SAPIENZA
UNIVERSITÀ DI ROMA

Network-Aware Recommendations in Online Social Networks

Department of Computer and System Sciences Antonio Ruberti

Dottorato di Ricerca in Computer Science and Engineering – XXVIII Ciclo

Candidate

Noor Aldeen Alawad
ID number 1568492

Thesis Advisor

Prof. Stefano Leonardi

Co-Advisor

Dr. Aris Anagnostopoulos

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science and Engineering

September 2016

Thesis defended on 20 February 2017
in front of a Board of Examiners composed by:

Prof. ALFONSO GEREVINI (chairman)
Prof. LAURA TARANTINO
Prof. MIRIAM DI IANNI

Network-Aware Recommendations in Online Social Networks
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: February 20, 2017

Author's email: alawad@dis.uniroma1.it

Abstract

Along with the rapid increase of using social networks sites such as Twitter, a massive number of tweets published every day which generally affect the users decision to forward what they receive of information, and result in making them feel overwhelmed with this information. Then, it is important for this services to help the users not lose their focus from what is close to their interests, and to find potentially interesting tweets. The problem that can occur in this case is called information overload, where an individual will encounter too much information in a short time period. For instance, in Twitter, the user can see a large number of tweets posted by her followees. To sort out this issue, recommender systems are used to give contents that match the user's needs.

This thesis presents a tweet-recommendation approach aiming at proposing novel tweets to users and achieving improvement over baseline. For this reason, we propose to exploit network, content, and retweet analyses for making recommendations of tweets.

The main objective of this research is to recommend tweets that are unseen by the user (i.e., they do not appear in the user timeline) because nobody in her social circles published or retweeted them. To achieve this goal, we create the user's ego-network up to depth two and apply the transitivity property of the *friends-of-friends* relationship to determine interesting recommendations. After this step, we apply cosine similarity and Jaccard distance as similarity measures for the candidate tweets obtained from the network analysis using bigrams. We also count the mutual retweets between the ego user and candidate users as a measure of shared similar tastes. The values of these features are compared together for each of the candidate tweets using pairwise comparisons in order to determine interesting recommendations that are ranked to best match the user's interests.

Experimental results demonstrate through a real user study that our approach improves the state-of-the-art technique. In addition to the efficiency of our approach in finding relevant contents, it is also characterized by the fact of providing novel tweets, which solves the over-specialization challenge or serendipity problem that appears when using content-based recommender systems as a stand alone approach of recommendation.

Acknowledgments

First of all, I would like to express my deepest gratitude and appreciation to my supervisor Prof. Stefano Leonardi for his help, wisdom, sagacious guidance and advice over the years. And I would also like to thank my collaborators: Dr. Aris Anagnostopoulos, Dr. Ida Mele, and Dr. Fabrizio Silvestri for their insightful comments, efforts and dedication to my research.

I would like to thank all the people who participated in the user studies of this research for their time and interest in my work. I also have to thank my wife for her support, encouragement, and patience during my study. Finally, I want to thank my late parents who assisted me a lot in my early life.

Contents

Chapter 1

Introduction

The World Wide Web (WWW) is a prominent resource of information that is accessible by network. It was initially created to take care of the requests from scientists in universities in order to share information automatically. Then it is widely spread across billions of people to connect on the Internet. The basic constructs of the WWW are the web pages that are accessible by web browsers. The webpages contents are text documents, images, sounds, and video.

From these huge resources of information, people can do activities like: reading documents, listening to music, watching videos. Therefore, in order to find information that is relevant to their demands, they need to get results of high quality according to their queries using information retrieval techniques. In these days, users interact with the web by adding, removing, and changing contents. For example, User-Generated Content (UGC) take the form of of blogs, wikis, or contents from online social networking sites like: Facebook, Twitter, where users contact with each other through posting or tweeting text, sending messages, posting photos or videos.

1.1 Online Social Networks

Online social networks allow people to connect to one another and to share information, opinions, and ideas. And, they will give an opportunity to find friends to connect among the network members (social relationship). Those friends will appear as links in the profile of the user. Based on the interaction between a user and her friends, they are mainly classified into five classes [38]: family, friends, neighbors, contacts, and co-workers as shown in Figure 1.1 .

Mislove et al. [80] show that online social networks are not new platforms, as for example, the graph that represent sending emails between users is considered as online social network. The online social network sites that are popular in term of size are Twitter, Facebook, Flickr, YouTube, etc. The users of these sites can register to interact with these networks, and they may specify some additional informations about themselves (e.g., their birthday, residence place, interests), and this will be part of their profiles. The social network consists of user accounts and links between them. Some sites, such as Twitter and Flickr, allow users to link to any other user, without obtaining permission from the link target. While other sites, such as Facebook, LinkedIn, need permission from both the creator and target before a link is established between them.

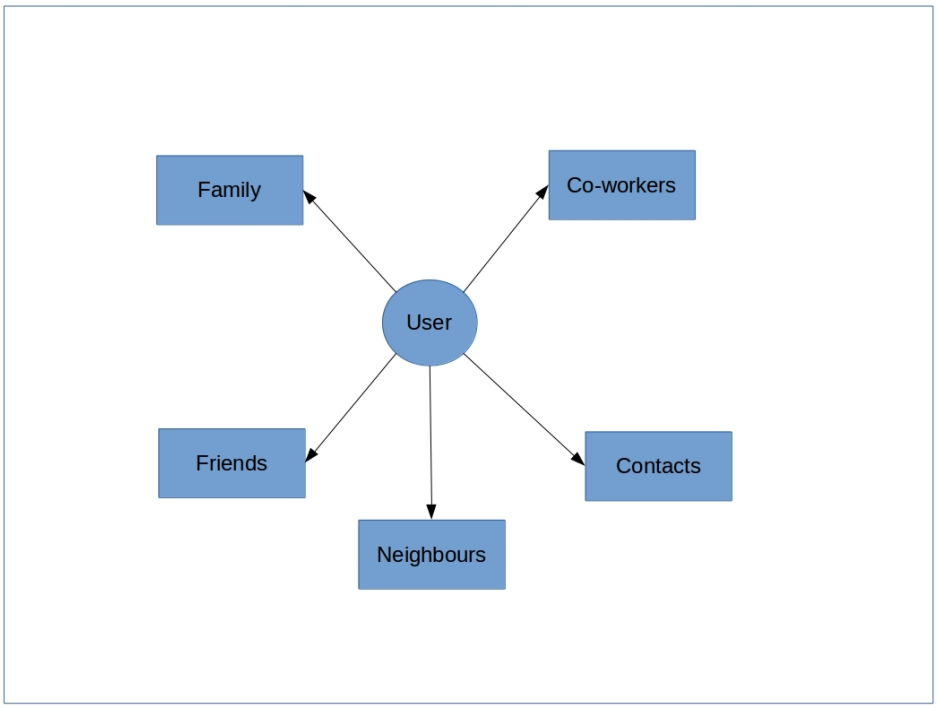

Figure 1.1. User-links classes

Due to the huge amount of data posted by users, there is an imminent need to filter contents that are useful to a target user, so that she will concentrate on interesting contents. The task that should be carried out in this case is information filtering to eliminate unnecessary or undesirable contents from an information stream before showing them to users

To solve this problem, there is a platform that is called "recommender system", which is a subclass of information filtering. Recommender systems are profit-oriented to online vendors like: Amazon.com (Internet-based retailer), Netflix.com (streaming media and video on demand provider), and IMDb.com (online database of information related to films, television programs and video games). These systems predict user preferences (ratings) for new items based on previous ratings on other items. The companies that are developing these systems focused on enhancing the accuracy of prediction, for instance, in October 2006, Netflix lunched an open tournament with a prize of one million dollars for the algorithm that best predicts user ratings for films[1]. The cause of conducting this tournament is that the best proposed algorithm will match the interests and needs of a large group of users and as a result, will raise profits for e-commerce websites [52].

Some companies exploit the trust level between users to spread their services. For example, Hotmail appends a promotion message at the bottom of every outgoing email: "Get your private, free email at http://www.hotmail.com.". A large number of persons who will receive such message in the email will sign up and then spread the message all around. Consequently, the number of Hotmail user accounts grew from zero to 12 million in 18 months on only a half million dollar advertising budget. By this, they found that their approach beat other marketing strategies [57].

[1] http://netflixprize.com/

1.1.1 Factors affecting the constructions of social networks

Social networks are constructed in various shapes. Nevertheless, they are affected by different factors. McCulloh et al. [76] define the mechanisms that govern the social relations between ties as following:

- **Homophily**: It is also known as the "birds of a feather flock together" where members are likely to create connections with others who are similar to them, with regard to socio-demographic attributes (i.e., age, gender, education), or shared interests or beliefs.

- **Social conformity**: It describes how people, groups, and organizations create, control, and delete links in the network.

- **Reciprocity**: The extent to which there are common ties between members of a network. For instance, if user (X) send en email to user (Y), then does user (Y) will send a reply to user (X)?

- **Proximity**: It predicts that there are inter-relations between members who are physically close to each other in a place.

- **Balance**: It proposes that two users are more likely to be connected if they have mutual friends (Triadic transitivity).

In [7], the authors studied the effect of homophily to predict future links. As they found that users with similar interests are more likely to connect with each other.

1.1.2 Twitter social network

Twitter is a popular micro-blogging system which was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass. In Twitter, users can post short messages, called tweets, whose length is at maximum 140 characters. Tweets typically consist of personal information, status updates, news, or links to webpages or other web content (e.g., images and videos). Twitter users, by following one another, define a social graph, where nodes are users, and a direct edge (u, v) represents the fact that the user u follows the user v.

The *tweets* posted or retweeted by a user are shown on the user's profile page as well as on the timeline of her followers. These users can *reply* to the author of the tweet, or they can *retweet* the original message, so that it is made visible to their followers, too. Retweets are very frequent, allowing the propagation of interesting information into the Twitter community, and, for this reason, the users follow news channels, favorite celebrities, or friends to obtain new information as soon as possible. Other actions that a user can make is *Mention* where a user tag another one. The *Like* action represents the user's appreciation for a tweet. Figure 1.2 show an example of tweet's components, where their descriptions are highlighted in red color.

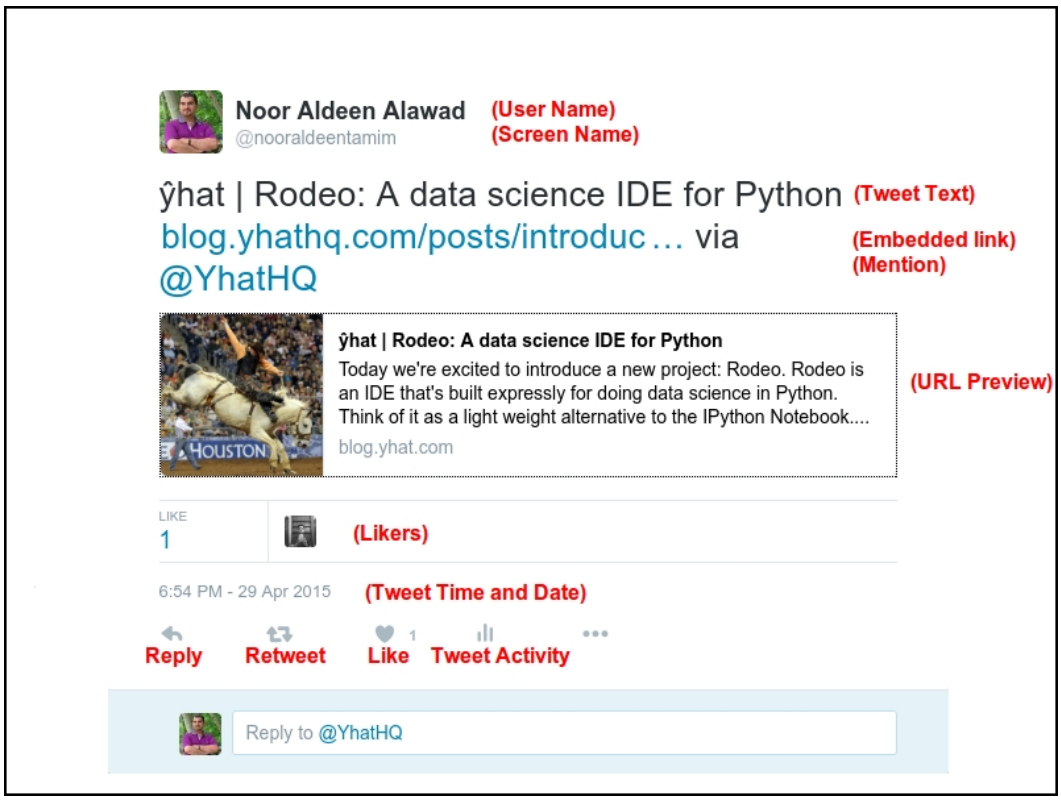

Figure 1.2. Tweet components

Figure 1.3 shows the tweet activity, which contains information about how impressive the tweet is:

- **Impressions**: Number of times people saw a tweet.
- **Total engagements**: Number of times people interacted with a tweet, which is a sum of the following measures.
  - **Detail expands**: Number of times people viewed the details about a tweet.
  - **Link clicks**: Number of times people clicks on the URL embedded in a tweet.
  - **Replies**: Number of replies to tweet.
  - **Likes**: Number of times people liked (favorited) a tweet.
  - **Retweets**: Number of times people retweeted a tweet.

Hashtags are phrases that have the number sign (#) as a prefix. For example: (#Sapienza, #SIGIR2016) are used to know what is happening regarding the terms 'Sapienza', and 'SIGIR2016'. A user who click on these hashtags will be redirected to a page that contains all the users who have included the hashtag in their own tweets. An individual can create her own hashtag, so that other users can interact with it and create conversations easily. When a hashtag become popular at a certain time, then it will be a *trend*, and it appears on the left side of a user's page. Hashtags can help everyone to keep up to date efficiently, so the people can contact each other during an event (i.e., conference).

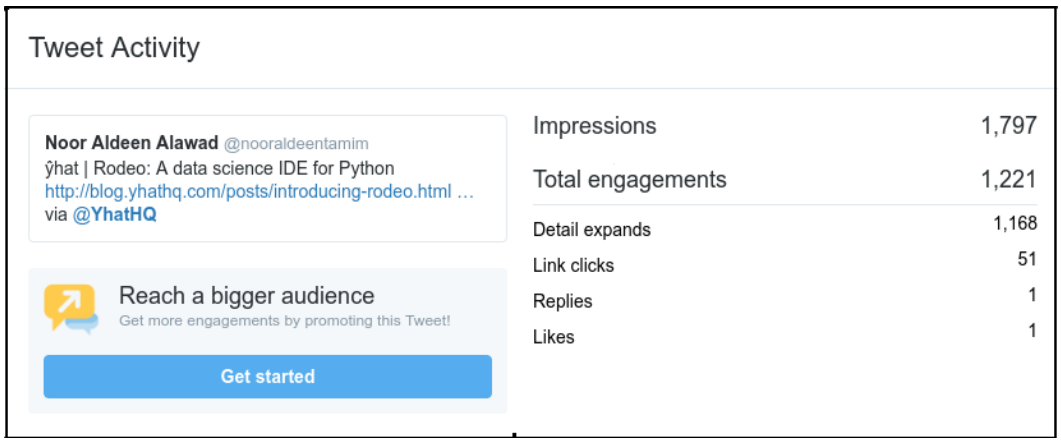

Figure 1.3. Tweet Activity

A user's *feed* is a list of (re)tweets that are posted periodically by the user and that are arranged chronologically with the most recent posts first. On the other hand, a user can send *direct message* (DM) to another one to participate in private conservation.

Twitter has millions of users who use its services everyday, and create huge amount of data. The following are recent statistics on Twitter [2] (Updated till August 2016):

- Number of Twitter users (monthly active users) is 313 million.
- Estimated total number of Twitter registered users is 1.3 billion.
- Unique monthly visitors to Twitter (desktop and mobile) is 120 million.
- Average number of monthly visitors to Twitter that do not log in is 500 million.
- Average number of followers per Twitter user is 208 user.
- Number of Twitter accounts that have ever sent a tweet is 550 million.
- Percentage of Twitter users that have tweeted, but not within the past year is 43%.
- Percentage of Twitter users that created an account and never sent a tweet is 44%.
- Number of Twitter users in the US is 66 million.
- Daily active Twitter users are 100 million.
- Percentage of active Twitter users that log onto it more than once a day is 34%.

Each user on Twitter can identify herself to world through providing information about herself once she signed up. These information include:

- **Name**: Identify the real name of a user with maximum of 20 characters long.
- **Location**: Determine the geographical location of a user.
- **Web link**: An external link to a personal webpage.

[2] http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/

- **Bio**: A small paragraph of less than 160 characters, to express the user's business, interests, etc.

Basically, Twitter specify two classes for the individuals around a user: people you follow (*followees*), and people who follow you (*followers*). And according to [60], Twitter community is classified into three groups:

- Twitter *friends* of a user: they are the people a user follow and follow her back.
- Twitter *fans* of a user: they are people who follow a user although they are not followed back.
- Twitter *inspiration*: They are people a user follows without following her.

As mentioned in [56], the users are also classified into three categories: information source, information seeker, and friends, where information source users are those who have large number of followers, and this class of users always posts important updates in a regular time, while information seeker users are those who posts rarely and follow other users, and the friend class which contains family members or the user colleagues who have the same domain of work. The user intentions on Twitter are mainly concentrated on the ordinary daily activities, and it is considered as the largest class of users.

Twitter is used by researchers to get help in their work, through tweeting a summary of their projects, and ask others for feedback. The followers would access these projects, by following the URLs inside their tweets, and sending their comments. And to fit under the restriction of the maximum characters limit (140 characters), a publisher could use one of the *URL shorteners* services [3],[4] to replace the original long URL.

Mollett et al. [81] show that Twitter give opportunities for 'crowd sourcing' research activities across the sciences, social sciences, history and literature through asking people to help with gathering information, making observations, launching data analysis, transcribing and editing documents. Other researchers have also used Twitter to help 'crowdsource' research funding from interested public bodies.

Burghardt [22] shows that Twitter attract researchers in various academic fields, and this happens, because of the following features:

- **Message size**: Twitter messages are relatively short, which almost bring out homogeneous datasets. While the posts of other social media are larger and of different lengths, which results in imbalanced data.
- **Sample size**: A huge number of messages are posted on Twitter daily, that make it a rich source of data.
- **Metadata**: Twitter messages give all types of metadata, such as username, creation date, language, geolocation, etc.
- **Availability**: Most of the data is public, so that, not only registered users can access it, but it is also accessible by all web users.

[3] http://tinyurl.com/
[4] https://bitly.com/

- **Accessibility**: The services provided by Twitter to access and download the data give flexibility to deal with it. These services are a pre-defined Application Programming Interfaces (API) which will be explained in Appendix B.

Moreover, in academia, Twitter data was analyzed during conferences, for instance, Letierce et al. [67] found that Twitter is effectively used in broadcasting scientific messages, and they conducted their research by:

1. Getting tweets that contain official *hashtags* of three conferences.
   (a) (#iswc2009). The International Semantic Web Conference (ISWC 2009).
   (b) (#online09). The Online Information Conference 2009 (Online 09).
   (c) (#estc2009). The European Semantic Technology Conference (ESTC 2009).
2. Identifying the way they do it.
3. Checking whether their tweets can reach other communities.

They found that studying streams of scientific conferences give intends to consider trend topics of the event, and they noticed that users whom have an authority during an event get a high score over the others. They conclude that Twitter remove barrier between researchers and broadcast the event to a larger set of audience.

In [121], the authors studied the similarity of information flows in Twitter communications by comparing two types of citations:

- URLs pointing to external resources.
- Retweets (RTs) that cite other users tweets.

They conducted citation analysis on collection of tweets that contain hashtags for two conferences:

- (#www2010). The World Wide Web Conference (WWW2010).
- (#mla09). The Modern Language Association Conference (MLA 2009).

They conclude that RTs may indicate the most well known twitterers, while URLs could be counted to specify the impact of referenced publications or presentation slides. In related research [122], the authors show that the kinds of Websites that the URLs refer to, the highly cited URLs from the conference datasets, and the number of retweets for the different datasets, are highly retweeed by users.

Priem et al. [89] studied the way and the cause of citing on Twitter, through answering the following questions:

- Do scholars cite on Twitter?
- If yes, how these citations occur on Twitter?
- Do these citations carry impact?

In their research, they form a sample of 28 academics, where they inspected their reactions to Twitter citation, as they answered the questions as following:

- Yes. The scholars use Twitter to cite articles.
- The following are about how Twitter citations appear:
  - It was noticed that half of these citations have a direct link to a resource, while others have intermediary links to the target resource.
  - Twitter citations are much faster than traditional citations, with 40% happen in the first week of the cited resource's publication.
- The participants said that Twitter citations transmit scholarly impact.

At last, they conclude that Twitter citations could enhance traditional citation analysis through presenting faster and broader metrics of scholarly communication.

1.2 Motivation and Research Objectives

The huge number of tweets posted everyday causes the problem of *information overload*. This problem was studied by Rodriguez et al. [94], they present to what degree the social network users are overwhelmed with the large amount of information posted in these networks. In addition to this, they explain how this problem affects the choices of users to share and spread information to other users, and they note the following:

- Through their experiments, they prove that there exist a limit on the size of information that are posted daily. They detect that there is a small number of Twitter users who approximately post more than 40 tweets per day.
- Although there exist limits on posting information, they discover that there are no restrictions on the information received by Twitter users. This information is proportional to number of followees. And as known, many Twitter users follow several hundreds to thousands of other users.
- They report a limit of approximately 30 tweets per hour, in which less than this limit, the likelihood that a user retweets a tweet remain steady. Conversely, above this limit, the likelihood that a user retweet a received tweet starts to drop considerably. Therefore, they conclude that this rate limit estimates the level of information a user will process, and as result, it will be possible to specify the users who got overloaded information.
- They notice that, at some point, when users are not overloaded, they handle and retweet tweets rapidly as they receive more information. In contrast, when users are overloaded, they take time to handle and spread information as well as they receive more and more information.
- Apparently, when users are overloaded, they seem to organize tweets from a chosen subset of users. They noticed that, although the overloaded users follow more users, the set of users whose tweets they retweet stay to be constant and increments gradually.

Therefore, and according to their study, receiving a large amount of information by users will affect their role in processing like: concentrating on certain sources, the volume of the information spread, and the speed of processing information.

Recommendation systems can play a key role in solving the information overload problem. In Twitter most of these systems are based on recommending users to follow [50], webpages to visit [127], or they consist in reranking the tweets appearing in the user's timeline [125], where the authors used a model to rank tweets and their authors simultaneously using several networks: the social network connecting the users, the network connecting the tweets, and a third network that ties the two together.

A new line of research is based on recommending concealed tweets, by which we mean, tweets that are not posted or retweeted by anybody in the user's social circles. As a result, users have access to additional tweets that may be of interest. Consider, for example, the following scenario: A user is interested in computer science and everyday she checks the tweets of the computer scientists she is following. Some of these tweets are originally posted by them, whereas others are just the result of a retweet. When nobody has retweeted a message that is potentially interesting for the user, such message remains concealed from the user's eyes, because it does not appear in her timeline.

In details, the contribution in Chapter 3 is about to employ the transitive property of ties in the ego networks of Twitter. The graph of users to make recommendations in Twitter is a directed graph, where nodes represent users and an edge between two users, u and v, indicate that u is following v. The graph is built for each ego node using data obtained from their networks.

In our approach, we first conduct a network analysis as in the following situation, suppose that there is a tie between X and Y and one between Y and Z, then it will be expected that X is connected to Z. If this link exist, then they form a closed triangle, otherwise, they form an open triangle. Closed triangles indicate that X can see updates from Z (Z is one of the friends of X). In contrast, in open triangles, Z updates are seen only from the set of Y users (Z is one of the 'friends of friends' of X), and consequently, unseen by X. To choose the most important candidates from Z, we used the number of links from Y to Z as a criteria . Based on the fact that the volume of data that we need to process is large, where the set of Z users are counted in millions, we used MapReduce platforms to extract and propose these candidate users through applying two rounds of map-reduce jobs.

Then, we used content analysis measures to find the similarity between the user X and the set of candidate users that resulted from the network analysis. In content analysis, we applied cosine similarity and Jaccard distance between the timeline of X and the timelines of the proposed users of Z using unigrams and bigrams. In addition to this, we used retweet analysis through counting the incommon retweets between the user X and the proposed users of Z.

The features that we computed their values through network analysis, content analysis, and retweet analysis were used to build the feature matrix. The rows represent the tweets, and the columns represent the features. A pairwise comparison was used to rank the tweets in order to show the highly relevant tweets to the user according to the strategy of providing the user with the tweets that have the best combination of features' values. Figure 1.4 shows an overview of the system structure.

We present a user study aimed at proving the quality of our recommender system. We contact people from network of our friends asking them for help in evaluating our approaches. We asked them some information about their twitter accounts, interests, etc. Then, we sent them a link that they can follow to obtain a set of tweets when they were ready. After that, the users were asked to evaluate based on how much interesting are the recommended tweets.

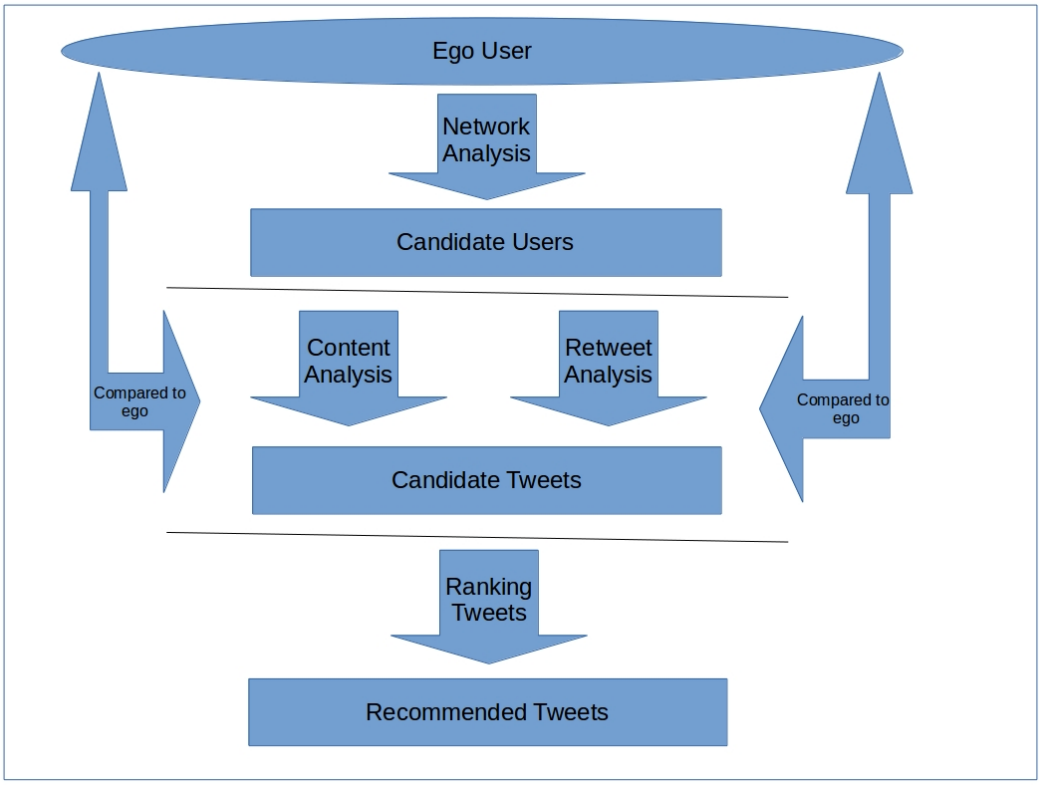

Figure 1.4. General Overview of the System Structure

The user study that we conducted is not anonymous, but the only personal information that we requested from users is minimal (gender, age range, topic interests, how much they use twitter, and in general how much they access the Internet), and it is optional. The interface of the recommender system is efficient, user-friendly and easy to use that describes the overall process of rating the recommended tweets. As a result, we received positive feedback from the users through interaction and participation, and they expressed their interest to know the outcomes of the study.

Pennacchiotti et al. [87] tackled the problem of recommending unseen tweets by proposing tweets whose content matches the user's interests. We follow and extend their idea of recommending hidden tweets that are potentially interesting to the user, but instead of using only the content analysis, we propose to exploit the structure of the network around the user as well as the analysis of mutual retweets; we demonstrate that this yields an improvement in the quality of the recommendations.

1.3 Thesis Outline

This thesis has the following structure:

- Chapter 2 contains the description of the paradigm, types, applications of recommender systems, and a review of recent literature on Twitter recommendations.
- Chapter 3 presents the structure of our recommender system by first describing the

retrieval process from Twitter, how to find candidate tweets based on the user's network, and then to rank and recommend subset of them through content and retweet analysis.

- Chapter 4 describes the users who participated in the user study, explains the results of the experiments that were conducted in this research, and demonstrates the superiority of our approach over the baseline.
- And finally, Chapter 5 concludes the thesis and proposes the directions for future work.

Chapter 2

Literature Review

Recommendation systems help the users to find products and services, music and videos as well as blogs and news articles, and they have been widely studied in the past [92, 39, 21].

2.1 Recommender Systems

There are two main types of recommender systems (personalized and non-personalized):

1. The personalized recommender systems examine the users' preferences in recommendation.

2. The non-personalized recommender systems will not benefit from user preferences. For instance, a recommender system will suggest the same recommendations for everyone. (i.e., Top-10 best seller goods in a market).

Basically, most of the recommender systems that is widely used are personalized recommenders [93]. As in Figure 2.1, the following approaches of recommendation are mainly used as personalized recommenders, and they will be described in details in the sections: 2.1.2, 2.1.3 and 2.1.4 :

- Collaborative Filtering (CF).
  - User-based CF.
  - Item-based CF.
  - Hybrid models (User-based CF and Item-based CF).
- Content-based.
- Hybrid models (CF and Content-based).

2.1.1 Recommender systems paradigm

The major goal of a recommender system is to suggest items to a user through applying transactions. The data that is used in a recommender system consists of three types [93]:

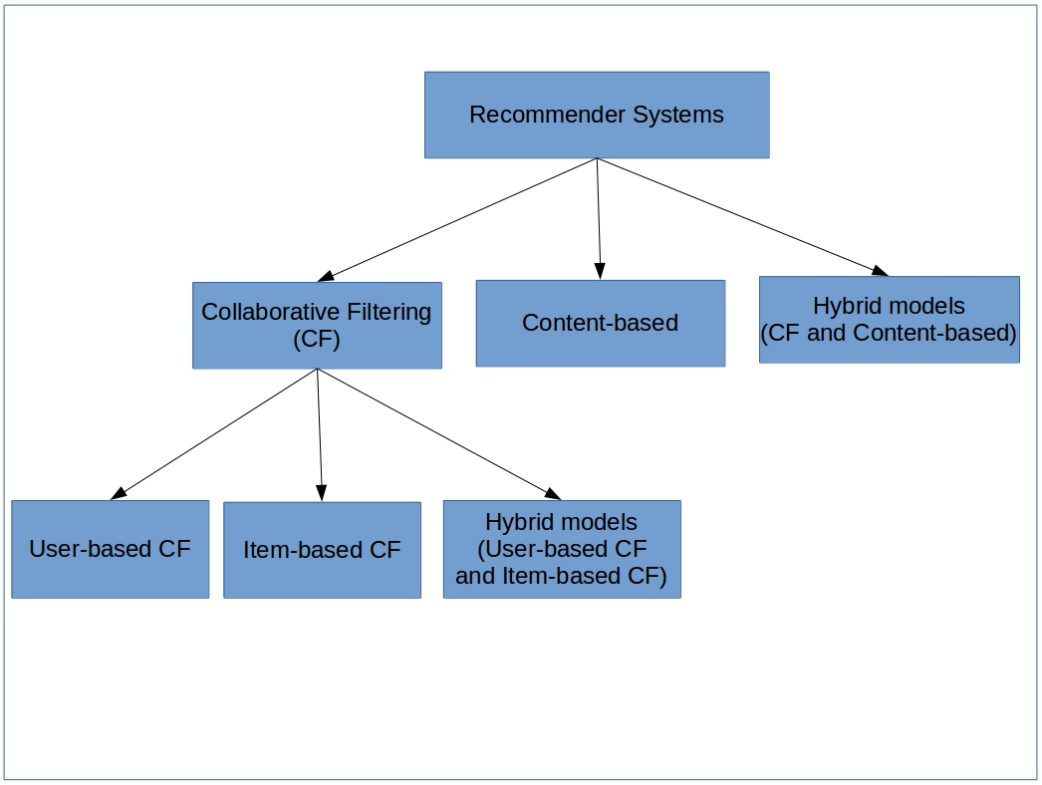

Figure 2.1. Recommender systems techniques

- Items: Items are the entities that are recommended (i.e., tweets, movies, restaurants, etc). The Items have descriptions that are used in the process (i.e., category, price, location, etc). The values of these items may be useful to the user or not, (positive or negative). However, providing users with useful contents will save their time in searching for related items from a very large number of items.

