from typical tweets by the existence of a "retweeted_status" attribute. This attribute contains a representation of the original tweet that was retweeted. Users can also unretweet a retweet they created by deleting their retweet. There are various properties [27] associated with the *Tweet* object which uniquely define and identify a tweet. Among these numerous properties associated with the *Tweet* object, we have used the following in our algorithm:
  - *Tweet_id* is the integer representation of the unique identifier for a tweet. Similar to *User_id*, it is recommended to use a signed 64 bit integer for storing this identifier. An example could be: "id":114749583439036416.
  - *Text* is the actual UTF-8 text of the status update. An example could be: "text": "Quebec values charter: is it a political game changer for the PQ? Bit.ly/189j36o".
  - *Created_on* refers to the UTC time when this tweet was created. An example is: "created_on":"Wed Aug 27 13:08:45 +0000 2008".

- *Retweet_count* is the number of times this tweet has been retweeted. An example is: "retweet_count":1585.

- *Entities* provide metadata and the additional contextual information about content posted on Twitter. There are four types of tweet entities, namely: hashtag, media, URLs, user_mentions [28].

  - *Hashtag* refers to the topics or keywords marked by a "#" symbol in a tweet, and it can be used to categorize messages by Twitter users. For Example, "hashtags":[{"indices": [32, 36],"text":"lol"}].
  - *Media* refers to the media elements uploaded with the tweet. For example, "media":[{"type":"photo","sizes":{"thumb":{"h":150,"resize":"crop","w":150}, "large":{"h":238,"resize":"fit","w":226}, "medium":{"h":238,"resize":"fit","w":226}, "small":{"h":238,"resize":"fit","w":226}}, "indices":[15,35],"url":"http:\/\/t.co\/rJC5Pxsu", "media_url":"http:\/\/p.twimg.com\/AZVLmp-CIAAbkyy.jpg", "display_url":"pic.twitter.com\/rJC5Pxsu","id":114080493040967680, "id_str":"114080493040967680","expanded_url": "http:\/\/twitter.com\/yunorno\/status\/114080493036773378\/photo\/1", "media_url_https":"https:\/\/p.twimg.com\/AZVLmp-CIAAbkyy.jpg"}].
  - *URLs* refer to the URLs included in the text of a tweet or within the textual fields of a *User* object. For example, "urls":[{"indices":[32,52],"url":"http:\/\/t.co\/IOwBrTZR",

"display_url":"youtube.com\/watch?v=oHg5SJ\u2026",

"expanded_url":"http:\/\/www.youtube.com\/watch?v=oHg5SJYRHA0"}].

- *User_mentions* represents other Twitter users mentioned in the text of the tweet. For Example,

  "user_mentions":[{"name":"TwitterAPI","indices":[4,15],

  "screen_name":"twitterapi", "id":6253282, "id_str":"6253282"}].

These are the most common concepts related to Twitter and its various data properties, and they are described here to give an understanding on the ranking factors we use in this work. These terms are also helpful to understand the Twitter API, which is essential for collecting the data for the later experiment.

3.2 System Architecture

Figure 3.1 shows the architecture model we use to implement our tweet re-ranking algorithm. A sleek multithreaded WPF-based GUI enables the user to input the query and provides results in the form of tweets listed in a ranked manner determined by the ranking algorithm. The Fetching module plays the most critical part in providing the fresh dataset available for processing. On runtime it collects data from Twitter based on user-keyword in a request-response manner with the help of Twitter API and the integration library. With Twitter imposing certain limits such as the number of requests per hour per account and the amount of data per request, a time lag was encountered towards the overall operation time.

A round-robin multi-account system was developed, which switches to the next available account as soon as the limit of the current account is reached; thus, we could always have an active account to perform further request-response operation with Twitter API. These limits are

imposed based on the time interval and are refreshed after a specific unit of time has elapsed. Thus, an exhausted account, after a certain period of time, becomes available. By making use of 20 authentication tokens from different accounts, we were able to create a system which always had an available account for operations.

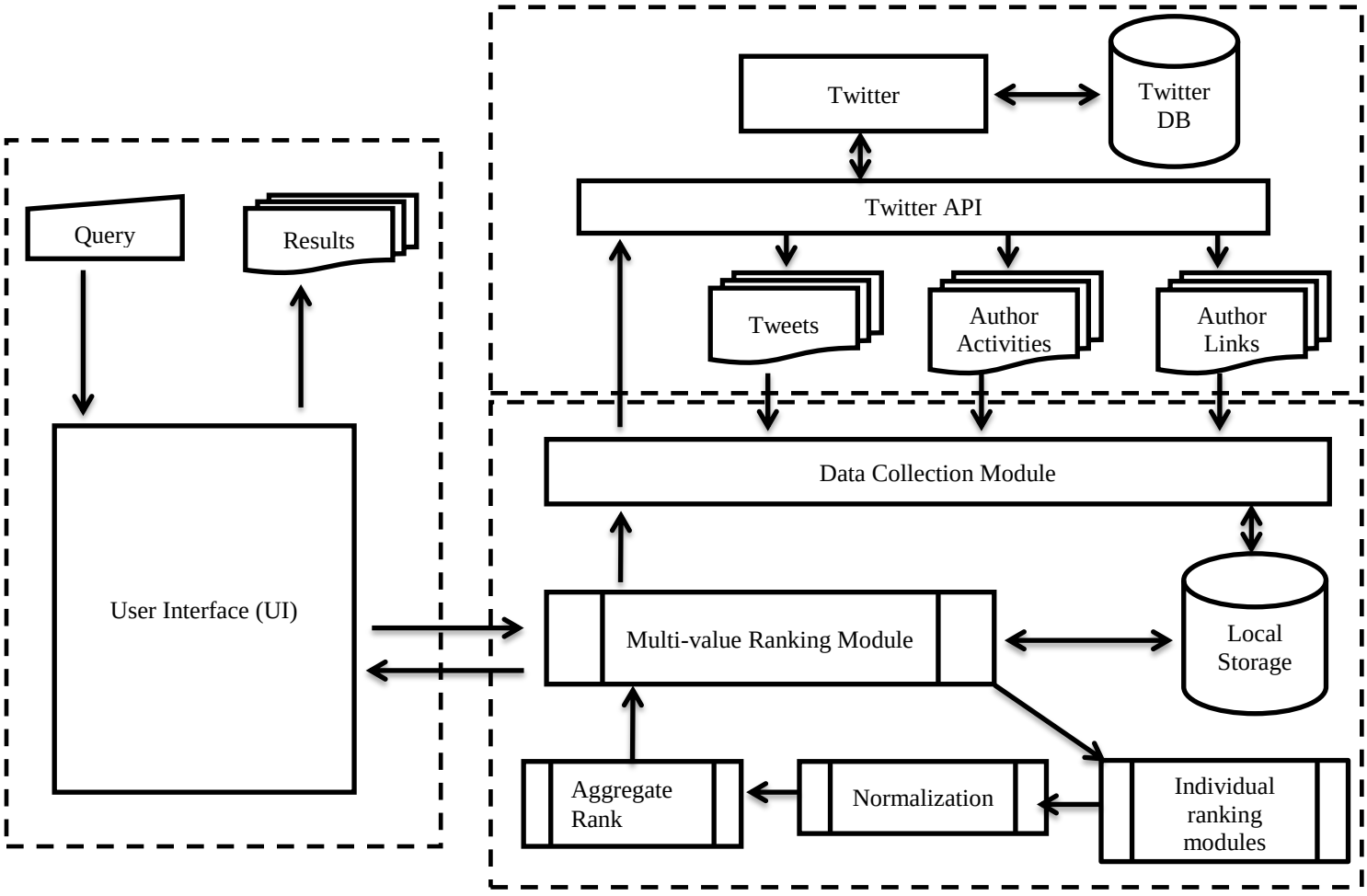


Figure 3.1. System Architecture

Although the problem of the operation limit has been tackled eliminating any waiting lag, there is still a considerable amount of time required to collect a complete dataset for one query. Hundreds of request-response operations are performed to complete one collection operation.

Each operation involves sending and receiving data over the internet and is affected by the network congestion and Twitter server load lag.

Having a static local dataset would completely eliminate any fetching time and would largely improve the rank computation time for a query; however, it would handicap the system to allow only those queries/keywords which have already been pre-fetched into static dataset to be processed and the tweets are not always fresh.

The Ranking module has two phases. Phase I, or the fetching phase, is responsible for collecting all the related information from Twitter. We are using the live data, not a pre-collected set of tweets. Live data is slow and restricted by limits set by Twitter but gives a wide range of keywords and data to test our algorithm.

In this step, multi-threading is done for background processing, for which we have to make an app on Twitter that provides a token and gives authentication for a JSON request. It interacts with the Twitter API library for authentication and data collection. It also performs internal cleaning and data filtering before storing the data into the database.

In general, Phase I performs the following tasks:

- Account authentication for API connectivity.
- Collection of tweets returned from Twitter based on the given query.
- Collection of list of friends and followers of users who are authors of the returned tweets.
- Collection of recent tweets of these users.

Data is freely available, but can be accessed only through authentication by using Twitter API and account tokens. The process is slow because data access has hourly limits and the rate of retrieval is also slow. When using application-only authentication, rate limits are determined globally for the entire application. If a method allows for 15 requests per limit window, then it

can make 15 requests per window on behalf of the application. More details can refer to [40]. Additionally, it depends on the network speed. So in general, phase I is slow. It could have taken hours, but we are using multiple account tokens and account switching, therefore, no waiting time is wasted. All the time taken in phase I is for the retrieval of the matching result.

