A CONTENT BASED MICROBLOGGER RECOMMENDATION MODEL

by

Hüseyin Burak Çelebi

B.S., Computer Science, İstanbul Bilgi University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2012

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Suzan Üsküdarlı, for her endless support, guidance, and understanding throughout this thesis. I could not have imagined having a better advisor and mentor for my M.Sc study. I am grateful to the members of my committee, Assoc. Prof. Yağmur Denizhan and Assist. Prof. Albert Ali Salah for their time and valuable critiques which have improved the quality of the work.

I am grateful to TÜBİTAK for supporting my graduate study with "The National Scholarship Program for M.Sc. Students - 2210". This work is also partially supported by B.U. Research Fund (5079).

I would also like to thank the members of SoSLab and CASLab for their friendship and support. I gratefully acknowledge the evaluators of the prototype application for the time they invested as well providing very useful feedback.

Finally, my special gratitude goes to my family for their unconditional love and endless support through my life.

ABSTRACT

A CONTENT BASED MICROBLOGGER RECOMMENDATION MODEL

Social networks are one of the most significant information sources on the Internet. People share information, their feelings, their opinions and interesting links. A microblogging system is a special kind of social network in which users post short but frequent update messages. Microbloggers subscribe (follow) to posts of others. However, finding relevant microbloggers to follow is a major problem, due to the massive quantity of users as well as the difficulty of mentally aggregating fragmented short contributions. In this thesis, a content based recommendation model is proposed, which given a query recommends a set of ranked microbloggers. This model focuses on the content of posts as well as other characteristics of microbloggers to evaluate the relevance of microbloggers to the query. This thesis describes the model and a prototype implementation. Finally the outcome of a test with 41 users is discussed along with observations and recommendations for improved recommendations.

ÖZET

MİKROBLOG SİSTEMLERİ İÇİN İÇERİK TABANLI BİR KULLANICI TAVSİYE MODELİ

Sosyal ağlar İnternet'teki en önemli bilgi kaynaklarından biridir. İnsanlar her hangi bir konudaki bilgilerini, hislerini, görüşlerini ve ilginç bulduğu İnternet adreslerini bu ağlarda paylaşırlar. Bir mikroblog sistemi, kullanıcılarının oldukça kısa uzunlukta fakat sıkça yazdıkları özel bir çeşit sosyal ağdır. Kullanıcılar diğerlerine abone olarak yazdıklarını takip edebilir. Ancak bir konu ile ilgili takip etmeye değer mikroblog kullanıcısı bulmak bu tip sistemler için ciddi bir problemdir. Bu sorunun kaynağı yüz milyonlarca mikroblog kullanıcısının bu sistemlerde yaptıkları katkının, sistemin doğası gereği sınırlı uzunluktaki yazılarında dağınık bir biçimde yer almasından kaynaklanmaktadır. Bu tezde bir sorguya karşılık, sıralanmış olarak mikroblog kullanıcısı öneren içerik bazlı bir tavsiye sistemi modeli sunarak bahsettiğimiz problemi çözmeye çalışıyoruz. Bu model içeriğe odaklandığı kadar tavsiye istenen konu ile ilgili mikrobloga yapılan katkıya ilişkin bazı ölçümleri de sürecin içine dahil etmektedir. Tezde modelin formal yapısını ve örnek bir uygulaması ortaya konmaktadır. Çalışmanın sonunda bir grup kullanıcının kendi sorgularına karşılık uygulamanın ürettiği çıktıları değerlendirmesi istenmiştir. Bu testin sonucu ile hem modelin başarısını ortaya koyuyor hem de gelişim alanlarını tartışıyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Fd_r	Feedness of reshared posts by a microblogger
Fd_s	Feedness of self posts of a microblogger
Ht_r	Hashtag Usage in reshared posts by a microblogger
Ht_s	Hashtag Usage in self posts of a microblogger
R	Worth Recommending
$\neg$R	Not Worth Recommending
$\neg\mathrm{R}_C$	Not Worth Recommending due to Low Content Based Score
$\neg\mathrm{R}_{TV}$	Not Worth Recommending due to Low Term Variation
Re_r	Number of reshares for reshared posts by a microblogger
Re_s	Number of reshares for self posts of a microblogger
So_r	Socialness in reshared posts by a microblogger
So_s	Socialness in self posts of a microblogger
Tv_r	Term Variation of reshared posts by a microblogger
Tv_s	Term Variation of self posts of a microblogger

LIST OF ACRONYMS/ABBREVIATIONS

API	Application Programming Interface
CBS	Content-based Score
Ev	User test evaluation
IDF	Inverse Document Frequency
IR	Information Retrieval
JSON	JavaScript Object Notation
NA	Not Applicable
NS	Not Satisfied
PS	Partially Satisfied
RDF	Resource Description Framework
Rec	Recommendation decision of Micromender
ReR	Ratio of reshared posts for a microblogger
REST	Representational State Transfer
S	Satisfied
SN	Screen name (username) of a microblogger
SPARQL	SPARQL Protocol and RDF Query Language
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
URL	Uniform Resource Locator

1. INTRODUCTION

Social media in the Internet today allow people to share their knowledge, experiences, articles, news, blogs they read, and photos, videos or any other digital source they like. The popularity of social applications increased after the advent of Web 2.0 [1]. Blogs played an important role throughout the evolution of web since their contribution of the *user generated content* on the Web. Microblogging is a special kind of blogging which allows a very limited number of characters per entry. The simplicity of microblogs presents an easy, fast, and open platform for information sharing.

Microblogs became the fastest growing type of social applications after the introduction of Twitter. Microblogs are now one of the mainstream information sources on the Internet. Millions of Twitter users share more than 300 millions of tweets[1] , i.e. microblog entries, everyday. Microbloggers subscribe others to follow their microblog posts. Sheer amount of content produced in a microblogging environment makes it very difficult to reach a microblogger to follow on a specific topic. Although such systems can list recent entries of microbloggers according to a user query, this is not enough to understand how much relevant and quality content these microbloggers share in general.

In this study, a content-based microblogger recommendation system is proposed. Given a user query, a ranked list of microbloggers worth following is returned for the user. A prototype application of the model is developed using real-time data from Twitter to measure the success of the work.

In Chapter 2, some important background concepts are described to provide a better understanding of the remainder of the thesis. Chapter 3 introduces a number of related academic studies and social applications supplied both from Twitter and other companies. In Chapter 4, formal representation of the proposed model is given alongside the data types and functions. details regarding the implementation of the model is

[1] *Twitter Turns Six*, 2012, `http://blog.twitter.com/2012/03/twitter-turns-six.html`, accessed at March 2012.

introduced in Chapter 5. Results of the evaluation which is performed using a live user test is discussed in Chapter 6. Finally, Chapter 7 contains discussion, conclusions, and future work.

2. BACKGROUND

This section presents some important background information about concepts and technologies used in this study to provide better understanding of the proposed model. Section 2.1 provides brief information about microblogging and Twitter, the most popular microblogging service on the Internet today. In Section 2.2, the definition of Collaborative Bookmarking is given and a sample platform, Delicious [2], which plays an important role in the implementation of our model. Section 2.3 consists of brief information about methods borrowed from the domain of Information Retrieval. Finally a brief overview of Semantic Web, which is used only experimentally throughout our study, is given in Section 2.4.

2.1. Microblogging and Twitter

A *blog* is a type of website which is updated whenever the owner of the blog wants to. It is composed of blog posts about anything. Personal blogs, corporate and organizational blogs are some of the common type of blogs on the Internet today. *Microblogging* is a special kind of blogging that allows a limited number of characters. Therefore it is very convenient to write a microblog using a mobile device.

Twitter [3] is a very popular microblogging service which is the 9$^{\text{th}}$ web site on the Internet in terms of traffic according to *Alexa*[2] . Twitter defines itself as a real-time information network that connects people to the latest information about what they find interesting[3] .

A microblog post in Twitter is called a *tweet* which contains 140 characters or fewer[4] . Posts are composed of plain text, URLs, and words having a special meaning in Twitter (hashtags, mentions, and retweets). The # symbol, called a *hashtag*, is used

[2] *Twitter.com Site Info on Alexa*, 2012, `http://www.alexa.com/siteinfo/twitter.com`, accessed at March 2012.

[3] *About Twitter*, 2012, `http://twitter.com/about`, accessed at March 2012.

[4] *Why 140 Characters?*, 2012, `http://support.twitter.com/articles/127856-about-tweets-twitter-updates`, accessed at March 2012.

to mark keywords or topics in a tweet. It was created organically by Twitter users[5] . A mention is any Twitter update that contains @username anywhere in the body of the tweet[6] . Twitterer means a Twitter user. Twitter supports only asymmetric follow relationships between users. More about the Twitter jargon can be found on the Twitter glossary page[7] .

As of April 2012, 140 million active microbloggers of Twitter produce more than 340 millions of tweets everyday[8] . Highest value for TPS (tweets per second) was hit for the Champions League match between Barcelona and Chelsea in April 2012. It was nearly 14000 Tweets per seconds[9] .

The vast amount of posts and microbloggers in Twitter is charming for researchers interested in social network analysis[10] . We also think that Twitter is an excellent choice to use it as a playground for our model. Other notable microblogging services are Google+ [4], Friendfeed [5], Plurk [6], StatusNet [7], and Identi.ca [8].

Unlike blogs, microblogs contain frequent but very short entries which makes understanding the scope of the content produced difficult. A blog usually contains *blogroll* which is a list of other blogs that the blogger recommends by providing links to them. Microbloggers, on the other hand, have a subscription (or following) mechanism which does not present a strong recommendation indicator as a blogroll. Subscribing others in a microblogger is much *cheaper* than to place a blog to a blogroll or to connect others in other social applications such as Facebook [9] and LinkedIn [10].

---

[5] *What Are Hashtags?*, 2012, `http://support.twitter.com/articles/49309-what-are-hashtags-symbols`, accessed at March 2012.

[6] *What Are Replies and Mentions?*, 2012, `http://support.twitter.com/articles/14023-what-are-replies-and-mentions`, accessed at March 2012.

[7] *The Twitter Glossary*, 2012, `http://support.twitter.com/articles/166337-the-twitter-glossary`, accessed at March 2012.

[8] *Shutting Down Spammers*, 2012, `http://blog.twitter.com/2012/04/shutting-down-spammers.html`, accessed at March 2012.

[9] *Goal*, 2012, `http://blog.uk.twitter.com/2012/04/goal.html`, accessed at March 2012.

[10] *Bibliography of Research on Twitter & Microblogging*, 2012, `http://www.danah.org/researchBibs/twitter.php`, accessed at March 2012.

2.2. Collaborative Bookmarking

Collaborative Bookmarking or *Social Bookmarking* is a term used for organizing online resources using bookmarks that reference them rather than the resources themselves. The term *social bookmarking* coined by Delicious [2] which allows collecting and sharing web bookmarks using tags. There exists several collaborative bookmarking sites[11] having different capabilities and target domains. Common feature of these services is *tagging*. A *tag* is a non-hierarchical term or keyword assigned to a digital information item by the creator or viewer of the item. Tagging became an important feature of Web 2.0 applications. Users can organize bookmarks, photos, videos, blog posts, articles, etc. in flexible ways via tagging and develop shared vocabularies known as folksonomies[12] .

2.2.1. Delicious

Delicious is one of the most popular social bookmarking site on the Internet. Each registered user has collection of links organized with tags. When a user wants to add a new link to his/her collection, previously used tags for that link are suggested to the user, if any. Figure 2.1 shows a sample form for saving a new link. Suggested tags are the most frequently used ones in the site for the link. The user may choose from suggested tags, and/or write a new tag.

Delicious provides a number of APIs for the data collected in the site[13] . The information is supplied in JSON format. This study uses the method for fetching recent bookmarks by any number of tags. For example, recent 40 links tagged with *green* and *network* are reached with this URL: `http://feeds.delicious.com/v2/json/tag/green+network?count=40`. The most frequently used tags for each link are also given. The implementation of the model proposed in this thesis is using these tags to enhance the user query for a recommendation. Figure 2.2 shows the co-occurred

[11] *List of Social Bookmarking Websites*, 2012, `http://en.wikipedia.org/wiki/List_of_social_bookmarking_websites`, accessed at March 2012.

[12] A term coined by Thomas Vander Wal, is a portmanteau of folk and taxonomy [11]

[13] *Developers Resources for Delicious*, 2012, `http://delicious.com/developers`, accessed at March 2012.