- Users: Users are the people who will receive the recommendations. They have different characteristics and tastes. These information are used to predict items of interests to them. A model should be built to take these information in consideration (i.e., demographic information like: age, gender, education).

- Transactions: They represent the interactions between the process objects and the recommender system. These transactions take three forms:

  - Transactions between users themselves, (i.e., a user follows another).
  - Transactions between users and the recommender systems, (i.e., a user makes a request).
  - Transactions between the recommender systems and users through items, (i.e., user's feedback).

Figure 2.2 shows the paradigm of a recommender system, where a user sends a request to the recommender system, and the system in its turn extracts the features of items to

recommend items that are of interest to users based on their descriptions and the relation between users.

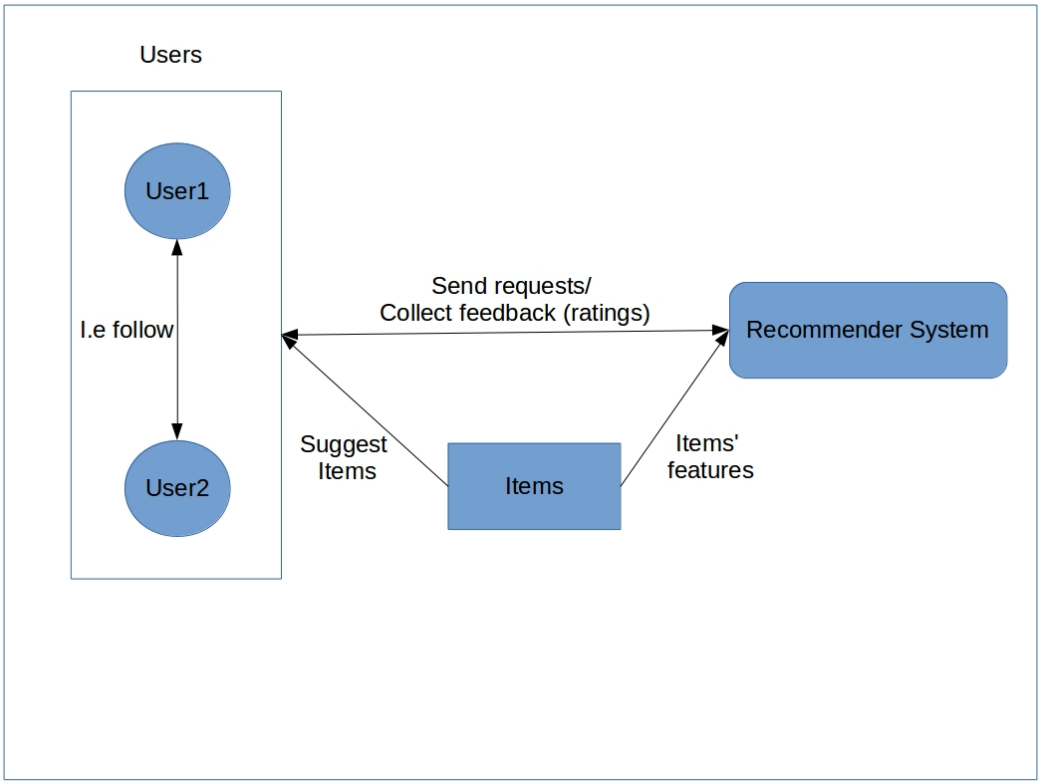

Figure 2.2. Recommender systems paradigm

2.1.2 Collaborative filtering recommendation

Collaborative filtering represents recommending items by referring to other users' activities and preferences in items. Collaborative filtering was widely studied in literature [97, 37, 43, 91]. Amazon company has been using collaborative filtering for long time to recommend products to their customers. Also, the Netflix Prize use collaborative filtering algorithm to predict user ratings for films.

The users preferences are called ratings, and they are represented in a triple as: (User, Item, Rating). And these ratings may appear in many patterns, based on the type of the considered system [53]:

- Real or integer-valued rating scales such as 0–5 stars.
- Binary (like/dislike) or ternary (like/dislike/unknown).
- Unary ratings (has purchased).

The set of all rating triples makes (User, Item) ratings pairs, such that the unknown values are marked as question marks as in the sparse matrix in Table 2.1.

Table 2.1. Four scales user ratings for items

	Item1	**Item2**	**Item3**	**Item4**	**Item5**
User1	1	?	3	3	2
User2	2	4	?	1	?
User3	?	1	3	2	3
User4	1	3	?	?	4

Collaborative filtering recommendation system is classified into the following algorithms [33, 108]:

1. **Memory based**: It recommends the items based on the past activities of users.
2. **Model based**: It recommends the items through a learned predictive model (machine learning model) by referring to the past activities of users in order to enhance prediction performance.
3. **Hybrid**: This technique incorporates both the memory-based and the model-based algorithms together. It enhances the quality of predictions and it solves problems that can be encountered when building collaborative filtering recommender systems like sparsity and information loss that will reduce accuracy.

There exist many forms of collaborative filtering systems [53], however the mostly used are user-based and item-based collaborative filtering.

2.1.2.1 User-based collaborative filtering

This filtering approach is also called "Nearest neighbor collaborative filtering", where users are expressed as neighbors. It is used to predict user preferences through the following two steps:

1. Find users who have similar ratings to a target user (the user who will receive predictions).
2. Use these ratings from those users to predict items that are of interest to the target user.

To calculate user similarities between the users' corresponding ratings, a lot of measures can be used, like: Pearson correlation, Cosine similarity.

Figure 2.3 shows an example that presents user-user collaborative filtering. In the figure, there are 4 users and 5 items, through which we can observe the following:

- $\{u1\}$ is interested in the items $\{i2, i3\}$, and $\{u3\}$ is interested in the items $\{i2, i3, i5\}$. *Therefore, items* $\{i2, i3\}$ *are shared interests - (indicated by orange arrows).*
- $\{u2\}$ is interested in the items $\{i1, i4\}$, and $\{u4\}$ is interested in the item $\{i4\}$. *Therefore, items* $\{i4\}$ *is a shared interest - (indicated by green arrows).*

From this, it is clear that there is a correlation between the users ($u1$ and $u3$), and between ($u2$ and $u4$). So in this case, $i5$ is predicted to be recommended to $u1$, and $i1$ is predicted to be recommended to $u4$. (Indicated by the purple arrows)

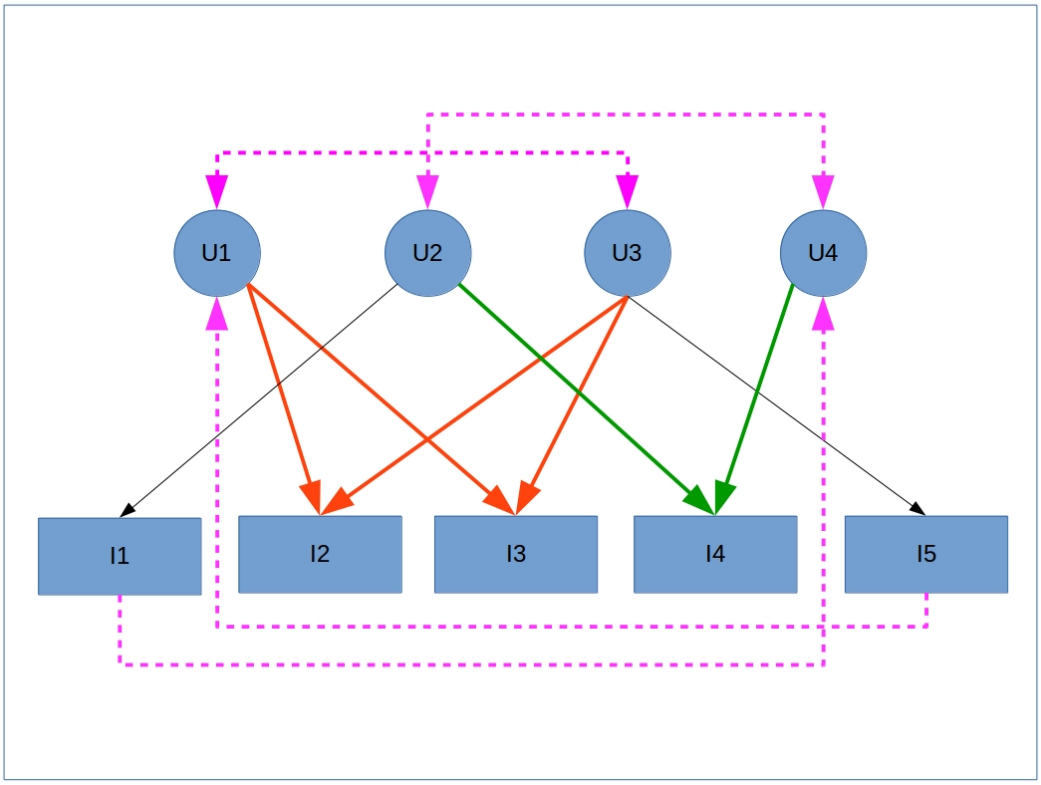

Figure 2.3. User-User CF

2.1.2.2 Item-based collaborative filtering

In contrast to the user-based approach, this one will make predictions according to similarities between items as in the following two steps:

1. Build an item-item matrix determining relationships between pairs of items.

2. Deduce the target user interests by investigating the matrix that are similar to her data.

These similarities can be calculated by finding the cosine similarity between items in a matrix of item-item.

Figure 2.4 shows an example that presents item-item collaborative filtering. In the figure, there are 4 users and 5 items, through which we can observe the following:

- $\{i2\}$ is one of the preferences of users $\{u1, u2, u3\}$, and $\{i4\}$ is one of the preferences of users $\{u1, u2, u3, u4\}$. *Therefore, users* $\{u1, u2, u3\}$ *have almost same interests - indicated by orange arrows for their interest in item* $\{i2\}$*, and green arrows for their interest in item* $\{i4\}$.

From this, it is clear that there is a correlation between the users $\{u1, u2, u3\}$ and user $\{u4\}$. So, in this case, $\{i2\}$ is predicted to be recommended to $\{u4\}$. (indicated by the purple arrows).

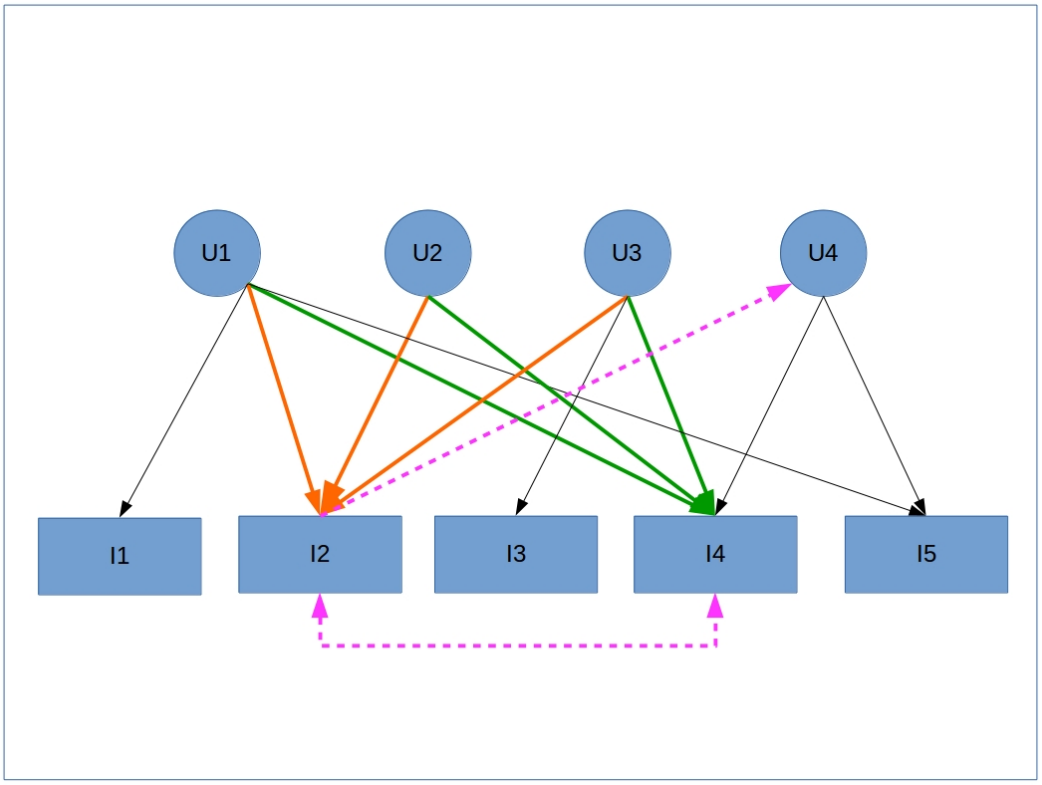

Figure 2.4. Item-Item CF

2.1.2.3 Collaborative filtering practical challenges

The two major challenges that may be encountered in this class of recommender systems are: **sparsity** and **scalability**.

Sparsity: There is a high possibility that (User, Item) matrix be large and sparse, which results in incomplete data that affect the quality of recommender systems when finding users of similar interests. Kumar et al. [63] overcome this problem using trust inference through a model that finds transitive user similarities. The trust inferences are explained in the following example:

1. Suppose that user1 and user2 rate Item1.
2. Suppose that user2 and user3 rate Item2.
3. Then, there may be a correlation between user1 and user3.

They infer the trust between the users (1 and 3) through user2. They found that their method adds new information to the collaborative filtering algorithm.

In another research, chen et al. [30] suggested to consider the direct similarity and the indirect similarity between users, and create the similarity matrix through the relative distance between the user's rating using association retrieval. This retrieval process aims to create a graph of documents, index terms and queries, and then to find the transitive associations among terms and documents using this graph. Their approach results in enhancing the quality of the recommendation system.

Sachan et al. [96] solved the sparsity problem by using fuzzy inference rule, and then they used k-nearest neighbor algorithm to compute the similarity between users. While Pechkis et al. [86] reduce the size of the sparse matrix by clustering the user's ratings through ordering the rows and columns of the matrix.

Scalability: This is another problem that is popular in recommender systems, when the matrix size increase because of its basic elements (Users, Items). The researchers in [124, 41] used the idea of clustering to solve this problem through grouping together similar users as well as items.

Pagare et al. [83] also propose two approaches to solve scalability issues:

1. They exploit the structure of cloud computing, such that, they apply their work on Map-Reduce on Hadoop cluster.

2. They propose a cluster-based collaborative filtering recommendation algorithm.

2.1.3 Content-based recommendation

This type of recommender systems recommend items to users based on the item's description and the users' interests through building profiles for those users [85]. The corresponded items to be recommended are subset of different candidate items that match the items that were already rated by the target user. The items may be documents, Web pages, tweets, news, etc. Content based recommendation was deeply studied in the literature [18, 79].

As depicted in Figure 2.5, content-based recommendation process passes in the following steps [108]:

1. Infer items' attributes through analyzing their characteristics (items' keywords).

2. Compare items' attributes with the the preferences of the target user which can be gained through:

   - Preference model of the user that represents her interests.
   - Interaction log of user with the recommender system to feed the profile by the profile learner, (user's feedback).

3. Get the filtered items to build the user profile and make recommendations through applying classification techniques like: decision trees, nearest neighbors, linear classifiers, etc.

The user's feedback can be positive (predicted items liked by the user) or negative (predicted items are not of the user's interest) as mentioned by Holte et al. [54]. And the feedback can be 'explicitly' taken from users response to the system (ratings, like/dislike, comments), or 'implicit' deduced from the activities of users, like save or bookmark a page. In this case, learning from feedback model will take in consideration the changes in user preferences.

2.1.3.1 Content-based recommenders advantages

The following are the benefits of applying content-based approach in recommendation [70]:

- **Independency**: It depends only on the ratings from the active user to build her profile, and it is not dependent on other users' ratings.

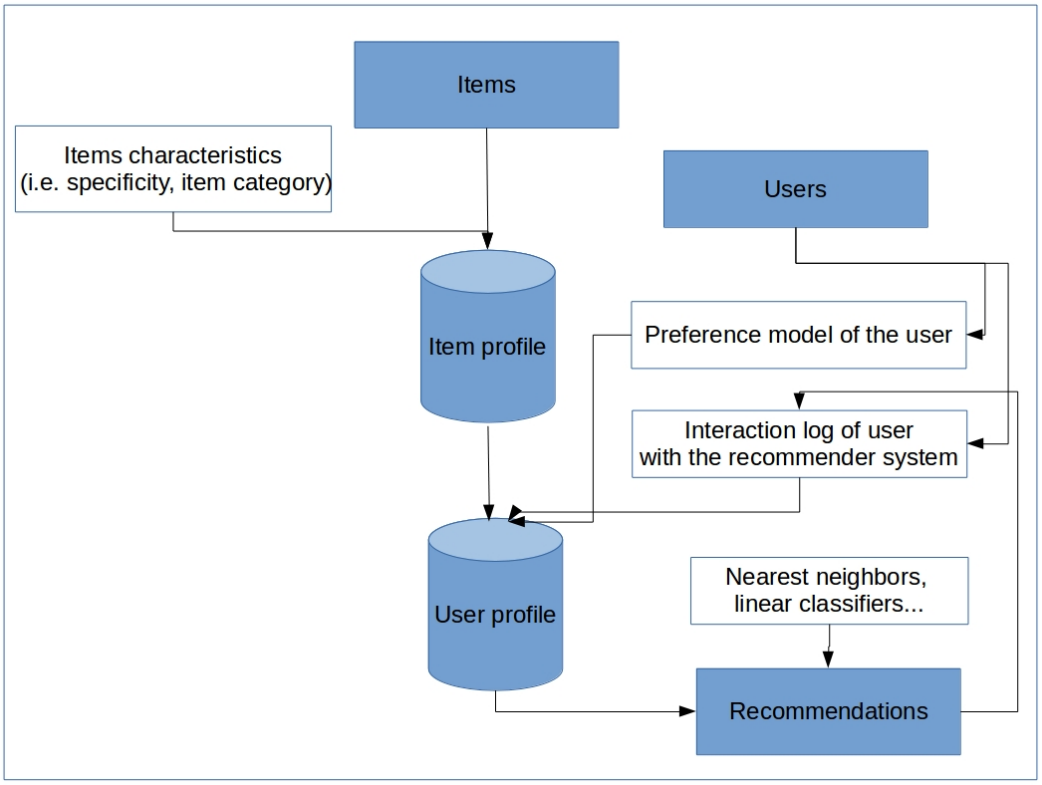

Figure 2.5. Content-based recommendation process

- **Transperancy**: The items' descriptions are clearly determined and used. They are open to active users, so they can be trusted.
- **New item**: There is an ability to recommend unrated new items without having these items rated by several users.

2.1.3.2 Content-based recommenders practical challenges

Nevertheless of the advantages, however there are some limitations that should be considered [70]:

- **Lack of Information**: Getting features automatically is not enough in recommendation, and it results in unsuitable suggestions, and this is because other supplementary information is needed. So, domain knowledge would be good to add more information in order to enhance the process.
- **Over specialization ('serendipity'):** The recommended items may not be novel to users, such that, it mainly depends on the candidate items with the highest scores, and consequently, it makes recommendation of certain type of items. In contrast, in our approach, we used hybrid approach (CF and content-based) to solve this problem, and recommend novel tweets to active user.
- **New user**: Inadequate ratings that are provided to the system will not give a clear picture about the user's interests, and eventually, recommending items to new users.

Yu et al. [129] proposed an approach to solve the problem of 'Over specialization'. Given that, the recommender system suggest similar items that have more overlap between them than others, their approach which is called 'recommendation diversification' is used to recommend dissimilar items. In their work, they define diversity as a measure of how much each item differs from the other terms (items that have higher ranks by the recommender systems while it contains few mutual explanations in common). They showed that their approach is effective in doing recommendations.

Also, in [55], the authors propose an extended content-based approach to introduce serendipity by doing a preprocessing step through representing the documents as concepts rather than keywords.

Other challenges can be addressed by building hybrid recommendation systems that will be explained in the next subsection 2.1.4. For instance, Basu et al. [16] recommend movies by using both ratings and content information through an inductive learning approach. This approach takes a user and a movie as input, and outputs a predict label (like/dislike) to classify the movies. They evaluated their system that uses content features, and observed better results than baselines.

Cantador et al. [25] show the evaluation of content-based recommendation models that are used as similarity measures between user and item profiles as following:

- TF-based Similarity.
- TF Cosine-based Similarity.
- TF-IDF Cosine-based Similarity
- BM25-based Similarity
- BM25 Cosine-based Similarity.

In which they add the Vector Space and Okapi BM25 (BestMatching25) ranking models to the traditional approaches, and they give precision values.

Wang et al. [117] use classification by concepts and instances to enhance content-based recommendation, and they find that their approach:

- Retrieve more explicitly and implicitly related items without affecting negatively the accuracy of the recommender system.
- Give serendipitous recommendations (new and interesting recommendations).
- Provide users with more useful and complete explanations for recommended items.

Martínez et al. [74] make a new content-based recommender system that give the opportunity to users to offer more informations about their tastes through providing a linguistic context, and they are conducted by the following steps:

1. Collect the preferences from different information sources.
2. Filter the items through computing the similarity between users' interests and the items' description.
3. Rank and select the most interesting items to users.

2.1.4 Hybrid recommendation

Hybrid recommender systems merge collaborative filtering and content-based filtering together to obtain better accuracy and performance in order to overcome the deficiencies and limitations that may be encountered if they are considered separately. A lot of research has been done during the last users regarding these types of systems [15, 20, 42].

2.1.4.1 Hybrid recoomender systems aggregation

Adomavicius et al. [6] explain the way that collaborative filtering and content-based approaches combined together:

- **Combining separate recommenders**: Collaborative filtering and content-based filtering are used in isolation, and then combined together through one of the following alternatives:
  - Combine the items' ratings that were gained from each of the recommender systems into one recommended list of items.
  - At some point, select the recommender system that is better than the other, based on its quality (i.e., accuracy).
- **Adding content-based characteristics to collaborative models**: Content-based technique is integrated into the collaborative filtering. This strategy will get over:
  - The cold start problem (Insufficient information about users) in collaborative filtering.
  - Overspecialization problem of content-based filtering.
- **Adding collaborative characteristics to content-based models**: Collaborative filtering is integrated into the content-based technique. (i.e., this strategy is done by grouping of content-based profiles).
- **Developing a single unifying recommendation model**: This strategy selects features from both content-based technique and collaborative filtering to form a model of recommendation.

2.1.4.2 Hybrid recommender systems classes

Burke [24] classified hybrid recommender systems into the following classes:

- **Weighted**: It combines scores of items from the outcomes of other recommendation techniques. As an example, Claypool et al. [31] combine both approaches in their system 'P-Tango' to strengthen predictions through adjusting the weights of these approaches to meet user's interests.
- **Switching**: It switches between recommendation techniques based on a specific criteria, where content-based approach is applied first, then, if it gives insufficient results, a collaborative filtering approach will be adopted. Tran et al. [111] choose the technique of recommendation that best match the historical ratings of a user.

- **Mixed**: It provides recommended items alongside in a merged list from recommendation systems. Cotter et al. [32] combines a list of recommended TV programs based on content-based techniques (textual descriptions of TV shows) and collaborative techniques (information about the preferences of other users).
- **Feature combination**: It inserts features of one recommendation technique 'contributer recommender' (such as collaborative recommendation) into another 'actual recommender' (such as content-based recommendation). This class of recommenders improves the quality of the system by adding new types of features. And as mentioned before, Basu et al. [16] build a movie recommender system that use both user ratings and content features which results in a reasonable enhancement of precision over collaborative filtering.
- **Feature augmentation**: It is similar to the feature combination class, but in this case, the 'contributer' provides novel features for each items through the logic of the domain. Sullivan et al. [103]. solved the sparsity of the data sets through using association rule mining over the collaborative data to deduce novel content characteristics (features) for content-based recommendation.
- **Cascade**: It gives a priority for one technique to be applied first, and provides a list of recommended items, then this list is refined by a second technique. Burke [23] proposed a system that is called 'Find-Me Systems', through which they used the collaborative filter as a post-filter for the knowledge-based recommender system.
- **Meta-level**: In this class the 'contributing recommenders' will not add new features as in 'feature augmentation' class, in contrast, it will replace the whole data of 'actual recommnders'. Littlestone [68] et al. propose a content-based model for each user to predict her interests in restaurants. Basically the model contains vectors of terms and weights to be compared among users in order to provide them with predictions.

  Figure 2.6 shows a summarization of hybrid recommender systems. It presents the set of classes that can be used to combine collaborative recommender systems and content-based recommender systems.

Grivolla et al. [45] propose a hybrid recommender system that combines user demographics and item characteristics, around a collaborative filtering core based on user-item interactions. They evaluated their approach on data sets from MovieLens, such that the input to their approach are:

1. User data.
2. Location data.
3. Offer data.
4. Coupon history.

Items {1, 2 and 3} that represent user demographics are extracted and processed from User Generated Contents (UGC) platforms. The coupon history are gathered from a discount coupon provider, and the data includes:

- User profiles.

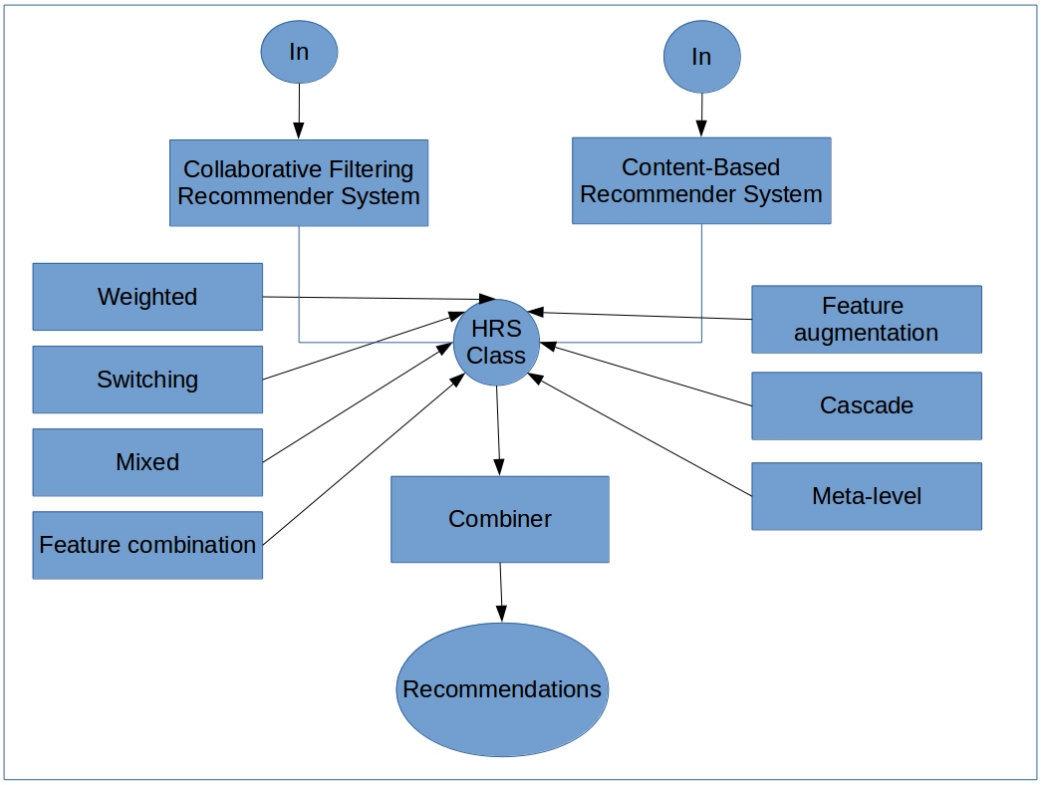

Figure 2.6. Hybrid recommendation system

- Offer descriptions.
- User-offer pairs.
- Additional info, such as interactions of user with offer (clicks).

From this data, an optimized preference matrix is created to predict unknown preference associations by the recommender. At last, they found that user and item information improve the performance of recommendation.

Kim et al. [59] confirmed that their recommender system was enhanced when combining social network analysis and collaborative filtering as in the following:

1. Choose subgroups of users through social network analysis (SNA),
2. Cluster the users into subgroups.
3. Recommend movies based on the clustering result.

2.1.5 Recommender systems applications

Recommender systems are used in different areas, where they are classified into the following domains as in [93]:

- Entertainment: Recommend movies, music, etc.

- Content: Recommend personalized newspapers, documents, Web pages, e-learning applications, and e-mail filters.
- E-commerce: Recommend products to buy (i.e., books, cameras, etc) for consumers.
- Services: Recommend travel services, experts for consultation.

Also, social networks recommendation is considered as an example of the 'content' class.

2.2 Twitter recommendations

With the rapid growth of Twitter, a lot of research has been focusing on analyzing Twitter data and the activities of its users for improving personalization and recommendations [65]. Recommendation tasks can be divided into the following categories:

- Followee recommendations.
- Tweet-based content recommendations.
- Tweet recommendations.
- Retweet recommendations

2.2.1 Followee Recommendations

A user in Twitter tend to follow new users who share the same interests as she has. The new users can be discovered through counting the common friends between them and the user, or comparing their profiles, or based on how popular the account is. In [13], the authors proposed an approach which recommends the people to follow by analyzing the topology of the network around the user through observing the set of followees of her colleagues who have the same followers as her, and those followees are proposed to be recommended to the user.

The authors in [10] suggested a followee recommender framework that as indicated by a heuristic methodology, investigates the topology of followers/followees system of Twitter to discover and recommend candidate users, and rank them by focusing on their similarities with the interests of the user, and they proposed three profiling techniques that are classified into two methodologies:

- The first one builds a model for a user through examining the content of her own tweets.
- The second one models the users through the tweets of their followees through:
  - Modeling a target user by the set of folowees' profiles.
  - Modeling a target user by a set of classes that can be found by bunching her followees in clusters as indicated by the content of their tweets.

As a result, the methodologies that utilize the posts of the followees of users either separately or gathered into classes for modeling their interests, showed remarkable degree of precision in recommendation. In addition to this, in [12], a technique was utilized to investigate Twitter with the objective of finding candidate users to recommend, based on the idea of that, if a user v is followed by both users u_1 and u_2, then followers of u_1 may be interested in following u_2. They detected the user's interests through building a content-based profile for the user.

Also in [11], the authors presented two recommendation methodologies:

- Collaborative filtering technique: It chooses candidate recommendations using just the network topology, where it investigates the connections from the target user to choose a set of candidate recommendations and ranks them based on a scoring function.
- Content-based technique: It uses the contents of the followees' tweets, where it makes a vector of terms that represents the target user's interests by referring to the tweets posted by her followees, and this vector is then used to find new users.