Phase II, or the calculation phase, operates on the collected data, calculating nine ranking scores based on different factors, which are later normalized and aggregated to produce the final ranking score. This final rank is used to order the results presented to the user.

At each step of this phase data is read and written into a database. A number of re-ranking algorithms are used. Phase II performs the following tasks:

- Calculation of Tweet Impact Rank.
- Calculation of Tweet Recency Rank.
- Calculation of Tweet Content Relevancy Rank.
- Calculation of Tweet Content Uniqueness Rank.
- Calculation of Tweet Media Rank.
- Calculation of User Connectivity Rank.
- Calculation of User Activeness Rank.
- Calculation of User PageRank.
- Calculation of User Hub Authority Rank.
- Normalization of all above ranking scores.
- Aggregating the calculated ranking scores to produce the final rank.

Screenshots for interfaces to collect Twitter data and calculate the ranks are included in Appendix A.

3.3 Our Proposed Re-Ranking Algorithm

Twitter shows a user's tweets in a reversed sequential order, which is not always the best way. Twitter is crowded with different types of messages. Informative tweets might be displayed at the bottom and some irrelevant tweets might be at the top.

This study proposes a re-ranking approach based on various tweet and user related properties. The proposed approach calculates ranking scores for individual properties, which are then normalized and added up to calculate the overall ranking score. Then the tweets are ordered on the basis of their calculated overall scores and presented as results.

3.3.1 Algorithm Overview

When a user searches for some keywords, he hopes to see the most relevant results returned, which can put an end to his information seeking process. For real time systems like Twitter, time is an important factor to evaluate the recency of the information, which, in a way, is necessary to determine the relevancy of the information too.

The aim in developing this algorithm is to integrate the information quality and relevancy into the search results and in the meantime keep the recency information.

The overall rank of a tweet in the result list is determined by the combination of different values. The main intention here is to provide results as relevant as possible, as recent as possible, and as authoritative as possible, to a user from the set of available data.

In the algorithm we mainly consider factors from two aspects, one is tweet, and the other is user. Tweet related factors include:

- **Time** - Is the tweet recent?
- **Popularity** – Has the tweet been retweeted and, if so, how many times?

- **Content** – Is the tweet relevant to the query, unique and does it contain information sources?

User related factors include:

- **Activeness** – Is the author active over a given time period?
- **Connectivity** – Is the author well connected to other users?
- **PageRank/Hub Authority** - Is the author a source of authoritative information?

In this approach, we believe a tweet should be ranked on the basis of its own quality, or the quality of its source, that is, the author who has posted it.

The following are the main factors we consider for ordering the tweets.

Tweet-based factors

i) Impact of tweet, which is measured by the retweet count.

ii) Recency, which is measured by the posting time of the tweet.

iii) Content relevancy, which is the relevancy of the content of the tweet to the query.

iv) Uniqueness, which is to measure how unique the tweet is compared with other tweets.

v) Media richness, which is measured by the count of media, URL, and video embedded in a tweet.

User-based factors

i) Number of followers who are not mutually followed.

ii) Activeness, which is measured by the number of tweets posted by the user in a given time period.

iii) PageRank score calculated on the link graph of the user following relationship.

iv) Hub/authority score calculated on the link graph of the user following relationship.

Tweets are then rearranged on the basis of the calculated scores of the above nine factors.

3.3.2 Details of the Factors Considered in the Algorithm

There are nine calculation steps based on different factors; below, we will discuss each of them.

i) Tweet Impact Rank (TIR)

In this step we count the number of times a particular tweet is retweeted among the users on Twitter. We give the most retweeted tweet the highest ranking score as it is considered as the most popular one. For instance, if a tweet is retweeted 5 times, its Impact Rank value will be proportional to 5. If it is not retweeted at all its Impact Rank value will be 0. This formula is applicable only on tweets having retweet count > 0. The formula to measure TIR of the current tweet tw_i is given below,

$$TIR_i = \frac{RC_{max} - RC_{min}}{(RC_{max} - RC_i) + 1} \quad (3.1)$$

where RC_{max} is the maximum retweet count, RC_{min} is the minimum retweet count and RC_i is the retweet count for the current tweet tw_i.

ii) Tweet Recency Rank (TRR)

In this step, we are checking the time the tweet was posted; the most recently posted tweet will have the highest-ranking score. This increases the probability that the tweets ranked high by the algorithm must be fresh and recent on Twitter. Below we list the formula to calculate TRR,

$$TRR_i = \frac{t_i - t_{oldest}}{(t_{latest} - t_{oldest}) + 1} \quad (3.2)$$

where t_i is the posting time of the current tweet tw_i, t_{latest} is the posting time of the latest tweet, t_{oldest} is the posting time of the oldest tweet.

iii) Tweet Content Relevancy Rank (TCRR)

This refers to the relevancy of the tweet to the given query, that is, the similarity between the tweet and the query keywords using the vector space model [2]. Tweets having a high similarity score to the query will have high TCRR values. The following steps have been used to calculate the TCRR value:

- Each tweet is considered as a document.
- The Vector Space Model is then applied to calculate the similarity between each tweet and the query.
- Tweets with high similarity values will have high TCRR values and are considered more relevant.

The formula for Tweet Content relevancy rank (TCRR) is as follows,

$$TCRR_i = Sim(tw_i, q) = \frac{\sum_{j=1}^{N} W_{ji} W_{jq}}{\sqrt{\sum_{j=1}^{N} W^2{}_{ji}} \sqrt{\sum_{j=1}^{N} W^2{}_{jq}}} \quad (3.3)$$

where tw_i is the current tweet, q is the user query, N is the total number of terms, W_{ji} is the weight of term j in tw_i, W_{jq} is the weight of term j in query q.

iv) Tweet Content Uniqueness Rank (TCUR)

In this step we consider the uniqueness of the tweet compared to other related tweets on the query keyword. The uniqueness of the tweet is measured by how rarely its terms appear in

other tweets on the given query keyword. Below is the formula to calculate Tweet Content Uniqueness Rank (TCUR):

$$TCUR_i = \sum_{j=1}^{M_i} log \frac{K}{df_j} \quad (3.4)$$

where M_i is the number of unique terms in tweet tw_i, K is the total number of tweets returned from Twitter on the given query, df_j is the number of tweets that contain term j.

v) Tweet Media Entity Rank (TMER)

In this step, we check if the tweet is containing any media file, URL, any video, or any other media content. Tweets having any of those types of media contents will have a high TMER score as they can be more informative than others. TMER measures how many media entities are present in the tweet. Below is the formula to calculate TMER,

$$TMER_i = media\_count_i + URL\_count_i \quad (3.5)$$

where $media\_count_i$ measures the number of media entities in tw_i and URL_count_i measures the number of URLs in tw_i.

vi) User Connectivity Rank (UCR)

In this step we look for the number of followers of the user whom are not mutually followed. These users are considered more informative because they are sources of information, not just friends.

The internal friendships and connections of a user denote his connectivity strength. The more people who follow a user, the more his tweets will be accessed, and the greater his popularity will be. However, a trend of mutual following prevails on Twitter; people follow a user just because that user is following them, not for the sake of information. Thus, we count the

number of mutually exclusive followers who follow an individual, but the individual does not follow back. The UCR is calculated using the formula below:

$$UCR_i = |F2_i - (F1_i \cap F2_i)| \quad (3.6)$$

where UCR is the count of users which follow the current user, but the user doesn't follows them back. $F1_i$ is the set of friends of current user who posts tw_i i.e. whom the user follows, $F2_i$ is the set of followers of current user, i.e. who follow the user. And $(F1_i \cap F2_i)$ gives a set of users who are both friends and followers.

vii) User Activeness Rank (UAR)

In this step, we check whether the user is active on the topic, as the user might have not tweeted about the topic in the last few days. The users who are more active have a high UAR score. Using this method preference is not given to those users who are not active on the topic.

The activeness rank of a user denotes his activeness in posting tweets on a specified keyword over a time period. This factor is very useful in measuring the overall contribution of a user on a particular keyword during a time period. The formula below shows the calculation steps,

$$UAR_i = \sum_{w \in q} tweet\_count(w) \quad (3.7)$$

where, *w* is a keyword in query *q*, *tweet_count(w)* measures the number of tweets posted on keyword *w* by the author of tweet tw_i.

Twitter API has a feature using which we can specify user id for which tweets are needed to be retrieved on given keyword.

viii) User PageRank (UPR)

In this step, we calculate the PageRank of the user by considering the followers and friends of the user. The PageRank of a user is a measure of reachability of the user in the user's following relationship graph. That is, it indicates what the chances are that a visitor will end up on that user while following through other users. To calculate the PageRank a numerical weight is assigned to it. The user is then ranked on the basis of that weight. The formula for User PageRank (UPR) [3, 4] is shown below,

$$UPR_i = \frac{1-D}{M} + \frac{\sum_{\forall A_j, A_j \to A_i} UPR_j}{out\_link_j} \quad (3.8)$$

where *D* is the damping factor which is set to 0.85, *M* is the number of all users in the following graph, A_j is following A_i, the author of tw_i, out_link_j is the number of users $A_j$follows.

ix) User Hub Authority Rank (UHAR)

Here, we calculate the hub and authority scores [2] for a user. The UHAR rank of a user specifies whether the user is a hub following many users and is considered an information source; or is an authority linking with many users following them.