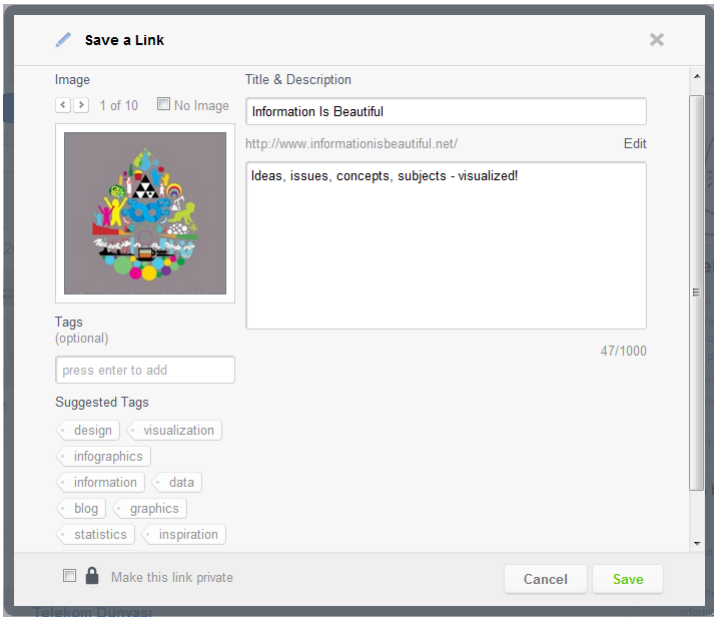


Figure 2.1. Saving A New Link In Delicious.

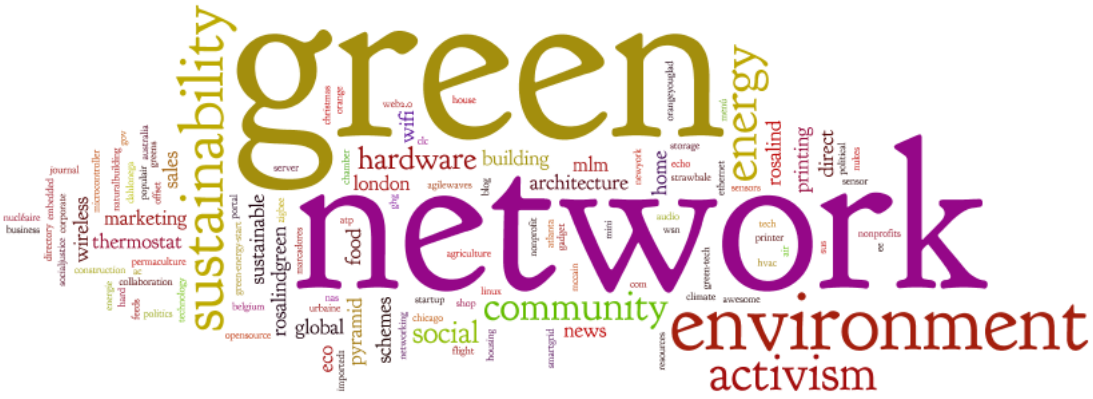


Figure 2.2. Tag Cloud For Green Network.

tags used for links tagged with *green* and *network* as a tag cloud. The size of a tag is proportional to the occurrences and colors are chosen randomly.

2.3. Information Retrieval

Many techniques of Information Retrieval (IR) is used in most of studies whose main concern is to process texts written in a natural language. Term Vector Model and Term Frequency-Inverse Document Frequency (TF-IDF) is two main concepts borrowed from IR domain used in the proposed model of this work.

2.3.1. Term Vector Model

Term Vector Model (or Vector Space Model) is a mathematical model for representing text documents. It was first introduced with the SMART system in 1950 [12] and became a popular model in the domain of Information Retrieval. In this model a document or a query is represented as a vector of values stating the weights of terms [13]. The definition of *term* depends on the context or application.

In this work, words and hashtags used by a microblogger is represented with Term Vector Model. Therefore, it becomes possible to perform search operations on a set of microblogs, rank them, and make content-based comparisons among each other.

2.3.2. Term Frequency - Inverse Document Frequency Weighting

Term Frequency - Inverse Document Frequency (TF-IDF) is one of the best known weighting schemes which can be used with Term Vector Model [14]. In this approach, the weight of a term is calculated using *term frequency* (TF) and *inverse document frequency* (IDF) as follows:

$$w_{t,d} = tf_{t,d} \times idf_t$$

where t is the term and d is the document. The term frequency (TF) of term t in document d is simply the number of times t occurs in d. However using raw term frequency does not work well when ranking documents since relevance does not increase proportionally with the term frequency. Therefore IDF, which considers the global weight of the term, is utilized. IDF is defined as

$$idf_t = \log\left(\frac{|D|}{|\{d' \in D \mid t \in d'\}|}\right)$$

where $|D|$ is the total number of documents in the document collection and denominator is the number of documents containing the term t. IDF increases weights of rare terms which are more informative than terms appearing in many documents. In other words, too frequent terms are not good discriminators between documents. By contrast, rare ones are assumed to be good discriminators since they appear in few documents [13].

TF-IDF is also preferred for the implementation of many proposed models dealing with microblogging systems [15–23]. Its simplicity, proven success, and suitability for the content-based approaches make TF-IDF a convenient choice for the proposed model in this study.

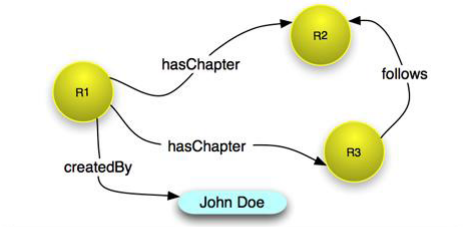
Theoretical background of TF-IDF is discussed in [24].

2.4. Semantic Web

"Semantic Web" means a Web with a meaning and making it possible for the machines to understand and process. The term was coined by Tim Berners-Lee, the inventor of the World Wide Web and it became a collaborative movement led by the *World Wide Web Consortium* (W3C). It is based on the *Resource Description Framework* (RDF), a W3C recommendation.

2.4.1. RDF

RDF is a family of specifications for representing information about resources on the Web and describing qualified relationships among them. The relations are built with *subject-predicate-object* expressions, known as *triples* in RDF terminology. In other words, a Web resource (the *subject*) is linked to another one (the *object*) with a directional relation (the *predicate*). An example is shown in Figure 2.3.



Subject	Predicate	Object
R1	hasChapter	R2
R1	hasChapter	R3
R3	follows	R2
R1	createdBy	"John Doe"

Figure 2.3. RDF Triples: Subject, Predicate, and Object.

2.4.2. Linked Data

Linked Data is a community effort that aims to publish the Web data using RDF and build the connections between related ones. Wikipedia defines Linked Data as *"a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF."*.

2.4.3. SPARQL

SPARQL (pronounced "sparkle") is a language designed to query data in RDF. SPARQL is a recursive acronym for "SPARQL Protocol and RDF Query Language". It became an official W3C Recommendation on 15 January 2008[14] and accepted as the query language for the Semantic Web. Tim Berners-Lee emphasises importance of it with the following sentence "Trying to use the Semantic Web without SPARQL is like trying to use a relational database without SQL." An sample SPARQL query for the question "What are all the country capitals in Africa?" is given in Figure 2.4.

[14] *SPARQL Is A Recommendation*, 2012, `http://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation`, accessed at March 2012.

```
PREFIX abc: <http://example.com/exampleOntology#$>
SELECT ?capital ?country
WHERE {
  ?x abc:cityname ?capital ;
     abc:isCapitalOf ?y .
  ?y abc:countryname ?country ;
     abc:isInContinent abc:Africa .
}
```

Figure 2.4. SPARQL Query Modeling “What are all the country capitals in Africa?”.

3. RELATED WORK

3.1. Academic Studies

With the popularity of social networks the data produced by social applications, especially microblogging systems, have received substantial attention from research communities. Academicians proposed several approaches addressing the common problems of this field.

In [19], John Hannon *et al.* evaluates a range of different profiling and recommendation strategies on Twitter. Twitterers are modeled using their tweets and follow relationships. This work includes 2 parts; search and recommendation. The first part is a content-based search on microbloggers with a plain text query. The output is a set of ranked microbloggers without ranking scores. Each microblogger's number of followers, following and tweets and 10 most frequent words are given are listed. On the other hand the recommendation part takes the system user as a seed user and suggests microbloggers based on the user's own tweets, friends or followers. There are 5 profiling strategies the system uses. They define 5 profiling strategies for the user: (i) own tweets, (ii) tweets of the followees, (iii) tweets of the followers, (iv) ids of the followees, (v) ids of the followers. Finally, they produced 9 strategies using the combinations of those five choices. They evaluated the strategies with a live user tests. They count how many of the recommendations are already in the user's known followees list. Note that this system uses previously collected 20000 microbloggers whereas the proposed model in this paper uses the search functionality of microblogging system. Therefore, more real-time recommendation become possible.

In [20], URLs shared in tweets used as items to make a recommendation to a microblogger. Microblogger's topic profile and social network is analysed according to the URLs shared. URLs are indexed to make both tweet and URL recommendation.

In [25], interestingness of tweets are studied. They ignore the retweet counts

and concentrated in the style and the content of a tweet. The set of features they considered are presence of exclamation and question marks, URLs, usernames and hashtags, positive and negative terms, emoticons, and sentiments. They utilize a Naive Bayes classifier to obtain the probability of being retweeted for an individual tweet.

In [26], given a user query they worked on retrieving relevant tweets. *Google Search API*[15] is used to make the query expansion. They collected the titles of last 64 pages retrieved and the 5 most frequent word-level n-grams ($n = 1, 2, 3$) were added to the original topic. The microblog dataset they use is supplied from TREC 2011 microblog track[16] . Our study differs from this work in a couple of ways. The dataset they use consists of news oriented tweets where the work presented in this thesis is evaluated using the real-time microblog data with no content type restriction. Moreover, we believe a social platform should be used for a query expansion targeting microblog environments. Therefore we decided to use a collaborative platform, delicious.com, for that purpose.

In [27], Nagmoti *et al.* studied ranking individual *tweets* retrieved with a query based search operation. They considered both properties of the twitterers and the tweets themselves. They found that using the social network properties of the authors as well as some properties of tweets such as the length of the microblog entry and the presence of a URL plays an important role in ranking tweets.

Akman in [28], proposes an approach for auto-tagging and comparison of microbloggers. The work reveals the contribution of a microblogger by processing weighted set of tokens in microblog posts. The importance of a word means the numbers of occurrences used by the microblogger. In our model, microbloggers are not analyzed individually, we have a set of microbloggers discovered from the entire Twitter space to be processed. That's why the importance of a word in the microblogger set is also important. So *TF-IDF* weighting scheme in which local and global weighting are

[15] *Google Web Search API*, 2012, `http://developers.google.com/web-search`, accessed at March 2012.

[16] *Microblog Dataset from TREC 2011*, 2011, `http://trec.nist.gov/data/tweets`, accessed at March 2012.

calculated together is used.

Yurtsever in [29], also tries to categorise microbloggers individually. Top keywords, which are found by calculating the frequency of words, are used to query semantic resources in DBPedia. The approach is based on the categorical hierarchy of those resources.

Degirmencioglu in [30], focuses on word-tag, word-user and tag-user networks in microblogging environments to identify contribution of microbloggers and communities of interest. Moreover, social network properties of these communities are analyzed.

[28], [29], and [30] lack the detailed microblogger characterization which effects the decision of following. Although all of them includes substructures for making recommendations, they do not introduce a complete model. They left recommendation module as a future work since the main concentration of these efforts are different.

Aslan in [31] proposes a model for extracting news contributions in microblogs. A pattern is used to identify the informative microblog posts. Moreover, an analysis made to study temporal and quantitative properties of such posts. Unlike [28], [29], and [30] who study microbloggers, i.e. twitterers, only individually, this study also uses *public timeline* of Twitter. The proposed model in our work uses the search feature of microblogging systems, so there is a similarity between his work and our work according to this aspect.

3.2. Solutions from Twitter

In this section, facilities supplied by Twitter for finding microbloggers easily are presented. Non of them actually presents a content-based solution similar to the work discussed in this thesis.

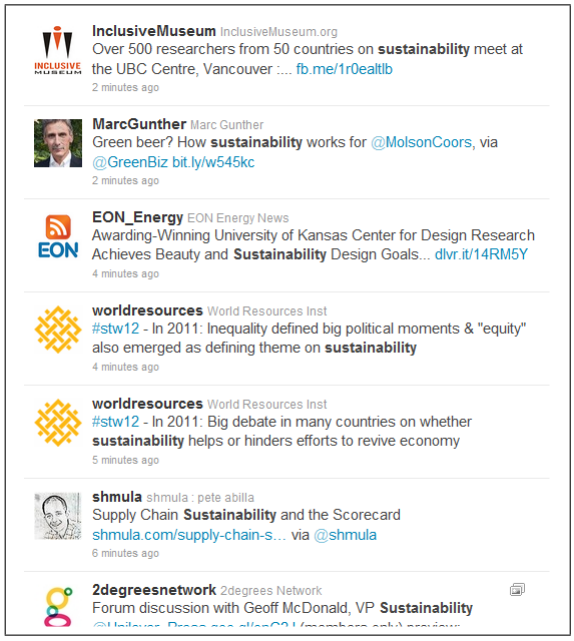


Figure 3.1. Twitter Search Result Page For The Query "sustainability".