And they found that the content-based technique is better at yielding good recommendations.

Other studies have based their followee recommendations on the popularity and activity of Twitter users, where in [27], both tweet content factor and social relation factor were taken in consideration to model inter-user preference which are affected by the user actions that include following users, retweeting tweets from followees and adding comments for the tweets of users. And in [40], the authors recommend followees based on the popularity and activity; such that if the ratio of followees and followers exceed a predetermined threshold then the user is considered as popular, while the activeness is measured through counting the number of the posted tweets since a user has created his/her account, and as a result it was noticed that considering these two features together is better than considering each of them individually.

As mentioned in [44], reciprocity is considered as a method of recommendation such that if a user (X) is interested in another user's (Y) posts, then there is a higher probability that (Y) is interested in user X's posts. And if two users have the same followees, then they mainly share the same interest; and so they will be recommended to each other. Also they are recommended if they have the same followers since they have the same audience. Moreover; those users who can be reached through group of users of her followees (i.e., followees of followees) are also recommended to that user.

Twittomender [50] is a famous system, which recommends followees based on the users' tweets and their relationships in the social graph. In detail, the approach concentrates on mining through Twitter API, the contents of the user's profile like the tweets, the Ids of her followers and followees. In which, a user asks the system through a search query to find users with similar profiles, or those users are recommended based on the tweets posted in the account. This research adopted seven strategies that are listed in the following two categories:

- Content-based:
    1. Users own tweets.
    2. Followee's tweets.
    3. Follower's tweets.

4. All tweets

- Collaborative-based:

1. Followee's Ids.
2. Follower's Ids,
3. All Ids.

Then, the users are represented by TF-IDF scores. A followee is considered as a best recommended user, if she is not followed by many users, which is represented through the IDF score. At last, they found that collaborative-based is better than content-based through which considering all Ids is better than followee's ids and also better than combining both followers/followees ids. Although our system does not recommend people to follow, but novel interesting tweets, there are some similarities with these approaches, because we also exploit the network structure of the social graph.

The researchers in [49] propose recommendations of followees to a user based on content and collaborative filtering methods. They built models for Twitter users from their tweets and relationships of their social graphs.

The researchers, too, in [51] give an opportunity for 'cold-start' users to discover new users to follow, who have a similar interest with them, and this is based on a pre collected historical data from Twitter, or through providing more information about the users to follow like: description of their profiles, the top terms they mention in their tweets or who they mention in their tweets.

In [62], the followee recommendations were done according to:

- The number of user's followers.
- The number of lists or groups that the user is listed in.
- The number of news related group the user is in.

They found that recommendation based on the number of followers is better than the number of lists the user is in.

In a study done in [109], the authors showed that the accuracy of followee recommendations can be enhanced through considering the personality factor (the combination of emotional, attitudinal and interpersonal that exist in an individual) in recommendation, such that through a data analysis, they showed that there exist relations among users characteristics.

The authors in [110] showed that the precision of recommendations can be enhanced when taking in consideration an adaptive technique for recommending followees by ideally combining diverse factors of recommendations based on the preferences of previously chosen followees. It is also fit to the changes of preferences overtime.

In [123], the authors predict if the user will follow others who are recommended from Twitter. The factors that are considered in prediction are:

- Item popularity.
- Item category.
- Followees' acceptance.

- Extracted keywords and actions: (retweet, mention, comment).

It was found that item popularity is the best feature that affect the recommendation positively, and also the extracted keywords and actions of the user.

In the research [47], the researchers enhanced the accuracy of recommendation systems through using implicit sentiment analysis. They stated a weighting function of user interests based on:

- (Sentiment): Which is the sentiment expressed by the user for a given concept.
- (Volume): How much she is interested in that concept.
- (Objectivity) How much she expresses objective comments on it.

To refine the content-based profiles, they built profiles for the user through the weighting function that consider SVO (Sentiment, Volume, Objectivity) to the user attitudes, and this yield in advantages to recommend followees in comparison to other methods.

Recommending followees was discussed through considering the new location of a user and her interests that were deduced from her previous tweets in order to look for new users to follow [9]. In details, their approach is composed of four components:

- User interests list: Extracting the nouns that appear in the last 150 tweets or retweets.
- Local tweets input: Retrieving local tweets and hashtags and analyzing items to find keywords, tags, category and topic.
- Conceptual fuzzy sets subsystem: Computing the similarity degree between local tweets and user interests list.
- Output trends list: Collecting the result of conceptual fuzzy sets subsystem.

The link structure is important in several recommendation systems as in [128], where an augmented social graph was built based on both user attributes and graph structure information, The authors applied a random walk algorithm on this graph to estimate the link relevance to the user.

2.2.2 Tweet-based Content Recommendations

Due to the lack of user-profiling information, recommending web content is often challenging, and Twitter has demonstrated to be a rich source from which is possible to obtain more information about web users. In particular, it has been proven that using Twitter data (e.g., tweets, retweets, hashtags of the user and of her friends) is possible to improve the URL recommendations [127].

And also, there is another research that is related to ranking URLs based on the users' interest [28]. The authors designed two approaches: one of them is through recommending URLs from the followees and followees of followees, and the other is the popularity score of URLs. Then, the proposed URLs are ranked through the topic-relevance and the social process. The topic-relevance was done through measuring the similarity between the tweets that contain these proposed URLs and the user's tweet, and between them and user's followees tweet, while in the social process, the proposed URLs are ranked based on the vote powers of the user who tweeted the URL, such that those vote powers are determined

through the users followers count, frequency of tweeting, etc. At last, it was discovered that the proposed URLs from the followees of followees are the most interesting. And that combining both methods which contain: the proposed URLs compared to user's tweets, and the vote powers give the best ranking.

Differently from these works, we do not recommend web content but focus on recommendations of tweets.

A content-based approach was suggested by [88] to recommend news that are ranked according to the users' tweets through her social graph (i.e., friends and followers,) or from the tweets of the public timeline of Twitter, and this were done by applying the following strategies:

1. Mine tweets from the public timeline, searches the user's index of RSS items.
2. Mine tweets from people the user follows, searches the user's index of RSS items.
3. Mine tweets from the public timeline, searches the entire space of RSS items gathered from all users' subscriptions.
4. Mine tweets from people the user follows, searches the entire space of RSS items gathered from all users' subscriptions.
5. Rank stories by recency.

The results show that the users prefer stories originated from their favorite RSS feeds, that are ranked by public tweets or the tweets of their social graph, than stories that from a wider community repository of RSS stories. However, in our approach, we do not recommend news but we recommend tweets.

Sun et al. [104] build a diffusion graph to recommend subsets of micro-blogs in emergency occasions, where they applied their approach on Twitter during the early spread of H1N1 disease, and they found that their approach is better than the other state of the art methods.

In [130], the authors collected data from Twitter to train probabilistic collaborative filtering models that are called Matchbox in order to predict individual retweets in Twitter. The learning algorithms are carried out using the following features:

- The tweet source (the tweeter).
- The user who is retweeting (the retweeter).
- The tweet content.

They found that the tweeter and retweeter were the most important features for prediction.

Ramage et al. [90] show a scalable implementation of a partially supervised learning model that maps the content of the Twitter feed into dimensions (substance, style, status, and social characteristics of posts). Such that, these dimensions help in discovering new users and for filtering tweets in the meantime. They proved the efficiency of these models in finding similarities in posts, and the performance of personalized feed re-ranking.

Naveed et. al [64], analyzed a set of high-level and low-level content-based features on a large collections of Twitter messages.

The low-level features can be obtained in straightaway from the text of the messages, and they include:

- The words contained in a tweet.
- The tweet being a direct message,
- The presence of URLs.
- The presence of hashtags.
- The presence of usernames.
- The presence of emoticons.
- The presence of question and exclamation marks.
- The terms with a strong positive or negative meaning.

The high-level features are gained from:

- Associating tweets to topics.
- Determining the sentiments of a tweet.

The authors trained a logistic regression analysis model to predict the chance of retweeting a tweet based on its contents. They found that there is a high probability for a tweet to be retweeted when its topic is general or public, and that bad news spread rapidly in Twitter. However, in our approach, we concentrate on recommending the tweet contents.

In [36], the researchers suggest an approach to rank tweets by training a learning-to-rank algorithm using the following features:

- Content relevance features: They represent the features that characterize the content relevance between queries and tweets, like:
  - Similarity of contents.
  - Tweet Length.
- Twitter specific features: They represent the features that characterize the particular characteristics of tweets, like:
  - Retweet count.
  - URLs shared in tweet.
- Account authority features: They represent the features that show the influence of authors of the tweets in Twitter, like:
  - Sum_mention: Sum of mention scores of users who published or retweeted the tweet.
  - First_list: List score of the user who published the tweet.
  - Important_follower: The highest follower score of the user who published or retweeted the tweet.

From this framework, it was found that Sum_mention, First_list, Important_follower, length and URL were the best features. In particular, the existence of URLs and the number of times the account is listed by other users were the most effective features.

The researchers in [5] considered the temporal dynamics of user profiles in Twitter that are figured out from users' activities. They defined time-sensitive user modeling strategies (hashtag-based and entity-based), and evaluated them in context of a recommender system that provides Web site recommendations on the Social Web. They gave the best accuracy and performance.

In [58], the authors showed a semantic web technique to filter public tweets that are analogous to the interests from personalized user profiles. They collected and clustered information about users from social media networks (i.e., Twitter, Facebook and LinkedIn). They selected the entities and interests from tweets, they modeled them, they grouped users based on them, and they recommended tweets with suitable interests to the users according to these groups.

The researchers in [115] suggest a new recommender system which is called whom-to-mention by training a machine learning ranking function through the following features:

- User interest match.
- Content-dependent user relationship.
- User influence.

They conducted a real user study, in which they found superiority of their approach in comparison to other baseline algorithms. Apart from what they did, we do not recommend persons to mention, we recommend new undiscovered tweets.

Tweets are used as a facility to recommend hashtags [100, 35] by classifying similar tweets based on their contents, and enhancing this process by following the documents that are linked to them.

In [131], the researchers trained a learning-to-rank algorithm using three categories of features: user-based, movie-based, and tweet-based, in which their approach beat the baselines.

The research that was done in [126] used a topic identification as a proxy to analyze huge number of tweets, and to measure the interestingness of a person's tweet by refering to its latent topics. They also showed that their approach outperform other baseline algorithms.

The authors in [120] propose a new method to enhance finding the news highlights through using microblogs. For a news article, they did the following two steps:

1. They found a set of indicative tweets, and use them to help in ranking the news sentences for extraction.
2. They extract the top ranked tweets as a substitute of sentence extraction.

They show that their approach outperforms state-of-the-art baseline approaches, and that the tweets extraction shows a good performance for generating highlights that are closer to the human compression.

2.2.3 Tweet Recommendations

Most of these approaches offer a ranking of tweets based on a query, or they re-rank the tweets appearing in the user's timeline. Yan et al. [125] rank tweets and their authors using hybrid networks. Uysal and Croft [113] proposed to rank the incoming tweets based on their probability to be retweeted by the users, and the ranking of tweets is done through a classifier that is trained with four features

- Author-based features: They are determined from the user profile: followers count, friends count, account age, statuses count, favorites count, average of tweets per week, etc.
- Tweet-based features: They are determined through hashtags, URLs, mentions, whether a tweet is retweeted, the length of the tweet, the tf-idf score of the tweet, etc.
- Content-based features: This is related to the information contained in the tweet, such that the filtered tweets are those with minimum cosine distance to other tweets for successive weeks in the user timeline.
- User-based features: These features are used to determine whether the author of the tweet is a follower or friend for the user who will receive the recommendation, and if they mention each other before or if the user retweeted a tweet from the author.

And through their work, it was noticed that the ranked tweets are adjusted based on the proposed users who have a higher probability to retweet.

Lumbreras et al. [72] build a model called 'MarkovTrust' that is applied to Twitter to guess trust among users according to their interactions. Their recommender system do the following actions:

1. Crawl the Twitter network by following the top trusted neighbors of target users,
2. Make tweet recommendations by depending on both past retweets and estimated trust.

The results show that recommendations are enhanced when using this approach and this system adds more accuracy to trust based recommender systems.

Chen et al. [26] suggest a real-time recommender system that is called 'TeRec', in which it models users' preferences on the fly and gives recommendations based on this. Their system gives opportunities for users to post tweets, receive recommended hashtags and provide feedbacks to update the system. Through experimental results, they proves dominance of their approach over baseline for real time hashtag recommendation for tweet streams.

In [114], the authors examined the way the audience spread information. They analyzed their approach upon social plugins, Facebook pages, and Twitter accounts, and they found that:

1. The audience shares online news content strongly through social plugins.
2. The activity of the news media in Facebook and Twitter impacts the activity of the audience.
3. The news media are more active on Twitter than on Facebook.

Krestel et al. [61] invented general approaches for recommending tweets to news articles' readers. In their system, they consider the following actions:

1. They invented and trained different models to recommend tweets to news articles.
2. From a dataset of tens of thousands of tweets and news articles, they gathered thousands of relevance judgments.
3. They conducted a large-scale user study to judge their invented models.

Through their research, they determined the related content tweets by handling language models and topic models to overcome the following challenges:

- The tweets should be as similar in content as the news article.
- The tweets should be posted in about the same time as the article.
- The recommended lists of tweets should be useful and not redundant.

They showed that using topic models is better than the language model approach to discover related tweets.

Lu et al. [71] present a system that depend on Wikipedia concepts and link structure, such that they consider the following two remarks:

- If the user u is following the user v and the user u has lots of communications with v, then u is interested in v's interests.
- If the user u is interested in the concept α, then u will be interested in the concepts that are closely related to the concept α.

In details, they re-rank tweets in the timeline of user through building a user profile based on the tweets of the user and how they are related to user preferences. This profile is defined as concepts through a random walk on Wikipedia graph by exploiting the inter-links between Wikipedia articles. They found that their model is impressive and efficient in recommending tweets to users.

Guo et al. [46] improve the quality of the contents of short text in social media like in Twitter, in which they proposed the following strategies:

1. They create a dataset that contain tweet-news pairs by linking tweets to news.
2. They suggest to build a graph based latent variable model that models the text-to-text information (correlations between texts) by considering the following features:
   (a) The tweet specific features like hashtags.
   (b) The news specific features like the named entities in a document.
   (c) The temporal information in tweets and news articles.

Mainly, they proposed to model the inter short text correlations (text-to-text information) in addition to text-to-word information. They show that their approach beats the baselines.

Wang et al. [116] propose a recommender system that recommend tweets to core users. They offer an algorithm that is combined with WAF (Word Activation Forces) to analyze the data and process the tweets of the users in order to recommend tweets to core users, and

moreover, they consider the emotional bias of each of the tweets' nouns in their algorithm. They prove the effectiveness of their approach in comparison to others.

Liu et al. [69] examined the retweeting patterns of users through modeling their retweeting behaviors (the retweeting history of each user) by considering 3W retweeting patterns:

- Temporal (When).
- Social (Who).
- Topical(What).

And also, they designed a time-aware recommendation method in order to recommend tweets during the online sessions of users. They showed that their results outperformed other baseline approaches.

Tang et al. [106] explore the way of exploiting social relations from local and global perspectives in recommender systems and suggest an approach that use both of them together as following:

- The local perspective of social relations reveals the correlations between users and their neighborhoods.
- The global perspective of social relations reveals the reputation of a user in the whole social network.

They conclude that ratings from users with high reputations are more likely to be trustworthy. The system that uses both of them for recommendation is called 'LOCABAL', and it beats other social recommender systems

The author in [73], combines active learning methodologies into tweet recommendation to discover and rank relevant tweets. The proposed method give a ranked list of most relevant and novel tweets at time T based on query Q. The novelty of the recommended tweets is considered by clustering the retrieved tweets.

Wang et al. [118] suggest two approaches for tweet recommendations:

- Recommend tweets based on the cosine similarity between topic-distributions of users and candidate tweets.
- Recommend tweets based on word ranking in each topic, by assuming that there is only one topic for each tweet.

The evaluation of their approach shows better performance in comparison to other baselines.

2.2.4 Retweet Recommendations

There are only some research papers that are interested in retweet recommendation. For instance, Zhao et al. [132] propose an online recommender system of retweets that are suitable for the followers. They show that a tweet will likely be retweeted, if the user retweet it earlier in a period of no more than two days of the tweet creation date, and at the same time, not to retweet many tweets. The issue here is to choose retweets as soon as they appear in the Twitter feed before seeing the following ones. They used two online algorithms:

- Real-time recommendation: It is to decide immediately to retweet a tweet when it is encountered through comparing its score with a threshold.
- Non-real-time recommendation: It is to take care of the selection quality, through looking at the sucessive tweets to decide to retweet in a chosen time window.

They found that the later algorithm (Semi-online) gave better results than the former one (Online).

Nasirifard et al. [82] check the relevancy of hashtags in tweets to the followers of a target user. And to recommend tweets which the user should retweet for his followers through the following two steps:

- Audience profiling that allows users to find the subset of their followers that will be tagged based on a term in tweet (i.e., it can be gained from the hashtags of tweet).
- Recommending well-connected topic-sensitive users for a tweet, who may retweet the tweet.

Their recommendations system 'Tadvise' is fruitful and impressive for followers to retweet a tweet.

Suh et al. [102] studied how the features of tweets will make them retweeted. They found that a tweet that contains contents like: URLs and hashtags have higher chance to be retweeted. And that other contextual features of an account like: age of account and number of followers affect the action of retweeting.

Lee [66] built a model to find the right persons at the right time on Twitter who probably will retweet a tweet and spread it. And another model was used to estimate the period of time an individual will retweet a post based on the past retweets. Through their approaches, they concluded that the retweeting acrivities were doubled in comparison to two baselines.

Our approach differs from these, because it aims at making recommendations of concealed tweets, which are tweets that do not appear in the user timeline, because they have not been posted or retweeted by anybody in the user's social circles.

Moreover, our approach is user-centric in the sense that the recommendations do not depend on a specific query but they rather match the user's interests inferred from previous user's (re)tweets. In this way, our recommendation algorithm is similar to [87] where the authors recommend unseen tweets based on the content similarity.

Differently from them, we exploit not only the analysis of tweet content but also the structure of the network around the users (ego-network) and the users' interest similarity which is mined from their common retweets. As we will explain in Chapter 3, we define a set of candidate tweets to recommend by creating the user's ego-network with depth two and exploiting the transitivity property of the following–follower relationships among users. These candidate tweets are then ranked by using content-similarity measures plus the users' interest similarity to best match the user's interests.

Chapter 3

Design and Methodology

In this chapter, we describe the structure of our recommendation approach. The approach is based on recommending tweets that cannot be seen by the user, for example, because nobody in her circles wrote or retweeted them. Twitter users who subscribe to our system receive recommendations of novel tweets, which do not appear in their timeline but which are of potential interest.

For making recommendations we applied network, content, and retweet analyses. The idea is to use the network structure around the user as a pre-filtering step to find candidate tweets to recommend. Then, these tweets are ranked by using the content-similarity features and the number of common retweets, which is an indication of how much the users' interests are similar.

To explore our method more deeply, when a user u subscribes to the service, the system uses the Twitter API to retrieve u's friends, friends of friends, and timeline (i.e., tweets and retweets). Given u's ego-network up to depth two, the recommendation algorithm exploits the transitivity property of the following–follower relationships. For example, assume that user u follows a set of users v_i who follow user z, and that u does not follow z, the (re)tweets of z are not visible to u, unless some v_i retweeted them. The idea is to use z's (re)tweets, which do not appear in u's timeline, as possible candidates for the recommendations. To weigh the importance of the tweets of z, we count the number of users v_i that are in between.

3.1 The Retrieval Process

At the beginning, for the user, the system is accessing the Twitter REST (Representational State Transfer) APIs [2] through the API keys in order to retrieve all her friends, friends of friends and her timeline including (re)tweets, retweeters list, and the user who issued the retweet. In Appendix A, there is more information about the way of creating applications from Twitter Developers site and how to establish connections, and how to determine the application type.

Twitter supports a few authentication methods [1], and with a range of OAuth [1] which is an open protocol to allow secure authorization in a simple and standard method from web, mobile and desktop applications. The authentication methods are OAuth signed authentication and Application-only authentication. However, Application-only authentication

[1] http://oauth.net/

has a rate limit higher than the other authentication methods, so we used it in our system. Appendix B shows more information about REST APIs and these authentication methods.

Twitter put limits on the download process for developers by restricting the number of requests in a time window. These requests could be of two main classes:

- GET, like statuses/home_timeline, statuses/retweets/:id, friends/list, etc.
- POST, like statuses/update, statuses/retweet/:id, direct_messages/new, etc.

Basically, our system uses four kinds of requests: GET friends/ids, GET statuses/user_timeline, GET statuses/retweeters/ids, and GET users/lookup, which they are explained in Appendix B:

In retrieving data, we used Twython which is a Python library that gives the ability to access Twitter data through Twitter REST API.

Twython [77] gives a facility to query data for:

- User information.
- Twitter lists.
- Timelines.
- Direct Messages.
- and anything found in the Twitter API docs as the requests in [2].

It also offers support for OAuth 2 for application authenticated calls to do read-only calls to Twitter, (i.e., searching, reading a public users timeline). Aftering installing Twython, it can be imported in Python through the following statement:

```
from twython import Twython
```

After registering application as in Appendix A, then you need the Consumer_Key, Consumer_Secret, Access_token, and Access_token_Secret to be passed to the library, and then it converts the retrieved JSON from Twitter into a Python object.

3.1.1 Retrieving friends and friends-of-friends

In our approach, this retrieval process goes through two passes. The application sends requests to retrieve the friends in the first pass, and friends-of-friends in the second pass as shown in Figure 3.1. Through this, it takes in consideration the rate limits in a time window.

In Algorithm 1, the user's friends will be retrieved and stored in the form of '(user-friend)' in the edges list 1, and the the user's friends of friends will be retrieved and stored in the form of '(friend-friend_of_friend)' in the edges list 2.

Algorithm 1 Retrieving accounts of distance 2 from a central node (u)

1: /* Input: A user u.
2: Output: Edges list1, 'edgesL1': (u, v_i), and edges list2, 'edgesL2': (v_i, w_j). */
3: $v_i \leftarrow ids\_of\_u's\_friends$
4: $edgesL1 \leftarrow (u, v_i)$
5: **for** $id \in v_i$ **do**
6: $\quad w_j \leftarrow ids\_of\_id's\_friends$
7: $edgesL2 \leftarrow (v_i, w_j)$

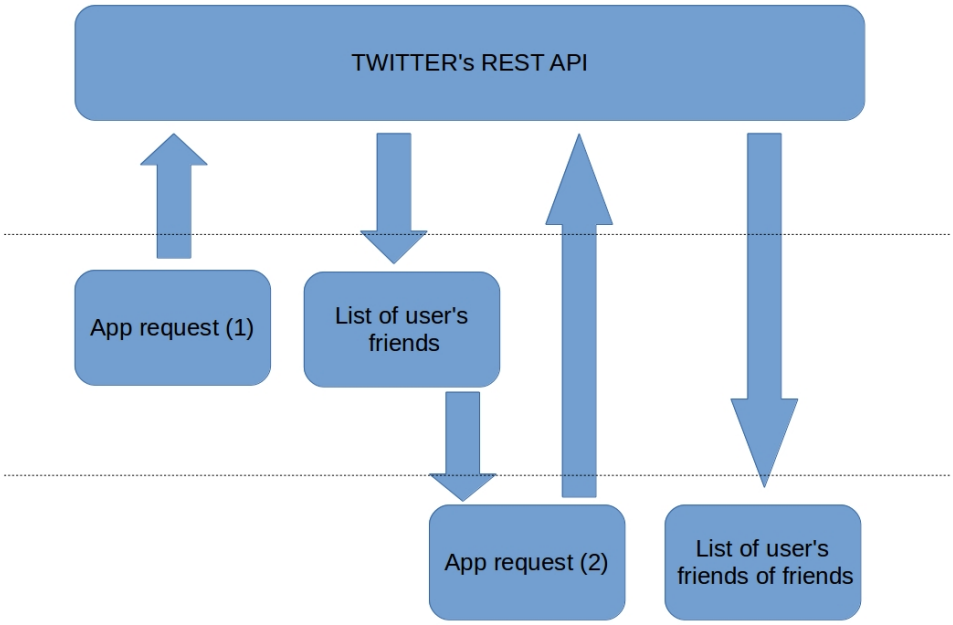

Figure 3.1. Flow diagram of friends and friends-of-friends list.

Figure 3.2 clarifies the process in an example, where the red edges are retrieved from the first pass, and the blue edges are retrieved from the second pass.

Through this process, there exist some private Twitter accounts that raise an error when crawled. In this case, Twitter API returns the following error:

- TwythonAuthError: Twitter API returned a 401 (Unauthorized), An error occurred processing your request.

This is because the user is a private account, and it is not permitted to access her friends, and so the program will skip this account and proceed to the next public accounts through an exception handling process.

3.1.2 Retrieving the timeline of the user

For the user, when a request is sent, the retrieved information of this user step encompasses:

- English (re)tweets posted by the user.
- Retweeters list for retweets.
- ID of the source user who generate the retweet.

As in Algorithm 2, the retweeters list will be stored for later use to find from distance two, whom from the users share the same retweets as the main user, so that this will give an indication of common interest.

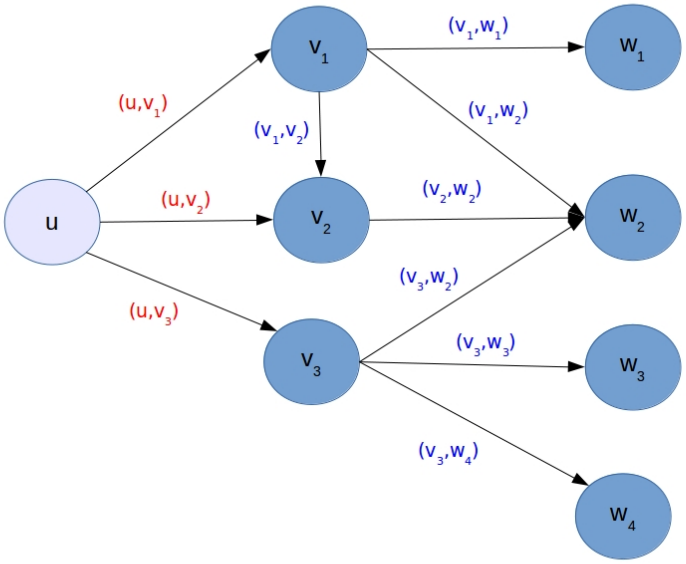

Figure 3.2. Example of friends and friends-of-friends edges in a graph.

We show in Figure 3.3 the flow of the retrieval process for the user timeline. Throughout this process, we dealt with encoding some (re)tweets in order to be stored properly. There are some issues to consider when working with timelines, such as dealing with large accounts, character encodings and socket timeout, and they are explained in Appendix C.

Algorithm 2 Downloading the time-line for a user (u)

```
/* Input: A user u.
Output: The (re)tweets posted by u, their ids, and retweets list. */
timlne ← the_timeline_of_u
RT ← A_retweet
for t_i ∈ timlne do
    txt_i ← text_of_t_i
    TID_i ← tweet_id_of_t_i
    if t_i = RT then:
        RetLst ← retweet_list_of_t_i
        RetId ← id_of_source_user
```

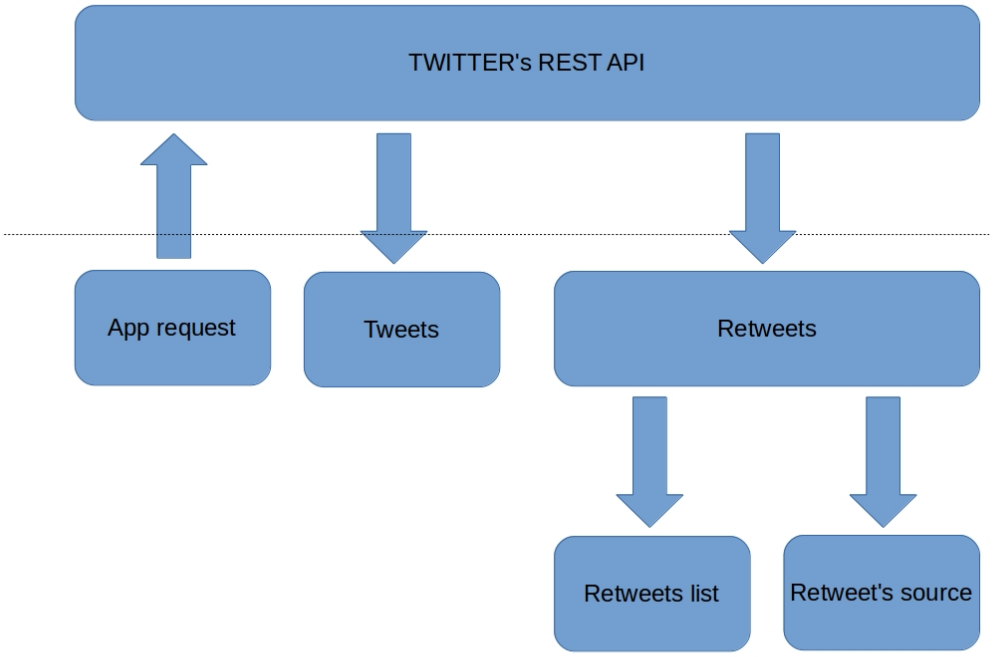

Figure 3.3. Flow diagram of the user timeline.

3.2 Network Analysis

The scenarios we have developed can be mapped to the problem of finding open triangles in the ego-network of a user. Inspired by the MapReduce approach of Suri et al. [105] for counting triangles, we designed and implemented a MapReduce algorithm to find open triangles.

3.2.1 Egocentric networks

Egocentric networks include nodes called “alters” who are connected to a central individual node called “ego”, The ties berween them are used to find the effect of alters on the ego. For example, a network of a user’s friends in Twitter would be an egocentric. Other types may extend from the ego friends to access friends of friends.

In [48], the author presents some definitions related to alters (i.e., “friends” in twitter) and their relations to each other:

- **Neighborhood**: It represents the ego and all nodes that are connected to it, which is a one-step from ego included; and it also includes all of the ties among all of the nodes that are connected to ego.
- **N-step neighborhood**: It expands the size of ego’s neighborhood by including all nodes that are connected to ego at a path length of N, and all the connections among all of these nodes.

- **“In” neighborhood**: For directed graph, an “in” neighborhood would include all the nodes who sent ties directly to ego.
- **“Out" neighborhood**: For directed graph, an “out” neighborhood would include all the nodes to whom ties are directed from ego.

In our system of recommendation, we used the “2-step-Out” neighborhood, where (“Out” neighborhood) as friends, and “2-step” neighborhood as friends-of-friends.

Mainly, egocentric networks are characterized by the number of degrees from ego as mentioned in the book: “ANALYZING SOCIAL MEDIA NETWORKS WITH NODEXL” [75]:

- The 1-degree ego network consists of the ego and their alters. 3.4 (a).
- The 1.5-degree ego network extends the 1-degree network by including connections between all of the alters (i.e., How the friends are related to each other) 3.4 (b).
- The 2-degree ego network include all of the alters’ own alters (i.e., friends of friends) 3.4 (c).