Below is the formula to calculate User hub authority rank (UHAR):

$$UHAR = \alpha . Auth_i + \beta . Hub_i \quad (3.9)$$

where, α, β is the weight on authority score of user who posted tw_i $Auth_i$, and hub score Hub_i. $Auth_i$ and Hub_i are calculated on the user following graph, and the formulas for $Auth_i$and Hub_i can refer to section 2.1.3 in chapter 2.

3.3.3 Final Ranking Function

Before we combine all the ranking scores together, we need to normalize them. Normalization is a process of rescaling numbers so that they fit into 0 to 1 scale. In order to achieve this scale, every number is divided by the highest number available.

$$N(v, V) = v/\max(V) \quad (3.10)$$

where *V* represents the set of all possible values for a certain variable, *v* represents a value from *V*, *max* is to get maximum value from the set.

After normalization, the final ranking score of a tweet is calculated as,

$$S_i = (TIR_i + TRR_i + TCRR_i + TCUR_i + TMER_i) * W_i + (UAR_{iu} + UCR_{iu} + UPR_{iu} + HAR_{iu}) * W_u \quad (3.11)$$

where W_i = Weight value for Tweet Ranks

W_u = Weight value for User Ranks

$W_i + W_u = 1$

Having weights on user related and tweet related scores gives the approach a flexibility of shifting/balancing importance between tweet factors and user factors by adjusting weight values. In the current implementation equal weights are assigned to user scores and tweet scores for the simplicity reason.

The algorithm and the actual code of one ranking score calculation – Tweet Impact Rank, is listed in Appendix B.

3.4 Summary

In this chapter, we have explained the Twitter related terms, the architecture of our system, as well as our proposed re-ranking algorithm. We also discussed, in detail, each factor considered in our proposed algorithm. We described all the formulas that are used in our proposed method. In the next chapter, we will discuss the system implementation and its results, as well as the user evaluation results.

CHAPTER 4

EXPERIMENTS

The main objective of our implementation and experiments is to test the working of the proposed approach and verify the accuracy of the results generated through a user evaluation.

4.1 Dataset

There is no standard benchmark dataset available to test ranking algorithms performed on Twitter data. So in this study, we decided to choose some queries and use Twitter API to fetch the relevant results returned from Twitter on each query, and then apply our proposed ranking algorithm to re-rank them and compare with the original ranked results through the user evaluation.

The sample test dataset contains tweet and user values for 20 general keywords/queries. These keywords were randomly chosen based on suggestions from various people. The dataset was collected by performing fetch operations for all specified queries. This required the use of Twitter API, C# API library and multiple twitter access accounts.

A list of 20 search queries were submitted to Twitter, and 200 tweets returned from Twitter on each query were collected; furthermore, all other tweet and user related data required by our algorithm were collected and stored into a local database for later processing and calculation. We chose these queries by asking some users to randomly give some topics they are interested in or have some knowledge of. The list of queries is:

1) Amitabh Bachchan
2) Apple

3) Barack Obama
4) Bank Of Canada
5) Blackberry
6) Boston Marathon
7) Cloud Computing
8) Cyber Security
9) FIFA World Cup
10) Global Warming
11) Health Canada
12) Human Rights
13) Immigration Canada
14) National Hockey League
15) Ryerson University
16) Russell Peters
17) Sachin Tendulkar
18) Solar Energy
19) Toronto Maple Leafs
20) US Tornadoes

4.2 Implementation

The discussed approach was implemented as a Windows based application, using C# WPF .Net technology. The configuration of the machine used to run the experiment is: Windows 7 Home premium, RAM 4GB, Hard Disk 596GB with a 64 bit operating system.

As illustrated in the system architecture shown in Figure 3.1, the application is divided logically and operationally into two modules, that is, a data fetching module and a re-ranking module. The data fetching module is responsible for communication with the Twitter service, account authentication, API request-response operation, and collecting the dataset for the specified query. And the following values are collected during the fetch operation:

- 200 tweets for each specified query.
- Related information for each tweet.
- Information of all authors of the listed tweets.

The re-ranking module is responsible for performing ranking calculations based on 9 tweet or user related factors, combining the ranking scores, and finally ordering the results in a ranked manner.

4.3 Experimental Results and Analysis

Running our program for 20 different queries presented us with 20 datasets, with tweets being calculated with ranking scores. In order to compare the re-ranking results with the original results, following the standard evaluation approach for the information retrieval research, when the benchmark dataset is not available, we need to have human evaluators to help us evaluate the relevancy of the results so as to measure the accuracy of each algorithm.

An online evaluation system was developed in C# ASP.net and SQL server. For each query, the top 20 tweets based on the original Twitter results and our algorithm results were mixed together without labeling their sources and then transferred to the user evaluation database and were linked to respective queries. A total of 20 different queries were loaded into the system,

with each having up to 40 linked tweets, which are classified as Twitter only, algorithm only, or common result. The classification is only known to us, but not to the evaluators. Also the duplicates are checked, so if there are overlapping tweets from the two result sets, only one copy is kept.

A total of 40 volunteer users from different locations and professions were invited to take our online user evaluation. They were requested to choose the ten tweets they considered the most relevant for a query. Each user was requested to perform the task for five queries out of the twenty queries. The selection of the query was left to the users based on whether they are familiar and comfortable with the query topic. In our evaluation system, we have implemented an adaptive ordering of queries so that the least selected queries are displayed on a higher position on the list in the hope that they have higher chance to be selected in the future.

A total of 40*5 = 200 unit user evaluation data was collected, with each unit containing evaluation results on 10 tweets. Based on these evaluations, points were allotted to either the original Twitter ranking or our algorithm. Twitter gets a point if a *twitter only* tweet is selected, algorithm gets a point if an *algorithm only* tweet is selected, and both get 1 point if a *common* tweet is selected. And the winner is decided based on the number of points earned. Suppose if a user selects 3 Twitter only tweets, 5 algorithm only tweets, and 2 common tweets, the score for this result will be Twitter Points = 3+2=5, Algorithm Points = 5+2=7. In this case, our algorithm wins because it has more points.

After taking all the user evaluation units into account, the final overall score was 1023 points for Twitter and 1205 points for our algorithm. This score indicates our approach as the winner based on our user evaluation results. Since this measurement is rather intuitive, we use other common ranking evaluation approaches for the comparison in the following sections.

4.3.1 Kendal Tau Comparison

Kendall tau correlation coefficient can be used to measure the difference between the two ranking results on the same dataset and is represented as "τ" in our later discussion. Its value varies from -1 to 1. This coefficient helps us calculate the similarity between the ranked datasets. The higher the value of "τ", the greater relationship is between the ranked datasets, whereas a smaller value represents a bigger difference between ranked datasets.

The formula to calculate the Kendall Tau is:

$$\tau = \frac{\#C - \#D}{\#C + \#D} \tag{4.1}$$

where #C represents the number of concordant pairs and #D represents the number of discordant pairs. If there are two pairs X_i, Y_i and x_j, y_j, concordant pairs are defined as:

$X_i > x_j$ and $Y_i > y_j$ OR $X_i < x_j$ and $Y_i < y_j$

The same relation must exist between each pair of elements; otherwise, the pair is the discordant pair.

In our proposed approach, we have Twitter ranked datasets and our proposed algorithm ranked datasets. The Kendall tau algorithm helps us check the similarity or dissimilarity between the two ranked datasets.

A sample query - cloud computing, was chosen and 20 tweets were obtained, giving two ranks for each tweet: - Twitter rank and algorithm rank. Their ranks are shown in Table 4.1.

Table 4.1. Ranking results from two algorithms for the sample query “Cloud Computing”

Tweets	Twitter Rank	Algorithm Rank
Tweet 1	1	3
Tweet 2	2	9
Tweet 3	3	5
Tweet 4	4	6
Tweet 5	5	8
Tweet 6	6	14
Tweet 7	7	4
Tweet 8	8	20
Tweet 9	9	18
Tweet 10	10	12
Tweet 11	11	11
Tweet 12	12	15
Tweet 13	13	19
Tweet 14	14	7
Tweet 15	15	1
Tweet 16	16	16
Tweet 17	17	10
Tweet 18	18	13
Tweet 19	19	17
Tweet 20	20	2

Using those ranks, we calculate the degree of similarity between the two ranked result sets using the Kendall tau algorithm. Based on Formula 4.1, for this example, we can get the following results:

$\#C$ = total number of concordant pairs = 110

$\#D$ = total number of discordant pairs = 80

τ = 110-80/110+80 = 30/190 = 0.157

Here, the τ is a positive value, which means the two ranking orders are somehow related, however, its small value indicates the similarity between the two is low.

We also did a significance test [29] to check whether both variables are statistically dependent. The formula is shown below,

$$Z = 3 * \tau\sqrt{n(n-1)}/\sqrt{2(2n+5)} \quad (4.2)$$

According to this formula, the value of *Z* is 0.973. Since the higher the value of *Z*, the more significant is the association between the two rankings, and here the value of *Z* is small, there is a less significant association between the two rankings. To check whether two variables are statistically dependent, we know that if Z >1.96 the dependency is more significant and if Z<1.96 it is less significant. Since, Z=0.973, it is two values are less significantly dependent and hence different. Table 4.2 shows the Kendall Tau values for all 20 queries.