3.2.1. Twitter Search

'Twitter Search"[17] is used to get latest or top tweets according to a search query. Figure 3.1 shows a result page for the query "sustainability". It also supports a number of operators for advanced search. Sample usages are listed in Table 3.1.

Although *Twitter Search* is good start point to discover new people on Twitter, it is impossible to make a decision about the relevance of Twitter users listed. The fragmented nature of contributions in microblogs does not allow to understand the value of the users. One should open all twitterers' pages, read the tweets on each page, and try to make a decision accurately. This process may cause a serious cognitive overload. Moreover, the person who wants to get a recommendation on a subject may not have competence in that area.

3.2.1.1. Twitter People Search. It is an extension of regular *Twitter Search*. It does not provide a content-based recommendation. The twitterers on the results also include people whose username, or "bio"[18] includes the query.

[17] *Twitter Search*, 2012, `http://twitter.com/search`, accessed at March 2012.
[18] *Description of Bio*, 2012, `http://bit.ly/L8ATGP`, accessed at March 2012.

Table 3.1. Twitter Search Operators.

Operator	**Finds tweets...**
twitter search	containing both “twitter” and “search”. This is the default operator.
“happy hour”	containing the exact phrase “happy hour”.
love OR hate	containing either “love” or “hate”(or both).
beer -root	containing “beer” but not “root”.
#haiku	containing the hashtag “haiku”.
from:alexiskold	sent from person “alexiskold”.
to:techcrunch	sent to person “techcrunch”.
@mashable	referencing person “mashable”.
“happy hour” near:“san francisco”	containing the exact phrase “happy hour” and sent near “san francisco”.
near:NYC within:15mi	sent within 15 miles of “NYC”.
superhero since:2010-12-27	containing “superhero” and sent since date “2010-12-27” (year-month-day).
ftw until:2010-12-27	containing “ftw” and sent up to date “2010-12-27”.
movie -scary :)	containing “movie”, but not “scary”, and with a positive attitude.
flight :(	containing “flight” and with a negative attitude.
traffic ?	containing “traffic” and asking a question.
hilarious filter:links	containing “hilarious” and linking to URLs.
news source:twitterfeed	containing “news” and entered via TwitterFeed

Figure 3.2. Sample Suggestion From Twitter Just After Following A Twitterer.

3.2.2. "You might also want to follow"

When a person starts to follow a Twitter account, a small user suggestion part is seen. Figure 3.2 shows the suggestion after following the user *shaunontheair*. We could not find any official declaration about the algorithm on the Internet, but we performed a number of tests and observed that there exists a mutual follow relationships between the user followed recently and the suggested users. This approach may offer both relevant and non-relevant people regarding the content.

3.2.3. Similar Users

Twitter allows listing similar people to any twitterer. For example, this link lists accounts similar to user "BurakCelebi": *http://twitter.com/similar_to/BurakCelebi*

Although the technical background of this feature is not declared, we observed that connections of the user's plays an important role. We think that no or poor content-based analysis is performed since the result page may also list people having no tweets at all.

3.2.4. Twitter Categories

"Twitter Categories"[19] includes topics like music, sports, funny, fashion, etc. which lists suggestions generated by Twitter itself.

[19] *Twitter Categories*, 2012, `http://twitter.com/who_to_follow/interests`, accessed at March 2012.

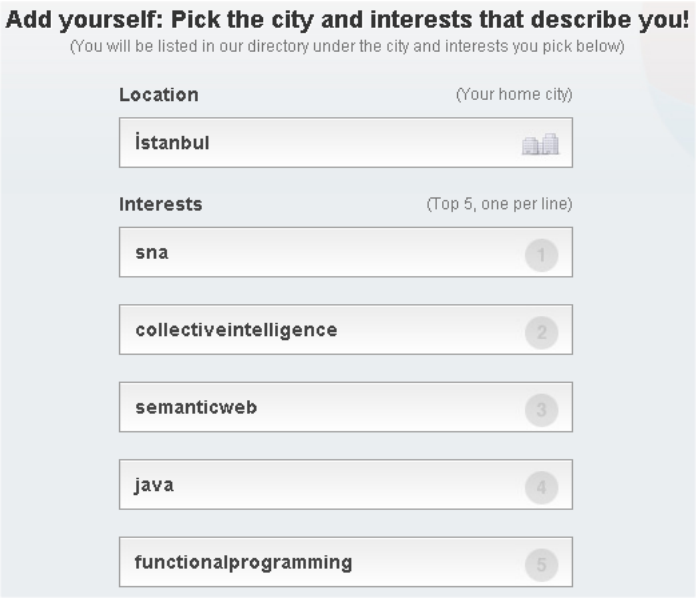


Figure 3.3. WeFollow - Adding Account.

3.3. Other Applications

3.3.1. WeFollow

WeFollow [32] is one of the earliest applications for grouping microbloggers according to their interests. Digg [33], a popular social news site, owns WeFollow. It is basically a tag based directory application that categorizes *twitterers*.

A user only can add twitter account she owns by using 5 tags which fits best according to her. Figure 3.3 shows an example form for adding a new user. Tags (i.e. interests) can be chosen from previously used ones or a new tag can also be written. After adding the account, only the owner can edit it. Any user can browse twitterers in a particular tag by choosing *most influential* or *most followers* lists. Figure 3.4 shows the page for tag *socialmedia*. The calculation of an influence is not known publicly. There is a search functionality which accepts only 1 previously used tag. There is no content based search or recommendation.

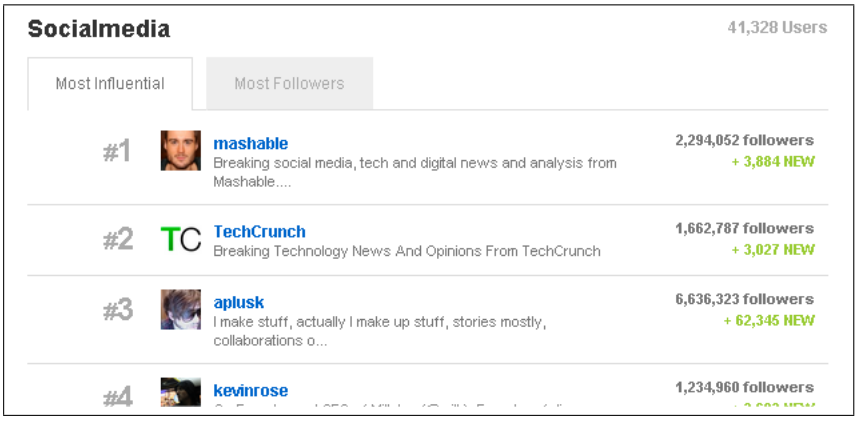


Figure 3.4. WeFollow Page For "socialmedia".

3.3.2. Klout

Klout [34] is a social media tool that measures the influence of a person across the Web by using her social media accounts such as Twitter, Facebook [9], Google+, LinkedIn [10], etc. While Klout calculates an overall score for each user, it also finds a user's most influential topics based on the engagement she receives from others for the content produced. The high level approach and scoring factors is partially declared on Klout's help pages[20] . While the site provides a search functionality, we do not think it works well with free-text queries unlike topics.

3.3.3. Twiends

Twiends [35] is a rather new social networking application for the categorization of people on Twitter, similar to WeFollow. Twitterers can only add their own accounts with at most 5 interests. There exists a search functionality that only gets the page of a previously defined interest. Microbloggers in an interest page are ranked according to their number of followers.

[20] *Understand Klout*, 2012, `http://klout.com/understand/klout`, accessed at March 2012.

4. MODEL

This chapter proposes a model for recommending a ranked set of microbloggers based on a user query. At the highest level, the recommendation model, called Micromender, can be viewed as follows:

$$User\ Query \Rightarrow \boxed{\text{Micromender System}} \Rightarrow List\ of\ Microbloggers$$

The highly fragmented nature of microblog contributions makes it challenging to assess the nature of a microbloggers contributions. The friend of a friend approach to recommendation has been employed in social networking environments, however, this approach is hardly useful in microblogger contexts due to high number of follow relationships in such environments.

This model suggests that microbloggers worth recommending are actively contribute content related to the user query. It must be noted that microblogging systems are very focused on the present time, therefore the notion of actively contribute may mean that they do so recently. However, users who contribute more persistently are deemed more relevant.

The main concern in trying to identify microbloggers to recommend is to try to determine the intention of the user query and how a microbloggers contributions may relate to a given query. Since user queries as well as microblog posts are very terse, the user query is enriched with additional related words. The aim is to be able to fetch more relevant posts.

For example, the user query *"semantic web"* may be extended to *"semantic web, metadata, rdf, tagging, microformats"*. The idea being, if someone were to contribute about semantic web, they would use terminology related to those terms. User queries are extended by retrieving related tags from social systems. The idea is to chose a

Table 4.1. PrimitiveDT: Primitive Data Types For Handling Microblogs.

Type Name	**Description**
Int	an integer value
Double	a double value
Text	a sequence of characters including white spaces
Word	a sequence of characters delimited by white space.
URL	uniform resource locator as defined by W3C

similar user base, names social media for identifying enriching terms.

Micromender proposes a content based microblogger recommendation by:

- processing and enriching the user query
- fetching related microbloggers
- fetching and processing the posts of these microbloggers to evaluate their relevance and identify other candidates for recommendation
- ranking the identified microbloggers

In order to describe the processing involved, we introduce several data types and functions. The data types we use may be considered as PrimitiveDT (and predefined) (Table 4.1), and MicroblogDT (Table 4.2) that describes microblogging system.

Several conventions are used to present types and functions. The processing of microblog posts involves numerous *sets*, *bags*, and *lists*. Such types are represented as Type_s for sets, Type_b for bags, and Type_l for lists of a given type. For example, Word_s denotes a set of *Words*. Types with "#' subscripts represent integer values. Often descriptive type names are introduced for the sake of readability, such as $\text{Followee}_{s\#}$ to indicate the cardinality of a followee set. A simple Integer type would be suffice, but would make it difficult to follow.

The main functions are listed in Table 4.6.

Table 4.2. MicroblogDT: Data Types Of A Microblogging System.

imports PrimitiveDT		
Type Name	**Specification**	**Notes**
MBlogSys	MBlog_s	A set of microblogs
MBlog	$<$ ScreenName, MBlogEntry_s, MBlogStats $>$	A microblog consisting of a name, a set of contributions, and statistics.
Post	Text	The text of the content in a microblog post.
MBlogEntry	$<$ ScreenName, Post, $\text{Retweet}_{\#}$ $>$	A microblog entry consisting of a name, post content, and statistics.
ScreenName	@[A-Za-z0-9_]+	Microblog user name.
MBlogStats	$<$ $\text{Followee}_{s\#}$, $\text{Follower}_{s\#}$, $\text{MBlogEntry}_{s\#}$ $>$	Information about a microblogger's posts and social network.
Token	Word | Hashtag | Mention | URL	Types of token in a post.
Hashtag	#[A-Za-z0-9_]+	A microblog tag.
Mention	ScreenName	A reference to a microblogger.
Follower	ScreenName	A microblogger subscribed to by this microblogger
Followee	ScreenName	A microblogger who subscribes this microblogger.

Table 4.3. TaggingDT: Data Types Of A System Which Has Tagging Feature.

Type Name	**Specification**	**Notes**
TaggingSys	SocialItem_s	A social system composed of a SocialItem_s.
SocialItem	$< \text{Item}, \text{Tag}_s >$	A social item is an internet resource, such as a URL, a video, or an image, tagged with a set of tags.

Table 4.4. Elements Of MetaCharacteristics.

Characteristics	**Description**
Socialness	Number of references (*mentions* in Twitter) to other microbloggers per post.
Feedness	Number of unique URLs shared per post.
HashtagUsage	Number of unique hashtags per post.
Retweeted	Average retweet count of a post.
TermVariation	Number of unique words per post.

Table 4.5. MicroblogProcessingDT: Data Types For Processing Microblogging System.

imports MicroblogSystemDT	
Type Name	**Specification**
MetaCharacteristics	$<$ Socialness, Feedness, HashtagUsage, Retweeted, TermVariation $>$
ProcessedMBlog	$<$ MBlog, MBlogWordContent, MetaCharacteristics, MetaCharacteristics, Double $>$
StopWord	Word
TokenizedPost	$<$ Word_b, Hashtag_b, StopWord_b, URL_b, Mention_b $>$
MBlogWordContent	A bag of words obtained from words and hashtags.
Query Processing	
Query	Word_s
Tag	Word+
WeightedTag	$<$ Tag,Weight $>$
Weight	Int
RelatedConcepts	WeightedTag_s
TopRelatedConcepts	WeightedTag_l
Ranking	
TfIdfMatrix	$<$ MBlog, $<$ Word, Double $>_s$ $>$
QueryVector	$<$ Word, Double $>_s$

Table 4.6. Microblog System Processing Functions.