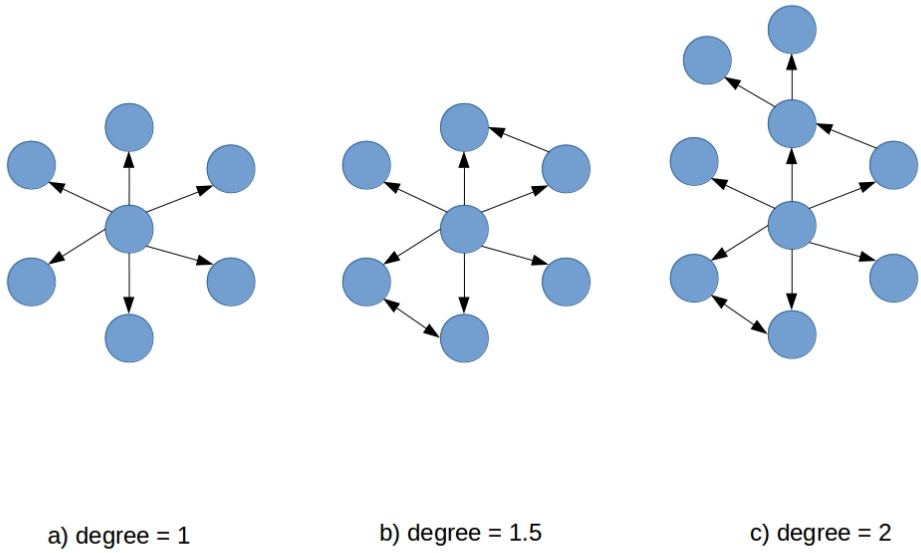

Figure 3.4. Number of degrees from ego

And again, in our system, we built a subset of the egocentric networks up to degree 2.

3.2.2 MapReduce framework

MapReduce is a programming model, and an associated implementation for processing and generating large data sets that was developed within Google, such that the computation

includes two functions Map and Reduce [34]:

- **Map**: It is written by the user that takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key (I), and passes them to the Reduce function.
- **Reduce**: It is also written by the user that accepts an intermediate key (I), and a set of values for that key, and then merges together these values to form a possibly smaller set of values.

The framework splits the data into smaller pieces that are processed in parallel on cluster of machines by the mappers. Then, the output from the mappers is compacted by reducers into desired output as in Figure 3.5 .

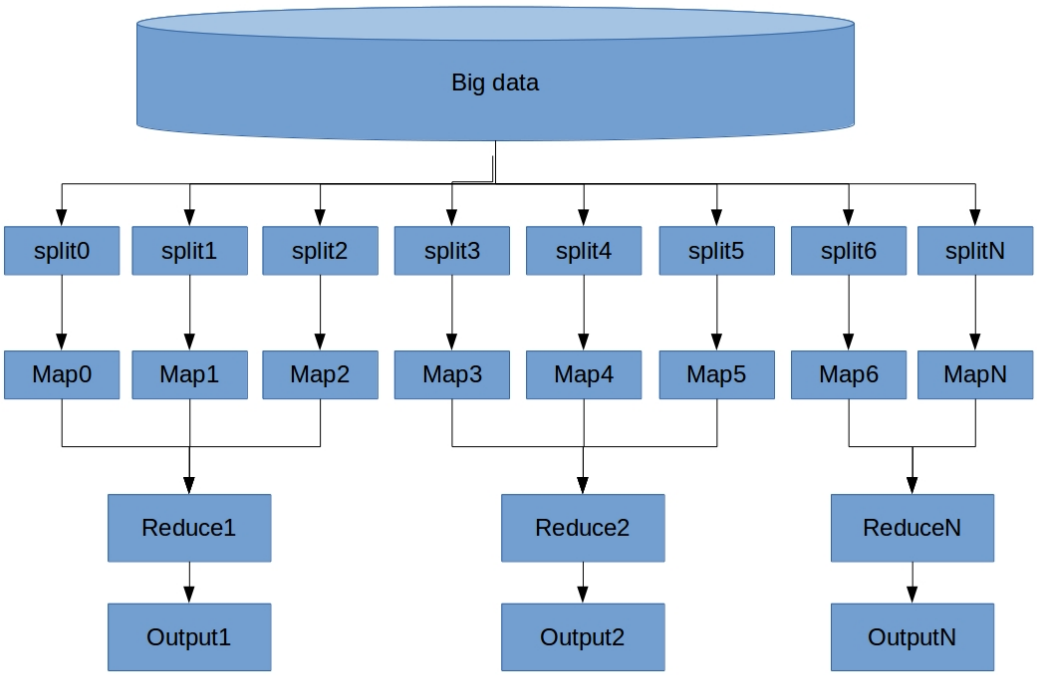

Figure 3.5. Mapreduce structure

MapReduce advantages

According to Google's developers, using MapReduce to process big data has the following benefits:

- **Simplicity**: MapReduce is considered as high level framework that keep the programmer apart form the low level details through which the programmers can write parallel programs easily.
- **Fault tolerance**: if one of the machines failed while executing a mapper, it will be assigned to another machine.

- **Locality of processing**:When running large MapReduce operations, most input data is read locally, and so, it prevents network bandwidth from being exhausted.

- **Smoothness in data distribution**: The number of mappers and reducers are more than machines, so that, it enhance dynamic load balancing, and hasten recovery when a machine fails.

3.2.3 Triangles in social networks

One of the social networks analysis tasks is triangles counting, that is considered as an essential task in graph mining in general. It is used to estimate the clustering coefficients and transitivity ratio of a graph [112].

Clustering coefficient is one of the graph properties that measures the degree to which nodes in a graph tend to cluster together. In other words, it measures the extent to which a user's friends are also friends of each other [119]. It can be computed for a node u by finding the fraction of the number of pairs of neighbors connected by edges to the number of pairs of neighbors. The clustering coefficient for a graph will be the average of the clustering coefficients for all the nodes of the graph.

3.2.3.1 Number of triangles in a graph (G)

According to the research related to finding, counting and listing triangles in large Graphs [99], and based on the following notations:

- $G = (V, E)$ is a graph with a set of nodes V and a set of edges E.

- $|n|$ is the number of nodes.

- $|m|$ is the number of edges.

Then, a triangle is a three-node subgraph of G which is fully connected. It will be clear that the number of triangles in a graph can be computed by enumerating the $\binom{n}{3}$ nodes, and count how many of them are fully connected.

In the Ph.D. thesis entitled as "Algorithmic Aspects of Triangle-Based Network Analysis" [98], the author presents the following main algorithms that count triangles efficiently with $\Theta(m^{3/2})$ running time:

- *"node-iterator-core"* algorithm: It starts from a node u that has the lowest degree by checking repeatedly whether for all the pairs of it's neighbors they belong to the edges of the graph, and then removes node u.

- *"edge-iterator"* algorithm: It goes over all edges and compares the neighborhood of the two incident nodes. For an edge $(u1, u3)$ the nodes $(u1, u2, u3)$ will form a triangle if and only if node $u2$ exists in both neighborhoods of $u1$ and $u3$.

- *"forward"* algorithm: It is a refinement of algorithm *edge-iterator*. It works by comparing a subset of the node's neighborhoods instead of all of them through pre-ordering the nodes in non increasing order of their degrees as a preprocessing step.

3.2.3.2 Triangles types in directed graphs

Given a directed graph G', then for a selected edge (n_1, n_2), there are four different kinds of directed triangles [17]:

- Cyclic: a 2-edge path from n_2 to n_1, as in Figure 3.6 (a).
- Transitive: a 2-edge path from n_1 to n_2, as in Figure 3.6 (b).
- Input: a 2-inliks to the nodes n_1, n_2, as in Figure 3.6 (c).
- Output: a 2-outliks from the nodes n_1, n_2, as in Figure 3.6 (d).

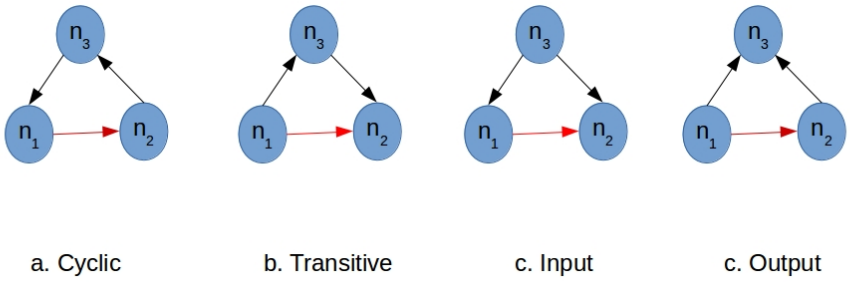

Figure 3.6. Directed graph triangles

3.2.3.3 Link recommendation based on triangles

Link recommendation is an essential application of counting triangles in online social networks as mentioned in [112], where the authors proposed a procedure to recommend the links which create as many triangles as possible.

Given the following notations:

- $G(V, E)$ is an undirected, unweighted graph.
- (u) is a node which will receive recommendation.

- $A(n)$ is the adjacency matrix of the node n.

The procedure to recommend links to the node (u) from other nodes except her neighbors, can be done as in the following:

1. Find $S = V - N$, where $N = j_1, ..., j_{d_u}$ is the set of the d_u neighbors of node u.
2. For every node $v \in S$, compute the inner product $\langle A(u), A(v) \rangle$.
3. Sort the $|S|$ inner products, and choose nodes $v_1, ..., v_k$ which result in the top-k inner products.

By this, the higher value of inner product between (u) and (v), the more common neighbors between them, and consequently the best link recommendation.

3.2.4 Our approach of finding and counting open triangles

Inspired by the MapReduce approach of Suri et al. [105] for counting triangles, we designed and implemented a MapReduce algorithm to find open triangles in the ego-network of a user.

In our algorithm, when user u follows v, and v follows z, the predicted link would be (u, z), and recommendations would go from z to u as if u is actually one of z's followers. We also count the missing edges that close triangles so as to rank the nodes at distance two from the ego based on how many incoming links they have.

Specifically, given the ego node u, let $\Gamma(u)$ be the set of u's friends (followees) and $\Gamma(\Gamma(u)) \setminus \Gamma(u)$ the set of u's friends of friends who are not friends of u. We define as the weight of user $z \in \Gamma(\Gamma(u)) \setminus \Gamma(u)$ to be the number of in-links $weight(z) = |(\Gamma(u), z)|$ as described in Algorithm 3. Nodes are then ranked based on decreasing values of their weights.

Figure 3.7 shows an example of traversing a graph to find the weights of the nodes at distance two from the ego u, as follows:

Given that:

- $\Gamma(u) = \{v_1, v_2, v_3, v_4\}$
- $\Gamma(\Gamma(u)) = \{z_1, z_2, z_3, z_4, z_5\}$
- $\Gamma(\Gamma(u)) \setminus \Gamma(u) = \{z_1, z_3, z_4, z_5\}$

Then the weights of the nodes z_i at distance two, such that the $z \in \Gamma(\Gamma(u)) \setminus \Gamma(u)$ are described as green out-links arrows to u:

- $weight(z1) = 1$
- $weight(z3) = 4$
- $weight(z4) = 2$
- $weight(z5) = 3$

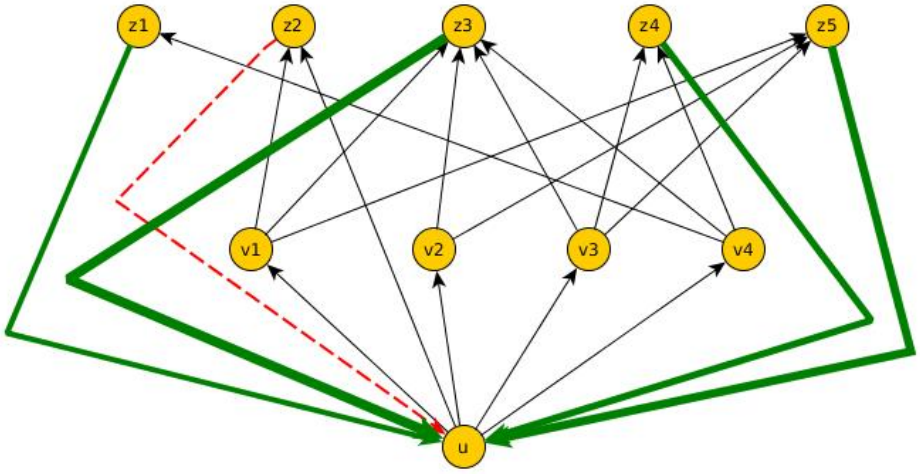

Figure 3.7. Finding and ranking missing triangles

Then, the nodes will be ranked as: z_3, z_5, z_4, z_1. However, z_2 is not considered, since even it can be reached through the friend v_1, it is a friend of u that can be reached directly. The red dashed arrow from z_2 to u indicates that the link is ignored.

In [105], the authors used MapReduce algorithms for counting triangles, where data are represented as a <key, value> pair, and passed through the following processes, that can be repeated for multiple rounds:

- **Map**: <key, value> $\rightarrow$ multiset of <key, value> pairs
- **Shuffle**: Aggregate all <key, value> pairs with the same key and it is executed by underlying system and sort them.
- **Reduce**: <key, multiset(value)> $\rightarrow$ <key, multiset(value)>

As Algorithm 3 shows, the implementation is applied in two passes:

1. **Pass 1**: Find the paths of distance two from the central node in parallel.
2. **Pass 2**: Filter off the paths that are already closed by an edge from the central node, and output the remaining ones.

The mappers and reducers in the algorithm will work as follows:

1. The first mapper identifies the set $\Gamma(ego)$.
2. The first reducer identifies the nodes in $\Gamma(\Gamma(ego)) \setminus ego$.
3. The second mapper identifies the set $\Gamma(\Gamma(ego)) \setminus (\Gamma(ego) \cup ego)$
4. The final reducer counts for each node $z \in \Gamma(\Gamma(ego)) \setminus (\Gamma(ego) \cup ego)$ how many nodes v_i such that $ego \rightarrow v_i \rightarrow z$ exist.

Algorithm 3 MR-Counting-Open-Triangles(V, E)

/* Let $G = (V, E)$ be the graph and $(u, v) \in E$ the edge from u to v. Let *ego* be the id of the ego node and let $\Gamma(v)$ be v's neighborhood */
Map 1: Input: $key, (u, v)$; *ego*
    if $u = ego$ **then**
        emit *key*, (u, v) // *key* is the default input key
Reduce 1: Input: *key*, $[(u, v1), (u, v2), \dots]$
// for each node at distance 2 from the ego, check cycles.
    for $(u, v) \in values$ **do**
        for $z \in \Gamma(v)$ **do**:
            if $u \neq z$ **then**:
                emit $(v, (u, z))$
Map 2: Input: $v, (u, z)$
/* check if the edge (u, z) that closes a triangle is open. If so, then u doesn't follow z and cannot see z's tweets. */
    if $(u, z) \notin E$ **then**:
        emit $((u, z), 1)$
Reduce 2: Input: $(u, z), [1, 1, \dots]$
/* sum the counts for each missing edge and emit a single key/value with the edge (u, z) and sum. */
    $sum \leftarrow 0$
    for $i \in values$ **do**
        $sum = sum + i$
    emit $(u, z), sum$

3.3 Content Analysis

Tweets that best match the user's interests can be discovered by applying the content similarity between a candidate tweet and the ego's (re)tweets as well as the similarity between ego's and candidate users' timelines.

3.3.1 Retrieving the profile properties and timeline of the top-k users

In this stage, from the top-k nodes that were ranked by the counting of open triangles, we downloaded their (re)tweets, and we ignored non-English tweets and outdated accounts that were not posted during the previous two months as a threshold.

In Algorithm 4, we retrieved the timeline, and some profiles properties of the top-K ranked nodes The value of k that we tried in our approach was 50. We used the function **GET users/lookup** to download the following profile properties for each user:

- User name (usr_nam).
- Profile image url (pro_pic).
- Screen name (scr_nam).
- Location (loc), this property is optional, and if it is empty, it will be set to "Unknown".

- Time zone (tim_zon).

We appended to these retrieved fields, the rank of the node (Rnk_J) from Algorithm 3 which represents the count of open triangles, that we will use later as a feature for the user.

For each user, we used the function **GET statuses/user_timeline** to download the timeline, and check if the top tweet in the timeline is created in not later than two months prior to the current date of retrieval (i.e., The value of THR is 60 days). We also reused the technique of "Working with timelines" [3] which is explained in Appendix C in order to download 40 more (re)tweets in addition to the 20 (re)tweets that can be downloaded as default. This will result in providing 60 (re)tweets in total per user. And so the information for a tweet would be:

- Tweet's text (txt_i).
- Date and time of the tweet (DaT_i).
- Tweet's language $(lang)$.

We formatted the date and time of the tweet in a way that is near to the tweets date and and time displayed in Twitter, and this was done by getting it from the timeline, and reformatting it as in the following: ('%a %b %d, %Y h. %H:%M:%S'), where:

- %a is the abbreviated weekday name.
- %b is the abbreviated month name.
- %d is the day of the month as a decimal number [01,31].
- %Y is the year with century as a decimal number.
- %H is the hour (24-hour clock) as a decimal number [00,23].
- %M is the minute as a decimal number [00,59].
- %S is the second as a decimal number [00,59].

For example, the retrieved data for a candidate user includes the fields: date, time, time zone, location and language, and it will be like the following:

Sat Oct 10, 2015 h. 04:24:09 - America/Los_Angeles timezone - Oakland, CA - en

Such that:

- *(Sat Oct 10, 2015 h. 04:24:09)* is the date and time.
- *(America/Los_Angeles timezone)* is the time zone.
- *(Oakland, CA)* is the location, (i.e., State of California. Oakland).
- *(en)* is the language, (English language).

Algorithm 4 Downloading tweets of the top-k ranked nodes

/* Input $N \leftarrow Top - k\_ranked\_list.$
Output: The timeline and profile details of these nodes. */
for $j \in N$ **do**
    $usr\_nam \leftarrow user\_name$
    $pro\_pic \leftarrow profile\_picture$
    $scr\_nam \leftarrow screen\_name$
    $loc \leftarrow location$
    $tim\_zon \leftarrow time\_zone$
    $Rnk_J \leftarrow Rank\_of\_j$
    if $loc = \phi$ **then**
        $loc =$ "$Unknown$"
    $timlne \leftarrow the\_timeline\_of\_j$
    for $t_i \in timlne$ **do**
        $cdt(t_i) \leftarrow creation\_date(t_i)$
        if $(current\_date - cdt(t_i)) <= THR$ **then**
            $txt_i \leftarrow t_i\_text$
            $DaT_i \leftarrow t_i\_date\_and\_time$
            $lang \leftarrow t_i\_language$

3.3.2 Tweets preprocessing

Tweets preprocessing is an essential part to eliminate the incomplete, noisy data that are considered unnecessary. Then, to stem the tweets' words after tokenizing them through the function *'word_tokenize'* from NLTK library in Python.

After removing non-English content, the (re)tweets are preprocessed by:

- Removing stop words like ('i', 'me', 'my', 'myself', 'we', 'our', etc). These words do not contain important implication, and they can be removed through passing the argument 'english' to the *'stopwords'* function from NLTK corpus in Python.
- Removing punctuations, and hypertext symbols (&amp, &lt, w/, etc).
- Removing retweets symbols (RT @, via @) and mentions (@username) and hashtag symbol (#).
- Removing abbreviations (I've, 'd, 'll, n't, etc).
- Removing URLs, (i.e., http, https, www).
- Removing short tweets (tweets with less than 25 characters, or those with less than 5 words, or with few nouns).

Stemming is an important step in natural language processing, that is used to convert a word to its root by removing suffixes, prefixes. The idea behind this is that the words that have the same root describe the same thing. As a result, this will reduce the computations and eliminate excessive overhead. For example, the words (collected, collection, collections, collective) have the same root (collect). We used *PorterStemmer* function in NLTK library in Python to do the process of deriving the words into their roots.

3.3.3 Content-similarity measures

As content-similarity measures we used the *cosine similarity* and *Jaccard distance*, which are typically employed for making recommendations of tweets [19], news articles [101], and research papers [84].

We computed the similarity of tweets by using single terms and bi-grams. Moreover, one type of similarity is computed between the timelines of u and of $z \in \Gamma(\Gamma(u))$, and another similarity is between the candidate tweet originated from z and u's (re)tweets. This gives us an overall number of six content-similarity features: both cosine similarity and Jaccard distance for the tweets, the timelines, and the timelines using bigrams.

3.3.3.1 Cosine similarity

Finding the similarity between two tweets is frequently used in information retrieval. The weighting heuristics used in its calculation is TFIDF.

The TFIDF weighting is the term frequency-inverse document frequency that represents a weight for ranking the words by taking in consideration the context of the text that contains this word.

- TF is the normalized term frequency as in Equation 3.1 :

$$TF(w_i) = \frac{c}{N} \tag{3.1}$$

  c : The number of times a word (w_i) occurs in the tweet t_j.
  N: The total number of words in the tweet t_j.

- IDF is the inverse document frequency, and in this case a document represents a tweet, as in Equation 3.2 :

$$IDF(w_i) = \log(\frac{T}{t_j}) \tag{3.2}$$

  T : The total number of recommended tweets from the proposed candidates.
  t_i: The number of tweets containing the given word (w_i).

In this way, rare words will get importance as well as frequent ones in the Twitter feed. By multiplying them together we will gain the TFIDF weighting, as in Equation 3.3 :

$$TFIDF(w_i) = TF(w_i) * IDF(w_i) \tag{3.3}$$

The following Equation 3.4 is used to find the cosine similarity:

$$CS(X,Y) = \frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}} \tag{3.4}$$

The attribute vectors X and Y are the TFIDF vectors of two tweets. The cosine similarity value is 1 when the two tweets are identical, and 0 if they are completely different.

3.3.3.2 Jaccard distance

Finding the distance between two tweets is computed based on the Jaccard distance measure, that is frequently used in information retrieval. It can be computed as in Equation 3.5:

$$JD(X,Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \tag{3.5}$$

The attribute vectors X and Y are the terms of the tweets. This formula is calculated through dividing the difference of the sizes of the union and the intersection of two set of tweets by the size of the union of them. The Jaccard distance value is 0 when the two tweets are identical, and 1 if they are completely different.

3.3.4 Using N-grams

Word level N-grams are widely used in recommenders systems, and natural language processing. They are mainly a sequence of consecutive fixed size of n words, where they are extracted by moving one word ahead each time. As in [14], we consider n-grams for values of n equal to 1 (uni-grams), and 2 (bi-grams). To find near-duplicate tweets, a fingerprint summary of the tweet's words will be constructed, and this fingerprint will be compared with other tweets.

The following example clarifies the concept of bigrams $(N = 2)$ for this sentence: "data mining is an interesting field". In this sentence, the bigrams would be 5:

- (data mining).
- (mining is).
- (is an).
- (an interesting).
- (interesting field).

As shown here. we moved from the first bigram (data, mining) to (mining, is) to (is, an), and so on. Basically, this is done by moving one word ahead to extract the following bigram. For tri-grams, where $(N = 3)$, the 3-grams would be 4:

- (data mining is).
- (mining is an).
- (is an interesting).
- (an interesting field).

So, when $N = 1$, the process is called unigrams, and it is basically the individual words in a sentence. When $N = 2$, it is called bigrams, and when $N = 3$ it is called trigrams. When $N > 3$ this is usually referred to as four grams (quad-grams) or five grams and so on.

To generalize, we can calculate the number of N-grams in a given sentence through the following:
Given that:

- $|W|$: the number of words in the sentence.
- N: the gram size.

Then, for a sentence s, the number of N-grams can be calculated by Equation 3.6:

$$Count\_s(Ngrams) = |W| - (N - 1) \tag{3.6}$$

For example, if we have a sentence of 15 words length, and we want to find it's quad-grams, then their count would be 12. In Algorithm 5, the tweets were sent as set of words, and the N value will be 2, so the algorithm will return list of bigrams.

Algorithm 5 N-grams tokenizer algorithm

```
/* Input: text ← set_of_words.
N ← gramsize.
Output: The list of bigrams. */
length ← len(text)
bigrams ← []
for c ∈ range(length − N + 1) do
    l ← text[c : c + N]
    bigrams.append(l)
```

This process make sense when the comparison is done between the timeline of the ego (the node to which recommendation goes) and the timeline of a candidate user that comes from the open triangles approach. This is because of the balance in the number of tweets between the users's timelines. N-grams can be tri-grams, quad-grams or more, but it was found that the best size of grams was of 2, because of the short text limitation of tweets.

And so, we computed the cosine similarity and Jaccard distance measures between the timelines of the ego and the candidate users from distance two using bigrams constructs. The following is an example of applying Jaccard distance to bigrmas:
Suppose that we have the following four sentences:

1. S1: minimum system recommended requirements.
2. S2: requirements, system recommended.
3. S3: writing comments beside commands are minimum system recommended requirements.
4. S4: writing comments beside statements make them more readable.

The bigrams will be:

1. b1 = [minimum system], [system recommended], [recommended requirements].
2. b2 = [requirements system], [system recommended].
3. b3 = [writing comments], [comments beside], [beside commands], [commands are], [are minimum], [minimum system], [system recommended], [recommended requirements].

4. b4 = [writing comments], [comments beside], [beside statements], [statements make], [make them], [them more], [more readable].

Then, the Jaccard distance will be:

1. Given that: $|b1 \cap b2| = 1$, and, $|b1 \cup b2| = 4$, then:

$$JD(S1, S2) = 1 - \frac{1}{4} = 0.75$$

2. Given that: $|b1 \cap b3| = 3$, and, $|b1 \cup b3| = 8$, then:

$$JD(S1, S3) = 1 - \frac{3}{8} = 0.625$$

3. Given that: $|b1 \cap b4| = 0$, and, $|b1 \cup b4| = 10$, then:

$$JD(S1, S4) = 1 - \frac{0}{10} = 1.0$$

4. Given that: $|b2 \cap b3| = 1$, and, $|b2 \cup b3| = 10$, then:

$$JD(S2, S3) = 1 - \frac{1}{10} = 0.9$$

5. Given that: $|b2 \cap b4| = 0$, and, $|b2 \cup b4| = 9$, then:

$$JD(S2, S4) = 1 - \frac{0}{9} = 1.0$$

6. Given that: $|b3 \cap b4| = 2$, and, $|b3 \cup b4| = 13$, then:

$$JD(S3, S4) = 1 - \frac{2}{13} \approx 0.846$$

The most similar sentences are those which have the lowest Jaccard distance (i.e., S1 and S3). In contrast, the less similar sentences are those which have the highest Jaccard distance (i.e., S1 and S4, S2 and S4).

3.4 Retweet Analysis

We include another feature based on the number of common retweets, which provides an indication of the similarity between the interests of the user u and her neighbors at distance two as shown in Algorithm 6.

Although a user's retweet will be viewed in the feed of her followers, it may reach other users who follow her followers, or their followers, etc. As a result, it has a higher probability to be reweeted from the followers of followers who find it useful. In this case, the user who share a retweet will appear in the retweeters list of the retweet status.

For example, in Figure 3.8 (a), the users $\{v1, v2, v3, v4\}$ who are a subset of z followers share retweets from z, while in Figure 3.8 (b), users $u1$ and $u2$ are followers of $v2$ and they share retweets from z through $v2$.

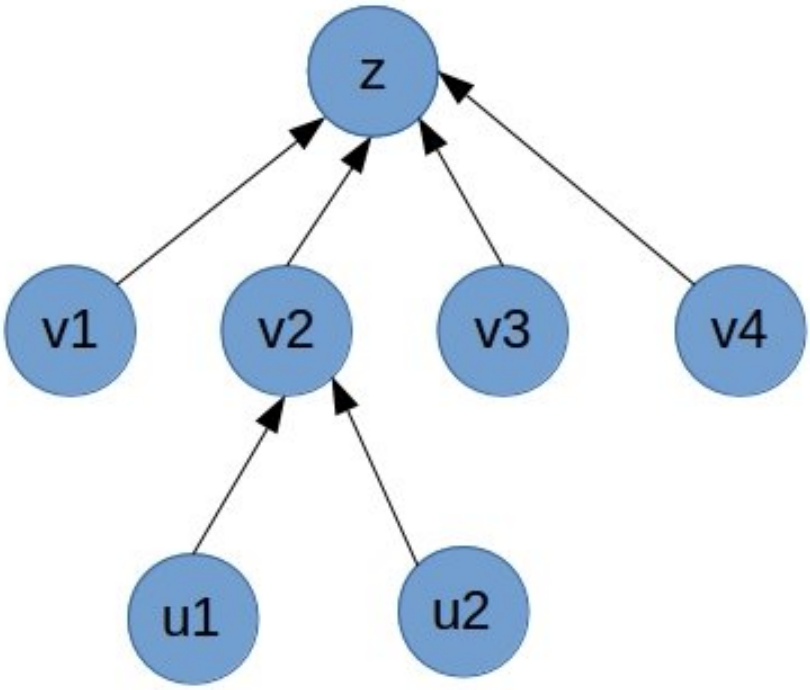

Figure 3.8. Network of retweets for two levels

Indeed, we noticed that users share on average 15 retweets, so we can use the number of mutual retweets to infer how close the users' interests are as in [29], where in this research, there is an assumption that says "users who have retweeted similar statuses in the past are likely to retweet similar statuses in the future". In this case, other related tweets which is not discovered by content-similarity methods can be found through retweet analysis.

Actually, retweeting action is a measure of personal usefulness to the user, where she read the tweet and found that its contents are interesting, and as a consequence, it is worth to spread it. The retweets count in general was also used as a feature in [95] to find the influence of user to her followers that help in deciding whether to follow her or not.

Tables 3.1 - 3.34 show the common retweets for selected users who participated in the experiment. We found that there exist strong ties between the user and the candidate users, that gives an evidence to its importance to be included as a feature in finding similarities.