Table 4.2. Kendall Tau values for 20 queries

Sr. No	Query	Kendall Tau Coefficient.
1.	Health Canada	0.147
2.	Solar Energy	0.031
3.	National Hockey League	0.073
4.	FIFA World Cup	0.178
5.	Sachin Tendulkar	0.063
6.	Russell Peters	0.252
7.	Apple	0.042
8.	Cloud Computing	0.157
9.	Cyber Security	0.189
10.	Toronto Maple Leafs	0.221
11.	Barack Obama	0.178
12.	Amitabh Bachchan	0.084
13.	Blackberry	0.031
14.	US Tornadoes	0.226
15.	Boston Marathon	0.105
16.	Immigration Canada	0.147
17.	Ryerson University	0.105
18.	Global Warming	0.147
19.	Human Rights	0.094
20.	Bank Of Canada	0.252

This table concludes that for all queries τ is positive but small hence values are somehow related but the similarity is low.

4.3.2 Precision of User Evaluation Results

Precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved [2]. Both precision and recall are therefore based on an understanding and measure of relevance. Since for the Twitter search, for each query, we don't have the ground truth of the complete relevant document set, in this experiment, we only consider the precision value. The formula below shows the definition of precision for a document retrieval problem.

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|} \quad (4.3)$$

In our case, Equation 4.3 can measure the ratio of the most relevant tweets over all the returned tweets for a given query. The relevancy judgment is given by the user who goes through the returned tweet list to find the ones which are considered most relevant to the query. And whether a tweet is relevant to a query or not is depending on the average opinion of all evaluators who are evaluating the query.

Based on this formula, we calculate the average precision value for 20 queries in our dataset with considering of all 40 users' evaluation. The average precision for Twitter is 0.274 and the average for our algorithm is 0.286. Our algorithm has a better precision value, however, the difference is very small. Therefore, we tried to remove some queries, or users, or both from our evaluation results if the evaluation results from those users or on those queries were largely different from the average values. They could be considered as outliers. After removing them, we got the following results.

For 30 users and 20 queries: Twitter: 0.254, algorithm: 0.310

For 40 users and 15 queries: Twitter: 0.253, algorithm: 0.317

For 30 users and 15 queries: Twitter: 0.244, algorithm: 0.331

Therefore, after removing outliers, the improvement from our algorithm over the original Twitter ranking is more obvious. Since in our experiment, there is no control on the quality of the user evaluation. Some users may not understand the query topic well so that they do not have good judgment on result relevancy, some users may not spend time on reading result tweets carefully to judge their relevancy, and some may accidentally pick some wrong options. By removing those evaluation data that are largely different from the rest, the remaining data could provide a better evaluation on the algorithm.

Figure 4.1-4.3 show more detailed comparison results on individual queries.

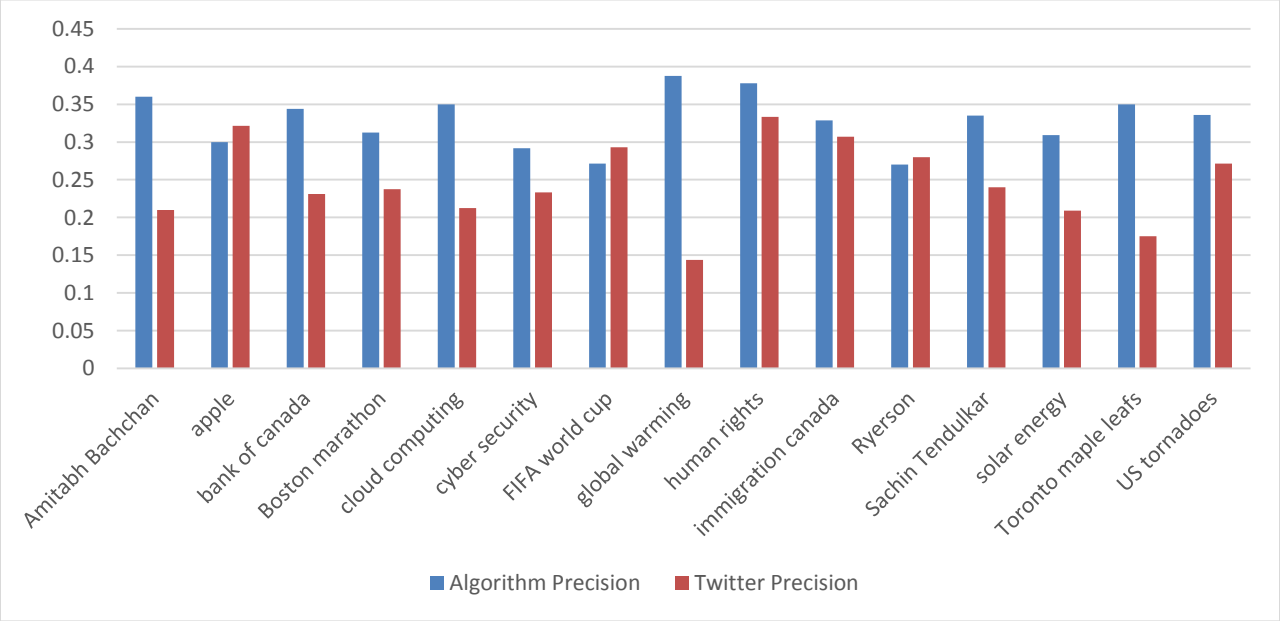


Figure 4.1 Graph Comparison for 30 users and 15 queries

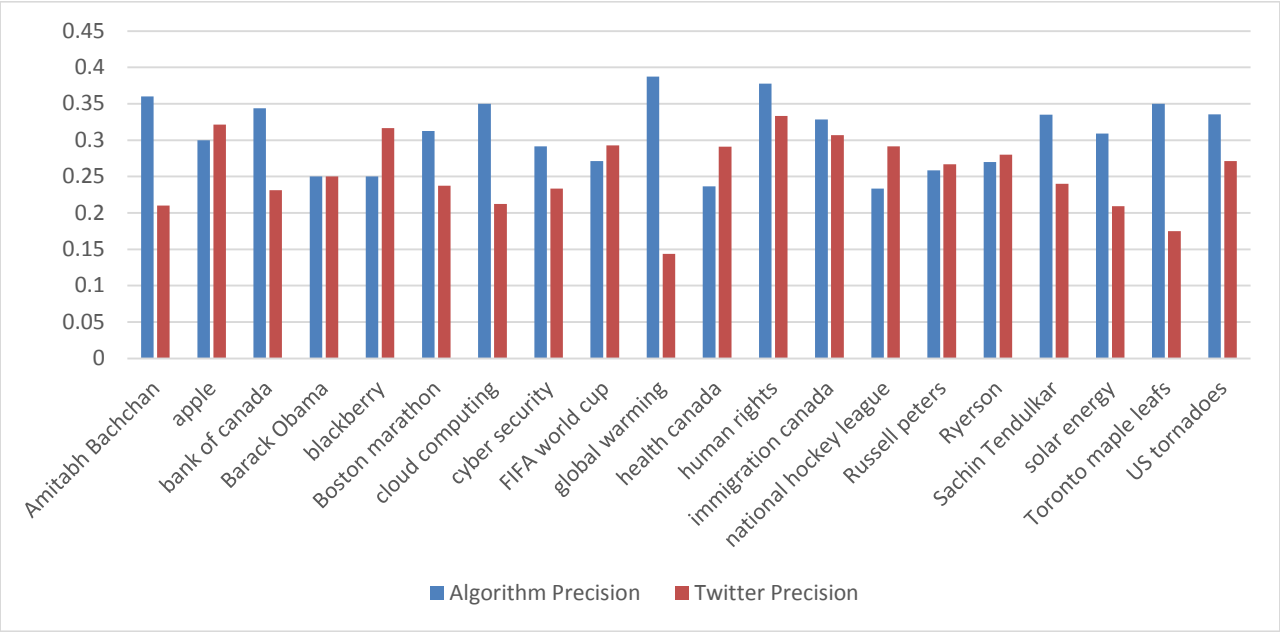


Figure 4.2 Graph Comparison for 30 users and 20 queries

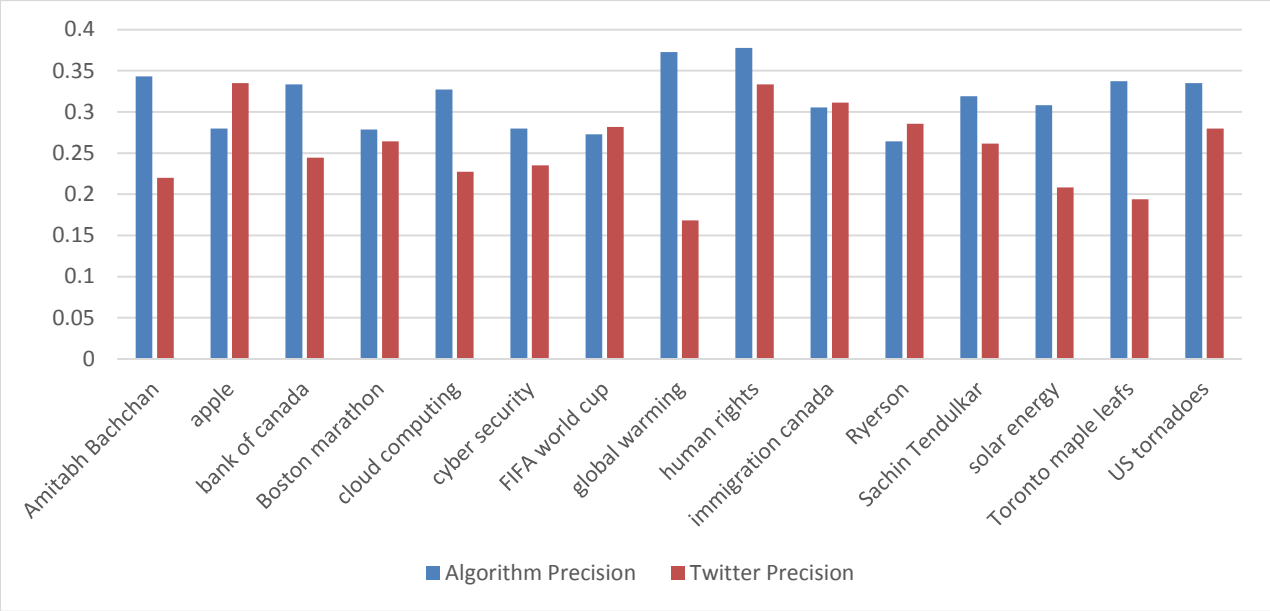


Figure 4.3 Graph Comparison for 40 users and 15 queries

These comparison charts are between precision values scored by the proposed algorithm and twitter results for each keyword/query. Blue colored (dark) bars represent algorithm values and red colored (light) bars represent twitter values. Three charts are drawn for results by varying the contribution count of users and queries. We could see that for the majority of the queries, the proposed algorithm achieves a higher precision than the original result. Among those queries that achieve a better result, the degree of improvement varies depending on the queries, some queries have a high percentage of precision increase, and the others have only a small increase.