Function	**Description**
GetQuery() : Query	Returns the user query.
Query Handling	
FindRelatedConcepts(Query, TaggingSys) : RelatedConcepts	Given a query and a tagging system, it returns tags related to the query.
FindTopRelatedConcepts(RelatedConcepts) : TopRelatedConcepts	Most frequent concepts in RelatedConceptsare found.
Selecting Potential Microblogs	
HashQuery(Query) : Hashtag	Joins query terms with spaces and prepend a # sign.
FindMBloggers(Word_s, MBlogSys) : ScreenName_s	Recent usernames of microbloggers who wrote posts containing the input text.
FindMBloggersHT(Hashtag, MBlogSys) : ScreenName_s	Recent usernames of microbloggers who wrote posts containing the input hashtag.
GetMBlog(ScreenName, MBlogSys) : MBlog	Returns the microblog of a microblogger.
GetMostValuableMicrobloggers(ProcessedMBlog_l) : ScreenName_s	Returns most contacted (includes retweeted microbloggers alongside the mentioned ones) microbloggers among the given MBlog_s.
MicroblogEntry Processing	
RemoveStopWords(Word_s, StopWord_s) : Word_s	Returns $(\text{Word}_s - \text{StopWord})_s$
TokenizePost(MBlogEntry) : TokenizedPost	Returns the tokenized form of the Post in a MBlogEntry.
FindWordContent(MBlogEntry) : MBlogWordContent	Returns the MBlogWordContent of a MBlogEntry.

Table 4.7. Content Based Ranking Functions.

Function	**Description**
Recommend(Query, MBlogSys, TaggingSys) : ScreenName_l	Main recommendation function. Given a user query, a microblogging system, and a social system, it returns a ranked list of microbloggers.
Content Based Ranking	
ConstructMatrix(ProcessedMBlog_s) : TfIdfMatrix	Constructs the TfIdfMatrix for a given ProcessedMBlog_s.
RankVectors(TfIdfMatrix, QueryVector, Double) : ScreenName_l	Ranks microblogger vectors according to RelatedConcepts found.
ProcessMBlogs(ScreenName_s, MBlogSys) : ProcessedMBlog_l	Builds a ProcessedMBlog_l according to a given ScreenName_s.
ContentBasedRanking(ProcessedMBlog_l, QueryVector) : ProcessedMBlog_l	Performs content based ranking of a given ProcessedMBlog_l according to the QueryVector. See Figure 4.6.

Table 4.8. Ranking Functions Related To Meta Characteristics.

Function	**Description**
Rank(ProcessedMBlog$_l$, TopRelatedConcepts) : MBlog$_l$	Ranks the given ProcessedMBlog$_l$.
FindRelevantMBlogEntries(ProcessedMBlog, TopRelatedConcepts, Query, Hashtag) : MBlogEntry$_s$	Returns MBlogEntry$_s$ of MBlogcontaining TopRelated-Concepts
CharacterizeMBlog(MBlog, MBlogEntry$_s$) : MetaCharacteristics	Given an MBlog, characteristics of it are calculated.
ExcludeLowTV(ProcessedMBlog$_l$) : ProcessedMBlog$_l$	Given a ProcessedMBlog$_l$, excludes each ProcessedM-Bloghaving low *term variation* value.
ComputeAvgMeta(ProcessedMBlog$_l$) : MetaCharacteristics	Computes average MetaCharacteristics of a given ProcessedMBlog$_l$
NormalizeMeta(MetaCharacteristics, MetaCharacteristics) : MetaCharac-teristics	Normalizes a given MetaCharacteristics according to the second MetaCharacteristics given.
ComputeScore(ProcessedMBlog) : Double	Computes the score based on meta characteristics.
ComputeSocialness(MBlog, MBlogEntry$_s$) : Double	Computes the Socialness of a microblogger using MBlogEntry$_s$.
ComputeFeedness(MBlog, MBlogEntry$_s$) : Double	Computes the Feedness of a microblogger using MBlogEntry$_s$.
ComputeHashtagUsage(MBlog, MBlogEntry$_s$) : Double	Computes the HashtagUsage of a microblogger using MBlogEntry$_s$.
ComputeRetweeted(MBlog, MBlogEntry$_s$) : Double	Computes the Retweeted of a microblogger using MBlogEntry$_s$.
ComputeTermVariation(MBlog, MBlogEntry$_s$) : Double	Computes the TermVariation of a microblogger using MBlogEntry$_s$.

Figure 4.2 shows the formal view of input and output of the recommendation system. Main parts of the model will be explained by using formal data types and functions in upcoming sections. Note that the proposed model works with a number of constants used by related functions. Names and descriptions of them are listed in Table 4.9.

The name of the main function in the model is *Recommend* (See Figure 4.1). Given the user query, the microblogging system, and a tagging system, it gives a ranked list of user names of microbloggers as a recommendation.

4.1. Discovering Related Concepts and Query Expansion

Microblogging systems have a large number of users. For example, Twitter has more than 300 millions registered microbloggers as of June 2011[21] . Twitter officially declared that there exists 140 million active users in March 2012[22] . In such a large space, it is clear that a rational selection of a subset is necessary. An intuitive way to reach people who is speaking about something is just using the search facility which is a common feature of many microblogging environments[23] . A typical search works with a user query and outputs a number of latest microblog entries containing words in the query. However, authorities about a subject in a microblogging environment seldom use the subject as a word while posting a message. They mostly use relevant concepts when they produce a content, i.e. compose a new microblog post. Therefore, a strategy for finding relevant concepts of a subject is needed for discovering microblogs having better quality of content.

The proposed model here utilizes an external system to reach related concepts or keywords for the user query. This system is a social media platform containing user generated contributions. Users use *tagging* to describe their contributions. Data types

[21] *Social Networking 'Utopia' Isn't Coming*, 2012, `http://articles.cnn.com/2011-06-27/tech/limits.social.networking.taylor_1_twitter-users-facebook-friends-connections?_s=PM:TECH`, accessed at March 2012.

[22] *Twitter Turns Six*, 2012, `http://blog.twitter.com/2012/03/twitter-turns-six.html`, accessed at March 2012.

[23] *Twitter Search*, 2012, `http://twitter.com/search`, accessed at March 2012.

```
function Recommend (Query query, MBlogSys mbloggingSys, TaggingSys taggingSys):
ScreenName_l

RelatedConcepts relatedConcepts
TopRelatedConcepts topRelatedConcepts
ProcessedMBlog_l pMblogs
ProcessedMBlog_l pMblogsForMVP
ScreenName_s screenNames
WeightedTag concept

relatedConcepts ← FindRelatedConcepts(query, taggingSys)
topRelatedConcepts ← FindTopRelatedConcepts(relatedConcepts)

for each concept in TopRelatedConcepts do
   screenNames.add( FindMBloggers(concept.getTag(), mbloggingSys) )
   screenNames.add( FindMBloggers(concept.getTag().add(query), mbloggingSys) )
end for

screenNames.add( FindMBloggers(query, mbloggingSys) )
screenNames.add( FindMBloggersStr(StrQuery(query), mbloggingSys) )
screenNames.add( FindMBloggersHT(HashQuery(query), mbloggingSys) )

pMblogs ← ProcessMBlogs(screenNames, mbloggingSys)
pMblogs ← ContentBasedRanking(pMblogs, queryVector)

pMblogsForMVP ← ProcessMBlogs(GetMostValuableMicrobloggers(pMblogs), mbloggingSys)

pMblogs ← ContentBasedRanking( pMblogs.add(pMblogsForMVP), queryVector )

return Rank(pMblogs, topRelatedConcepts)
```

Figure 4.1. Recommendation Algorithm.

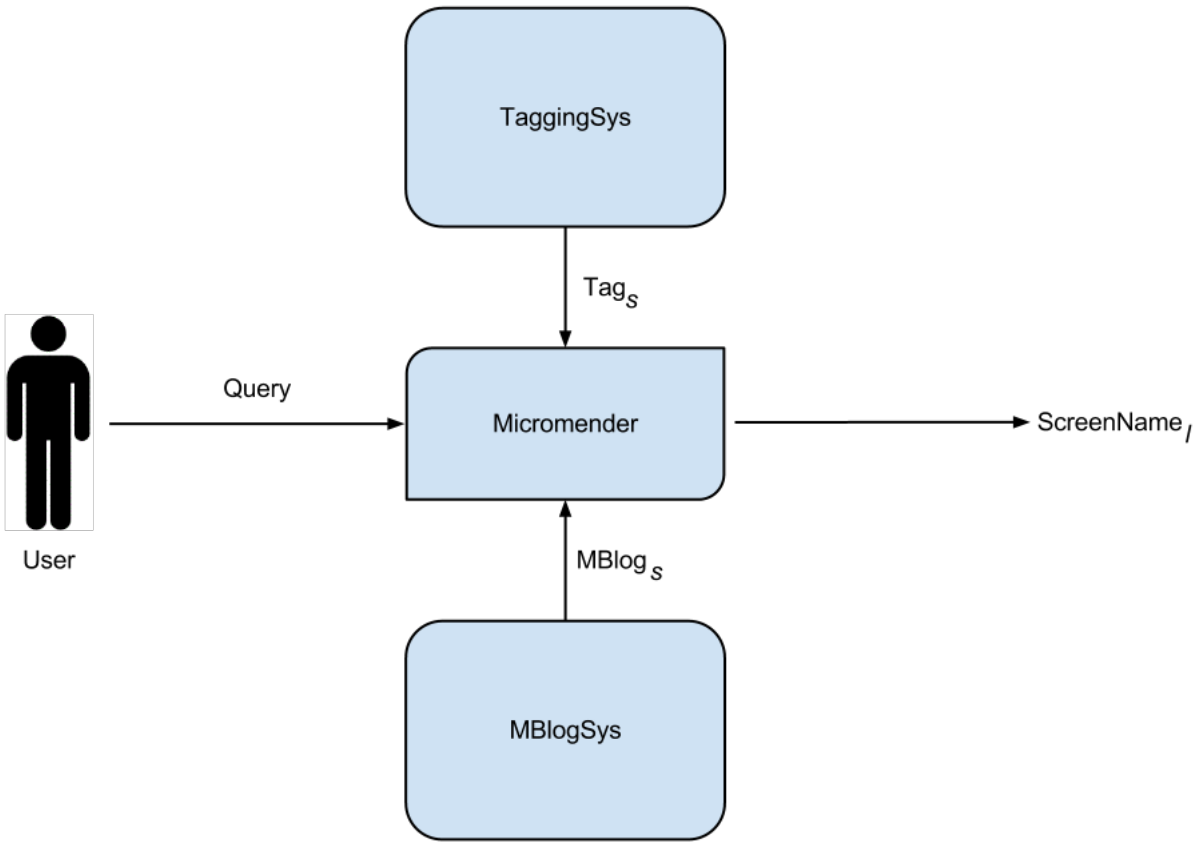


Figure 4.2. Formal Input And Output Of The System.

Table 4.9. Model Constants.

Constant	**Description**
MAX_SOCIAL_ITEMS	Maximum number of social items to fetch tags
MOST_FREQ_CONCEPTS	Number of tags in TopRelatedConcepts
RECENT_MBLOGGERS	Number of microbloggers fetched for each microblog search
RECENT_MBLOG_TWEETS	Number of posts fetched for each microbloggers
TOP_MBLOGGERS	Number of microbloggers who get highest content-based score
NUM_OF_ITERATIONS	Number of iterations for getting referenced microbloggers
MIN_CONTENT_BASED_SCORE	Minimum content-based score for microbloggers to be recommended.
MIN_TERM_VARIATION	Minimum term variation value for microbloggers to be recommended.
RETWEET_RATIO_THRESHOLD_TV	Minimum ratio for the number of posts retweeted. Term variation constraint is check with either microlbogger's self posts or posts retweeted by him/her according to this value.

of these systems are defined as in Table 4.3. Only the function GetSocialItems is defined in our model. It returns a set of social items (SocialItem_s) which are tagged with a set of tags (Tag_s). Micromender utilizes this function in FindRelatedConcepts function whose algorithm is given in Figure 4.3. It takes a *Query* and a *TaggingSys* as inputs and outputs *RelatedConcepts*. We also define *TopRelatedConcepts*, a subset of RelatedConcepts, in Table 4.5.

4.2. Selecting Potential Microblogs

After gathering TopRelatedConcepts, the function FindMBloggers is called once for each kind of queries given in Table 4.10. The output of the function FindMBloggers is a ScreenName_s.

```
function FindRelatedConcepts (Query query, TaggingSys taggingSys):

RelatedConcepts relatedConcepts
SocialItem_s socialItems
SocialItem socialItem
Tag_b tagBag = {}
Tag tag
query ← RemoveStopWords(query)
socialItems ← GetSocialItems(query)
for all socialItem in socialItems do
   for all tag in socialItem.GetTags() do
      tagBag.add(tag)
   end for
end for
relatedConcepts ← ToRelatedConcepts(tagBag)

return relatedConcepts
```

Figure 4.3. Finding Related Concepts.

Table 4.10. Inputs Of FindMBloggers Function.