Algorithm 6 Finding the common retweets between the central node (u) and a candidate user

```
/* Input: Retweeters list (RetLst) from algorithm 2, and the candidate users (N) from algorithm 3.
Output: The count of the common retweets for each of the candidate users (N) */
    for j ∈ N do
        if N[j] ∈ RetLst then
            count = count +1
```

Table 3.1. Common retweets (u1)

candidate users	common retweets
u1_v1	1
u1_v2	5
u1_v3	7
u1_v4	1
u1_v5	3
u1_v6	2
u1_v7	14
u1_v8	1
u1_v9	1
u1_v10	3
u1_v11	1
u1_v12	6
u1_v13	1
u1_v14	5
u1_v15	5
u1_v16	12
u1_v17	1
u9	14
u1_v18	7

Table 3.2. Common retweets (u2)

candidate users	common retweets
u2_v1	2
u2_v2	1

Table 3.3. Common retweets (u3)

candidate users	common retweets
u3_v1	2
u3_v2	6
u3_v3	9
u3_v4	7
u3_v5	7
u3_v6	1

Table 3.4. Common retweets (u4)

candidate users	common retweets
u1_v12	1

Table 3.5. Common retweets (u5)

candidate users	common retweets
u5_v1	1
u5_v2	1
u1	2

Table 3.6. Common retweets (u6)

candidate users	common retweets
u6_v1	5
u6_v2	4
u1	8
u6_v3	1
u6_v4	2
u6_v5	3
u6_v6	2
u6_v7	1
u6_v8	1
u6_v9	1

Table 3.7. Common retweets (u7)

candidate users	common retweets
u7_v1	1
u7_v2	1

Table 3.8. Common retweets (u8)

candidate users	common retweets
u6_v2	1

Table 3.9. Common retweets (u9)

candidate users	common retweets
u9_v1	1
u9_v2	2
u9_v3	4
u9_v4	1
u9_v5	1
u9_v6	1
u9_v7	10

Table 3.10. Common retweets (u10)

candidate users	common retweets
u10_v1	4
u10_v2	1
u10_v3	1
u10_v4	9

Table 3.11. Common retweets (u11)

candidate users	common retweets
u11_v1	1
u11_v2	2
u11_v3	1
u11_v4	1
u6_v1	5
u11_v5	1
u11_v6	1
u11_v7	2
u11_v8	1
u11_v9	1
u11_v10	4
u11_v11	3
u11_v12	6
u6_v2	2
u11_v13	1

Table 3.12. Common retweets (u12)

candidate users	common retweets
u12_v1	3
u12_v2	1
u1_v13	1
u12_v3	1
u12_v4	2
u12_v5	1
u9_v1	1
u12_v6	1
u9_v6	1
u12_v7	2
u12_v8	1
u12_v9	1
u1_v4	1

Table 3.13. Common retweets (u13)

candidate users	common retweets
u13_v1	3
u6_v1	1
u13_v2	1
u13_v3	1
u13_v4	2
u9_v7	2

Table 3.14. Common retweets (u14)

candidate users	common retweets
u14_v1	1
u14_v2	1
u12_v7	1
u9_v1	1
u14_v3	1
u14_v4	1
u14_v5	1

Table 3.15. Common retweets (u15)

candidate users	common retweets
u6_v1	5
u6_v2	1
u1	4
u15_v1	1
u15_v2	1

Table 3.16. Common retweets (u16)

candidate users	common retweets
u16_v1	1

Table 3.17. Common retweets (u17)

candidate users	common retweets
u17_v1	1
u17_v2	1
u17_v3	3
u17_v4	1
u17_v5	3

Table 3.18. Common retweets: (u18)

candidate users	common retweets
u18_v1	1
u18_v2	2
u6_v5	1
u18_v3	1
u18_v4	1

Table 3.19. Common retweets (u19)

candidate users	common retweets
u33	1

Table 3.20. Common retweets (u20)

candidate users	common retweets
u20_v1	13
u20_v2	2
u20_v3	5
u6_v1	21
u14_v3	1
u1	16
u11_v7	1
u20_v4	1
u20_v5	7
u11_v11	1
u20_v6	1
u15_v2	4
u6_v3	1
u20_v7	2
u20_v8	2
u20_v9	3
u20_v10	2
u20_v11	2
u15_v1	1
u20_v12	4

Table 3.21. Common retweets: (u21)

candidate users	common retweets
u2_v1	3
u21_v1	1
u21_v2	1
u33	1
u21_v3	1
u20_v3	1
u11_v7	1
u26	1
u21_v4	1
u21_v5	1
u6_v1	4

Table 3.22. Common retweets (u22)

candidate users	common retweets
u5_v2	1
u9	1
u22_v1	2
u20_v4	1

Table 3.23. Common retweets (u23)

candidate users	common retweets
u14_v5	2
u23_v1	1
u23_v2	1
u23_v3	2
u23_v4	1
u23_v5	1
u23_v6	1
u23_v7	1
u23_v8	2
u23_v9	1
u23_v10	2

Table 3.24. Common retweets (u24)

candidate users	common retweets
u2_v1	1
u6_v1	2
u24_v1	1
u24_v2	1
u1	8
u33	1
u6_v6	2
u14_v3	1

Table 3.25. Common retweets (u25)

candidate users	common retweets
u25_v1	1
u25_v2	1
u25_v3	1

Table 3.26. Common retweets (u26)

candidate users	common retweets
u14_v3	2
u20_v9	12
u13_v1	1
u26_v1	1
u11_v7	1
u26_v2	1
u6_v1	7
u13_v4	3
u20_v3	17
u26_v3	4
u14_v1	1
u26_v4	2
u20_v6	3
u1	42
u12_v6	1
u26_v5	8
u26_v6	1

Table 3.27. Common retweets (u27)

candidate users	common retweets
u11_v11	1
u11_v12	2
u6_v2	1
u11_v8	1
u27_v1	1
u6_v3	1
u21_v4	1
u27_v2	2
u15_v1	3

Table 3.28. Common retweets (u28)

candidate users	common retweets
u28_v1	1
u28_v2	1
u28_v3	1
u28_v4	3
u28_v5	2

Table 3.29. Common retweets (u29)

candidate users	common retweets
u25_v3	9
u29_v1	5

Table 3.30. Common retweets (u30)

candidate users	common retweets
u30_v1	1
u30_v2	1

Table 3.31. Common retweets (u31)

candidate users	common retweets
u31_v1	1
u31_v2	1
u31_v3	3
u31_v4	2

Table 3.32. Common retweets (u32)

candidate users	common retweets
u15_v2	1
u6_v1	1
u20_v3	1
u32_v1	2
u11_v1	1

Table 3.33. Common retweets: (u33)

candidate users	common retweets
u11_v1	3
u6_v1	6
u20_v7	1
u20_v9	14
u26_v2	1
u13_v1	2
u12_v6	1
u26_v3	3
u6_v3	1
u21_v3	4
u1_v13	1

Table 3.34. Common retweets (u34)

candidate users	common retweets
u20_v7	2
u20_v1	10
u20_v9	3
u34_v1	1
u26_v5	1
u20_v11	2
u34_v2	1
u6_v1	2
u34_v3	3
u34_v4	1
u26_v3	1
u34_v5	1
u9_v6	2

3.5 Ranking of Recommendations

We obtained the ranked list of tweets to recommend by using the pairwise comparison [107]. For two items, (a, b), a pairwise comparison check if item a is ranked higher than item b or not. In more detail, we create a matrix of tweets and features (we assume that all the features have the same importance) as Figure 3.9 shows.

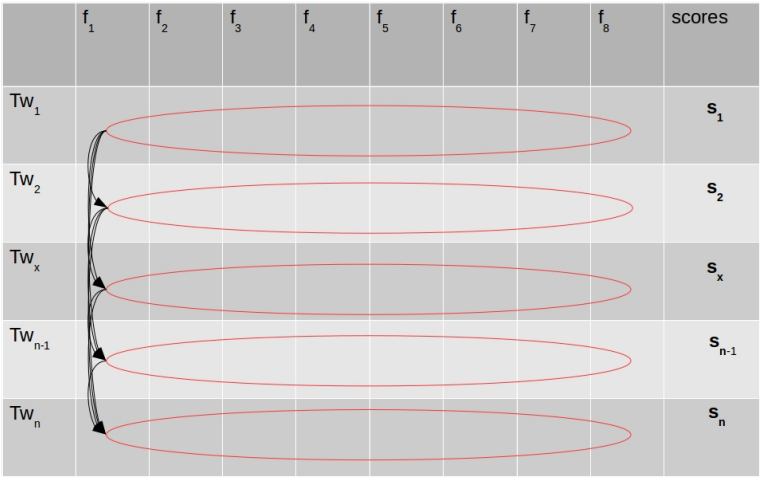

Figure 3.9. A tournament to compute the top-k tweets.

As in Algorithm 7 , we consider all pairwise combinations of candidate tweets and we compare them with respect to the features. A tweet beats another one if it has a better value for more features. At the end, each tweet has a number of wins against the rest of the tweets, which induces a ranking among all the tweets. Tweets with higher number of wins are more likely to be relevant to the user's interests and are shown on the top of the ranked list of recommendations.

Algorithm 7 The pairwise-comparison algorithm to compute the top-k tweets

```
/* Input: Tweets_features matrix (Tweets: T, Features: F).
Output: Ranked list of tweets. */
n ← count(tweets)
T_C ← combinations(n, 2)
for (t_i, t_j) in enumerate(T_C) do
    for k ∈ F do:
        if t_i[F[k]] > t_j[F[k]] then
            counts[t_i] = counts[t_i] + 1
        else:
            counts[t_j] = counts[t_j] + 1
Sort tweets based on the counts list
```

Table 3.35 shows an example of a pairwise comparison matrix for ranking 10 tweets, where each value in a cell represents the id of the tweet that has better features' values than the other one.
In this operation, the first tweet is compared to the other 9 tweets, the second is compared to the rest, and so on. Finally, they get scores accordingly.

The scores would be calculated as following:

1. The score of the 1^{st} tweet can be computed through counting its occurrences in the first row.

2. The score of the 2^{nd} tweet can be computed through counting its occurrences in the second row and the second column.

3. The score of the 3^{rd} tweet can be computed through counting its occurrences in the third row and the third column.

4. The score of the 4^{th} tweet can be computed through counting its occurrences in the forth row and the forth column.

5. The score of the 5^{th} tweet can be computed through counting its occurrences in the fifth row and the fifth column.

6. The score of the 6^{th} tweet can be computed through counting its occurrences in the sixth row and the sixth column.

7. The score of the 7^{th} tweet can be computed through counting its occurrences in the seventh row and the seventh column.

8. The score of the 8^{th} tweet can be computed through counting its occurrences in the eighth row and the eighth column.

9. The score of the 9^{th} tweet can be computed through counting its occurrences in the ninth row and the ninth column.

10. The score of the 10^{th} tweet can be computed through counting its occurrences in the tenth column.

The tweets are ranked based on these scores, where the first is the one with highest score (i.e., Tweet $\#6$), and the last one is that with lowest score (i.e., Tweet $\#5$).

Table 3.35. Pairwise comparison matrix

Tweets	1	2	3	4	5	6	7	8	9	10	Score	Rank
1		1	1	4	1	6	7	8	1	10	4	**6**
2			2	4	2	6	7	8	2	10	3	**7**
3				4	3	6	7	8	3	10	2	**8**
4					4	6	4	8	4	4	7	**3**
5						6	7	8	9	10	0	**10**
6							6	6	6	6	9	**1**
7								8	7	7	6	**4**
8									8	8	8	**2**
9										10	1	**9**
10											5	**5**

From the previous example, it is clear that we used the upper triangle of the matrix above the main diagonal, and this is because the matrix is square symmetric, such that comparing tweet X with Y is the same as comparing Y with X. And also, tweets are not comparable to itself as shown in the main diagonal. Generally, the number of comparisons is $\binom{n}{2}$, where n is the number of tweets.

Chapter 4

Experimental Results

In this chapter, we present the experiment that we performed to validate our methodology and its results. As our approach does not recommend (re)tweets that are visible to the user, because it aims at recommending concealed tweets, we could not use retweets to assess if the recommendations are interesting or not. Therefore, we conducted a user study, where we proposed to real Twitter users our recommended tweets and collected their feedback.

4.1 User Study Evaluation

In the user study, we involved 42 active Twitter users to test the quality of our recommender system by judging the tweets that are supposed to be the most interesting to them. They expressed their interaction by participating in the experiment as soon as they were invited.

4.1.1 User study design

The users could participate to our experiment by registering to our system using their screen name (the Twitter handle of the user) and email as shown in Figure 4.1 . Other optional information could also be provided:

- Real name: First name and last name.
- Gender.
- Age range: The range is divided into intervals, as their bounds like this: (<=19, 20-29, 30-39, 40-49, 50-59, >=60).
- Twitter usage: It asks for how often do the user login in Twitter. And the scale of the response is: (every day, once in a week, once in a month, and infrequently).
- Internet usage: It asks for how often do the user connected to Internet. The scale of the response is also: (every day, once in a week, once in a month, and infrequently).

The user can select her interests from a set of check-boxes, or provide them in a text area if they were not listed in the check-boxes.

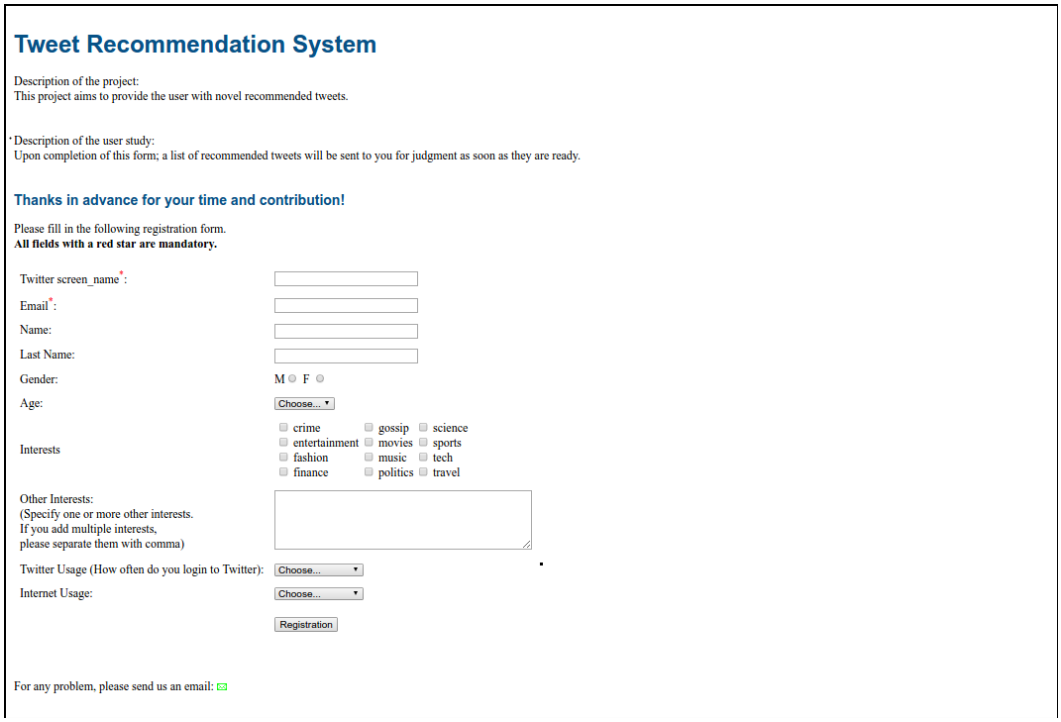

Figure 4.1. Registration form of the experiment

4.1.2 Demographic information of the participants

The participants of our user study were recruited from the list of the authors' followers [8]. Most of them are young (between 20 and 30 years old) as presented in the bar chart of Figure 4.2:

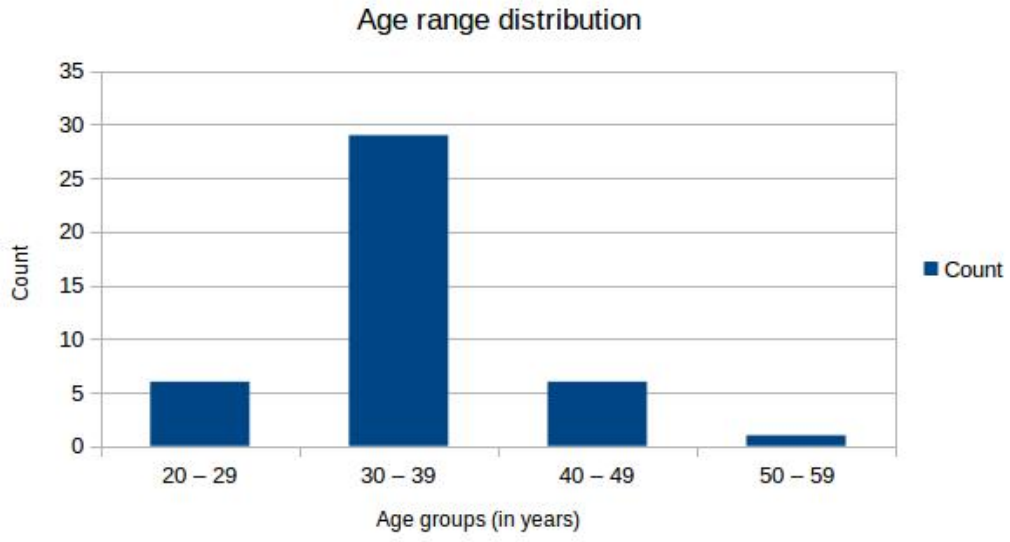

Figure 4.2. Age range distribution among the participants

The users who participated in the experiment are researchers, coming from different countries, affiliations, and research areas. The selected users are very active online (all of them use Internet everyday and 81% of them access Twitter frequently) as presented in the bar chart of Figure 4.3.

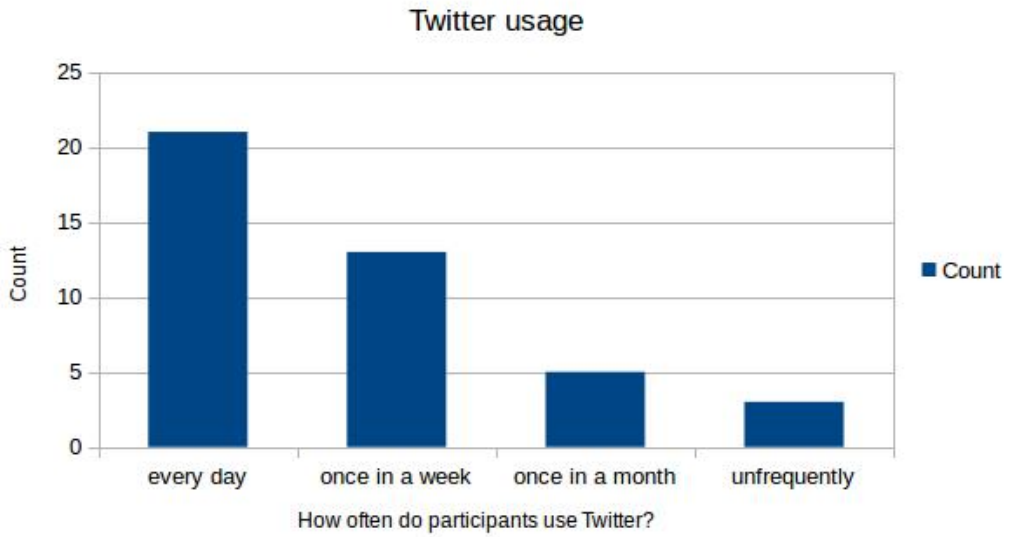

Figure 4.3. Twitter usage of the participants

After the registration, the user will receive an acknowledgment of successful registration, and then, the system will retrieve the user's information needed to make the recommendations, and once these were ready, the user was notified by an email and could rate the list of recommended tweets by logging into the system through the screen name as in Figure 4.4.

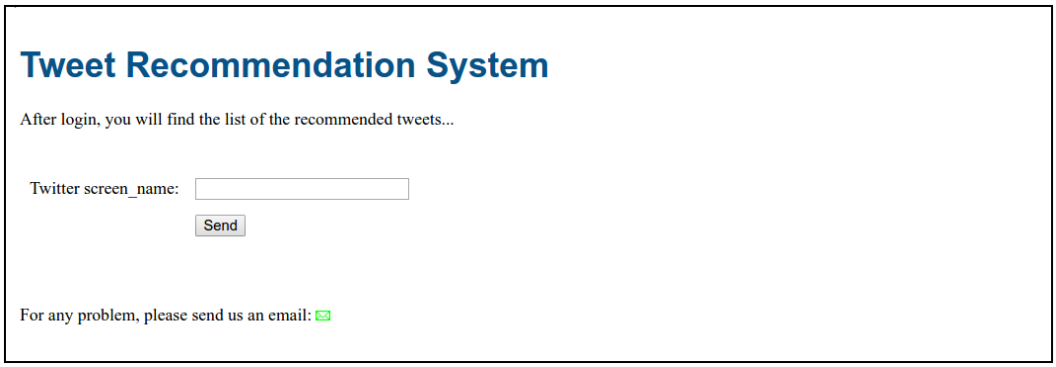

Figure 4.4. User's login screen

4.1.3 Characterization of the participants

The participants are active on Twitter in term of the frequency of tweets / retweets they posted. In Figures 4.5 and 4.6 , the participants are grouped based on the count of their (re)tweets into five classes:

- Greater than or equal to 1 thousands (re)tweets, $(>= 1000)$.
- Greater than or equal to 500 and less than 1 thousands (re)tweets, $(>= 500 \;\&\&\; < 1000)$.
- Greater than or equal to 100 and less than 500 (re)tweets, $(>= 100 \;\&\&\; < 500)$.
- Greater than or equal to 50 and less than 100 (re)tweets, $(>= 50 \;\&\&\; < 100)$.
- Less than 50 (re)tweets, (< 50).

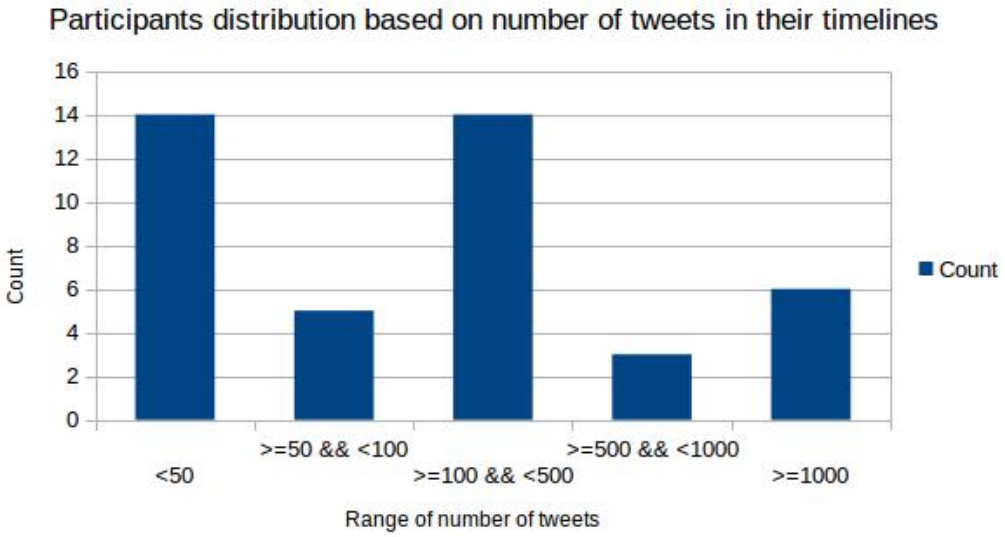

Figure 4.5. Participants groups according to their activity of tweeting in their timelines

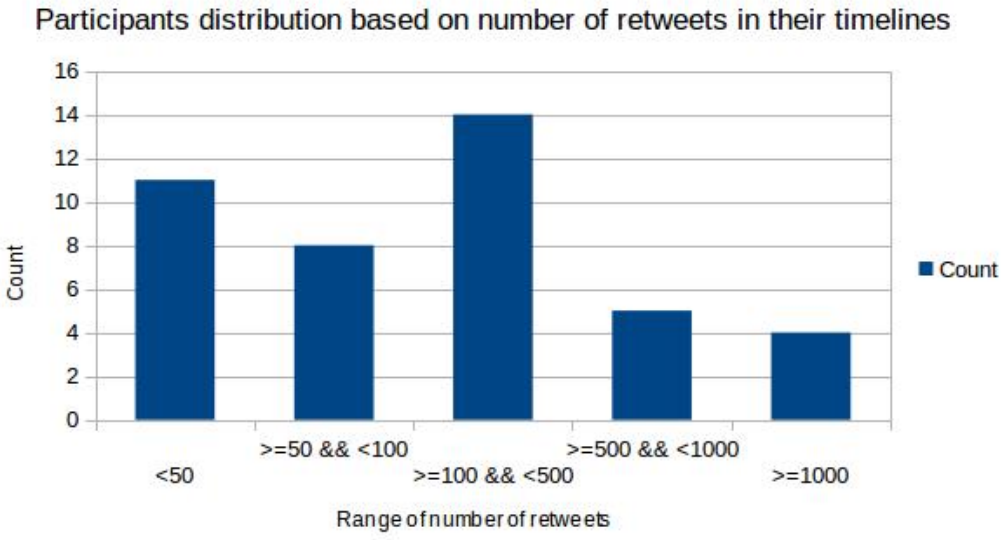

Figure 4.6. Participants groups according to their activity of retweeting in their timelines

The percentage of participants who have more than 50 tweets in their timelines is 67% as in Figure 4.5, and the percentage is 74% for the participants who have more than 50 retweets as in Figure 4.6.

Figure 4.7 shows the sizes of the 2-hop ego networks of the participants, in which they are grouped into five classes:

- Greater than or equal to 1 million edges, $(>= 1M)$.
- Greater than or equal to 500 thousands and less than 1 million edges, $(>= 500K \;\&\&\; < 1M)$.
- Greater than or equal to 100 thousands and less than 500 thousands edges, $(>= 100K \;\&\&\; < 500K)$.
- Greater than or equal to 50 thousands and less than 100 thousands edges, $(>= 50K \;\&\&\; < 100K)$.
- Less than 50 thousands edges, $(< 50K)$.

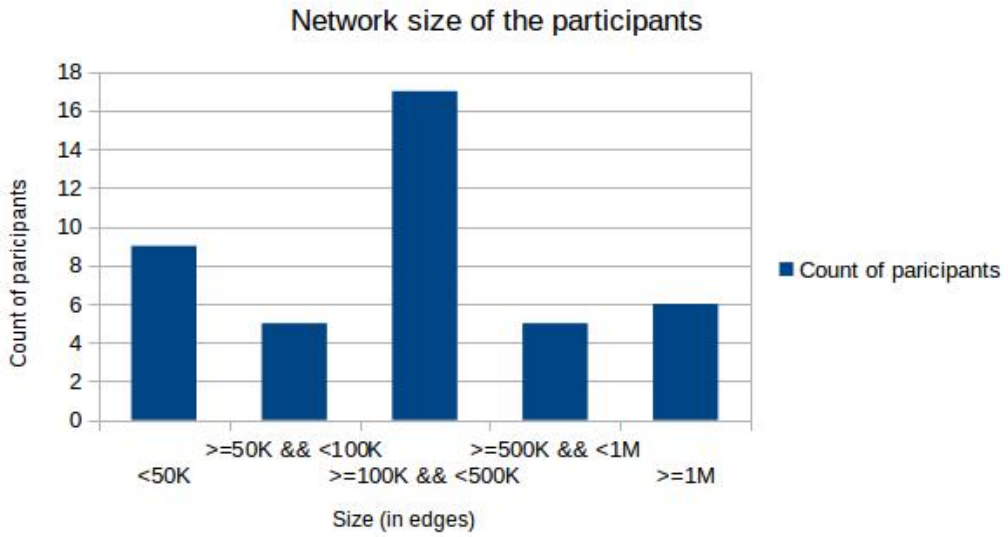

Figure 4.7. Network size of the participants

From Figure 4.7, it is clear that the percentage of the participants whom their network size is greater than or equal to 50 thousands is 78.6%, and that the major class of them have their network size between 100 thousands and half million with a percentage of 40.5%

4.1.4 Tweets rating

A total of 420 tweets from Oct. 7 to Nov. 17, 2015 were rated by our users. We compare the precision of our recommendations against a baseline approach for recommending concealed tweets. Since our aim is to recommend not visible tweets, we cannot compare our performance against algorithms that re-rank tweets appearing in the user's timeline. As a

baseline, we used the approach presented by Pennacchiotti et al. [87], which exploits the content similarity among tweets, and, to the best of our knowledge, it is the only work on recommendation of unseen tweets.

We adopted their approach based on the cosine similarity, and for the evaluation we proposed the top-5 recommended tweets from our approach and the top-5 recommendations from the baseline. The two rankings were presented to the users, in a way that the user could not identify what system was used for creating the corresponding ranking. Following the same experiment of [87], our users could rate the proposed recommendations using a four-grade scale:

- *Excellent* (the tweet is very interesting/informative w.r.t. her interests),
- *Good* (the tweet is interesting/informative w.r.t. her interests),
- *Fair* (the tweet is somehow interesting/informative w.r.t. her interests),
- *Bad* (the tweet is not interesting/informative at all).

Figure 4.8 shows an example of 10 recommended tweets for the user (u13).

4.2 Candidate Users Timelines

The candidate users are the users at distance two from the participants. A large portion of them have similar interests, where their timelines have a high percentage of common retweets, or their tweets were retweeted by the participants, and most of their (re)tweets were written in English.

4.2.1 Tweets and retweets of candidate users

In the experiment, we noticed that most of the participants had retweeted retweets from candidate users at distance two with a percentage of 81%, and that the percentage of participants who had retweeted a tweet originated form candidate users at distance two is 69% as in Figure 4.9 and Figure 4.10 respectively.

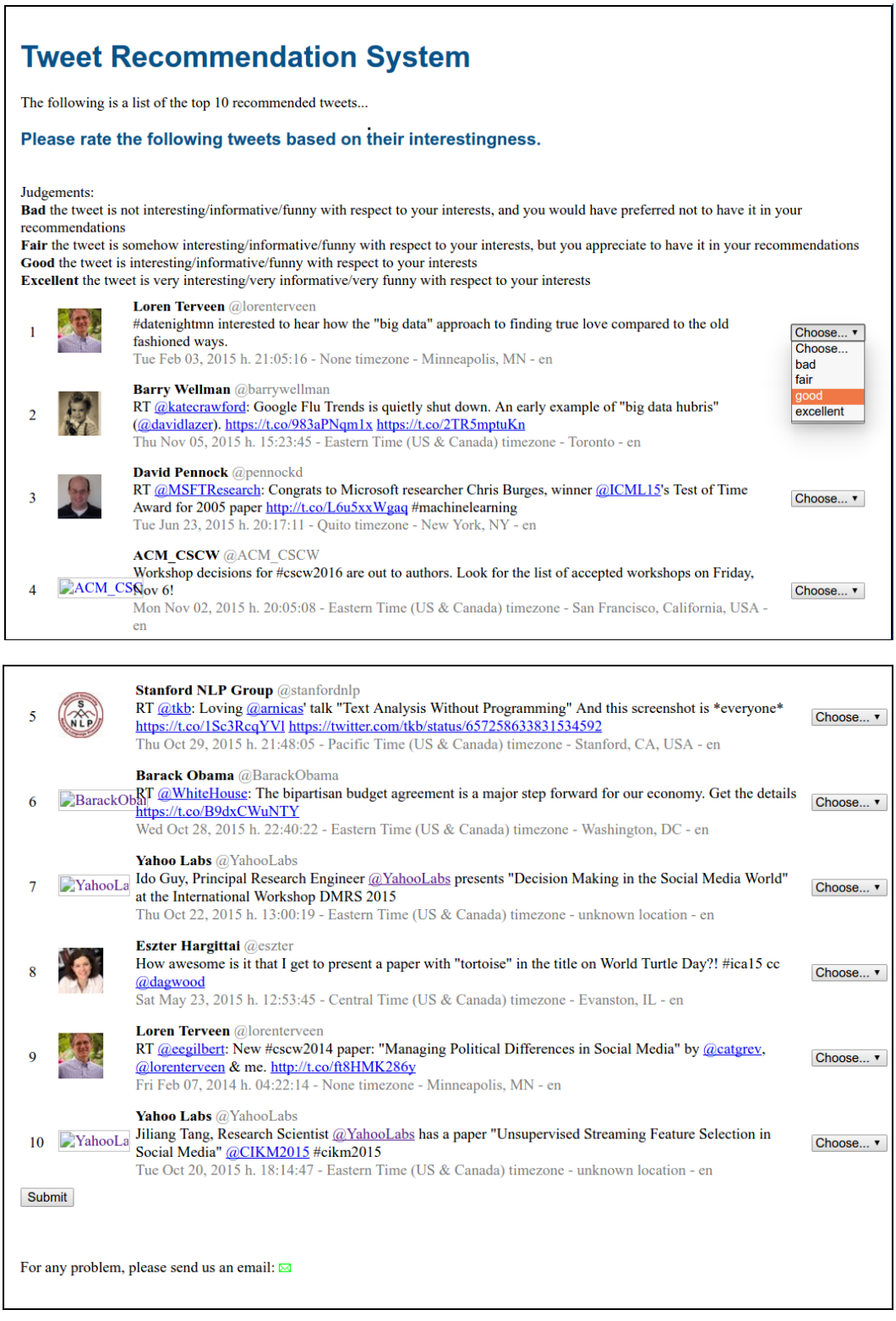

Figure 4.8. A list of 10 recommended tweets to the users (u13).