Table 4.3 shows the comparison between the precision values from the original Twitter rank and the proposed algorithm and the percentage of the difference. It shows the average precision on all considered queries. It also shows the difference before and after we remove outliers. We could see that after outliers are removed, the precision increase is more obvious.

Table 4.3. Precision results

	PAA	**PAT**	**Diff%**
40u 20q	0.286	0.274	2.18%
30u 20q	0.310	0.254	9.98%
40u 15q	0.317	0.253	11.23%
30u 15q	0.331	0.244	15.12%

PAA = Average precision for algorithm

PAT = Average precision for Twitter

SUM = PAA+PAT

Difference % = ((PAA-PAT) / SUM)*100

The complete precision results are included in Appendix C.

4.4 SUMMARY

In this chapter, we explained our experiment design, the datasets we used, the user evaluation process, as well as some of the implementation details. We have evaluated and analyzed our algorithm and proved that our algorithm worked better than the original Twitter ranking results. We also performed Kendall tau comparison between the two ranked lists from our algorithm and from Twitter, and the result value is small, which shows a small degree of similarity between two ranking algorithms.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

Social feed search engines, such as twitter, respond to users' search queries by providing or displaying a collection of result-objects (tweets in this case) in a sorted or ranked manner mainly based on the content and recency of the objects. In this study, we looked at this problem from a broader perspective and consider a few important factors that play an important role in calculating the rank of the tweet, that is, the position in which it is displayed in the results.

The focus of this thesis is to enhance the result accuracy, relevancy and uniqueness by making use of a set of information available via tweets' properties. Our approach use both tweets' and authors' information and properties to generate a different set of ranks. This set of ranks for each tweet results in an overall rank, which is used as a deciding factor for the ordering of tweets in the result set.

In this study, we have not only taken a tweet and its properties into account but have also made use of characteristics and properties of its author. In other words, for us, not only is what is posted (the tweet) important, but also who has posted it (the author) is equally important. We have tried to cover all the important factors such as when the tweet was posted, how many times it has been retweeted, how relevant its content is to the query, how unique its content is among the tweet dataset, whether it contains any embedded links or media. Furthermore, many properties related to the author such as how many posts the author posted in the past few days (e.g., last month); how many users follow him without being followed back; how well he is

connected to other users; whether he is a good source of information; and whether he is someone who can lead us to a source of information. All of these are taken into account and included as part of our algorithm. The algorithm's accuracy has been proven by the user evaluation results.

Our algorithm not only covers a wide range of mentioned factors, but its structure also provides a flexible platform so that other factors and processing algorithms when and as available can be integrated into it and existing ones can be removed, replaced or updated.

There are mainly two contributions of this work:

- We proposed a re-ranking algorithm for Twitter search based on multiple features, including a few novel ones;
- Our re-ranking algorithm consider many different tweet related properties and user related properties, and the final ranking is a combination of all features.

5.2 Limitations of our proposed algorithm

There are some limitations in our proposed algorithm as listed below.

a. Individual factors are not tested separately.
b. Implementation is highly dependent on twitter API to fetch data. If twitter API does not work or when there is any problem, then our algorithm may not work.
c. Our result is highly depending on the original twitter result. If the original search result from Twitter is not good, ours may also not be good.
d. Efficiency could be a concern, because the re-ranking is performed on top of the twitter results, and all the data is collected on-the-fly, which could be time consuming.

5.3 Possible future directions

A few directions we would like to work on in the future include:

Firstly, for fast calculation we can analyse contribution of each algorithm involved and then accordingly remove or replace weakly performing ones. Also, we would like to make changes in slower algorithms to improve the speed but still keep the original logic intact.

We can also implement a cache mechanism to store frequently used data and possible implementation of step ranking and elimination so as to taper off the amount of data passed on to next part can also speed up the calculation.

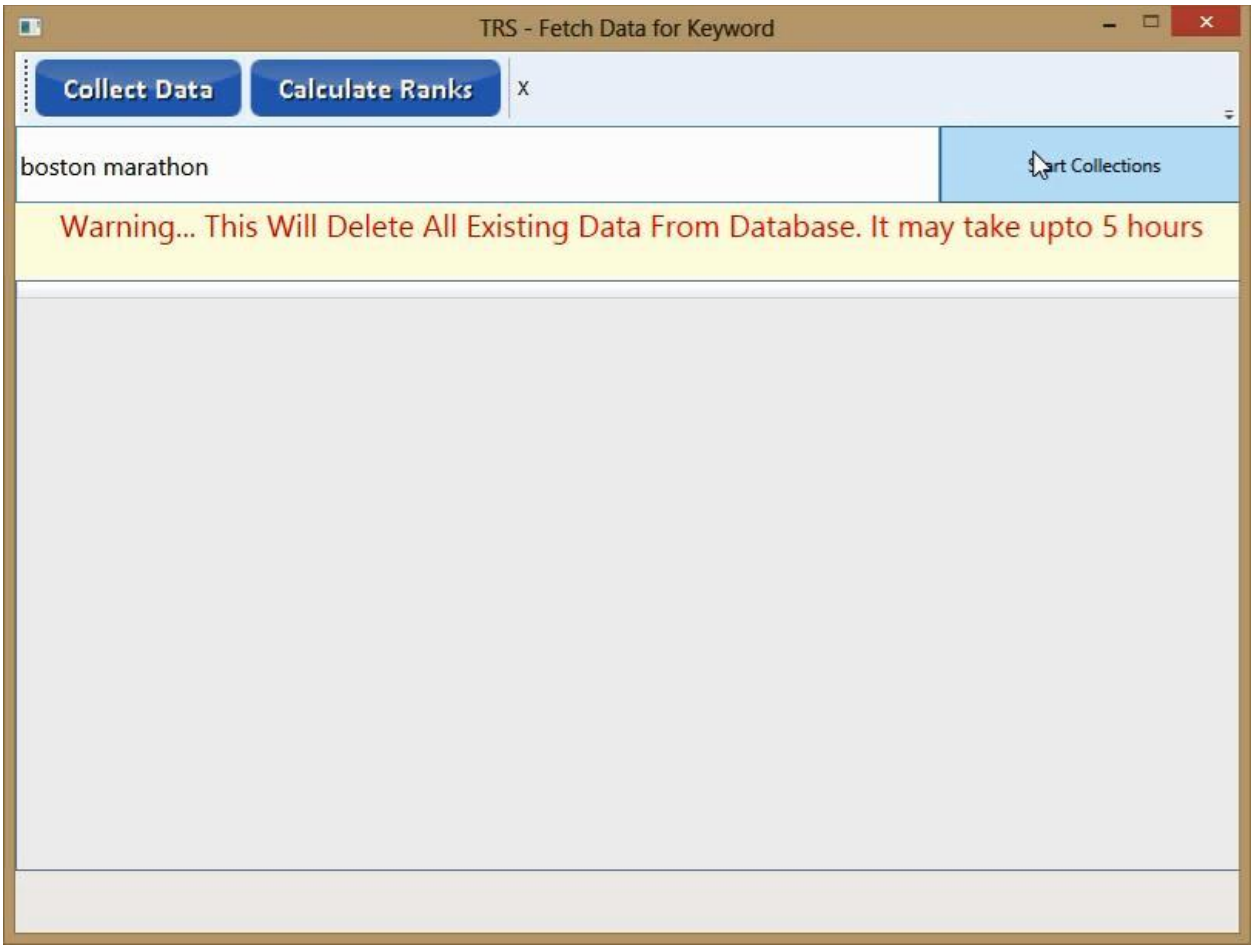
Secondly, dividing calculations in part by implementing precognition to allot a base rank as soon as tweet is posted can also give another factor to search on. Precognition base rank will be based on time. User activeness at that point and tweet content will help in this. And also direct access to data repository for unlimited time and faster access will help a lot. That is, it will reduce data fetch time, remove limit on number of tweets and values available, more data to be analyzed, so faster and improved ranks.