Input	**Example**
Pure Query.	semantic web
Query as a string.	"semantic web"
The Query as a hashtag.	#semanticweb
Each *Tag* in the *TopRelatedConcepts*.	rdfa ontology w3c linked data . .
Query and each *Tag* in the *TopRelatedConcepts*.	semantic web rdfa semantic web ontology semantic web w3c semantic web linked data . .

4.3. Analysis of Microblogs

According to the view of the proposed model, contents of latest microblog entries reflects the microblog's interests. Therefore entries of candidate microblogs should be collected to perform the analysis.

After having all potential microbloggers screen names, the function ProcessMBlogs is called (See Table 4.7 and Figure 4.5). Using each *ScreenName* information found in last step, functions *GetMBlog* is called. At the end of this procedure, candidate microblogs for recommendation are ready to be processed.

4.3.1. Indexing Contributions

Having latest microblog entries of candidate microblogs, it is time to tokenize the contents of entries. The function *TokenizePost* takes a MBlogEntryas an input and decomposes the contents to get tokens as a Word_b, Hashtag_b, StopWord_b, URL_b, and Mention_b.

Excluding stopwords, the model defines the contribution of a microblog entry with the words and hashtags it has. The function *FindWordContent* is used to determine the *MBlogWordContent* of a microblogger. The functions converts the words to their lowercase form and ignores the # sign in hashtags for unification.

The Figure 4.4 shows a sample microblog post and the contribution of it. Notice that stopwords are removed. Moreover, *usability* written twice since there also exists a hashtag for it.

4.3.2. Extending Potential Microblogs

Giving references to other microblogs is one of the useful features of microblog systems. Microbloggers have opportunity to communicate with other people on the system using this feature. Microbloggers are able to reply any other microblog entry

Post:	8 Advantages of Standardized Usability Questionnaires #usability http://goo.gl/r1iS7 (http://measuringusability.com)
	⇓
Contribution:	[advantages, standardized, usability, questionnaires, usability]

Figure 4.4. An Example For The Contribution Associated With A Post.

of other microbloggers easily. For example, microbloggers on Twitter can use @replies and mentions to interact with others.

If a microblogger is mentioned by others many times, this situation is noteworthy and he/she could be an interesting microblogger. After finding notable microblogs in terms of their contributions by using *Content Based Ranking*, references to other microbloggers are found in top ranked microblogs. The model also uses reshared (*retweeted* in Twitter) microblog entries.

The function *GetMostValuableMicrobloggers* is used to reach such microbloggers. It takes a list of microblogs and as an input, outputs the screen names of most referenced microbloggers. Apart from microbloggers found with *FindMBloggers* in Section 4.2, microbloggers found with this function are also subjected to analysis and ranking.

4.3.3. Calculation of Meta Characteristics

The model introduces 5 meta characteristics to analyse microblog contributions (Table 4.4). The characteristics are calculated separately for the microblogger's self relevant posts and related relevant reshared by him/her. Relevant entries are identified with the function *FindRelevantMBlogEntries*. It simply selects entries which includes terms in the user query, or user query as a hashtag (see function HashQuery in Table 4.6), or any Tag found via function *FindTopRelatedConcepts* as a *contribution*.

Meta characteristics of a microblogger consists of *Socialness*, *Feedness*, *Hashta-*

gUsage, *Retweeted*, and *TermVariation*.

- *Socialness* is used to measure how much social a microblogger is. It equals to the number of references to other microbloggers per microblog post. The more the references the higher the socialness value. We think that good microbloggers refer to others and this creates an opportunity for users to discover new relevant microbloggers. So we have considered this property as a valuable property of a microblogger you may wish to follow.
- *Feedness* refers to the number of external web resources (i.e. URLs) per post[24] . A good microblogger tends to post lots of useful URLs. Therefore a microblogger having high feedness value allows people to move other relevant external resources on the web.
- *HashtagUsage* is the number of unique hashtags per post. Hashtags are used to group a set of relevant posts written by various microbloggers. It is a very common usage in microblogging systems. Using hashtags is a deliberate effort to provide visibility to the posts. Hashtags used for a particular subject act as an up-to-date common vocabulary of that subject among mictobloggers. That's why we wanted to give credits to them.
- *Retweeted* is the average number of times a post has been reshared (retweeted in Twitter). It is a very commonly used feature by microbloggers. A microblogger reshares posts he/she finds valuable with the followers. If this value of a post is high, this means it is a notable one and may be worth reading which is already passed from many filters (i.e. microbloggers). Therefore, it is an important measure for recommending a microblogger.
- *TermVariation* is the number of unique words per microblog post. This number acts as a power of microblogger vocabulary on the subject. The model defines "terms" used by a microblogger as the union of words (excluding stopwords) and words used as a hashtag. We also observed that this value can be used to filter spammers, some kind of bots, some of microbloggers having poor quality content. So we also define a threshold (MIN_TERM_VARIATION) to be used when deciding whether the microblogger is worth recommending or not.

[24] *Influence Metrics of Infochimps*, 2012, `http://bit.ly/Pb15Pg`, accessed at March 2012.

4.4. Ranking

The proposed model applies 2-phase ranking to the candidate microblogs:

(i) Content Based Ranking
(ii) Ranking based on Meta Characteristics

In the first ranking step, the microblogs are analysed and ranked according to their contribution related to the user query. Using the content score calculated, a subset of top-ranked microblogs become eligible for the second ranking step which is based on *Meta Characteristics*. The recommendations are given according to second step.

4.4.1. Content Based Ranking

The meaning of the *content* was defined at Section 4.3.1. In this step, the content, i.e. contribution, is used to rank microblogs to reveal how much they are related to the user query.

4.4.1.1. Construction of TfIdfMatrix. TF-IDF, as stated in Section 2.3.2, is a numerical statistic which shows the importance of a word to a text document. This method is borrowed from *information retrieval* domain to measure the contribution of a microblogger in accordance with the user query.

The *TfIdfMatrix* represents a mathematical matrix whose rows are the contributions collected from the candidate microbloggers, and columns are the *ScreenName*s of the candidate microbloggers. The values show the *importance* of each contribution word made by each microblogger. If a microblogger do not have any contribution for a specific word, then the corresponding value will be 0 for him/her.

4.4.1.2. Construction of QueryVector. QueryVector represents a one dimensional vector whose rows are the contributions collected from the candidate microbloggers, as in

```
function ProcessMBlogs (ScreenName_s sns, MBlogSys mbloggingSys): ProcessedMBlog_l

MBlog mblog
MBlogWordContentmblogWordContents
ProcessedMBlog_l processedMblogs
for all ScreenName sn in sns do
   mblog ← GetMBlog(sn, mbloggingSys)

   for all MBlogEntry mbe in mblog.MicroblogEntry_s do
      mbe.tp ← TokenizePost(mbe)
      mblogWordContents.add(FindWordContent(mbe))
   end for

   processedMblogs.add(< mblog, wordContents, null, null >)
end for

return processedMblogs
```

Figure 4.5. Processing Microblogs

```
function ContentBasedRanking (ProcessedMBlog_l processedMblogs, QueryVector queryVector): ProcessedMBlog_l

TfIdfMatrix matrix
matrix ← ConstructMatrix(processedMblogs)
processedMblogs ← RankVectors(matrix, queryVector)

return processedMblogs
```

Figure 4.6. Content-based ranking

TfIdfMatrix. The values int this vector are calculated according to the user query and related concepts found using the function FindRelatedConcepts.

4.4.1.3. Calculation of Similarity. Similarity between QueryVector with each microblogger vector in TfIdfMatrix how much relevant contributions a microblogger has according to the user query. Cosine Similarity is chosen as a similarity measure in this model where cosine of the angle between two vectors gives the similarity. Similarity value is higher when the angle is smaller.

```
function Rank (ProcessedMBlog_l processedMblogList, TopRelatedConceptstopRelatedConcepts): ScreenName_l

MBlogEntry_s relevantEntries
for all ProcessedMBlog pMblog in processedMblogList do
   relevantEntries ← FindRelevantMBlogEntries(pMblog, topRelatedConcepts, GetQuery(), HashQuery(query))
   MetaCharacteristics meta ← CharacterizeMBlog(mblog, relevantEntries)
   pMblog.metaCharacteristics ← meta
end for

processedMblogList ← ExcludeLowTV(processedMblogList)
MetaCharacteristics avgMeta
avgMeta ← ComputeAvgMeta(processedMblogList)
for all ProcessedMBlog pMblog in processedMblogList do
   pMblog.metaCharacteristics ← NormalizeMeta(pMblog.metaCharacteristics, avgMeta)
   pMblog.score ← ComputeScore(pMblog)
end for
return SortByScore(processedMblogList)
```

Figure 4.7. Ranking Based On Meta Characteristics.

4.4.2. Meta Characteristics Oriented Ranking

Calculation of meta characteristics are given in Section 4.3.3. The function ComputeScore is called in function Rank computes the final score for ranking (Table 4.8). Figure 4.7 shows the whole process beginning from finding relevant entries. The score of a microblogger is calculated using meta characteristics of both self-posts and posts reshared by him/her.

5. IMPLEMENTATION

The prototype application of the proposed model, named *Micromender*, consists of two main parts: The front-end application to submit a query and a back-end application which fulfills the recommendation process. The front-end application also acts a monitoring tool to investigate the details of the recommendation. A relational database is used as a data persistence layer. When a query is submitted, a new recommendation order regarding the query is created in the database. The back-end part listens the related table in the database for a new request to fire the corresponding recommendation process. The front-end application also serves as a monitoring application which publishes the details of the recommendation orders finished.

5.1. The Front-End

The front-end of Micromender is a web application written in PHP. A new query can be submitted using this application. Figure 5.1 shows the query form. A regular recommendation order is submitted using the first input box. The second one, advanced search, is used to select a semantic web resource from Freebase [36] as a query. Note that this part is remained as an experiment and not part of the current model proposed in the model.

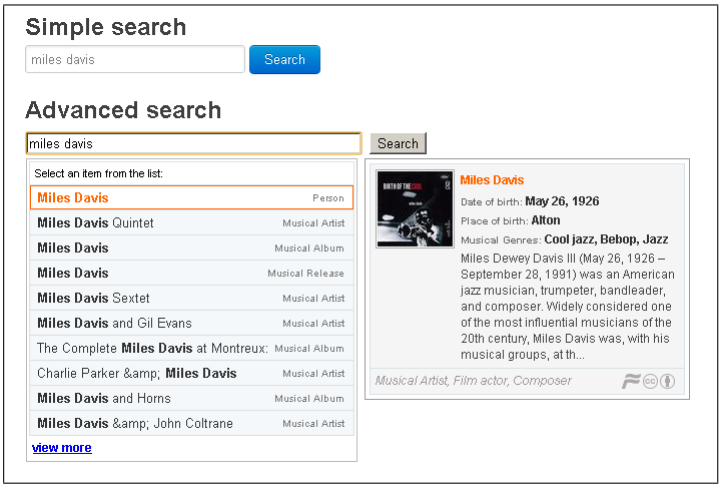


Figure 5.1. Query Screen In Micromender.

Query	Freebase Id	Freebase Type	Start Time	End Time	Status
graphic design			2012-03-31 07:00:02	2012-03-31 07:33:06	FINISHED
summer olympics			2012-03-31 06:26:44	2012-03-31 06:52:44	FINISHED
Java	/m/07sbkfb	Programming Language	2012-03-31 06:19:25	2012-03-31 06:25:37	FINISHED
global financial crisis			2012-03-31 05:46:12	2012-03-31 06:12:01	FINISHED
honor killings			2012-03-31 05:15:48	2012-03-31 05:39:58	FINISHED

Figure 5.2. Queries Submitted.

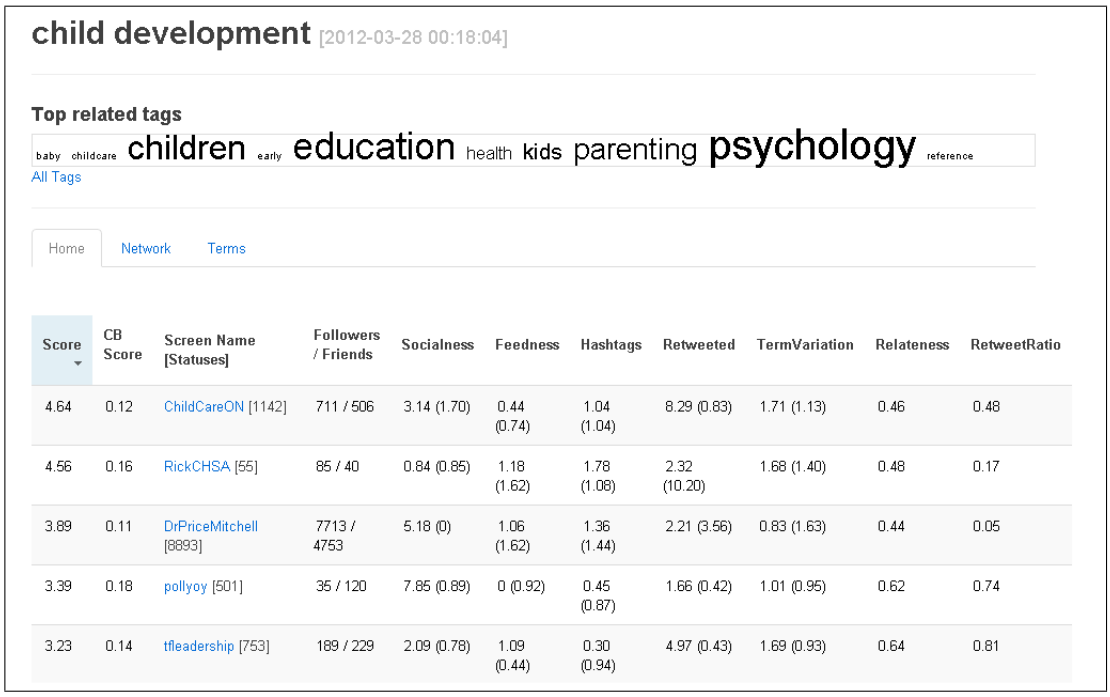


Figure 5.3. Main Page For A Recommendation.