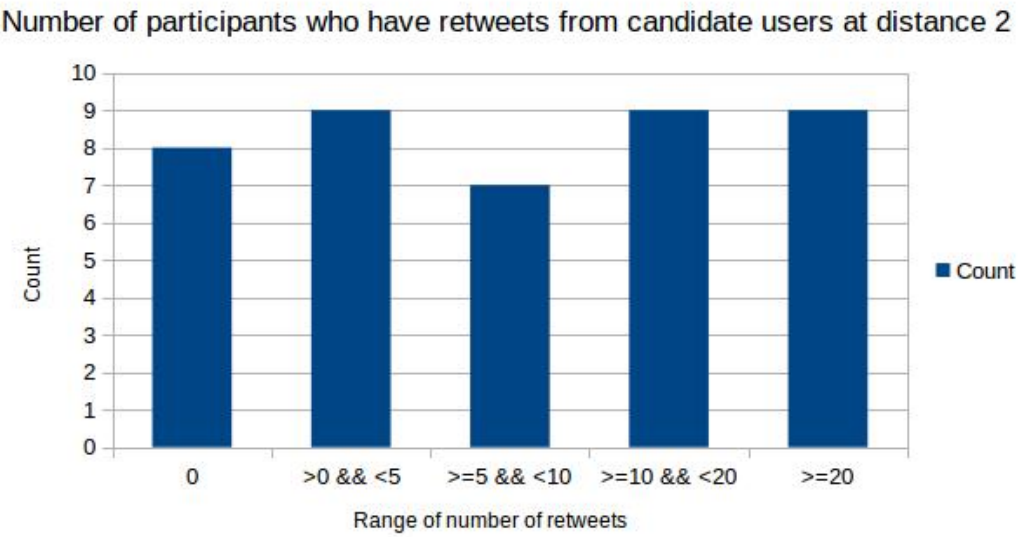

Figure 4.9. Number of participants who have retweets from candidate users at distance 2

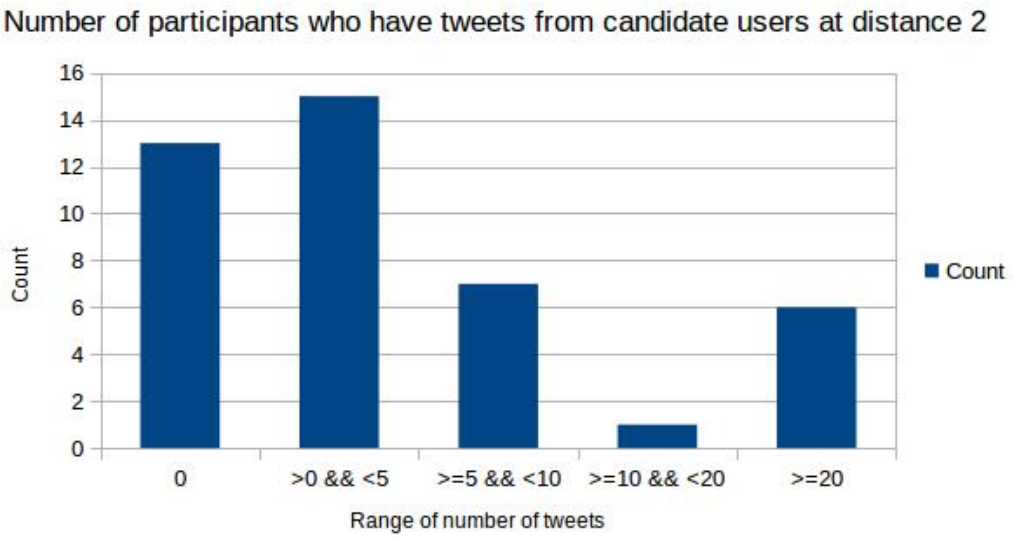

Figure 4.10. Number of participants who have tweets from candidate users at distance 2

As shown in Figures 4.9 and 4.10, about 60% of the participants had at least one retweet and less than 20 retweets from candidate users at distance two. This percentage is about 55% for tweets from candidate users at distance two.

Figure 4.11 shows that the candidate users may contain:

- Tweets that were tweeted by the ego.
- Retweets that were retweeted by the ego.

- Tweets and retweets that were retweeted by the ego.

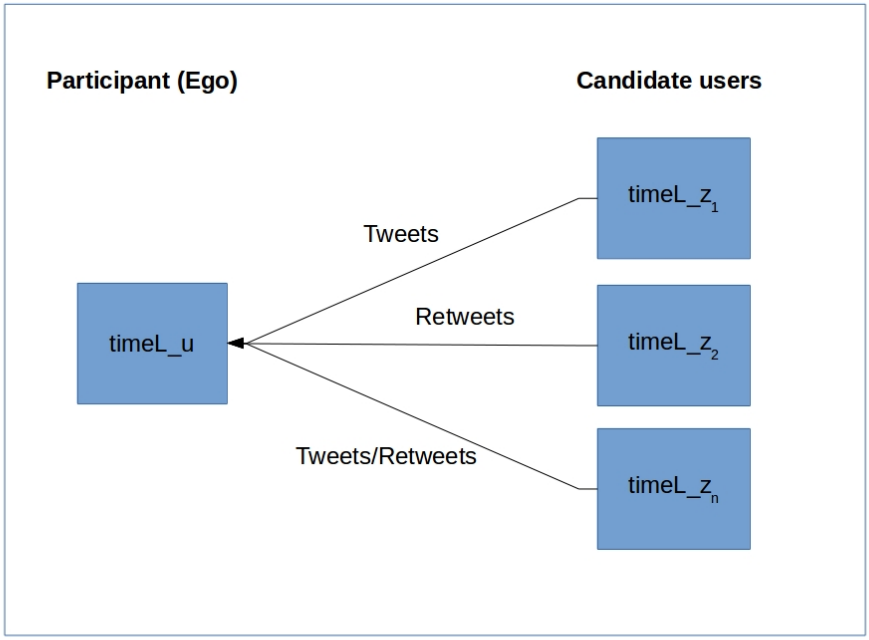

Figure 4.11. (Re)tweets of ego that were posted in the candidate users' timelines

Figures 4.12 - 4.21 show tweets and retweets of candidate users for 10 participants. For example, in Figure 4.12, the participant (u1) had retweeted:

- 10 tweets from the candidate user 'WIRED'.
- 1 tweet from the candidate user 'MIT Tech Review'.
- 2 tweets and 5 retweets from the candidate user 'Randy Olson'.

It is clear from the figure that the candidate users 'WIRED', 'MIT Tech Review', and 'Randy Olson' appear in the seventh, eighth, and ninth locations of the candidates list.

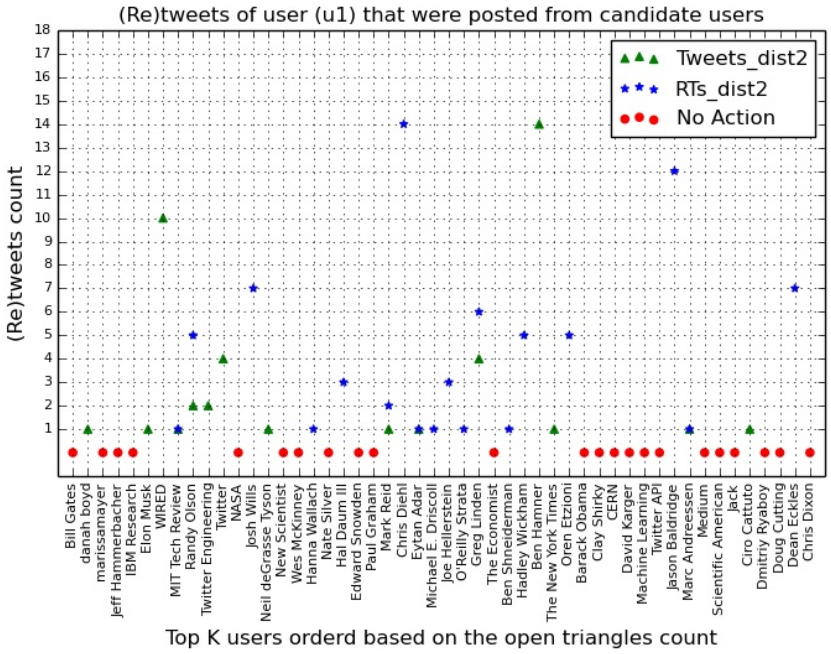

Figure 4.12. (Re)tweets of user (u1) that were posted from candidate users

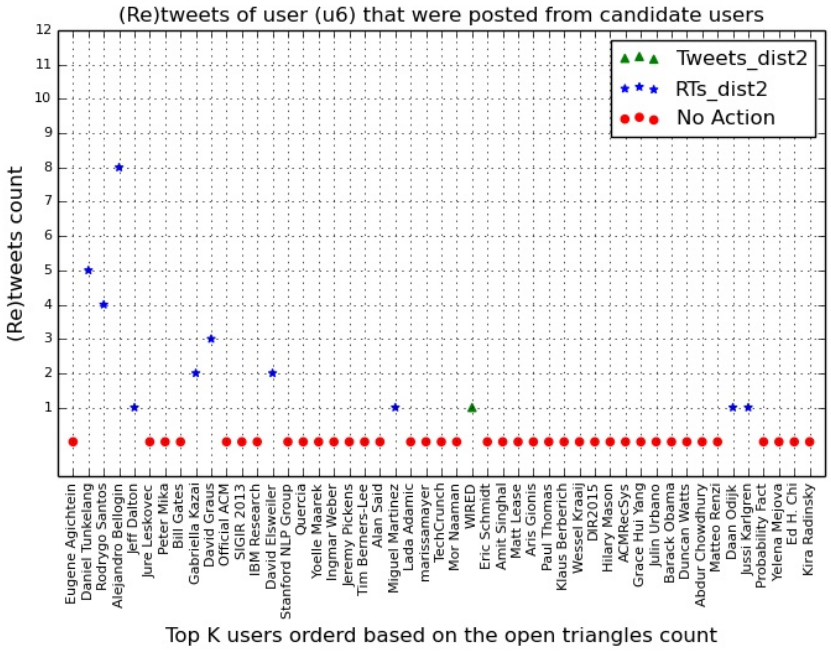

Figure 4.13. (Re)tweets of user of user (u6) that were posted from candidate users

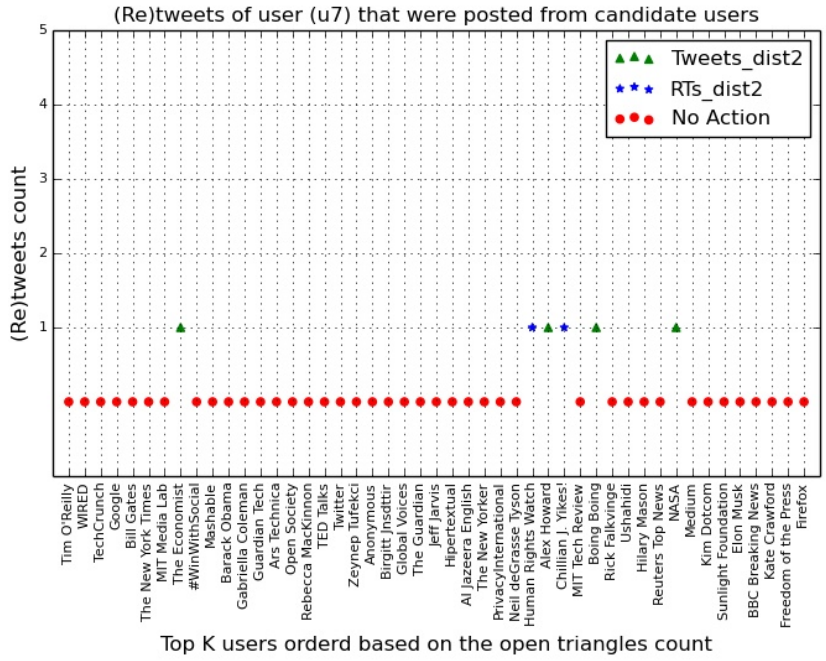

Figure 4.14. (Re)tweets of user (u7) that were posted from candidate users

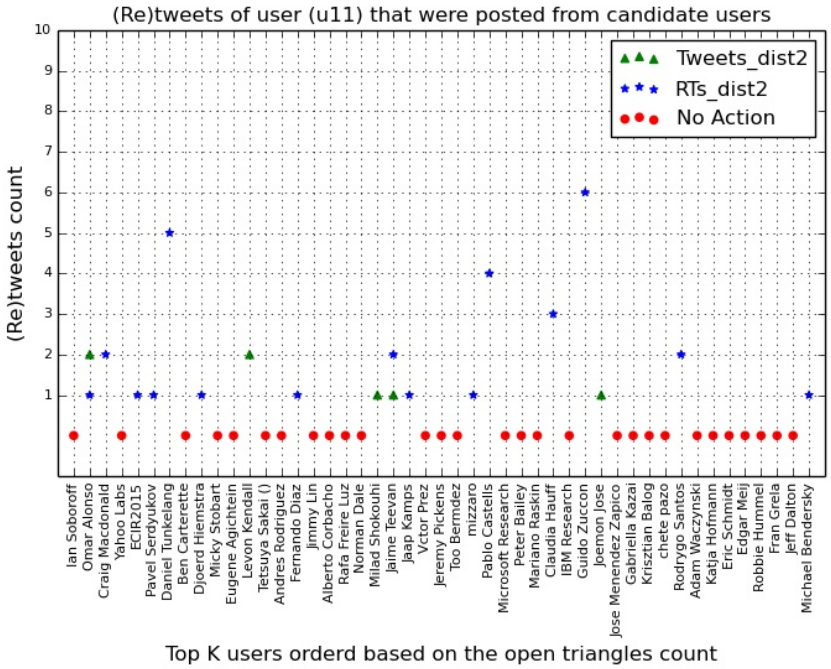

Figure 4.15. (Re)tweets of user (u11) that were posted from candidate users

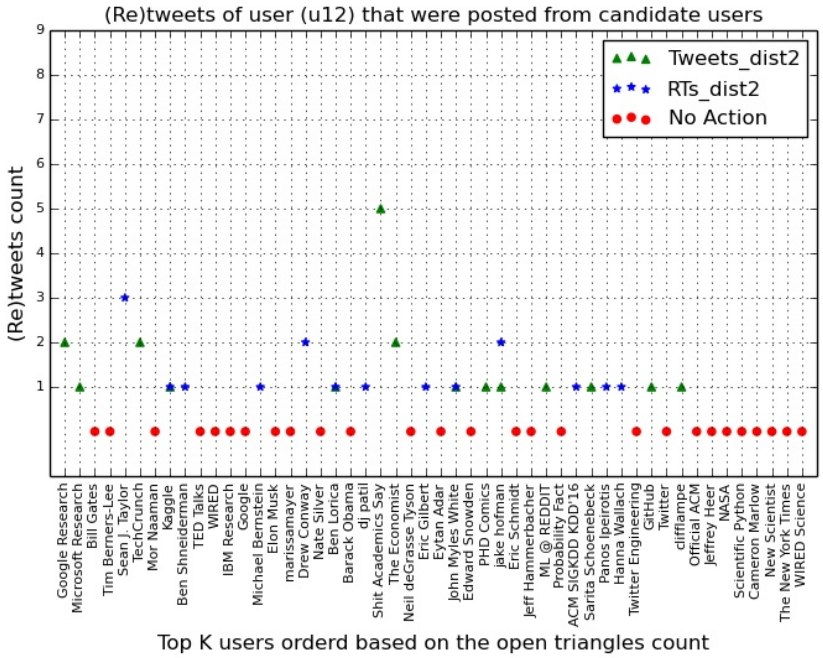

Figure 4.16. (Re)tweets of user (u12) that were posted from candidate users

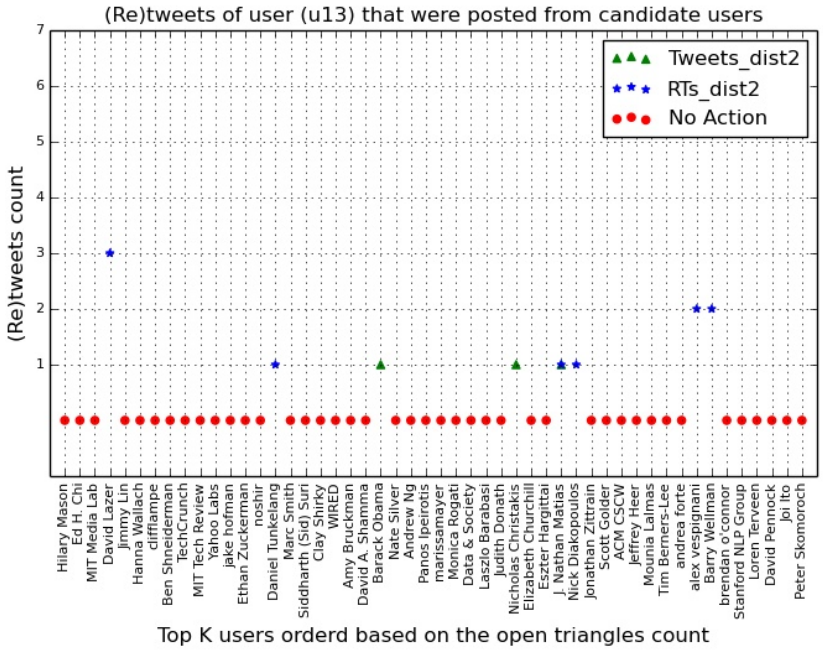

Figure 4.17. (Re)tweets of user (u13) that were posted from candidate users

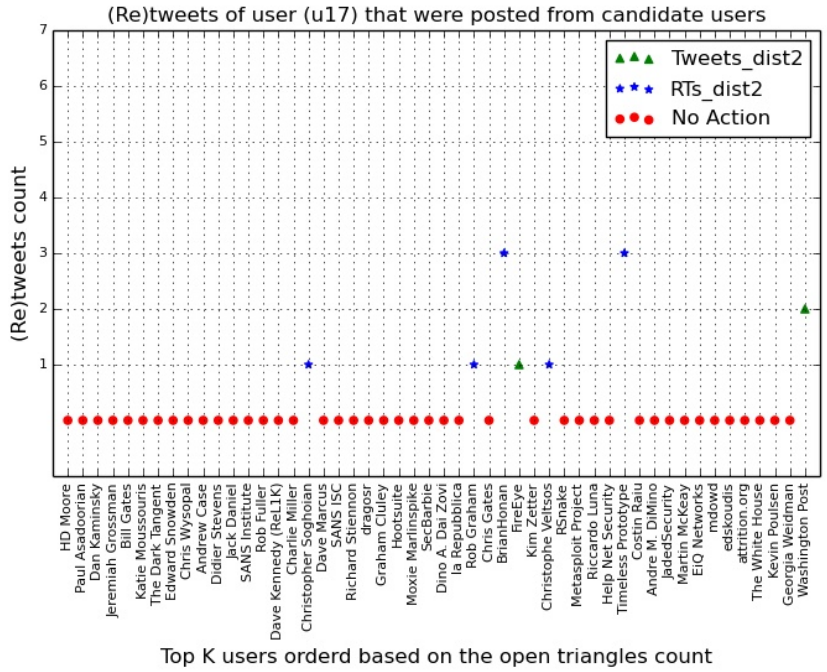

Figure 4.18. (Re)tweets of user (u17) that were posted from candidate users

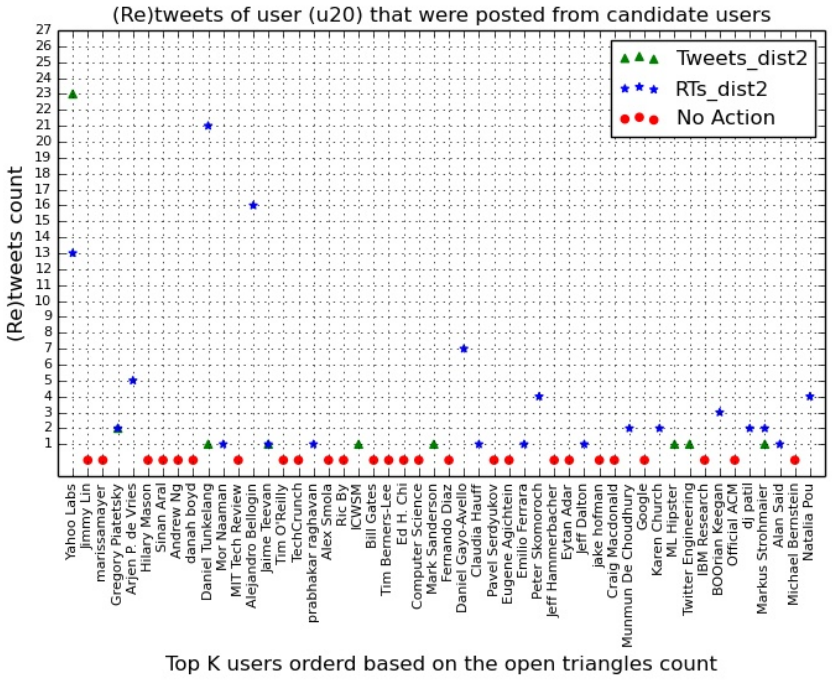

Figure 4.19. (Re)tweets of user (u20) that were posted from candidate users

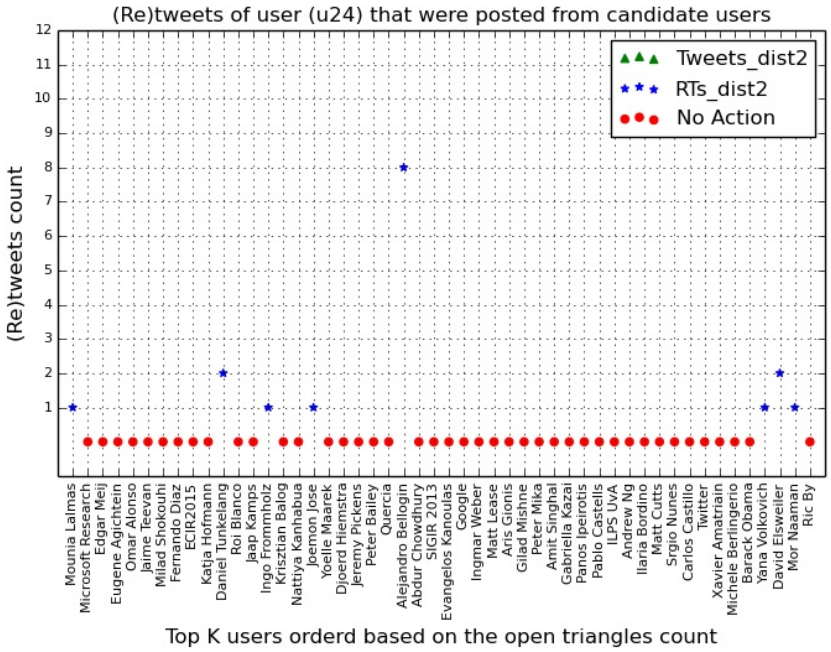

Figure 4.20. (Re)tweets of user (u24) that were posted from candidate users

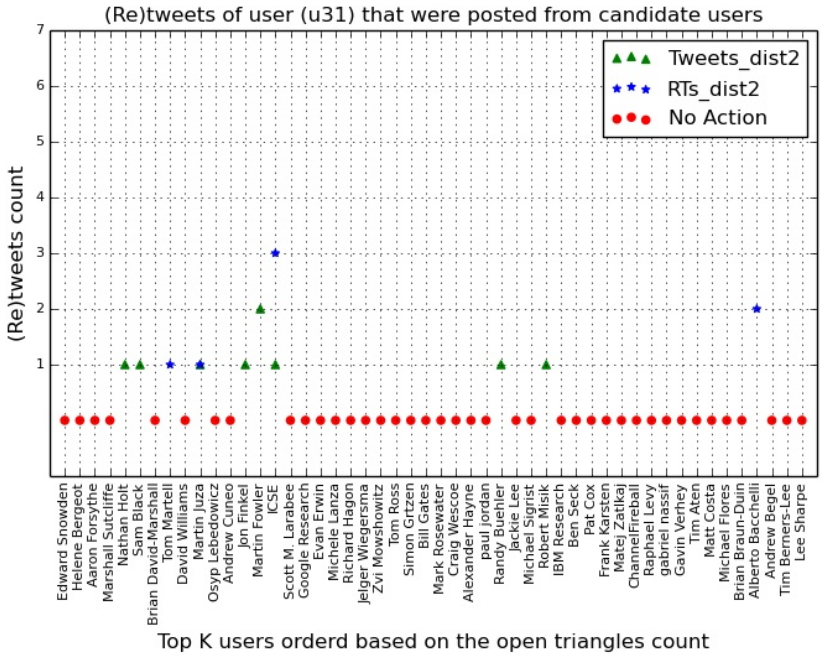

Figure 4.21. (Re)tweets of user (u31) that were posted from candidate users

4.2.2 Outdated candidate users

Most of the candidate users at distance two are active, such that the other outdated accounts were removed from the retrieval process as mentioned in subsection 3.3.1. Figure 4.22 shows that 40.5% of ego users do not have outdated accounts in the top-50 retrieved candidate users. In general, about $(\frac{2}{3})$ of ego users have at most one outdated candidate account in the top-50 retrieved candidate users.

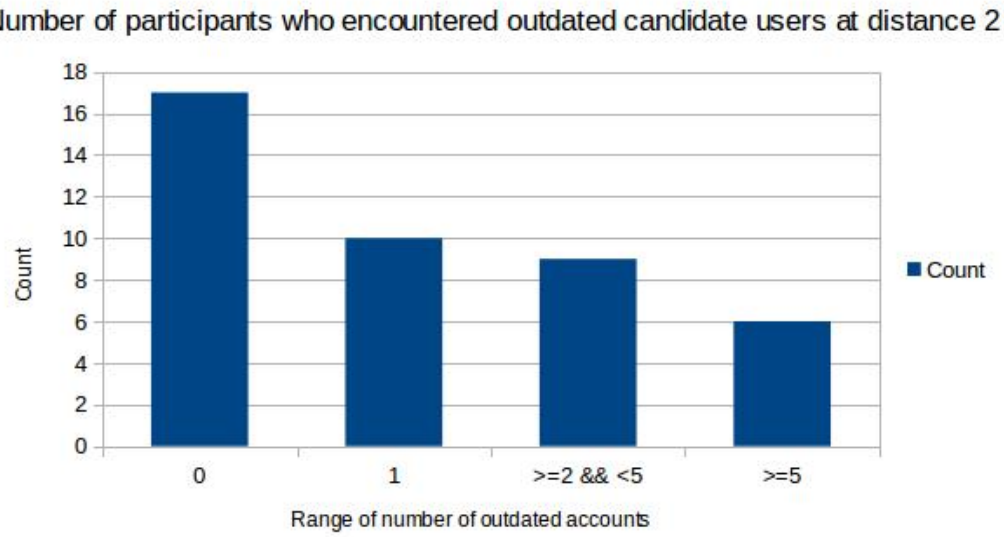

Figure 4.22. Number of participants who encountered outdated candidate users at distance 2

4.2.3 Preprocessed (re)tweets of candidate users

Throughout the preprocessing step as in subsection 3.3.2 , we noticed that the (re)tweets of candidate users were mostly written in English, such that more than $(\frac{3}{4})$ of the participants retrieved and stored at least $(\frac{2}{3})$ of the candidate users' timelines as in Figure 4.23. Also, most of the (re)tweets of candidate users were retrieved and stored to be recommended later, where more than $(\frac{3}{4})$ of the participants filtered out at most 700 (re)tweets from the candidate users' timelines as in Figure 4.24.

4.3 Assessing the Performance of the Recommendation System

For assessing the performance of our recommendation algorithm, we computed the following measures:

- Precision.
- Normalized Discounted Cumulative Gain (nDCG).
- Reciprocal Rank (RR).

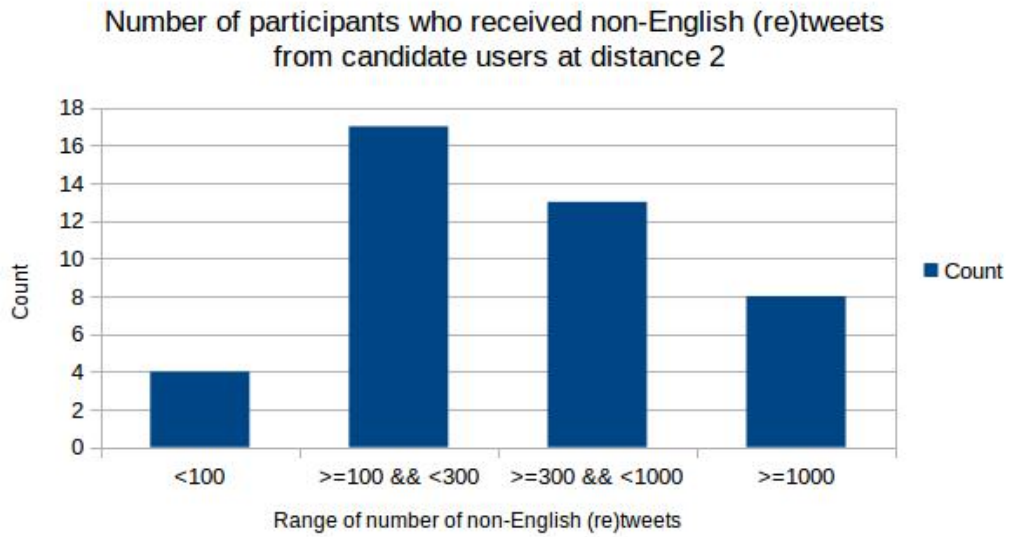

Figure 4.23. Number of participants who recieved non-English (re)tweets from candidate users at distance 2

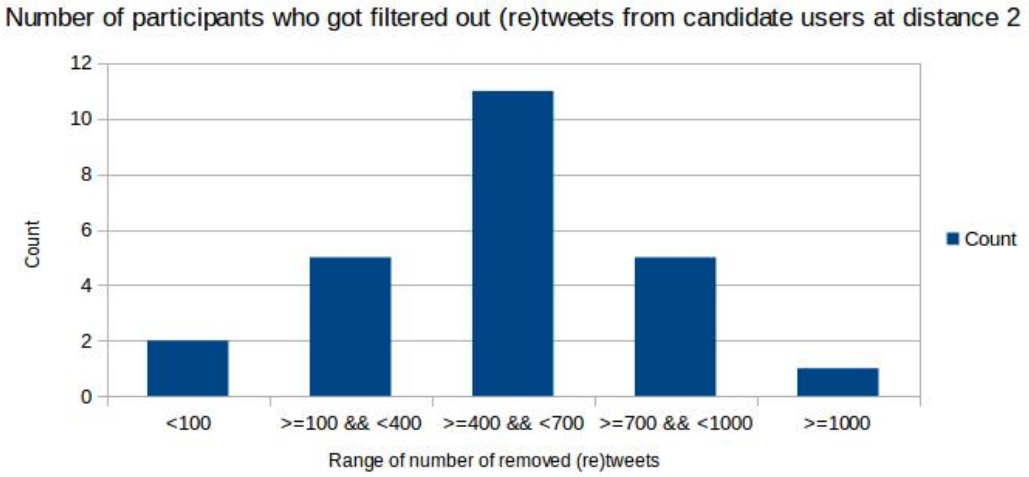

Figure 4.24. Number of participants who got filtered out (re)tweets from candidate users at distance 2

And we evaluate our approach against the baseline based on these measures. We found that our approach have higher values.

4.3.1 Precision

The precision at k metric (p@k) is the percentage of relevant tweets found in the top-k ranked tweets as in Equation 4.1.

$$Precision = \frac{Relevant\_recommended\_tweets\_in\_top - k}{k\_recommended\_tweets} \tag{4.1}$$

To apply this measure, we casted the four-grade scale to a binary score (1 for interesting and 0 for uninteresting). In particular, user's answers *Excellent*, *Good*, and *Fair* correspond to 1 and *Bad* to 0. We obtain similar results if we map *Excellent* and *Good* to 1 and *Fair* and *Bad* to 0.

4.3.2 Normalized Discounted Cumulative Gain (nDCG)

It measures the performance of a recommendation system based on the graded relevance of the recommendations. We used this metric to asses the effectiveness of our recommendation algorithm using the actual non-binary rates. It compares the ranking of tweets based on relevance scores:the recommendation scores and the ranking based on the user grades.