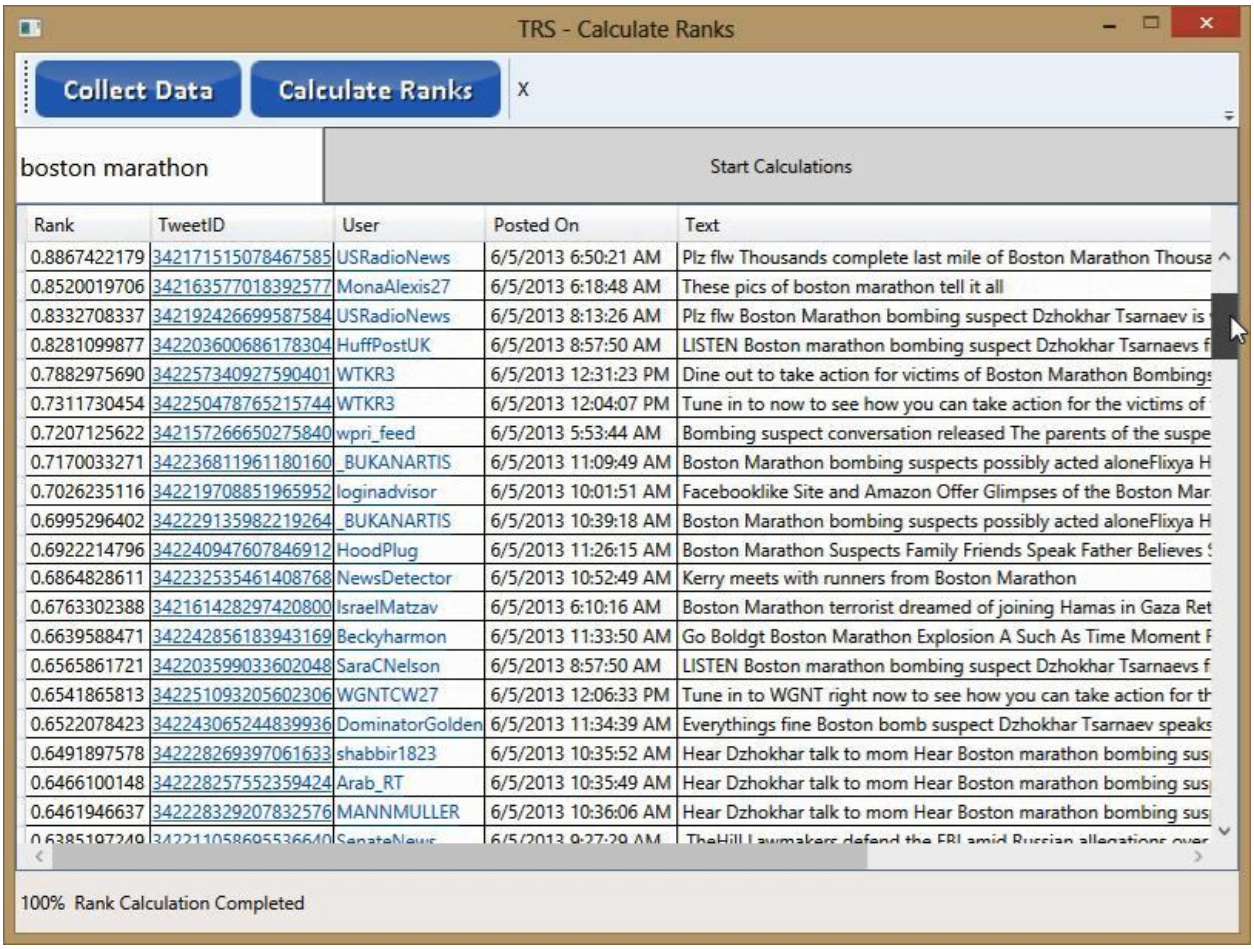
Thirdly, we can build the user following graph off-line, update it regularly, and save time for on-the-fly data collection.

Fourthly, we can apply the rank aggregation technique on the rankings so there is no need to calculate the content-based scores.

Finally, adding a learning module will be helpful to improve result accuracy and find the most effective factors for individual users.

APPENDIX A- Screenshot for user interface




TRS - Calculate Ranks
Collect Data
Calculate Ranks
X
boston marathon
Start Calculations
Rank
TweetID
User
Posted On
Text
0.8867422179 342171515078467585 USRadioNews 6/5/2013 6:50:21 AM Plz flw Thousands complete last mile of Boston Marathon Thousa
0.8520019706 342163577018392577 MonaAlexis27 6/5/2013 6:18:48 AM These pics of boston marathon tell it all
0.8332708337 342192426699587584 USRadioNews 6/5/2013 8:13:26 AM Plz flw Boston Marathon bombing suspect Dzhokhar Tsarnaev is
0.8281099877 342203600686178304 HuffPostUK 6/5/2013 8:57:50 AM LISTEN Boston marathon bombing suspect Dzhokhar Tsarnaevs f
0.7882975690 342257340927590401 WTKR3 6/5/2013 12:31:23 PM Dine out to take action for victims of Boston Marathon Bombings
0.7311730454 342250478765215744 WTKR3 6/5/2013 12:04:07 PM Tune in to now to see how you can take action for the victims of
0.7207125622 342157266650275840 wpri_feed 6/5/2013 5:53:44 AM Bombing suspect conversation released The parents of the suspe
0.7170033271 342236811961180160 _BUKANARTIS 6/5/2013 11:09:49 AM Boston Marathon bombing suspects possibly acted aloneFlixya H
0.7026235116 342219708851965952 loginadvisor 6/5/2013 10:01:51 AM Facebooklike Site and Amazon Offer Glimpses of the Boston Mar
0.6995296402 342229135982219264 _BUKANARTIS 6/5/2013 10:39:18 AM Boston Marathon bombing suspects possibly acted aloneFlixya H
0.6922214796 342240947607846912 HoodPlug 6/5/2013 11:26:15 AM Boston Marathon Suspects Family Friends Speak Father Believes
0.6864828611 342232535461408768 NewsDetector 6/5/2013 10:52:49 AM Kerry meets with runners from Boston Marathon
0.6763302388 342161428297420800 IsraelMatzav 6/5/2013 6:10:16 AM Boston Marathon terrorist dreamed of joining Hamas in Gaza Ret
0.6639588471 342242856183943169 Beckyharmon 6/5/2013 11:33:50 AM Go Boldgt Boston Marathon Explosion A Such As Time Moment F
0.6565861721 342203599033602048 SaraCNelson 6/5/2013 8:57:50 AM LISTEN Boston marathon bombing suspect Dzhokhar Tsarnaevs f
0.6541865813 342251093205602306 WGNTCW27 6/5/2013 12:06:33 PM Tune in to WGNT right now to see how you can take action for th
0.6522078423 342243065244839936 DominatorGolden 6/5/2013 11:34:39 AM Everythings fine Boston bomb suspect Dzhokhar Tsarnaev speaks
0.6491897578 342228269397061633 shabbir1823 6/5/2013 10:35:52 AM Hear Dzhokhar talk to mom Hear Boston marathon bombing sus
0.6466100148 342228257552359424 Arab_RT 6/5/2013 10:35:49 AM Hear Dzhokhar talk to mom Hear Boston marathon bombing sus
0.6461946637 342228329207832576 MANNMULLER 6/5/2013 10:36:06 AM Hear Dzhokhar talk to mom Hear Boston marathon bombing sus
100% Rank Calculation Completed

APPENDIX B – CODE FOR INDIVIDUAL RANK CALCULATION

Tweet Impact Rank

```
ObjectSet<tStatus> tTweets = oTwitterEntity.tStatus;
decimal maxRC = tTweets.Max(t => t.retweetcount).Value; // get max retweet count
int? iMinRetweetCount = tTweets.Where(t => t.retweetcount > 0).Min(t => t.retweetcount);
decimal minRC = iMinRetweetCount.HasValue ? iMinRetweetCount.Value : 0; // get minimum retweet count non zero
foreach (tStatus tTs in tTweets)
{
 if (maxRC > minRC)
    tTs.impactrank = (maxRC - minRC) / ((maxRC - tTs.retweetcount) + 1); // calculate impact rank for each tweet
else
   tTs.impactrank = 0;
}
  oTwitterEntity.SaveChanges();
```

Tweet Recency Rank

```
ObjectSet<tStatus> tTweets = oTwitterEntity.tStatus;
        DateTime dtNow = DateTime.Now.ToUniversalTime();
          decimal minME = (decimal)(dtNow - tTweets.Max(t => t.createdon).Value).TotalMinutes; // get minimum minutes elapsed from most recent tweet
          decimal maxME = (decimal)(dtNow - tTweets.Min(t => t.createdon).Value).TotalMinutes; // get maximum minutes elapsed from least recent tweet
        foreach (tStatus tTs in tTweets)
        {
                decimal iMinElapsed = (dtNow - tTs.createdon.Value).Minutes; // get minutes elapsed for current tweet
                 tTs.recencyrank = (maxME - iMinElapsed) / (maxME - minME) + 1; // calculate recency rank.
               }
        oTwitterEntity.SaveChanges();
```

Tweet Content Relevancy Rank

```
ObjectSet<tStatus> tTweets = oTwitterEntity.tStatus;
        Dictionary<string, string> docs = new Dictionary<string, string>();
        foreach (tStatus tTs in tTweets)
        {
           docs.Add(tTs.tweetid, tTs.text); // add each tweet as document to document dictionary
        }

         Dictionary<string, double> vResult = VectorSpaceModel.Calculate(docs, sKeyword); // run VSM
                                                                    algorithm on docs n keyword, calculate vsm value for each tweet

        foreach (tStatus tTs in tTweets)
        {
           if (vResult.Keys.Contains(tTs.tweetid))
           {
              tTs.relevancyrank = (decimal)vResult[tTs.tweetid]; // update relevancy rank for each tweet
           }
        }
        oTwitterEntity.SaveChanges();
```