Recommendation orders created are listed to monitor their statuses. Figure 5.2 shows a sample screen shot from the related page. If a query is submitted using the advanced search ID and type of the Freebase resource is also given.

Figure 5.3 shows a sample page of a finished recommendation order submitted with the query "child development". According to the page *TopRelatedConcepts* defined in Section 4.1 are *baby*, *childcare*, *children*, *early*, *education*, *health*, *kids*, *parenting*, *psychology*, and *reference* with their weights.

There are 3 tabs in this screen. In the first tab, microbloggers are listed according to their tanking score. Their content-based score, number of friends, followers, and statuses are also given alongside with values for characterizations such as socialness,

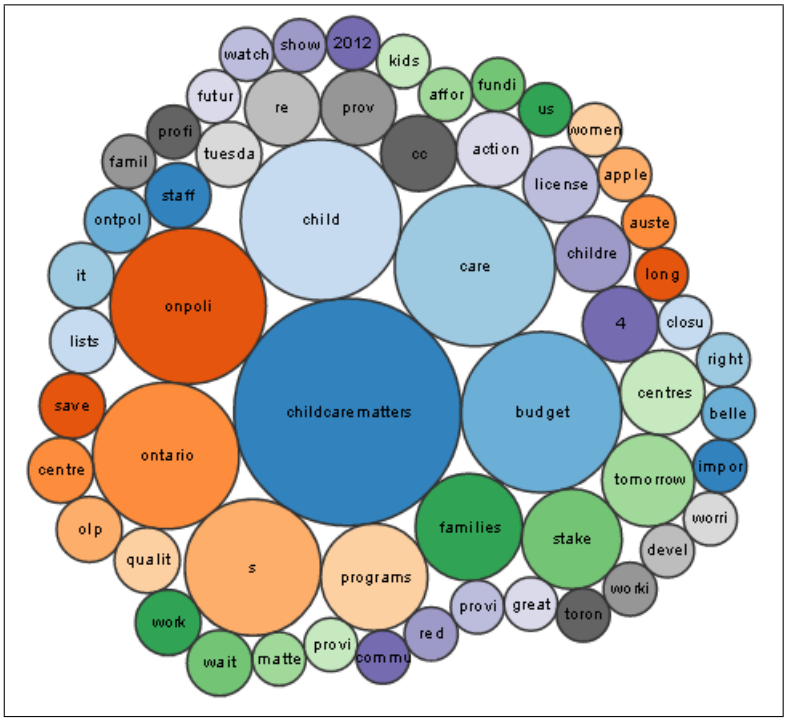


Figure 5.4. Contributions Of A Twitterer.

hashtag usages, feedness, term variation, etc. When the username of a twitterer is clicked, the user page is opened. User page includes basic information about the user such as name, description (bio), location, number of friends, followers, and statuses. Tweets of the user fetched during the recommendation process is also listed with their retweet counts.

Contribution of the user is also presented with a bubble chart. Figure 5.4 shows a sample chart. Each circle represents a distinct word proportional to the number of occurrences. Note that colors are randomly chosen during the generation of the chart.

Another informative chart provided is a pie chart which shows the users being communicated. They includes the mentioned users and retweeted users. Figure 5.5 shows a sample chart.

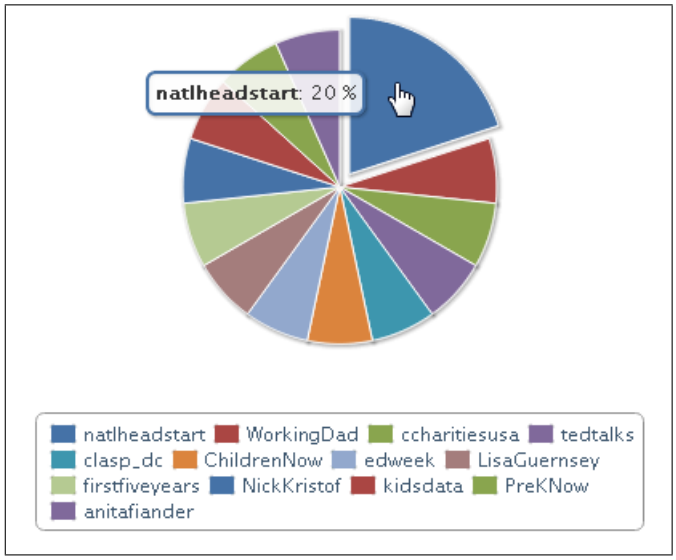


Figure 5.5. Mentioned And Retweeted Users.

5.1.1. Main Technologies

This part uses a number of technologies and libraries. Notable ones are listed below.

- The front-end of the application is written in PHP language[25] . It provides a productive environment especially for Web development.
- JQuery[26] is used to simplify JavaScript development. It allows HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.
- Bootstrap[27] , a set of CSS and JavaScript files from Twitter designed to kick-start development of Web applications. It includes base CSS and HTML for typography, forms, buttons, tables, grids, navigation, etc.
- Highcharts[28] , D3[29] and Raphael[30] are the JavaScript libraries used for the data visualization in front-end.
- Lastly, the front-end runs on a Apache Web server.

[25] *PHP*, 2012, `http://www.php.net`, accessed at March 2012.
[26] *JQuery*, 2012, `http://jquery.com`, accessed at March 2012.
[27] *Twitter Bootstrap*, 2012, `http://twitter.github.com/bootstrap`, accessed at March 2012.
[28] *Highcharts*, 2012, `http://www.highcharts.com`, accessed at March 2012.
[29] *D3*, 2012, `http://mbostock.github.com/d3`, accessed at March 2012.
[30] *Raphael*, 2012, `http://raphaeljs.com`, accessed at March 2012.

5.2. Back-end

Micromender application is written in Java programming language. It uses a MySQL database to persist the data collected from Twitter. Any piece of information used during analysis and ranking stages are also inserted into the same database.

5.2.1. Database Schema

Micromender application persists its data in MySQL database. The database schema created for this application is also called Micromender and it contains the following tables.

- *t_raw_tweet:* This table keeps the raw data, which is in JSON format, of the tweets fetched by Micromender application. A sample raw data can be seen by calling the related REST API of Twitter[31] .
  (i) *id:* This is the id of the tweet given by Twitter.
  (ii) *screen_name:* Twitter user who composed the tweet.
  (iii) *query_date:* Micromender query date for the tweet.
  (iv) *raw_json:* Tweet data in JSON format.
- *t_raw_user:* This table keeps the raw data, which is in JSON format, of the twitter users fetched by Micromender application. A sample raw data can be seen by calling the related REST API of Twitter[32] .
  (i) *id:* This is the id of the tweet given by Twitter.
  (ii) *screen_name:* Twitter user.
  (iii) *query_date:* Micromender query date for the tweet.
  (iv) *raw_json:* Twitter user data in JSON format.
- *t_search:* This table keeps a recommendation order from the user. The lifecycle of a recommendation is managed with the value of *status* column. Average network properties and meta characteristics calculated is also reflected to this table.

[31] *A Sample Raw Tweet Data*, 2012, `http://api.twitter.com/1/statuses/show.json?id=174870369474318338&include_entities=true`, accessed at March 2012.

[32] *A Sample Raw User Data*, 2012, `https://api.twitter.com/1/users/show.json?screen_name=burakcelebi&include_entities=true`, accessed at March 2012.

(i) *id:* The recommendation ID.

(ii) *start_time:* Start time of recommendation.

(iii) *query:* User query.

(iv) *semantic_search_id:* Semantic search ID.

(v) *finish_time:* Finish time of recommendation.

(vi) *avg_in_degree:* Average in-degree value.

(vii) *avg_out_degree:* Average out-degree value.

(viii) *avg_betweenness:* Average betweenness value.

(ix) *avg_closeness:* Average closeness value.

(x) *socialness_own_avg:* Average *socialness* of users' original tweets.

(xi) *socialness_rts_avg:* Average *socialness* of retweets by users'.

(xii) *feedness_own_avg:* Average *feedness* of retweets by users'.

(xiii) *feedness_rts_avg:* Average *feedness* of users' original tweets.

(xiv) *hashtags_own_avg:* Average *hashtagUsage* of users' original tweets.

(xv) *hashtags_rts_avg:* Average *hashtagUsage* of retweets by users'.

(xvi) *rating_own_avg:* Average *retweet count* of users' original tweets.

(xvii) *rating_rts_avg:* Average *retweet count* of retweets by users'.

(xviii) *term_variation_own_avg:* Average *term variation* of users' original tweets.

(xix) *term_variation_rts_avg:* Average *term variation* of retweets by users'.

(xx) *status:* Current status of the recommendation order. Possible values are 0 (invalid), 1 (not started), 2 (processing), and 3 (finished). They can also be found with *select * from t_ref where id = "t_search__status"*.

- *t_search_extended:* This table is used to store related keywords which are introduced in Section 4.1 on page 27.

  (i) *search_id:* Recommendation ID.

  (ii) *tag:* Related keyword.

  (iii) *weight:* Weight of the keyword.

  (iv) *type:* Type of the keyword. Possible values are 1 (top keyword), 2 (keyword).

- *t_search_result:* This table keeps any tweet reached for a recommendation order. Results of the Twitter searches for selecting potential microbloggers, which is mentioned in Section 4.2 on page 30, are recorded to this table. Also, microblogger

tweets to be analysed are kept here according to the recommendation order ID.

(i) *search_id:* Recommendation ID.

(ii) *tweet_id:* Tweet ID.

(iii) *screen_name:* Microblogger user name of the tweet owner.

(iv) *type:* Type of the search result. 1 is used for Twitter search results. 2 and more are used for user microblogger tweets according to iteration. 2 is used for the initial iteration, 3 for the second, and so on.

- *t_token_hashtag:* This table keeps any hashtag found in microblogger tweets. A unique *id* is assigned to each hashtag.

  (i) *id:* Unique hashtag ID.

  (ii) *hashtag:* Hashtag itself.

- *t_token_url:* This table keeps any URL found in microblogger tweets. A unique *id* is assigned to each URL.

  (i) *id:* Unique URL ID.

  (ii) *url:* URL itself.

- *t_token_word:* This table keeps any word (except stopwords) found in microblogger tweets. A unique *id* is assigned to each word.

  (i) *id:* Unique word ID.

  (ii) *word:* Word itself.

- *t_tweet:* This table holds information for each microblogger tweet. They are all supplied by Twitter API[33] and fetched via Twitter4J library except columns *query_date* and *is_related*. For example, to fetch last 50 tweets of user *burakcelebi*:

```
Twitter twitter = TwitterFactory().getInstance();
Paging paging = new Paging(1, 50);
List<Status> result = twitter.getUserTimeline("burakcelebi", paging);
```

  (i) *id:* Tweet ID.

  (ii) *user_id:* User ID of the tweet owner.

  (iii) *screen_name:* Screen name of the tweet owner.

  (iv) *tweet:* Text of the tweet.

[33] *User Timeline API of Twitter*, 2012, `https://dev.twitter.com/docs/api/1/get/statuses/user_timeline`, accessed at March 2012.

(v) *date:* Creation date of the tweet.

(vi) *query_date:* Query date of the tweet.

(vii) *is_retweet:* 1 if it is a retweet, otherwise 0.

(viii) *retweeted_id:* Original tweet ID if it is a retweet.

(ix) *retweeted_screen_name:* Screen name of the tweet owner if it is a retweet.

(x) *reply_id:* ID of the replied tweet if it is a reply.

(xi) *reply_screen_name:* Screen name of the replied tweet's owner if it is a reply.

(xii) *retweet_count:* Retweet count of the tweet, i.e. number of times it is retweeted.

(xiii) *latitude:* Latitude information if it is supplied by the user when tweeting.

(xiv) *longitude:* Longitude information if it is supplied by the user when tweeting.