Equation 4.2 shows this measure which is considered as a favored metric because it includes the order of the retrieved tweets. In this equation, "Discounted Cumulative Gain" (DCG) can be computed by Equation 4.3, where:

- rel_i: is the graded relevance of the result at position i.
- Ideal DCG ($IDCG$): is the ideal result list which was sorted by relevance.

$$nDCG_k = \frac{DCG_k}{IDCG_k} \tag{4.2}$$

$$DCG_k = \sum_{j=1}^{k} \frac{2^{rel_j} - 1}{\log_2(j+1)} \tag{4.3}$$

4.3.3 Reciprocal Rank (RR)

The Reciprocal Rank (RR), is also a measure of the performance of top-k recommendation, through finding how early in the proposed list the *first* relevant recommended tweet is ranked, which is the inverse of the ranking position of the first relevant tweet, as in equation 4.4 .

$$RR = \frac{1}{Rank\_of\_highest\_ranking\_relevant\_tweet} \tag{4.4}$$

Where: a RR value equal to 1 is the best, and 0 is the worst. If none of the recommendations is correct, then the reciprocal rank will be set to 0.

4.3.4 Results and discussion

We calculated the average of the precision and nDCG metrics over all the users who subscribed to the system. In Table 4.1, we show a comparison of our approach and the baseline for the average precision and nDCG of the top-k tweets (with $k = 1, ..., 5$). Compared to the baseline we could observe an average improvement of 12.4% for Precision@k and of 1.6% for nDCG.

Table 4.1. Comparison between our approach and the baseline.

Precision					
	p@1	p@2	p@3	p@4	p@5
Our Approach	**0.85**	**0.82**	**0.80**	**0.79**	**0.80**
Baseline	0.73	0.67	0.67	0.68	0.69
nDCG					
	@1	@2	@3	@4	@5
Our Approach	**0.75**	**0.79**	**0.84**	**0.88**	**0.92**
Baseline	0.74	0.77	0.82	0.86	0.91

We run t-test and could observe that the results were statistically significant at $p < 0.1$ for $p@1$ and at $p < 0.05$ for the other precisions as shown on Table 4.2.

Table 4.2. t_test for the top-k recommendations

Top-K	Mean Difference	t_value	P	Statistically sig.
top-1	0.122	1.36	0.087	<0.10
top-2	0.146	2.01	0.023	<0.05
top-3	0.138	2.07	0.021	<0.05
top-4	0.116	1.95	0.027	<0.05
top-5	0.102	1.78	0.039	<0.05

We computed the following statistics of the top-5 ranked tweets for our approach and the baseline as following:

- **Minimum**: The least value in the data.
- **First quartile (Q1)**: 25% of data fall below the lower quartile.
- **Median (Q2)**: 50% of all data are below or above it.
- **Third quartile (Q3)**: 75% of all data are below the upper quartile.
- **Maximum**: The greatest value in the data.

The range of scores from lower to upper quartile is referred to as the inter-quartile range (IQR).

In Table 4.3, the statistics for the top-1 recommended tweets show that 25% of the data are less than 1 for our approach, while 25% of the data are less than 0 for the baseline, which gives an evidence of the goodness of our approach.

Table 4.3. The statistics of both methods for the top-1 recommended tweets

Stats	Our Approach	Baseline
Minimum	0	0
Q1	1	0
Median	1	1
Q3	1	1
Maximum	1	1

In Table 4.4, the statistics for the top-2 recommended tweets show that half of the data are less than 1 for our approach, while for the baseline half of the data are less than 0.5, so that our approach is better too.

Table 4.4. The statistics of both methods for the top-2 recommended tweets

Stats	Our Approach	Baseline
Minimum	0	0
Q1	0.5	0.5
Median	1	0.5
Q3	1	1
Maximum	1	1

In Table 4.5, the statistics for the top-3 recommended tweets show that 25% of the data are less than 0.667 for our approach, while 25% of the data are less than 0.333 for the baseline. And that half of the data are less than 1 for our approach, while for the baseline half of the data are less than 0.667, so that our approach still dominates as the best one.

Table 4.5. The statistics of both methods for the top-3 recommended tweets

Stats	Our Approach	Baseline
Minimum	0	0
Q1	0.667	0.333
Median	1	0.667
Q3	1	1
Maximum	1	1

In Table 4.6, the statistics for the top-4 recommended tweets show that the average minimum value is 0.25 for our approach, while it is 0 for the baseline.

Table 4.6. The statistics of both methods for the top-4 recommended tweets

Stats	Our Approach	Baseline
Minimum	0.25	0
Q1	0.5	0.5
Median	0.75	0.75
Q3	1	1
Maximum	1	1

In Table 4.7, the statistics for the top-5 recommended tweets show that the average minimum value is 0.2 for our approach, while it is 0 for the baseline.

Table 4.7. The statistics of both methods for the top-5 recommended tweets

Stats	Our Approach	Baseline
Minimum	0.2	0
Q1	0.6	0.6
Median	0.8	0.8
Q3	1	1
Maximum	1	1

We also computed the Reciprocal Rank (RR), such that we considered the tweet which has been rated as *Excellent*, *Good*, or *Fair* by the user as the first relevant tweet. The average RR over all users is of 91% for our approach and 83% for the baseline.

We report in Figure 4.25 the summary statistics with respect to the Likert scale judgements for both methods. We can observe that there is a larger number of tweets rated as *Excellent* and *Good*, while the number of tweets rated as *Bad* is lower compared to the baseline.

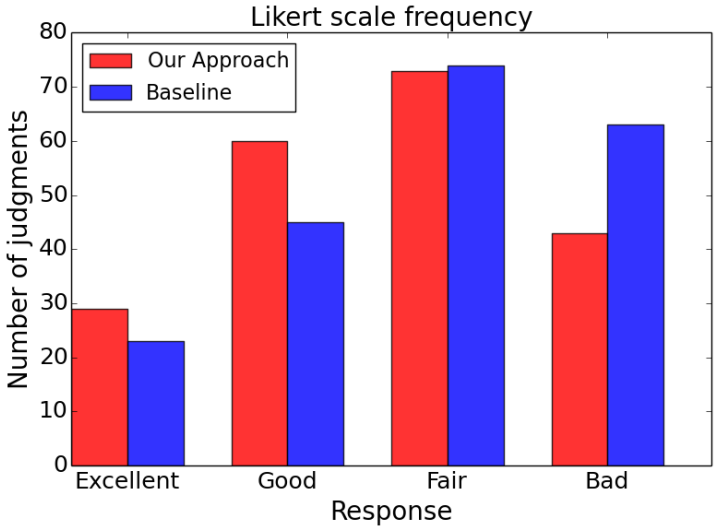

Figure 4.25. Comparison between our approach (red) and the baseline (blue) for the Likert scale.

Finally, in Figure 4.26 there are the percentages of tweets rated as *Excellent*, *Good*, *Fair*, and *Bad*, respectively. From these figures, it is clear that our method outperforms the baseline with higher percentage value.

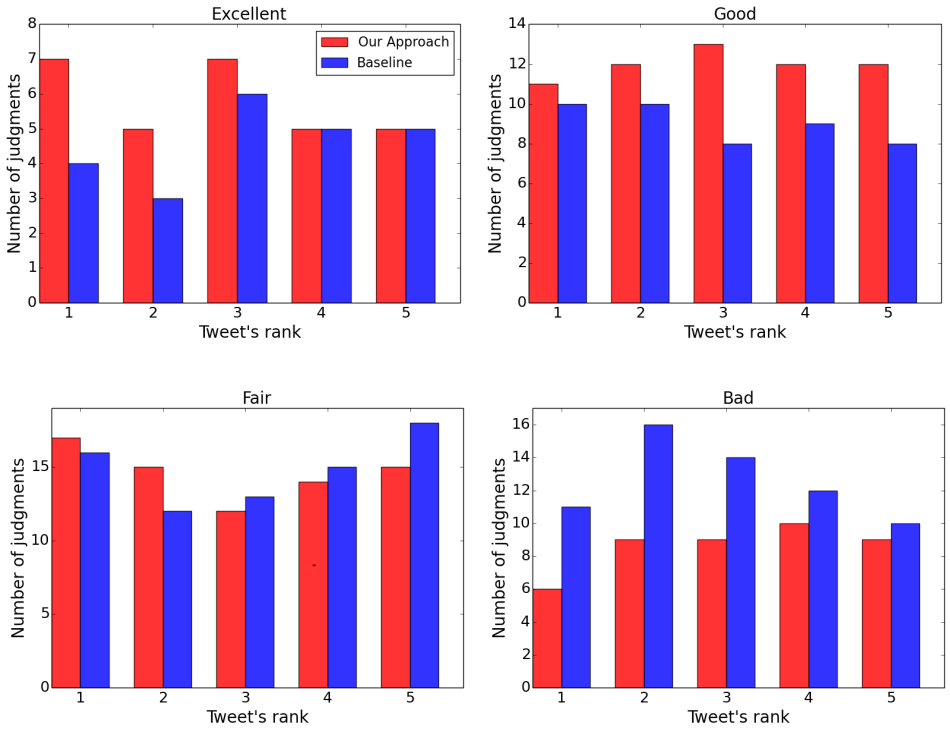

Figure 4.26. Comparison between our approach (red) and the baseline (blue) for each value of the Likert scale.

Chapter 5

Conclusions and Future Work

Recommender systems for the social web are turning out to be progressively important to find useful information from the huge amount of information that are produced by users everyday. This research exploits the advantages of using MapReduce in processing the massive amount of data which are crawled from the user's network through Twitter APIs platforms.

In addition to the feature of collaborative-based filtering, the data is also analyzed based on the features extracted form content-based similarity and incommon user's actions. These features are used together to recommend new interesting tweets to the user.

In more detail, this research have presented a novel approach for recommending concealed tweets. It first builds the network with two-hop distance from the user, such that the obtained graph will contain the out-links from the user (*friends*), and the out-links from her out-links (*friends-of-friends*). Then, two rounds of MapReduce algorithms are applied to find the links that close open triangles. And, as a feature, it makes sense to consider the count of these links to rank the nodes at 2-hop distance from the user.

After that, the timelines of the top list of best nodes are crawled to get the candidate tweets from which we want to recommend to the user. We found that the chronological order of these tweets may not be the best choice, so we applied content similarity measures: the cosine similarity and Jaccard distance between them and the user's tweets using bigrams constructs. The bigrams are applied at the timelines level between the ego and the candidate users. As a result, these features, enhance the quality of recommendations. Moreover, we used a retweet analysis of shared retweets between the ego user and the candidate users.

Finally, the tweets are ranked through pairwise comparison of the features' values in order to propose most interesting ones to the user that otherwise would remain hidden from her.

In experimental results, we conducted a real user study, and confirmed that our recommendation system outperforms the existing approach for recommending unseen tweets with an improvement of 12.4% in the precision.

As a final remark, this research add valuable insight to the recommender systems, through providing the user with most interesting contents from hidden environment, so that she will not miss important information which is mainly different from existing approaches that only recommend contents from direct connections that are seen by the user. As well as it implicitly gives a chance for the ego user to follow new people whenever she rate the recommended tweets.

For future work, we would like to extend our study in order to consider also the feedback from the users, and collect more data about their interests. In this case, the users will provide opinions about the quality of the recommendations such as: the way of determining the tweets that are auto-generated through bots, or how to exclude some candidate users, etc. It would also be interesting to recommend tweets to new Twitter users who just joined the network (cold start recommendations).

Another interesting extension of our approach consists in using the early adopter definition for Twitter users. The definition of early adopter is similar to the one presented in Mele et al. [78] for news recommendations. In that work, the pages visited by users who clicks on a new interesting page early are recommended to the other users. Similarly, we can assume that if a user retweets a tweet before the others and this happens frequently, she is the early adopter for the tweet. Such user appears early in the retweeters' list, so she gets a higher score relative to the retweeters' list length. Preliminary experiments conducted using retweet analysis have shown good performance. So, we think that the approach of early adoption for tweets is promising and deserves further investigation, for example, we plan to conduct a user study and collect users' feedback.

At last, here we considered only a particular graph property, but we plan to try other approaches to find and weighting potential tweets (e.g., from the followers), so that the network analysis will be a stronger interface to find new interesting candidates to the user, and result in a scalable system that have more diverse options.

We will apply an algorithm to learn weights of the features, in order to have better features combination and to find their importance related to the user, and consequently better ranking of the tweets. And also, we will consider other structural features that will enhance the quality of recommendations, and make it more efficient. All this will allow to make better predictions of tweets.

Appendix A

Registering a Twitter Application

Twitter give users the ability to create applications, and these applications have interfaces to work with which are called APIs. On June 11th, 2013, the version 1.0 of Twitter REST API (v1.0) retired and the programmers migrated to v1.1, and if someone use his application to access Twitter data through v1.0, he will get the following error:

The Twitter REST API v1 is no longer active. Please migrate to API v1.1. https://dev.twitter.com/docs/api/1.1/overview.

The characteristics of v1.1 are:

- By using this version a user can create sessions that are SSL (Secure Sockets Layer).
- The data format is just JSON (JavaScript Object Notation)
- The used authentication is OAuth.

In other words, more restrictions are added to give the users permission to write data like tweeting on behalf of them. After sign in, the user can create a new App, and access this webpage:

https://apps.twitter.com/

and press “Create New App” button, as in Figure A.1:

Figure A.1. Creating New Application

Then the will be directed to the application details as Figure A.2 shows:
To create an application, the user will be asked to fill the following fields:

- **Name**: The application name is a required field which must be unique name (not registered before).
- **Description**: This field is also required so the user can choose descriptive names for his applications to distinguish them from each other.

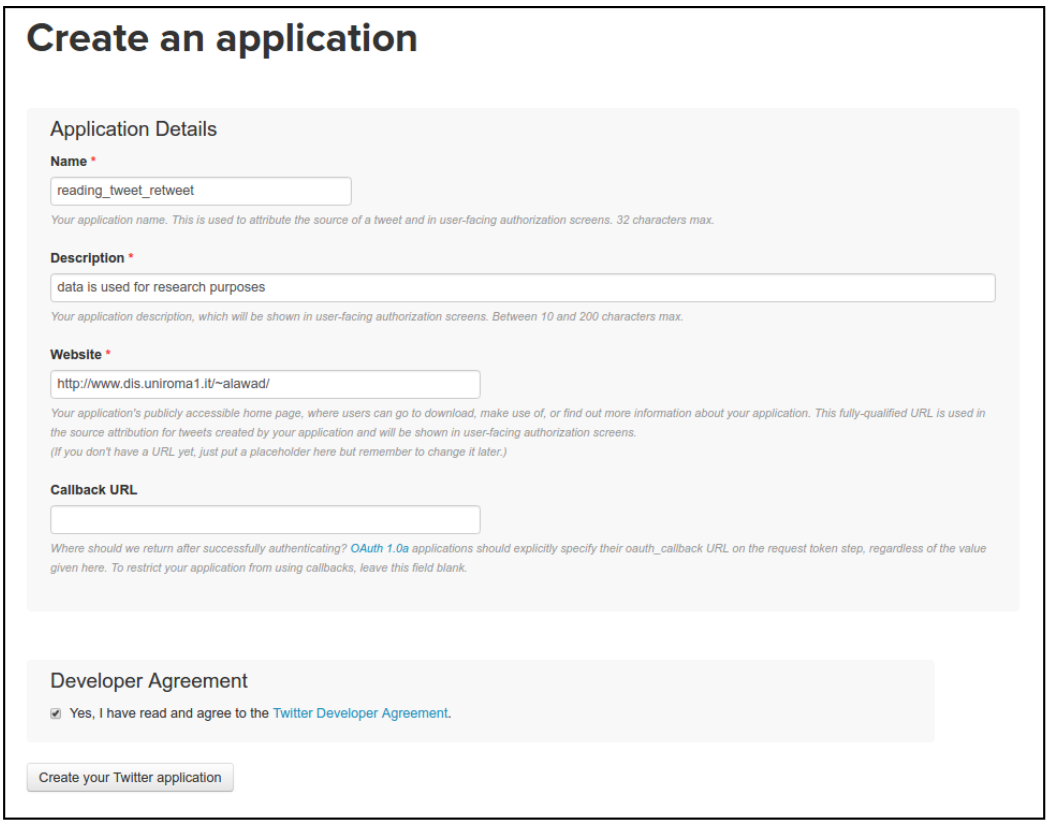

Figure A.2. Application details

- **Website**: This is the website address where the application will be hosted and accessed publicly, and it is also a required field.
- **Callback URL**: This is an optional field, which gives permission to other users to authenticate themselves through log into your App.

And then, the user will create his application after reading the agreement and agree on it. The application settings and details will be displayed in the first two tabs, while the third tab will contain the consumer key, consumer secret that identify the application, as in Figure A.3.

A user can also choose the access level from the 'Permission' tab as in Figure A.4), which have three types:

- **Read only**.
- **Read and Write**.
- **Read, Write and Access direct messages**.

In fact, these access levels are used to give more permissions to the App to add/remove tweets, etc.

It would be better to select the desired access level permission before creating access token keys, because if you change the permission you should regenerate them again to get the effect.

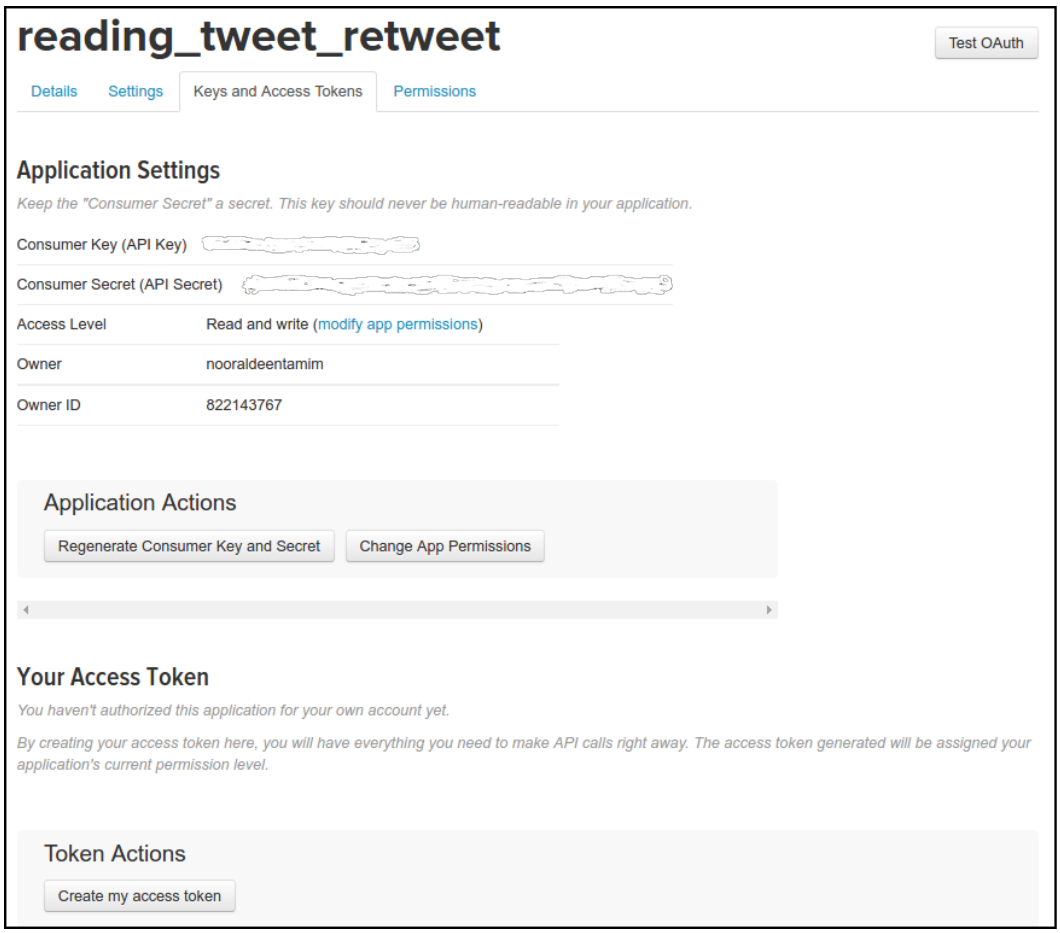

Figure A.3. Application keys

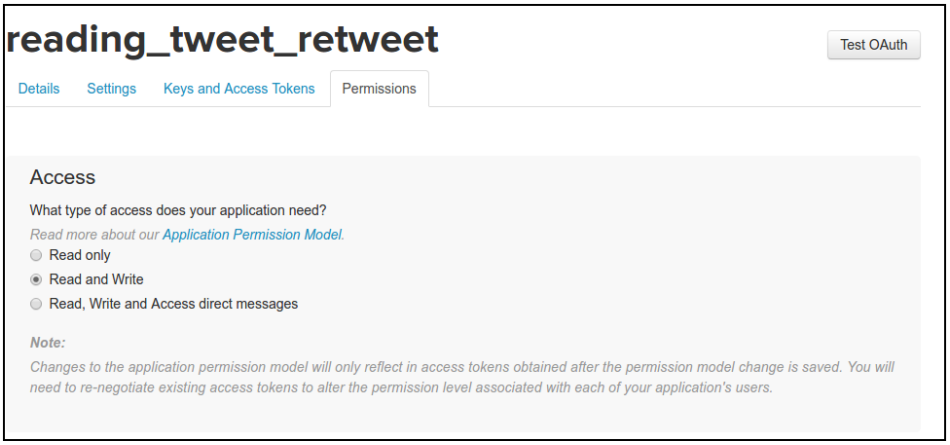

Figure A.4. Access tokens permissions

Figure A.5 shows the access token value and secret that are used to authorize the Twitter application.

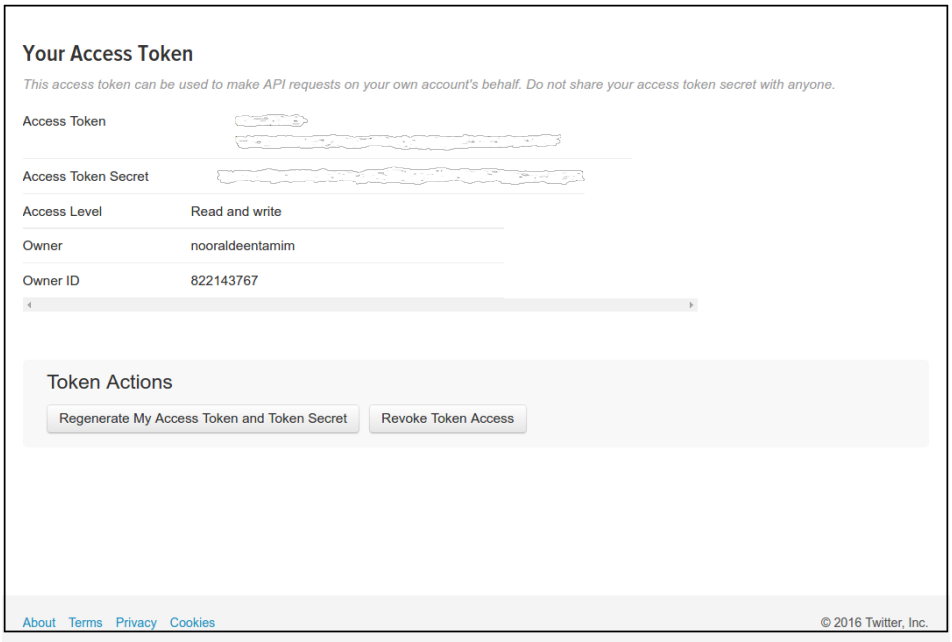

Figure A.5. Access token value and secret

To conclude, you need the following values from your app to access Twitter API:

- **Consumer Key**.
- **Consumer Secret**.
- **Access Token**.
- **Access Token Secret**.

These secret keys should be private to the account owner, because if they were known, any one can access her Twitter account.

Appendix B

Twitter's REST API

The REST APIs [2] provide programmatic access to read and write Twitter data: Author a new Tweet, read author profile and follower data, and more.

B.1 Authentication methods

Twitter supports two authentication methods which are OAuth signed authentication and Application-only authentication.

OAuth signed authentication.

In order to make authorized calls to Twitter's APIs, the application must first obtain an OAuth access token on behalf of a Twitter user to do the following:

- Give a facility to add "Sign in with Twitter" button on your website.
- If you want to read or post Twitter data on behalf of visitors to your website.
- When a mobile, desktop, or embedded app can't access a browser.
- If you want to access the API from your own account.
- When you need to use usernames/passwords and have been approved for xAuth.
- If you offer an API where clients send you data on behalf of Twitter users
- Want to issue authenticated requests on behalf of the application itself

Application-only authentication

Twitter offers applications the ability to issue authenticated requests on behalf of the application itself (as opposed to on behalf of a specific user). In other words, the app will not gain the context of an authenticated user like OAuth signed authentication functions (i.e., Posting, updating tweets, etc). The application will give the user the ability to:

- Pull user timelines.
- Access friends and followers of any account.
- Access lists resource
- Search in tweets.
- Retrieve any user information.

B.2 API requests

In our research, we focus on four methods for collecting data from Twitter:

- **GET friends/ids**.
- **GET statuses/user_timeline**.
- **GET statuses/retweeters/ids**.
- **GET users/lookup**.

The following sections explain the returned data from Twitter, that is applied to my account. The extracted values of fields are highlighted in colors.

B.2.1 GET friends/ids

It returns a list of the IDs of the followees 'friends' for a user (u), sorted with the most recently following user first. They are grouped into 5000 user IDs, and are ordered with the most recent following first. The total number of requests is 15 per 15-min window. The next_cursor values can be used to retrieve more IDs (if exist). The URL of this request is:

https://api.twitter.com/1.1/friends/ids.json?cursor=-1&screen_name=nooraldeentamim &count=5000

Table B.1 is an example of applying this request.

B.2.2 GET statuses/user_timeline

It returns the user's most recent tweets (up to $3,200$ tweets) posted by the user, and can be retrieved by her screen_name or user_id parameters, and it is identical to the contents of her profile on Twitter. The total number of requests is 300 per 15-min window. (In user-auth the number of requests is only 180). The URL of this request is:

https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=nooraldeentamim &count=1

The content in Tables B.2 and B.3 is an example of applying this request. It shows the recent tweet from my timeline.

B.2.3 GET statuses/retweeters/ids

It returns a list of at most 100 user IDs who retweetes a tweet that is given to the API through an id parameter. This id can be extracted from the user_timeline of the user B.2.2. The total number of requests is 60 per 15-min window. (In user-auth the number of requests is only 15). Table B.4 shows an example of the list of retweeters of a reweet.

The URL of this request is:

https://api.twitter.com/1.1/statuses/retweeters/ids.json?id=489278285525032960&count=100 &stringify_ids=true

Table B.1. Friends ids of my account

Field	Value
previous_cursor	0
previous_cursor_str	0
next_cursor	0
ids	[119512412, 15455450, 118197667, 2273201773L, 3128829965L, 146195370, 130671142, 127894322, 17157367, 15937226, 114485232, 20167623, 14538236, 14065835, 214272214, 2998896042L, 16868154, 14771839, 170281514, 436609034, 4833032980L, 720119788060721153L, 2478543656L, 70478255, 26261214, 22674817, 4829277184L, 54817041, 36412963, 204297410, 14826566, 229814338, 17960107, 234017215, 19739240, 486002273, 481094172, 227097055, 40216901, 19659370, 176757073, 400804254, 153327232, 30339571, 53125982, 123375238, 50692734, 108885204, 1105671, 382393, 13386092, 113125584, 2178067878L, 15089078, 5734242, 91697719, 607216969, 46131106, 91367103, 561201648, 2354542969L, 117553058, 12031032, 13568892, 115058238, 59504716, 15494489, 17078917, 32942338, 106064139, 16534969, 4038531, 78969930, 80422885, 20926161, 234321271, 17707080, 10095482, 14344469, 57741058, 5082531, 16067035, 283937225, 144776261, 10587552, 3080761, 633, 8143682, 17396121, 147949965, 42388966, 87208346, 427431943, 22994885, 250344120, 2510162390L, 45820175, 11990112, 2207892020L, 12179872, 398500188, 2663672256L, 2493328698L, 114565226, 1128974390, 216098950, 16958875, 2215006026L, 54833072, 997784960, 19388339, 2190723061L, 227458257, 125483940, 61814526, 23484381, 1455272023, 1562578862, 85648537, 9316452, 47885616, 12, 68678236, 93473293, 1134349284, 220145170, 620167024, 305474242, 242868086, 831420816, 138556170, 11892372, 1270104007, 220041728, 1527607790, 21457289, 16319797, 783214, 14749606, 6844292, 267283568, 372131355, 116455109, 6253282, 511402689, 1304717864, 146994016, 16687314, 72299308, 318310715, 31311757, 50090898, 33838201, 88727377, 105768989, 26000689, 10876852, 20536157, 4641021, 50393960, 1373460025, 547871798, 18189723, 115151170]
next_cursor_str	0

Table B.2. [Part1]: GET statuses/user_timeline

Field	Value
contributors	None
truncated	False
text	Twitter spikes after Microsoft buys LinkedIn for $26 billion https://t.co/xj1DeRCAgz via @themoneygame
is_quote_status	False
in_reply_to_status_id	None
id	742341578170245122
favorite_count	1
source	<a href="“http://twitter.com” rel="“nofollow” Twitter Web Client/>
retweeted	False
coordinates	None
symbols	[ ]
user_mentions	[{u‘id’: 65448361, u‘indices’: [89, 102], u‘id_str’: u‘65448361’, u‘screen_name’: u‘themoneygame’, u‘name’: u‘BI Markets’}]
hashtags	[ ]
urls	[{u‘url’: u‘https://t.co/xj1DeRCAgz’, u‘indices’: [61, 84], u‘expanded_url’: u‘http://read.bi/1Ol4Gx4’, u‘display_url’: u‘read.bi/1Ol4Gx4’}]
in_reply_to_screen_name	None
in_reply_to_user_id	None
retweet_count	0
id_str	742341578170245122
favorited	False
follow_request_sent	False
has_extended_profile	False
profile_use_background_image	True
default_profile_image	False
id	822143767
profile_background_image_url_https	https://abs.twimg.com/images/themes/theme1/bg.png
verified	False
profile_text_color	333333
profile_image_url_https	https://pbs.twimg.com/profile_images/378800000074791549/6b26302825ae09c0e46042cf9cfb3e03_normal.jpeg
profile_sidebar_fill_color	DDEEF6
urls	[{u’url’: u’http://t.co/q8TPBqT4dW’, u’indices’: [0, 22], u’expanded_url’: u’http://www.dis.uniroma1.it/∼alawad/’, u’display_url’: u’dis.uniroma1.it/∼alawad/’}]
followers_count	47

Table B.3. [Part2]: GET statuses/user_timeline

Field	Value
profile_sidebar_border_color	C0DEED
id_str	822143767
profile_background_color	C0DEED
listed_count	1
is_translation_enabled	False
utc_offset	7200
statuses_count	181
description	When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one that has opened for us. - Ale
friends_count	184
location	Sapienza University of Rome
profile_link_color	0084B4
profile_image_url	http://pbs.twimg.com/profile_images/ 378800000074791549/ 6b26302825ae09c0e46042cf9cfb3e03_normal.jpeg
following	False
geo_enabled	True
profile_banner_url	https://pbs.twimg.com/profile_banners/ 822143767/1368978123
profile_background_image_url	http://abs.twimg.com/images/themes/theme1/bg.png
screen_name	nooraldeentamim
lang	en
profile_background_tile	False
favourites_count	130
name	Noor Aldeen Alawad
notifications	False
url	http://t.co/q8TPBqT4dW
created_at	Thu Sep 13 20:32:17 +0000 2012
contributors_enabled	False
time_zone	Rome
protected	False
default_profile	True
is_translator	False
geo	None
in_reply_to_user_id_str	None
possibly_sensitive	False
lang	en
created_at	Mon Jun 13 13:03:18 +0000 2016
in_reply_to_status_id_str	None
place	None

Table B.4. The list of retweeters for a retweet

Field	Value
previous_cursor	0
previous_cursor_str	0
next_cursor	0
ids	[40965059, 178483403, 2417186826, 2412270288, 2417347170, 2411206362, 2415480745, 196194980, 812380152, 2412016100, 395197255, 23179623, 216706300, 367900071, 359324126, 2397301877, 63566265, 15327664, 100491371, 1117650571, 822143767, 784984, 585857069, 2456496284,116549673, 1243357730, 22622136, 16376537, 14752306, 503674501, 131448307, 57651821, 15189144, 176741459, 2371756192, 149808271, 2188107351, 362633864, 138516722, 904834980, 2374704301, 336240028, 2380259712, 59263014, 267342991, 19029792, 2638353259, 19932305, 606362845, 2498788723, 2160201530, 16840386, 21575173, 8674902, 14681121, 2330128201, 2615209200, 179205109, 1930579800, 22514183, 348967193, 79184517, 61775988, 48115420, 2255492856, 131786415, 2344725759, 2396892662, 209605296, 396578935, 24869669, 711364834, 17022290, 246191393, 15203929, 85996257, 1672681, 1300455596, 1635006739, 2167208841, 430096982, 85118086, 236287900, 328926221, 1573319048, 429707643, 275315306, 988437164]
next_cursor_str	0

B.2.4 GET users/lookup

It returns the user objects for at most 100 users in a request, they can be passed to the user_id and/or screen_name parameters through comma-separated values. The contents in Tables B.5 and B.6 show an example of applying this method. We are interested in retrieving the user's name, screen name, profile image url, location, and time zone. The total number of requests is 60 per 15-min window.