Tweet Uniqueness Rank

```
ObjectSet<tStatus> tTweets = oTwitterEntity.tStatus;

        Dictionary<string, string> docs = new Dictionary<string, string>();
        foreach (tStatus tTs in tTweets)
        {
           docs.Add(tTs.tweetid, tTs.text); // add each tweet as document to document dictionary
        }

             Dictionary<string, double> vResult = Uniqueness.Calculate(docs); // run Uniqueness
                                                                    algorithm on docs, calculate uniqueness value for each tweet

        foreach (tStatus tTs in tTweets)
        {
           if (vResult.Keys.Contains(tTs.tweetid))
           {
                    tTs.uniquenessrank = (decimal)vResult[tTs.tweetid]; // update relevancy rank for
                                                                                                each tweet
           }
        }
        oTwitterEntity.SaveChanges();
```

Tweet Media Entity Rank

```
ObjectSet<tStatus> tTweets = oTwitterEntity.tStatus;
        foreach (tStatus tTs in tTweets)
        {
            tTs.entityrank = int.Parse(tTs.entitycount.Split('-')[1]) + int.Parse(tTs.entitycount.Split('-')[3]);
                                                                    //0-hash,1-media,2-mention,3-url
        }
        oTwitterEntity.SaveChanges();
```

User Connectivity Rank

```
ObjectSet<tUser> tUsers = oTwitterEntity.tUser;
        foreach (tUser tTu in tUsers)
        {
           try
           {

              // list followers
              ListFollowerIdsOfOptions oFolOp = new ListFollowerIdsOfOptions() { UserId =
Convert.ToInt64(tTu.userid) };
              List<long> uFollowers = new List<long>();
              TwitterCursorList<long> uFollowerIds;
              do
              {
                 uFollowerIds = oTwitterService.ListFollowerIdsOf(oFolOp);
                 uFollowers.AddRange(uFollowerIds);
                 oFolOp.Cursor = uFollowerIds.NextCursor;
                 CheckLimit();
              }
              while (uFollowerIds.NextCursor != 0 && uFollowers.Count < 5000);
              tTu.followers = string.Join(",", uFollowers);

              //list friends
              ListFriendIdsOfOptions oFriOp = new ListFriendIdsOfOptions() { UserId =
Convert.ToInt64(tTu.userid) };
              List<long> uFriends = new List<long>();
              TwitterCursorList<long> uFriendIds;
              do
              {
                 uFriendIds = oTwitterService.ListFriendIdsOf(oFriOp);
                 uFriends.AddRange(uFriendIds);
```

```
            oFriOp.Cursor = uFriendIds.NextCursor;
            CheckLimit();
          }
          while (uFriendIds.NextCursor != 0 && uFriends.Count < 5000);
          tTu.friends = string.Join(",", uFriends);

          tTu.connectivityrank = uFollowers.Where(u => !uFriends.Contains(u)).Count();

          oTwitterEntity.SaveChanges();
        }
        catch { }
      }
```

User Activeness Rank

```
string sKeywordTemplate = sKeyword + " from:{0} +exclude:retweets";
      ObjectSet<tUser> tUsers = oTwitterEntity.tUser;
      foreach (tUser tTu in tUsers)
      {
        try
        {
          sKeyword = string.Format(sKeywordTemplate, tTu.screenname);
          SearchOptions oSO = new SearchOptions() { Q = sKeyword, Count = 100, Resulttype =
TwitterSearchResultType.Recent, Lang = "en" };
          IEnumerable<TwitterStatus> tResults = oTwitterService.Search(oSO).Statuses;
          tTu.activenessrank = tResults.Count();
          oTwitterEntity.SaveChanges();
          CheckLimit();
        }
        catch { }
      }
```

User Page Rank

```
ObjectSet<tUser> tUsers = oTwitterEntity.tUser;
      int iUserCount = tUsers.Count();

      foreach (tUser tp in tUsers)
      {
        tp.pagerank = (decimal)1 / iUserCount;
      }
      oTwitterEntity.SaveChanges();
```

```
        decimal dDampingFactor = new decimal(0.85);

        for (int iIteration = 0; iIteration < 10; iIteration++)
        {
           foreach (tUser tp in tUsers)
           {
              decimal dLinkedPr = 0;

              IEnumerable<tUser> myFollowers = tUsers.Where(u => u.listedfriends.Contains(tp.userid));
              foreach (tUser myFollower in myFollowers)
              {
                 int iLinksto = myFollower.listedfriends.Split(new char[] { ',' },
StringSplitOptions.RemoveEmptyEntries).Count();
                 iLinksto = iLinksto <= 0 ? 1 : iLinksto;
                 dLinkedPr += myFollower.pagerank.Value / iLinksto;
              }
              tp.pagerank = (1 - dDampingFactor) / iUserCount + dDampingFactor * dLinkedPr;
           }
        }

        oTwitterEntity.SaveChanges();
```

User Hub Authority Rank

```
ObjectSet<tUser> tUsers = oTwitterEntity.tUser;

        foreach (tUser tu in tUsers)
        {
           tu.hubrank = 1;
           tu.authorityrank = 1;
        }

        for (int iIteration = 0; iIteration < 10; iIteration++)
        {
           double dNorm = 0;

           // authority rank
           foreach (tUser tTu in tUsers)
           {
              tTu.authorityrank = 0;
              tUser[] myFollowers = tUsers.Where(u => u.listedfriends.Contains(tTu.userid)).ToArray();

              foreach (tUser tu in myFollowers)
              {
                 tTu.authorityrank += tu.hubrank;
              }
              dNorm += (double)(tTu.authorityrank * tTu.authorityrank);
           }
           dNorm = Math.Sqrt(dNorm);
```

```
        if (dNorm > 0)
            foreach (tUser tTu in tUsers)
            {
                tTu.authorityrank /= (decimal)dNorm;
            }


        //hub rank
        dNorm = 0;
        foreach (tUser tTu in tUsers)
        {
            tTu.hubrank = 0;
            tUser[] myFriends = tUsers.Where(u => tTu.listedfriends.Contains(u.userid)).ToArray();
            Array.ForEach(myFriends, delegate(tUser tu)
            {
                tTu.hubrank += tu.authorityrank;
            });
            dNorm += (double)(tTu.hubrank * tTu.hubrank);
        }
        dNorm = Math.Sqrt(dNorm);
        if (dNorm > 0)
            foreach (tUser tTu in tUsers)
            {
                tTu.hubrank /= (decimal)dNorm;
            }
    }

    oTwitterEntity.SaveChanges();
```

Normalization

```
ObjectSet<tStatus> tTweets = oTwitterEntity.tStatus;
    decimal dMaxEntityRank = tTweets.Max(d => d.entityrank).Value;
    decimal dMaxImpactRank = tTweets.Max(d => d.impactrank).Value;
    decimal dMaxRecencyRank = tTweets.Max(d => d.recencyrank).Value;
    decimal dMaxRelevancyRank = tTweets.Max(d => d.relevancyrank).Value;
    decimal dMaxUniquenessRank = tTweets.Max(d => d.uniquenessrank).Value;
    foreach (tStatus ts in tTweets)
    {
        if (dMaxImpactRank > 0)
            ts.impactrank /= dMaxImpactRank;

        if (dMaxRecencyRank > 0)
            ts.recencyrank /= dMaxRecencyRank;

        if (dMaxRelevancyRank > 0)
            ts.relevancyrank /= dMaxRelevancyRank;
```

```
    if (dMaxUniquenessRank > 0)
        ts.uniquenessrank /= dMaxUniquenessRank;

    if (dMaxEntityRank > 0)
        ts.entityrank /= dMaxEntityRank;
}
oTwitterEntity.SaveChanges();

ObjectSet<tUser> tUsers = oTwitterEntity.tUser;
decimal dMaxActivenessRank = tUsers.Max(d => d.activenessrank).Value;
decimal dMaxAuthorityRank = tUsers.Max(d => d.authorityrank).Value;
decimal dMaxHubRank = tUsers.Max(d => d.hubrank).Value;
decimal dMaxConnectivityRank = tUsers.Max(d => d.connectivityrank).Value;
decimal dMaxPageRank = tUsers.Max(d => d.pagerank).Value;

foreach (tUser tu in tUsers)
{
    if (dMaxConnectivityRank > 0)
        tu.connectivityrank /= dMaxConnectivityRank;

    if (dMaxActivenessRank > 0)
        tu.activenessrank /= dMaxActivenessRank;

    if (dMaxPageRank > 0)
        tu.pagerank /= dMaxPageRank;

    if (dMaxAuthorityRank > 0)
        tu.authorityrank /= dMaxAuthorityRank;

    if (dMaxHubRank > 0)
        tu.hubrank /= dMaxHubRank;
}

oTwitterEntity.SaveChanges();
```

APPENDIX C - FOR SURVEY DATA

FOR 40 USERS AND 15 QUERIES:

Query	Algorithm Precision	Twitter Precision
cloud computing	0.3273	0.2273
solar energy	0.3083	0.2083
cyber security	0.28	0.235
Toronto maple leafs	0.3375	0.1938
FIFA world cup	0.2727	0.2818
Sachin Tendulkar	0.3192	0.2615
Amitabh Bachchan	0.3433	0.22
apple	0.28	0.335
bank of canada	0.3333	0.2444
US tornadoes	0.335	0.28
global warming	0.3727	0.1682
immigration canada	0.3056	0.3111
human rights	0.3778	0.3333
Boston marathon	0.2786	0.2643
Ryerson	0.2643	0.2857