(xv) *is_related:* 0 for unrelated tweets and 1 for related tweets. They can also be found with *select * from t_ref where id = "t_tweet__is_related"*

- *t_tweet_hashtag:* This table is used to make a relation between hashtags found in a tweet, if any.

  (i) *tweet_id:* Tweet ID.

  (ii) *hashtag_id:* Hashtag ID from t_token_hashtag.

- *t_tweet_mention:* This table is used to state references to other microbloggers found in a tweet, if any.

  (i) *tweet_id:* Tweet ID.

  (ii) *mention:* Screen name of the referenced microblogger.

- *t_tweet_url:* This table is used to make a relation between URLs found in a tweet, if any.

  (i) *tweet_id:* Tweet ID.

  (ii) *url_id:* URL ID from t_token_url.

- *t_tweet_word:* This table is used to make a relation between words found in a tweet, if any.

  (i) *tweet_id:* Tweet ID.

  (ii) *word_id:* Word ID from t_token_word.

Table 5.1. Database Tables.

Table	**Description**
t_search	user query, network properties
t_search_extended	related concepts with weights
t_search_result	search - tweet relation
t_tweet	tweet info and statistics
t_user	twitterer info, characteristics, network properties
t_token_hashtag	found hashtags with IDs
t_token_url	found URLs with IDs
t_token_word	found words with IDs
t_tweet_hashtag	tweet - hashtags relation
t_tweet_mention	tweet - mentions relation
t_tweet_url	tweet - URLs relation
t_tweet_word	tweet - words relation
t_raw_tweet	raw tweet data in JSON format
t_raw_user	raw user data in JSON format

5.2.2. Application Layer

There are many libraries used for different aims in Micromender application. Here is the full list of libraries required for Micromender.

- Twitter4J is an unofficial Java library for the Twitter API. Any Java application using Twitter4J can easily integrate with Twitter. Twitter4J provides a clean object-oriented model which is very helpful for researchers and application developers.
- Spring Framework is an application framework and Inversion of Control container for the Java platform.
- JSON-simple is a library to deal data in JSON format.
- Jena is a semantic web framework for Java applications.
- Jung provides classes for the modeling, analysis, and visualization of data that can be represented as a graph or network.
- Apache Commons Math contains mathematics and statistics components address-

Table 5.2. Java Classes In The Micromender Application.

Package	**Classes**
indexer	UserVectors, IndexerUtil, IdfIndexer, LsiIndexer, TfIndexer
labeler	Labeler, HashTagLabeler, MentionLabeler, NonAsciiLabeler, StopwordLabeler, UrlLabeler, WordLabeler, MultiLabeler
network	Edge, NetworkUtils
search	Conf, Dao, DaoImpl, Search, SearchService, SearchServiceImpl, UserThread
semantic	DbPediaClient, DbpediaResource, DeliciousClient, FlickrClient, FreeBaseClient, FreeBaseTopic, KwMapKeywordFinder, Query, RelatedKeywordsFinder, RelatedTagsFinder, SemanticDao, SemanticDaoImpl, SemanticService, Tag, TagResourceQueryResult, WikiMetaExtractor, YoutubeClient
similarity	AbstractSimilarity, CosineSimilarity, JaccardSimilarity, Searcher, Searcher, SearchResult
stemmer	PorterStemmer
tokenizer	Label, Token, Tokenizer
twitter	MyTweet, MyUser, TwitterClient
util	AppContextUtil, JdbcUtil, JsonUtil, LogUtil, MathUtil, MicromenderConfigurator, TimeUtil, Util

ing the most common problems.

- Apache Commons Collections provides data structures that accelerate development of most significant Java applications.
- MySQL Connector is a standards-based drivers for JDBC to build database driven applications.

Micromender consists of several Java packages and classes. They are coded in object-oriented fashion. A special effort is made to apply practical design patterns where possible and take care of code reusability. The Java classes written inside the packages of Micromender is given in Table 5.2.

6. RESULTS AND THEIR EVALUATION

In this chapter, we evaluate our model by using *Micromender* application. We have asked 41 people to give us at least 5 queries (by e-mail) in any subject they want. After running Micromender application, we asked them to rate how relevant they find each microblogger.

Evaluation application lists queries of users and shows corresponding Twitter accounts for evaluation. They can rate one of these choices:

- Satisfied
- Partially satisfied
- Not satisfied
- Not Applicable

Remember that some constants are defined in Table 4.9 when introducing the proposed model. Values given for those constants during the evaluation are listed in Table 6.1. These values and the formulation inside the function *ComputeScore* are set according to our subjective observations. The final score is calculated with meta-characteristics for two sets of relevant posts (mc_{mb} and mc_o). This means that there exists $2 \times 5 = 10$ meta characteristic scores. In this evaluation, the formula of the final relevance score is as follows:

$$\begin{aligned} score_{mb} = 0.27 \times (socialness_{mb} + feedness_{mb} + termVar_{mb}) \\ +0.13 \times (socialness_o + feedness_o + termVar_o) \\ +0.05 \times hashtags_{mb} + 0.03 \times hashtags_o \\ +0.30 \times retweeted_{mb} + 0.22 \times retweeted_o \end{aligned} \tag{6.1}$$

Table 6.1. Model Constant Values Used For The Evaluation.

Constant	**Value**
MAX_SOCIAL_ITEMS	40
MOST_FREQ_CONCEPTS	10
RECENT_MBLOGGERS	10
RECENT_MBLOG_TWEETS	50
TOP_MBLOGGERS	5
NUM_OF_ITERATIONS	5
MIN_CONTENT_BASED_SCORE	0.1
MIN_TERM_VARIATION	3
RETWEET_RATIO_THRESHOLD_TV	0.7

The weights of the characteristics used in the formula were determined by personal experimentation and observations. The general idea is treat the posts created by the and those that are retweeted differently. In this case, the original posts are considered more important than retweeted posts, although retweeting good tweets would be considered highly valuable. For both sets of posts the meta characteristics are computed. However, they have different weights reflecting the value assigned to them.

We would have liked to further investigate the impact of different values for the parameters, but due to time constraints did not permit such an undertaking. First of all, the execution time for a regular user query takes too much time ($\sim$ 30 minutes) to complete. Moreover, we experienced significant difficulties in convincing a user to participate and complete a user test. Numerous people did not respond to our call for participation. A number of volunteers whose recommendations were presented to them, did not evaluate the recommended users even for a single query.

There were a total of 233 queries, for which 2346 microbloggers were evaluated by 41 test users. The whole process of evaluation took 1 month to complete.

According to the query results, the evaluation application may list not only recommended microbloggers but also microbloggers which does not conform to the Micromender recommendation criteria. Therefore, we also had the opportunity to see

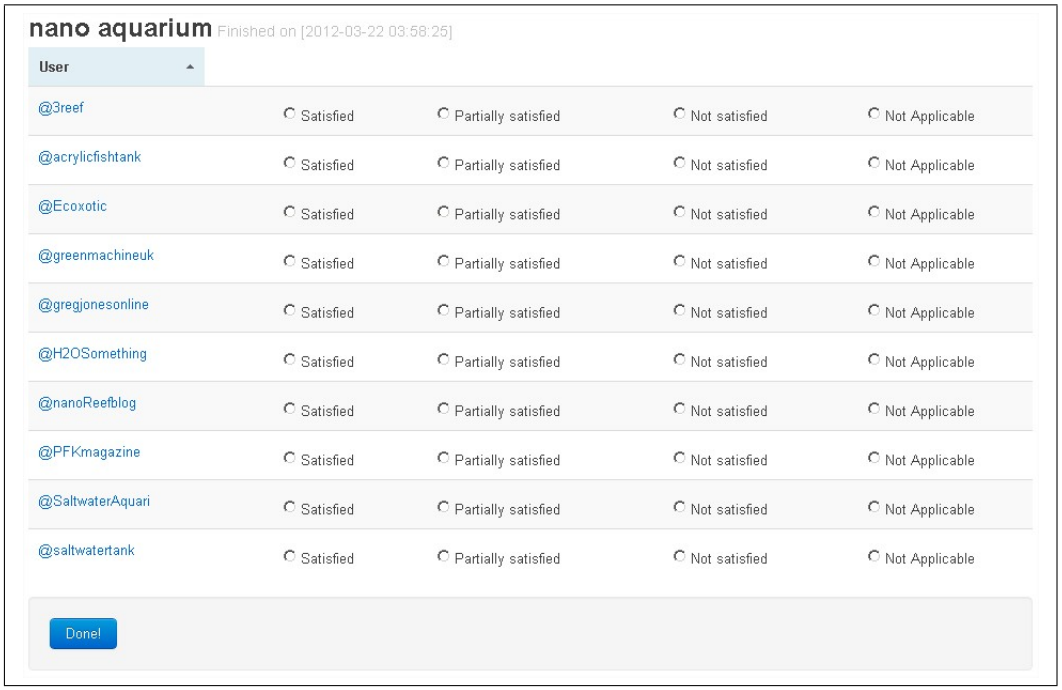


Figure 6.1. Evaluation Screen For The Query “nano aquarium”.

how much test users agree on our criteria for a recommendation.

6.1. Test User Profiles

In this section, we want to give some statistical information about test users. Users are categorized according to their ages, gender, education level, working status, and profession.

Although we did not analyse the results according to those categories, we think that it is valuable to study style of writing queries, diversity of query categories and recommendation results based on user profiles.

In Table 6.2, users are grouped according to their age intervals. The youngest test user is 22 and the oldest is 55 years old. Most of users are between 20 and 30 years old as you can see the table.

Most of the testers are male. Table 6.3 shows the number of users according to their gender.

Table 6.2. Test Users Ages.

Age Interval	**Count**	%
20-30	24	58.5
30-40	14	34.3
40-50	2	4.8
50-60	1	2.4

Table 6.3. Test Users Genders.

Gender	**Count**	%
Male	32	78
Female	9	22

We also categorized test users according to their education levels. Table 6.4 shows highest level of education completed. Note that test users completed high school level are actually senior university students.

Table 6.5 summarizes the occupation statuses of test users. 53.7% of them are student and others are actively working professionals.

People participated our evaluation is from 12 different professions. 8 out of 41 have a background from social sciences. Professions of users are summarized in Table 6.6.

6.2. Test User Queries

Although we wanted at least 5 queries from the test users, 3 of them could not complete all the queries. Anyway their evaluations are also included in the results. All queries are manually categorized for a better understanding of the results. In Table 6.7, distribution of query categories are given. "Miscellaneous" includes categories

Table 6.4. Test Users' Education Levels.

Education Level	**Count**	%
High School	3	7.2
Bachelor	19	46.4
Master	18	44
PhD	1	2.4

Table 6.5. Occupation Statuses Of Users.

Status	**Count**	%
Professional	19	46.3
Student	22	53.7

Table 6.6. Test Users Professions.

Profession	**Count**	%
Computer Science	18	43.9
Software Development	13	32
Psychology	3	7.2
Miscellaneous	7	16.9

academic, *activism*, *activity*, *art*, *books*, *business*, *category*, *consumer goods*, *culture*, *design*, *environmentalism*, *fashion*, *films*, *finance*, *food*, *game*, *geographical location*, *health*, *human condition*, *jobs*, *literary genre*, *method*, *music*, *news*, *person*, *personal development*, *politics*, *profession*, *software*, *sport*, *sports*, *study*, *technology*, *travel*.

We categorized users according to the number of distinct query categories. We put people whose queries belong to 5 or more categories into the same group. Figure 6.2 shows the distribution. Values on slices show the number of people.

6.3. Test User Evaluation Results

As we stated before, evaluation application also shows microbloggers which are not worth recommending according to Micromender. The algorithm may find a microblogger not worth recommending because of two reasons:

(i) **Content:** Content Based score of the microblogger defined in Section 4.4 may

Table 6.7. Broad Categories.

Broad Category	**Count**	%
academic	62	26.6
technology	24	10.3
health	18	7.7
music	16	6.9
person	15	6.4
miscellaneous	98	42.1

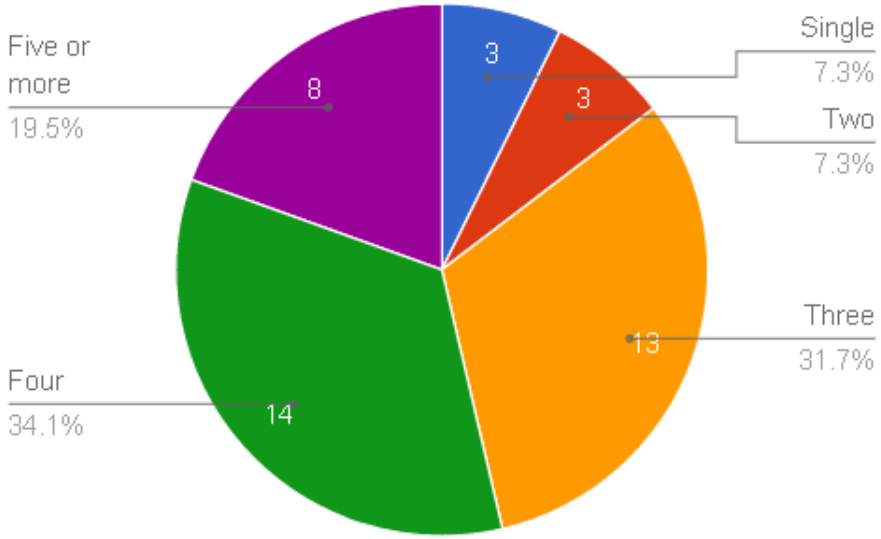


Figure 6.2. Number of Query Categories Per User.