The URL of this request is:

https://api.twitter.com/1.1/users/lookup.json?screen_name=nooraldeentamim'

Table B.5. [Part1]: GET users/lookup

Field	Value
follow_request_sent	False
has_extended_profile	False
profile_use_background_image	True
default_profile_image	False
id	822143767
profile_background_image_url_https	https://abs.twimg.com/images/themes/theme1/bg.png
verified	False
profile_text_color	333333
profile_image_url_https	https://pbs.twimg.com/profile_images/378800000074791549/6b26302825ae09c0e46042cf9cfb3e03_normal.jpeg
profile_sidebar_fill_color	DDEEF6
urls	[{u'url': u'http://t.co/q8TPBqT4dW', u'indices': [0, 22], u'expanded_url': u'http://www.dis.uniroma1.it', u'display_url': u'dis.uniroma1.it/'}]
followers_count	47
profile_sidebar_border_color	C0DEED
id_str	822143767
profile_background_color	C0DEED
listed_count	1
contributors	None
truncated	False
text	Twitter spikes after Microsoft buys LinkedIn for $26 billion https://t.co/xj1DeRCAgz via @themoneygame
is_quote_status	False
in_reply_to_status_id	None
id	742341578170245122
favorite_count	1
source	 Twitter Web Client
retweeted	False
coordinates	None
symbols	[]
user_mentions	[{u'id': 65448361, u'indices': [89, 102], u'id_str': u'65448361', u'screen_name': u'themoneygame', u'name': u'BI Markets'}]
hashtags	[]
urls	[{u'url': u'https://t.co/xj1DeRCAgz', u'indices': [61, 84], u'expanded_url': u'http://read.bi/1Ol4Gx4', u'display_url': u'read.bi/1Ol4Gx4'}]

Table B.6. [Part2]: GET users/lookup

Field	Value
in_reply_to_screen_name	None
in_reply_to_user_id	None
retweet_count	0
id_str	742341578170245122
favorited	False
geo	None
in_reply_to_user_id_str	None
possibly_sensitive	False
lang	en
created_at	Mon Jun 13 13:03:18 +0000 2016
in_reply_to_status_id_str	None
place	None
is_translation_enabled	False
utc_offset	7200
statuses_count	181
description	When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one that has opened for us. - Ale
friends_count	164
location	Sapienza University of Rome
profile_link_color	0084B4
profile_image_url	http://pbs.twimg.com/profile_images/ 378800000074791549/ 6b26302825ae09c0e46042cf9cfb3e03_normal.jpeg
following	False
geo_enabled	http://t.co/True
profile_banner_url	https://pbs.twimg.com/profile_banners/822143767/1368978123
profile_background_image_url	http://abs.twimg.com/images/themes/theme1/bg.png
screen_name	nooraldeentamim
lang	en
profile_background_tile	False
favourites_count	130
name	Noor Aldeen Alawad
notifications	False
url	http://t.co/q8TPBqT4dW
created_at	Thu Sep 13 20:32:17 +0000 2012
contributors_enabled	False
time_zone	Rome
protected	False
default_profile	True
is_translator	False

Appendix C

Issues Related to Retrieving a User's Timeline

C.1 Dealing with large accounts

For large accounts we worked with timelines like the technique in [3] through iterating over the timeline using cursoring to get the complete list of (re)tweets and avoiding redundancy.

In details, there exist a parameter "max_id" that works with streams of data called cursoring. Because of the dynamic changes in posting tweets frequently for accounts, this request parameter is used to track the tweets through their IDs instead of the paging technique that are used for other methods.

The application sends a request to a timeline with the maximum retrieval rate limit. Then, for subsequent requests, the lowest ID of the tweet will be be passed as the value of the max_id parameter for the next request, and so it will return Tweets with IDs lower than or equal to the value of the max_id parameter inclusive. In this case, the last tweet will be returned again, so we can retrieve for a request, all the tweets except the last one as in figure C.1.

C.2 Dealing with character encodings

When retrieving a tweet, it may contain non-ASCII characters (that is outside the 7-bit ASCII (0-127)), for example, these are non-ASCII characters: "ï", "é", etc. The Unicode string that contain these characters can be encoded to ASCII and the errors will be ignored by using the following statement:

twt = twt.encode('ascii', 'ignore')

Also, when retrieving a tweet, it may contain a backslash that may be considered as an escape character. These characters can be replaced through:

twt = twt.replace('\n', '').replace('\r', '')

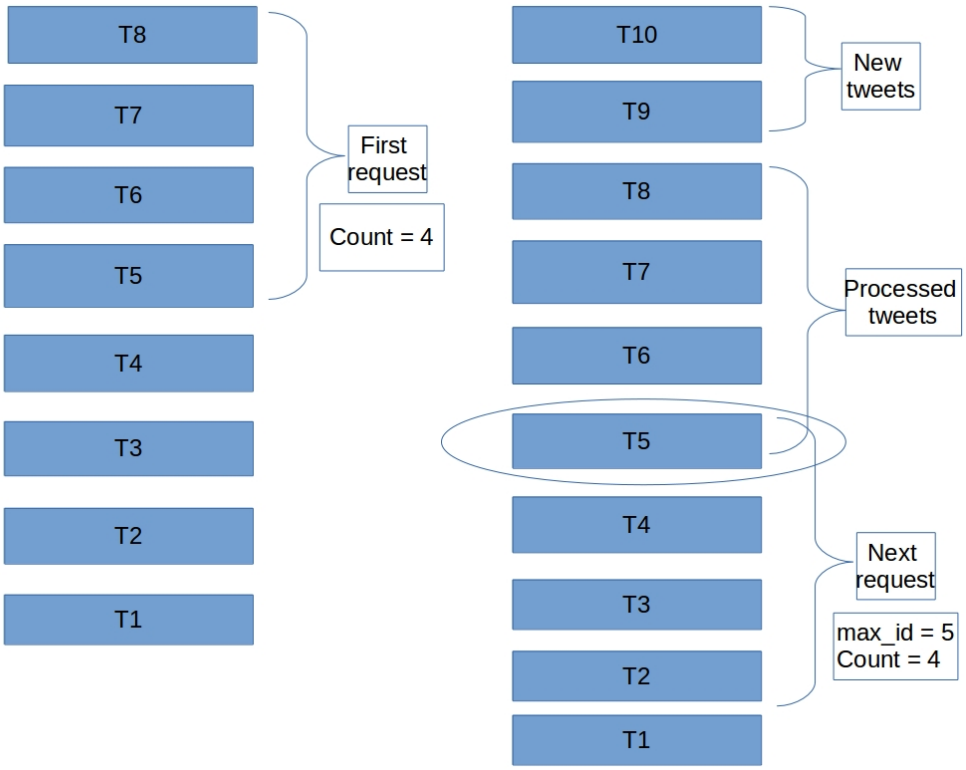

Figure C.1. Using the max_id parameter in timeline retrieval.

C.3 Dealing with socket timeout

Socket timeouts were occurred when attempting to connect to REST API during fetching retweeters list alongside with the user's timeline. After importing socket, the function we used to create socket object is:

socket.setdefaulttimeout(timeout)

Where the 'timeout' can be a nonnegative floating point number expressing seconds, or None [4]:

- If a non-zero value is given, subsequent socket operations will raise a timeout exception if the timeout period value has elapsed before the operation has completed (Where operations fail if they cannot be completed within the timeout specified for the socket) and they raise a timeout exception or the system returns an error.
- If zero is given, the socket is put in non-blocking mode, where operations fail if they cannot be completed immediately: functions from the select can be used to know when and whether a socket is available for reading or writing.
- If None is given, the socket is put in blocking mode, where operations block until complete or the system returns an error (such as connection timed out).

We set a value of the argument of 5 minutes. 300 second timer to fetch data, and if it pass without data reception, then it will send a keep-alive new request to prevent the process from timing out the connection.

Bibliography

[1] Authentication & authorization - twitter developers. Available from: `https://dev.twitter.com/oauth/overview/authentication-by-api-family`.

[2] Rest apis. Available from: `https://dev.twitter.com/rest/public`.

[3] Working with timelines. Available from: `https://dev.twitter.com/rest/public/timelines`.

[4] 5_18.1. socket low-level networking interface python 3.5.2 documentation_2016 (2016). Available from: `https://docs.python.org/3/library/socket.html#socket-timeouts`.

[5] ABEL, F., GAO, Q., HOUBEN, G.-J., AND TAO, K. Analyzing temporal dynamics in twitter profiles for personalized recommendations in the social web. In *Proceedings of the 3rd International Web Science Conference*, p. 2. ACM (2011).

[6] ADOMAVICIUS, G. AND TUZHILIN, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, **17** (2005), 734.

[7] AIELLO, L. M., BARRAT, A., SCHIFANELLA, R., CATTUTO, C., MARKINES, B., AND MENCZER, F. Friendship prediction and homophily in social media. *ACM Transactions on the Web (TWEB)*, **6** (2012), 9.

[8] ALAWAD, N. A., ANAGNOSTOPOULOS, A., LEONARDI, S., MELE, I., AND SILVESTRI, F. Network-aware recommendations of novel tweets. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 913–916. ACM (2016).

[9] ALMESHARY, M. AND ABHARI, A. A recommendation system for twitter users in the same neighborhood. In *Proceedings of the 16th Communications & Networking Symposium*, p. 1. Society for Computer Simulation International (2013).

[10] ARMENTANO, M. G., GODOY, D., AND AMANDI, A. Recommending information sources to information seekers in twitter. In *International workshop on social web mining* (2011).

[11] ARMENTANO, M. G., GODOY, D., AND AMANDI, A. Towards a followee recommender system for information seeking users in twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, vol. 730, pp. 27–38 (2011).

[12] ARMENTANO, M. G., GODOY, D., AND AMANDI, A. A. Followee recommendation based on text analysis of micro-blogging activity. *Information systems*, **38** (2013), 1116.

[13] ARMENTANO, M. G., GODOY, D. L., AND AMANDI, A. A. A topology-based approach for followees recommendation in Twitter. In *ITWP'11*.

[14] ASHOK, B., JOY, J., LIANG, H., RAJAMANI, S. K., SRINIVASA, G., AND VANGALA, V. Debugadvisor: a recommender system for debugging. In *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 373–382. ACM (2009).

[15] BALABANOVIĆ, M. AND SHOHAM, Y. Fab: content-based, collaborative recommendation. *Communications of the ACM*, **40** (1997), 66.

[16] BASU, C., HIRSH, H., COHEN, W., ET AL. Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*, pp. 714–720 (1998).

[17] BATAGELJ, V. AND MRVAR, A. Pajek—analysis and visualization of large networks. In *Graph drawing software*, pp. 77–103. Springer (2004).

[18] BEEL, J. AND LANGER, S. Research paper recommender systems: A literature survey.

[19] BENZARTI, S. AND FAIZ, R. EgoTR: Personalized tweets recommendation approach. In *CSOC'15*.

[20] BILLSUS, D. AND PAZZANI, M. J. User modeling for adaptive news access. *User modeling and user-adapted interaction*, **10** (2000), 147.

[21] BUETTNER, R. A framework for recommender systems in online social network recruiting: An interdisciplinary call to arms. In *2014 47th Hawaii International Conference on System Sciences*, pp. 1415–1424. IEEE (2014).

[22] BURGHARDT, M. Introduction to tools and methods for the analysis of twitter data. Available from: `https://www.researchgate.net/publication/281785732_Introduction_to_Tools_and_Methods_for_the_Analysis_of_Twitter_Data`.

[23] BURKE, R. Integrating knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce*, pp. 69–72 (1999).

[24] BURKE, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, **12** (2002), 331.

[25] CANTADOR, I., BELLOGÍN, A., AND VALLET, D. Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 237–240. ACM (2010).

[26] CHEN, C., YIN, H., YAO, J., AND CUI, B. Terec: A temporal recommender system over tweet stream. *Proceedings of the VLDB Endowment*, **6** (2013), 1254.

[27] CHEN, H., CUI, X., AND JIN, H. Top-k followee recommendation over microblogging systems by exploiting diverse information sources. *Future Generation Computer Systems*, **55** (2016).

[28] CHEN, J., NAIRN, R., NELSON, L., BERNSTEIN, M., AND CHI, E. Short and tweet: experiments on recommending content from information streams. In *CHI'10*.

[29] CHEN, K., CHEN, T., ZHENG, G., JIN, O., YAO, E., AND YU, Y. Collaborative personalized tweet recommendation. In *SIGIR'12*.

[30] CHEN, Y., WU, C., XIE, M., AND GUO, X. Solving the sparsity problem in recommender systems using association retrieval. *Journal of computers*, **6** (2011), 1896.

[31] CLAYPOOL, M., GOKHALE, A., MIRANDA, T., MURNIKOV, P., NETES, D., AND SARTIN, M. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, vol. 60. Citeseer (1999).

[32] COTTER, P. AND SMYTH, B. Ptv: Intelligent personalised tv guides. In *AAAI/IAAI*, pp. 957–964 (2000).

[33] DAS, A. S., DATAR, M., GARG, A., AND RAJARAM, S. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pp. 271–280. ACM (2007).

[34] DEAN, J. AND GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, **51** (2008), 107.

[35] DOVGOPOL, R. AND NOHELTY, M. Twitter hash tag recommendation. *arXiv preprint arXiv:1502.00094*, (2015).

[36] DUAN, Y., JIANG, L., QIN, T., ZHOU, M., AND SHUM, H.-Y. An empirical study on learning to rank of tweets. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 295–303. Association for Computational Linguistics (2010).

[37] EKSTRAND, M. D., RIEDL, J. T., AND KONSTAN, J. A. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, **4** (2011), 81.

[38] ESTER, M. Recommendation in social networks. In *RecSys*, pp. 491–492 (2013).

[39] FRIEDRICH, G. AND ZANKER, M. A taxonomy for generating explanations in recommender systems. *AI Magazine*, **32** (2011), 90.

[40] GARCIA, R. AND AMATRIAIN, X. Weighted content based methods for recommending connections in online social networks. In *Workshop on Recommender Systems and the Social Web*, pp. 68–71. Citeseer (2010).

[41] GEORGE, T. AND MERUGU, S. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 4–pp. IEEE (2005).

[42] GHAZANFAR, M. A. AND PRUGEL-BENNETT, A. A scalable, accurate hybrid recommender system. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pp. 94–98. IEEE (2010).

[43] GOLDBERG, D., NICHOLS, D., OKI, B. M., AND TERRY, D. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, **35** (1992), 61.

[44] GOLDER, S. A., YARDI, S., MARWICK, A., AND BOYD, D. A structural approach to contact recommendations in online social networks. In *Workshop on search in social media, SSM* (2009).

[45] GRIVOLLA, J., CAMPO, D., SONSONA, M., PULIDO, J.-M., AND BADIA, T. A hybrid recommender combining user, item and interaction data. In *Computational Science and Computational Intelligence (CSCI), 2014 International Conference on*, vol. 1, pp. 297–301. IEEE (2014).

[46] GUO, W., LI, H., JI, H., AND DIAB, M. T. Linking tweets to news: A framework to enrich short text data in social media. In *ACL (1)*, pp. 239–249. Citeseer (2013).

[47] GURINI, D. F., GASPARETTI, F., MICARELLI, A., AND SANSONETTI, G. A sentiment-based approach to twitter user recommendation. In *RSWeb@ RecSys* (2013).

[48] HANNEMAN, R. A. AND RIDDLE, M. Introduction to social network methods (2005).

[49] HANNON, J., BENNETT, M., AND SMYTH, B. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pp. 199–206. ACM (2010).

[50] HANNON, J., MCCARTHY, K., AND SMYTH, B. Finding useful users on Twitter: Twittomender the followee recommender. In *ECIR'11*.

[51] HANNON, J., MCCARTHY, K., AND SMYTH, B. The pursuit of happiness: searching for worthy followees on. *Computer Networks and ISDN Systems*, **30** (1998), 107.

[52] HE, J. AND CHU, W. W. *A social network-based recommender system (SNRS)*. Springer (2010).

[53] HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G., AND RIEDL, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, **22** (2004), 5.

[54] HOLTE, R. C. AND YAN, J. N. Y. Inferring what a user is not interested in. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pp. 159–171. Springer (1996).

[55] IAQUINTA, L., DE GEMMIS, M., LOPS, P., SEMERARO, G., FILANNINO, M., AND MOLINO, P. Introducing serendipity in a content-based recommender system. In *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pp. 168–173. IEEE (2008).

[56] JAVA, A., SONG, X., FININ, T., AND TSENG, B. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pp. 56–65. ACM (2007).

[57] JURVETSON, S. What exactly is viral marketing. *Red Herring*, **78** (2000), 110.

[58] KAPANIPATHI, P., ORLANDI, F., SHETH, A. P., AND PASSANT, A. Personalized filtering of the twitter stream. (2011).

[59] KIM, K.-J. AND AHN, H. Hybrid recommender systems using social network analysis. *World Academy of Science, Engineering and Technology*, **64** (2012), 879.

[60] KINGSTON, C. Twitter for beginners. New York: media DIY workshop (2011).

[61] KRESTEL, R., WERKMEISTER, T., WIRADARMA, T. P., AND KASNECI, G. Tweet-recommender: Finding relevant tweets for news articles. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 53–54. ACM (2015).

[62] KRUTKAM, W., SAIKEAW, K., AND CHAOSAKUL, A. Twitter accounts recommendation based on followers and lists. *3rd Joint International Information and Communication Technology*, (2010).

[63] KUMAR, A. AND SHARMA, A. Alleviating sparsity and scalability issues in collaborative filtering based recommender systems. In *Proceedings of the International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, pp. 103–112. Springer (2013).

[64] KUNEGIS, N., GOTTRON, T., KUNEGIS, J., AND ALHADI, A. Bad news travel fast: A content-based analysis of interestingness on twitter. In *Proceedings of ACM Web Science Conference* (2011).

[65] KYWE, S. M., LIM, E.-P., AND ZHU, F. A survey of recommender systems in Twitter. In *SocInfo'12*.

[66] LEE, K., MAHMUD, J., CHEN, J., ZHOU, M., AND NICHOLS, J. Who will retweet this?: Automatically identifying and engaging strangers on twitter to spread information. In *Proceedings of the 19th international conference on Intelligent User Interfaces*, pp. 247–256. ACM (2014).

[67] LETIERCE, J., PASSANT, A., BRESLIN, J., AND DECKER, S. Understanding how twitter is used to spread scientific messages. (2010).

[68] LITTLESTONE, N. AND WARMUTH, M. K. The weighted majority algorithm. In *Foundations of Computer Science, 1989., 30th Annual Symposium on*, pp. 256–261. IEEE (1989).

[69] LIU, G., FU, Y., XU, T., XIONG, H., AND CHEN, G. Discovering temporal retweeting patterns for social media marketing campaigns. In *2014 IEEE International Conference on Data Mining*, pp. 905–910. IEEE (2014).

[70] LOPS, P., DE GEMMIS, M., AND SEMERARO, G. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pp. 73–105. Springer (2011).

[71] LU, C., LAM, W., AND ZHANG, Y. Twitter user modeling and tweets recommendation based on wikipedia concept graph. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence* (2012).

[72] LUMBRERAS, A. AND GAVALDA, R. Applying trust metrics based on user interactions to recommendation in social networks. In *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*, pp. 1159–1164. IEEE Computer Society (2012).

[73] MAKKI, R. Active tweet recommendation based on user interest profiles.

[74] MARTÍNEZ, L., PÉREZ, L. G., AND BARRANCO, M. A multigranular linguistic content-based recommendation model. *International Journal of Intelligent Systems*, **22** (2007), 419.

[75] MATEI, S. Analyzing social media networks with nodexl: Insights from a connected world by derek hansen, ben shneiderman, and marc a. smith: Burlington, ma: Morgan kauffman, 2011. 284 pages. isbn: 978-0-12-382229-1. *Intl. Journal of Human–Computer Interaction*, **27** (2011), 405.

[76] MCCULLOH, I., ARMSTRONG, H., AND JOHNSON, A. *Social network analysis with applications*. John Wiley & Sons (2013).

[77] MCGRATH, R. twython. Available from: `https://media.readthedocs.org/pdf/twython/latest/twython.pdf`.

[78] MELE, I., BONCHI, F., AND GIONIS, A. The early-adopter graph and its application to web-page recommendation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 1682–1686. ACM (2012).

[79] MELVILLE, P., MOONEY, R. J., AND NAGARAJAN, R. Content-boosted collaborative filtering for improved recommendations. In *Aaai/iaai*, pp. 187–192 (2002).

[80] MISLOVE, A., MARCON, M., GUMMADI, K. P., DRUSCHEL, P., AND BHATTACHARJEE, B. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 29–42. ACM (2007).

[81] MOLLETT, A., MORAN, D., AND DUNLEAVY, P. Using twitter in university research, teaching and impact activities. (2011).

[82] NASIRIFARD, P. AND HAYES, C. Tadvise: A twitter assistant based on twitter lists. In *International Conference on Social Informatics*, pp. 153–160. Springer (2011).

[83] PAGARE, R. AND PATIL, S. A. Study of collaborative filtering recommendation algorithm-scalability issue. *International Journal of Computer Applications*, **67** (2013).

[84] PATEL, A. B., SUTHAR, N. B., AND DHOBI, J. S. Recommending top-n research papers (based on with, without and boolean items preferences: An user base collaborative filtering approach in mahout). In *ICCCIT'12*.

[85] PAZZANI, M. J. AND BILLSUS, D. Content-based recommendation systems. In *The adaptive web*, pp. 325–341. Springer (2007).

[86] PECHKIS, B. J. AND LEE, E.-J. Isolating matrix sparsity in collaborative filtering ratings matrices. In *Proceedings of the International Conference on Data Mining (DMIN)*, p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2013).

[87] PENNACCHIOTTI, M., SILVESTRI, F., VAHABI, H., AND VENTURINI, R. Making your interests follow you on Twitter. In *CIKM'12*.

[88] PHELAN, O., MCCARTHY, K., BENNETT, M., AND SMYTH, B. Terms of a feather: Content-based news recommendation and discovery using twitter. In *European Conference on Information Retrieval*, pp. 448–459. Springer (2011).

[89] PRIEM, J. AND COSTELLO, K. L. How and why scholars cite on twitter. *Proceedings of the American Society for Information Science and Technology*, **47** (2010), 1.

[90] RAMAGE, D., DUMAIS, S. T., AND LIEBLING, D. J. Characterizing microblogs with topic models. *ICWSM*, **10** (2010), 1.

[91] RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., AND RIEDL, J. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186. ACM (1994).

[92] RESNICK, P. AND VARIAN, H. R. Recommender systems. *Commun. ACM*, **40** (1997).

[93] RICCI, F., ROKACH, L., AND SHAPIRA, B. *Introduction to recommender systems handbook*. Springer (2011).

[94] RODRIGUEZ, M. G., GUMMADI, K., AND SCHOELKOPF, B. Quantifying information overload in social media and its impact on social contagions. *arXiv preprint arXiv:1403.6838*, (2014).

[95] ROWE, M., STANKOVIC, M., AND ALANI, H. Who will follow whom? exploiting semantics for link prediction in attention-information networks. In *International Semantic Web Conference*, pp. 476–491. Springer (2012).

[96] SACHAN, A. AND RICHHARIYA, V. Reduction of data sparsity in collaborative filtering based on fuzzy inference rules. *International Journal of Advanced Computer Research*, **3** (2013), 101.

[97] SCHAFER, J. B., FRANKOWSKI, D., HERLOCKER, J., AND SEN, S. Collaborative filtering recommender systems. In *The adaptive web*, pp. 291–324. Springer (2007).

[98] SCHANK, T. Algorithmic aspects of triangle-based network analysis. *Phd in computer science, University Karlsruhe*, **3** (2007).

[99] SCHANK, T. AND WAGNER, D. Finding, counting and listing all triangles in large graphs, an experimental study. In *International Workshop on Experimental and Efficient Algorithms*, pp. 606–609. Springer (2005).

[100] SEDHAI, S. AND SUN, A. Hashtag recommendation for hyperlinked tweets. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pp. 831–834. ACM (2014).

[101] SOTSENKO, A., JANSEN, M., AND MILRAD, M. Using a rich context model for a news recommender system for mobile users. In *NRA'14*.

[102] SUH, B., HONG, L., PIROLLI, P., AND CHI, E. H. Want to be retweeted? large scale analytics on factors impacting retweet in twitter network. In *Social computing (socialcom), 2010 ieee second international conference on*, pp. 177–184. IEEE (2010).

[103] SULLIVAN, D. O., SMYTH, B., AND WILSON, D. Preserving recommender accuracy and diversity in sparse datasets. *International Journal on Artificial Intelligence Tools*, **13** (2004), 219.

[104] SUN, A. R., CHENG, J., AND ZENG, D. D. A novel recommendation framework for micro-blogging based on information diffusion. In *19th Annual Workshop on Information Technolgies & Systems (WITS'09)* (2009).

[105] SURI, S. AND VASSILVITSKII, S. Counting triangles and the curse of the last reducer. In *WWW'11*.

[106] TANG, J., HU, X., GAO, H., AND LIU, H. Exploiting local and global social context for recommendation. In *IJCAI*, pp. 264–269 (2013).

[107] TANNENBAUM, P. AND ARNOLD, R. *Excursions in modern mathematics*. Prentice Hall (1995).

[108] THORAT, P. B., GOUDAR, R., AND BARVE, S. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, **110** (2015).

[109] TOMMASEL, A., CORBELLINI, A., GODOY, D., AND SCHIAFFINO, S. Exploring the role of personality traits in followee recommendation. *Online Information Review*, **39** (2015), 812.

[110] TOMMASEL, A. AND GODOY, D. An adaptive technique for weighting multiple factors in followee recommendation algorithms. In *Proceedings of the 2015 International Conference on Constraints and Preferences for Configuration and Recommendation and Intelligent Techniques for Web Personalization-Volume 1440*, pp. 32–33. CEUR-WS. org (2015).

[111] TRAN, T. AND COHEN, R. Hybrid recommender systems for electronic commerce. In *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press* (2000).

[112] TSOURAKAKIS, C. E., DRINEAS, P., MICHELAKIS, E., KOUTIS, I., AND FALOUTSOS, C. Spectral counting of triangles via element-wise sparsification and triangle-based link recommendation. *Social Network Analysis and Mining*, **1** (2011), 75.

[113] UYSAL, I. AND CROFT, W. B. User oriented tweet ranking: a filtering approach to microblogs. In *CIKM'11*.

[114] VILLI, M., MATIKAINEN, J., AND KHALDAROVA, I. Recommend, tweet, share: User-distributed content (udc) and the convergence of news media and social networks. In *Media Convergence Handbook-Vol. 1*, pp. 289–306. Springer (2016).

[115] WANG, B., WANG, C., BU, J., CHEN, C., ZHANG, W. V., CAI, D., AND HE, X. Whom to mention: expand the diffusion of tweets by@ recommendation on micro-blogging systems. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 1331–1340. ACM (2013).

[116] WANG, X., XIAO, B., LIN, Z., AND LU, Y. A tweets recommendation algorithm based on user relationship and text emotional tendentiousness. In *2012 3rd IEEE International Conference on Network Infrastructure and Digital Content*, pp. 237–241. IEEE (2012).

[117] WANG, Y., WANG, S., STASH, N., AROYO, L., AND SCHREIBER, G. Enhancing content-based recommendation with the task model of classification. In *International Conference on Knowledge Engineering and Knowledge Management*, pp. 431–440. Springer (2010).

[118] WANG, Z. AND IWAIHARA, M. Cross-lingual tweet recommendation based on user interest using bilingual lda.

[119] WATTS, D. J. AND STROGATZ, S. H. Collective dynamics of 'small-world'networks. *nature*, **393** (1998), 440.

[120] WEI, Z. AND GAO, W. Utilizing microblogs for automatic news highlights extraction. In *COLING*, pp. 872–883 (2014).

[121] WELLER, K., DRÖGE, E., AND PUSCHMANN, C. Citation analysis in twitter: Approaches for defining and measuring information flows within tweets during scientific conferences. In *# MSM*, pp. 1–12 (2011).

[122] WELLER, K. AND PUSCHMANN, C. Twitter for scientific communication: How can citations/references be identified and measured? (2011).

[123] WU, H., SORATHIA, V., AND PRASANNA, V. K. Predict whom one will follow: followee recommendation in microblogs. In *Social Informatics (SocialInformatics), 2012 International Conference on*, pp. 260–264. IEEE (2012).

[124] XUE, G.-R., LIN, C., YANG, Q., XI, W., ZENG, H.-J., YU, Y., AND CHEN, Z. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 114–121. ACM (2005).

[125] YAN, R., LAPATA, M., AND LI, X. Tweet recommendation with graph co-ranking. In *ACL'12*.

[126] YANG, M.-C. AND RIM, H.-C. Identifying interesting twitter contents using topical analysis. *Expert Systems with Applications*, **41** (2014), 4330.

[127] YAZDANFAR, N. AND THOMO, A. Link recommender: Collaborative-filtering for recommending urls to Twitter users. *ANT'13*.

[128] YIN, Z., GUPTA, M., WENINGER, T., AND HAN, J. Linkrec: a unified framework for link recommendation with user attributes and graph structure. In *Proceedings of the 19th international conference on World wide web*, pp. 1211–1212. ACM (2010).

[129] YU, C., LAKSHMANAN, L. V., AND AMER-YAHIA, S. Recommendation diversification using explanations. In *2009 IEEE 25th International Conference on Data Engineering*, pp. 1299–1302. IEEE (2009).

[130] ZAMAN, T. R., HERBRICH, R., VAN GAEL, J., AND STERN, D. Predicting information spreading in twitter. In *Workshop on computational social science and the wisdom of crowds, nips*, vol. 104, pp. 17599–601. Citeseer (2010).

[131] ZAMANI, H., SHAKERY, A., AND MORADI, P. Regression and learning to rank aggregation for user engagement evaluation. In *Proceedings of the 2014 Recommender Systems Challenge*, p. 29. ACM (2014).

[132] ZHAO, X. AND TAJIMA, K. Online retweet recommendation with item count limits. In *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01*, pp. 282–289. IEEE Computer Society (2014).