FOR 30 USERS 20 QUERIES:

Query	Algorithm Precision	Twitter Precision
health canada	0.2364	0.2909
cloud computing	0.35	0.28
solar energy	0.3091	0.28
cyber security	0.2917	0.28
national hockey league	0.2333	0.28
Toronto maple leafs	0.35	0.28
FIFA world cup	0.2714	0.28
Barack Obama	0.25	0.28
Sachin Tendulkar	0.335	0.28
Amitabh Bachchan	0.36	0.28
Russell peters	0.2583	0.28
blackberry	0.25	0.28
apple	0.3	0.28
bank of canada	0.3438	0.28
US tornadoes	0.3357	0.28
global warming	0.3875	0.28
immigration canada	0.3286	0.28
human rights	0.3778	0.28
Boston marathon	0.3125	0.28
Ryerson	0.27	0.28

FOR 30 USERS AND 15 QUERIES:

Query	Algorithm Precision	Twitter Precision
cloud computing	0.35	0.2125
solar energy	0.3091	0.2091
cyber security	0.2917	0.2333
Toronto maple leafs	0.35	0.175
FIFA world cup	0.2714	0.2929
Sachin Tendulkar	0.335	0.24
Amitabh Bachchan	0.36	0.21
apple	0.3	0.3214
bank of canada	0.3438	0.2313
US tornadoes	0.3357	0.2714
global warming	0.3875	0.1438
immigration canada	0.3286	0.3071
human rights	0.3778	0.3333
Boston marathon	0.3125	0.2375
Ryerson	0.27	0.28

REFERENCES

[1] Busch, M., Gade, K., Larson, B., Lok, P., Luckenbill, S., and Lin, J.: Earlybird: Real- Time Search at Twitter. In: Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, pp. 1360-1369 (2012)

[2] Yates, R.B., and Neto, B.R.: Modern Information Retrieval (the concepts and technology behind search) (2011)

[3] Brin, S., and Page, L.: The anatomy of a large-scale hypertextual Web search engine. In: Journal of the Computer Networks and ISDN Systems, Vol. 30 Issue 1-7, pp. 107-117 (1998)

[4] L. Page, S. Brin, R. Motwani, and T. Winograd.: The PageRank Citation Ranking: Bringing Order to the Web, Technical Report 1999-66, Stanford University (1998)

[5] Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: Journal of the ACM, Vol. 46 Issue 5, pp. 604-632 (1999)

[6] Pal, A., and Counts, S.: Identifying Topical Authorities in Microblogs. In: Web Services Distributed Management, pp. 45-54 (2011)

[7] Castillo, C., Mendoza, M., and Poblete, B.: Information Credibility on Twitter. In: World Wide Web – Session: Information Credibility, pp. 675-684 (2011)

[8] Bakshy, E., Hofman, J.M., Mason, W.A., and Watts, D.J.: Everyone's an Influencer: Quantifying Influence on Twitter. In: Web Services Distributed Management, pp. 65-74 (2011)

[9] Yang, M.C., Lee, J.T., Lee, S.W., and Rim, H.C.: Finding Interesting Posts in Twitter Based on Retweet Graph Analysis. In: Proceedings of the 35th international conference on Research and development in information retrieval. pp. 1073-1074 (2012)

[10] Meeyoung, C., Haddadi, H., Benevenuto, F., and Gummadi, K.P.: Measuring User Influence in Twitter: The Million Follower Fallacy. In: Proceedings of 4th International Conference on Weblogs and Social Media (2010)

[11] Duan, Y., Jiang, L., Qin, T., Zhou, M., and Y.Shum, H.: An Empirical Study on Learning to Rank of Tweets. In: Proceedings of the 23rd International Conference on Computational Linguistics pp. 295-303 (2010)

[12] Zhang, X., He, B., Luo, T., and Li, B.: Query-biased learning to Rank for Real-time Twitter Search. In: Proceedings of the 21st ACM international conference on Information and knowledge management, pp. 1915-1919 (2012)

[13] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC 2011 microblog track. In TREC, 2011.

[14] Sarma, A.D., Sarma, A.D., Gollapudi, S., and Panigrahy, R.: Ranking Mechanisms in Twitter-like Forums: In: Proceedings of the third ACM international conference on Web search and data mining, pp. 21-30 (2010)

[15] Shoutvelocity. http://shoutvelocity.com.

[16] Teevan, J., Ramage, D., and Morris, M.R.: #TwitterSearch: A Comparison of Microblog Search and Web Search. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp 35-44 (2011)

[17] Shen, k., Wu, J., Zhang, Y., Han, Y., Yang, X., Song, L., and Gu, X.: Reorder's User's Tweets. In ACM Transactions on Intelligent Systems and Technology, Vol. 4, No. 1, Article 6 (January 2013)

[18] Feng, W., and Wang, J.: Retweet or not? Personalized Tweet Re-ranking. In: Proceedings of the sixth ACM international conference on Web search and data mining, pp. 577-586 (2013)

[19] Chen, K., Chen, T., Zheng, G., Jin, O., Yao, E., and Yu, Y.: Collaborative Personalized Tweet Recommendation. In: Proceedings of the 35th international Special Interest Group of Information Retrieval conference on Research and development in information retrieval, pp. 661-670 (2012)

[20] Chaoji, V., Ranu, S., Rastogi, R., and Bhatt, R.: Recommendations to Boost Content Spread in Social Networks. In: Session: Information Diffusion in Social Networks, Proceedings of the 21st international conference on World Wide Web, pp. 529-538 (2012)

[21] Phelan, O., McCarthy, K., Bennett, M., and Smyth, B.: On using the Real-time Web for News Recommendation & Discovery. In: Proceedings of the 20th international conference companion on World Wide Web, pp. 103-104 (2011)

[22] Chen, J., Nairn, R., and Chi, Ed H.: Speak Little and Well: Recommending Conversations in Online Social Streams. In: Proceedings of the Special Interest Group on Computer Human Interaction Conference on Human Factors in Computing Systems, Session: Twitter Systems, pp. 217-226 (2011)

[23] Lin, J., and Efron, M.: Temporal Relevance Profiles for Tweet Search. In: SIGIR Workshop of Time Aware Information Access (2013)

[24] I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. Overview of the TREC-2011 Microblog Track. In: Proceedings of the Twentieth Text Retrieval Conference, (2011)

[25] Asadi, N., and Lin, J.: Document Vector Representation for Feature Extraction in Multi-Stage Document Ranking. In: Information Retrieval, Vol. 16, Issue 6, pp. 747-768 (December 2013)

[26] https://dev.twitter.com/docs/platform-objects/users, last accessed on 15 Aug 2013

[27] https://dev.twitter.com/docs/platform-objects/tweets, last accessed on 15 Aug 2013

[28] https://dev.twitter.com/docs/platform-objects/entities, last accessed on 15 Aug 2013

[29] Helsel, D. R., and Hirsch, R. M.: Statistical Methods in Water Resources. Chapter-12(12.4.1), pp. 338-340, (2002)

[30] Kwak, H., Lee, C., Park, H., and Moon, S.: What is Twitter, a Social Network or a News Media? In: Proceedings of the 19th international conference on World Wide Web, pp. 591-600 (2010)

[31] Tinati, R., Carr, L., Hall, W., and Bentwood, J.: Identifying Communicator Roles in Twitter. In: Proceedings of the 21st international conference companion on World Wide Web, pp. 1161-1168 (2012)

[32] Bourke, S., O'Mahony, M.P., Rafter, R., and Smyth, B: Ranking in Information Streams. In: Proceedings of the International Conference on Intelligent User Interfaces companion, pp. 99-100 (2013)

[33] Asadi, N., and Lin, J.: Fast Candidate Generation for Real-Time Tweet Search with Bloom Filter Chains. In: ACM Transactions on Information Systems, Vol. 31, No. 3, Article 13 (July 2013)

[34] Safar, M., Farahat, H., and Mahdi, K.: Analysis of Dynamic Social Network: E-mail Messages Exchange Network. In: Information Integration and Web-based Applications &

Services, In: Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services, pp. 41-48 (2009)

[35] Besmer, A., Lipford, H.R., Shehab, M., and Cheek, G.: Social Applications: Exploring A More Secure Framework. In: Proceedings of the 5th Symposium on Usable Privacy and Security Article No. 2, (2009)

[36] Schneider. F., Feldmann, A., Krishnamurthy, B., and Willinger, W.: Understanding Online Social Network Usage from a Network Perspective. In: Proceedings of the 9th ACM Special Interest Group on Data Communications conference on Internet measurement conference, pp. 35-48 (2009)

[37] Oskouei, R.J.: The Role of Social Networks on Female Students Activities. Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India, Article No. 26 (2010)

[38] Morris, M.R., Teevan, J., and Panovich, K.: What Do People Ask Their Social Networks, and Why? A Survey Study of Status Message Q&A Behavior. In: Proceedings of the Special Interest Group on Computer Human Interaction Conference on Human Factors in Computing Systems, pp. 1739-1748 (2010)

[39] Lee, D.H., and Brusilovsky, P.: Improving Recommendations Using WatchingNetworks a Social Tagging System. In: Proceedings of the iConference, pp 33-39 (2011)

[40] https://dev.twitter.com/docs/rate-limiting/1.1, last accessed on 15 Aug 2013