Table 6.8. Micromender and Evaluation Abbreviations.

Abbreviation	**Description**
R	Recommended
¬R	Not Worth Recommendation
$\neg R_C$	Not Worth Recommendation due to Low Content Based Score
$\neg R_{TV}$	Not Worth Recommendation due to Low Term Variation
S	Satisfied
PS	Partially Satisfied
NS	Not Satisfied
NA	Not Applicable

not be high enough for the second phase of ranking, *ranking based on meta-characteristics*.

(ii) **Term Variation:** We use the term variation as an indicator for the content quality. If the threshold defined with MIN_TERM_VARIATION, we conclude that the microblogger is incompetent or maybe a spammer.

Table 6.8 shows abbreviations used in next tables regarding evaluation results. The first part of the table is used for possible recommendation status of a microblogger. The second part shows possible reactions of users against a microblogger found for their queries.

Table 6.9. Summary Of User Evaluations.

	Microbloggers	**R**		**¬R**	
Evaluation	#	#	%	#	%
Satisfied	1000	847	85	153	15
Partially Satisfied	528	393	74	135	26
Not Satisfied	739	348	47	391	53
Not Applicable	79	38	48	41	52
	2346	1626	69	720	31

Table 6.10. Satisfaction of Micromender vs User.

	S	**PS**	**NS**	**NA**
R	52	25	21	2
¬R	21	19	54	6
$\neg R_C$	18	19	59	4
$\neg R_{TV}$	35	19	39	7

Table 6.9 shows the distribution of number of microbloggers recommended and not worth recommendation among users' evaluation choices, *satisfied*, *partially satisfied*, *not satisfied*, and *n/a*. Nearly 85% of *satisfied* microbloggers and 75% *partially satisfied* microbloggers were recommended by Micromender. Nearly 55% of *not satisfied* microbloggers were actually found *not worth recommending* by Micromender.

Table 6.10 presents another view for the evaluation results. It shows the distribution of choices for evaluation among microbloggers recommended and not worth recommendation. 52% of recommended microbloggers are found *satisfied* by the test users. 54% of microbloggers not worth recommendation are found *not satisfied*.

We have also analysed reactions of test users for recommended microbloggers[34] . Queries are aggregated based on their broad categories.

6.3.1. Specific Queries

During the evaluation, we saw that some user queries were too specific to get a recommendation of a microblogger. Table 6.11 shows the queries qualified as specific with their broad categories and expansion score they get. Notice that none of the result

[34] *Broad Category Results*, 2012, `http://bit.ly/M3Twu3`, accessed at March 2012.

Table 6.11. Specific Queries.

Query	**Broad Category**	**Expansion Score**
wheel of time	books	partially satisfied
toyota gt86	consumer goods	partially satisfied
hunger games	films	partially satisfied
spartacus	films	partially satisfied
starwars	films	not satisfied
mission to mars	films	not satisfied
pes game	game	partially satisfied
high stakes poker	game	partially satisfied
the man with the horn	music	not satisfied
bitches brew	music	not satisfied
guyamas sonora	music	n/a

Table 6.12. Summary Of User Evaluations For Specific Queries.

	Total	**R**		**¬R**	
Evaluation		#	%	#	%
Satisfied	27	24	0,89	3	0,11
Partially Satisfied	26	21	0,81	5	0,19
Not Satisfied	50	25	0,5	25	0,5
Not Applicable	3	2	0,67	1	0,33
Total	106	72	0,68	34	0,32

of their expansion is found satisfied. Table 6.12 and 6.13, gives the summary of the evaluation results for these queries.

6.3.2. Iteration Impact

Remember that the algorithm expands the set of potential microbloggers as defined in Section 4.3.2. Number of iteration was set to 5 in Micromender. In table 6.14, distribution of iteration number for microbloggers *satisfied* by the users is listed. The first iteration consists of microbloggers reached with function *FindMBloggers* defined in Section 4.2 and others with function *GetMostValuableMicrobloggers*. As you can see

Table 6.13. Satisfaction of Micromender vs User For Specific Queries.

	S	**PS**	**NS**	**NA**
R	0,33	0,29	0,35	0,03
¬R	0,09	0,15	0,74	0,02
$\neg R_C$	0,07	0,1	0,79	0,03
$\neg R_{TV}$	0,2	0,4	0,4	0

Table 6.14. Impact Of Iteration.

Iteration	**Count**
1	658
2	109
3	69
4	49
5	45
6	53

Table 6.15. Impact Of Query Expansion.

Query Expansion Score	**S**	**PS**	**NS**	**NA**
satisfied	58.8	22.0	16.6	2.7
partially satisfied	37.7	25.9	33.1	3.3
not satisfied	16.6	19.2	58.9	5.3
n/a	14.5	7.9	73.7	3.9

0.34 of microbloggers are reached using this expansion strategy.

6.3.3. Impact of Query Expansion

The motivation for the *query expansion* was to reach more content related microbloggers even do not have microblog entries containing the query terms.

We have manually evaluated the *TopRelatedConcepts* found for each user query by Micromender application. We choosed *satisfied*, *partially satisfied*, *not satisfied*, or *n/a* to rate each *TopRelatedConcepts* for user queries. When no query expansion was made, *n/a* is chosen.

We want to see the relation between query expansion quality and satisfaction of test users for the microbloggers listed in the evaluation application. Results are given in Table 6.15.

It is nice to see nearly 60% of microbloggers found with queries having good query expansion are evaluated as *satisfied* by test users. Again nearly 60% of microbloggers found with queries having not good query expansion are evaluated as *not satisfied*. When no query expansion can be done, almost 75% of microbloggers are evaluated as

Table 6.16. Queries With No Query Expansion.

Query	**Broad Category**
do it yourself	activity
net art	art
gold price predict	finance
french songs for kids	music
guyamas sonora	music
jon wayne and the pain	music
personal development	personal development
black power salute	politics

not satisfied. These queries are listed in Table 6.16. These results show that query expansion is an invaluable part of our model. Any contribution to have better related keywords directly effects the satisfaction of the recommendation results.

When we examine the evaluation results for queries having *partially good* query expansion, we see that values for *satisfied*, *partially satisfied*, and *not satisfied* evaluations are very near to each other. We think that this is another indicator for the importance of query expansion.

6.3.3.1. Using Different Tag Resources. During the evaluations we saw that delicious.com, the social web application we used for *query expansion*, may not be suitable for some type of queries. For example, we found the results for queries related to entertainment inadequate. An alternative approach would be using different strategies and resources for different types of queries. Youtube[35] , the popular video-sharing website, also supports tagging likewise Delicious [2].

For example, query expansion with Delicious produced no output for the query "jon wayne and the pain". However if Youtube were used the result would be as in Table 6.17.

[35] *YouTube*, 2012, `http://www.youtube.com`, accessed at March 2012.

Table 6.17. Query Expansion For "jon wayne and the pain" Using Youtube.

Youtube Tag	**Weight**
reggae	11
music	9
jam's space	6
jwp	6
ska	6
sublime	6
minneapolis	5
10klf	5

6.3.4. Content

Writing too much microblog entries about something does not always means that the microblogger is an interesting one for people seeking microblogger recommendation about the subject. For example, in query "world history" the application found 3 students (beza_seyoulater, NelcyIsMyName, sanderskaylee) who writes about their world history exams. Only one of them was found worth recommending by Micromender where 3 of them was evaluate as *not satisfied* by the user.

6.3.4.1. Disambiguation. We saw that a disambiguation method should be utilized in such a recommender system. Some results of query expansions and recommended microbloggers clearly supports the requirement.

The query "wadl" was submitted by one of our test users intending the "Web Application Description Language"[36] . One of microblogger reached by Micromender was the official account of Wadl TV (WADLTV38), a broadcast television station in the Midwestern United States[37] . This account was not found worth recommending by Micromender due to its low *term variation* value which was 2.1. However, if we had chance to know the user intention and employ a disambiguation method, this account may be elected beforehand since its content is content is irrelevant.

[36] *W3C Member Submission for Web Application Description Language*, 2012, `http://www.w3.org/Submission/wadl/`, accessed at March 2012.

[37] *WADL TV*, 2012, `http://www.wadldetroit.com`, accessed at March 2012.

Some results of query expansion also show the need for disambiguation.

- *metal:* a genre of rock music, metallicity of an object.
- *wheel of time:* a religious concept, a film, a book
- *birdy:* A film, a singer. The keywords found in query expansion includes *music*, *nicolas*, and *cage*. Micromender reached a microblogger, staceypatton, having such an interesting retweet: *"Singer Birdy to marry Nic Cage's son Weston - will legally be Birdy Cage. #FollowMeTMZ"*
- *software specification:* Query expansion found keyword *design* which is used in various domains. Recommendations of Micromender includes microbloggers writing about user interface design of software components.
- *design:* As a query *"design"* should be disambiguated to understand the user intention since it may refer to *fashion design*, *game design*, *graphic design*, *interaction design*, *interior design*, *product design*, *web design*, or *service design*, etc.
- *isometry:* This query was asked by a mathematics engineer who works as an e-learning professional. He is also interested in ActionScript which is a scripting language for Flash platforms. *"Isometry"* is both a mathematical concept and a type of library ing Flash programming. This example shows not only need for disambiguation, but also profiling user's interests to understand the intent of a user may not be sufficient.

6.3.5. Meta Characteristics

6.3.5.1. Selecting Related Microblog Entries. Identifying related microblog entries is an important step since final score is calculated using microblog entries found related. In Micromender, we used words and hashtags to choose related microblog entries. However, URLs in a microblog entry may also keep valuable information regarding the content. Assume that a user wants to get a recommendation using the query *"steve jobs"*. The tweet in Figure 6.3 has a link to the famous speech of Steve Jobs at Stanford university's 114^{th} commencement in 2005. It is obvious that the tweet is related to Steve Jobs. However just by using words in it, it cannot be figured out. But, the shortened URL in

Stay hungry. Stay foolish. http://bit.ly/pRHaXR

Figure 6.3. A Sample Tweet Hiding Information.

Table 6.18. Microbloggers Having Low Term Variation.

Query	**Twitterer: Sample Tweet**	**TV**
fitness	Quotes24T: @alliesawka followwww @ExerciseExpert for tipssss on #exercise #gym #motivation #diet He inspires me when it comes to exercise	0.14
history	spiritmound: Available on Kindle: Aliens in American History http://t.co/437zHn6d 99 cent Kindle non-fiction	2.35
graphic design	ExposedINK: Wanna #win $500+? Enter @ExposedINK tshirt #design #contest http://t.co/bzsjsP3W	1.32

the post have important clues because the real URL[38] already includes *"steve jobs"*. When a query is performed with *Twitter Search API*, it also finds such kind of tweets including query words in URLs. Therefore, URLs should also be used when identifying related tweets.

Remember that a tweet is decided as *related* when it contains one of the words in the user query. We observed that while this method is useful for many cases, it may also lead to bad recommendations. For example, the twitterer *3days_in_london* has got the highest score for the query *"london olympics 2012"* since the microblogger has a tweet including the word *London* retweeted 1445 times. Although she is not tweeting about *london olympics 2012*, it is recommended by Micromender. This case is also an example using retweet counts directly in scoring discussed in Section 6.5.1.

6.3.5.2. Term Variation. We use term variation value as an indicator for the quality of the tweets. It is also a convenient measure for electing spammers. They frequently write microblog entries with the same or similar set of words. Table 6.18 lists some of such microbloggers. TV is used for *term variation* as an acronym in the third column title.

[38] `http://busyteacher.org/7002-steve-jobs-connecting-dots-video-activity.html`, accessed at March 2012.

6.3.6. Selecting Potential Microbloggers

Micromender performs Twitter searches using related keywords alongside the query itself as written in Section 4.2. We observed that valuable hashtags and URLs are can be found in related tweets. We think that they can also be used to reach other potential microbloggers.

Another enhancement may be done using mentioned and retweeted microbloggers by the top microbloggers after the final score. This is exactly what is done after content-based ranking discussed in Section 4.3.2.

6.4. Sentiment Analysis

Sentiment Analysis is an application of natural language processing which can identify subjective information supplied by the author. This study does not include any sentiment analysis. Therefore we do not know the attitude of a microblogger with respect to the user query.

In a microblogging environment, people may also want to follow microbloggers against their opinion on particular subject. For example, the tester of *"intelligent design"* also *satisfied* for mirobloggers who is against *intelligent design*. It may be valid for queries on politicians, political opinions, or celebrities.

6.5. Ranking Score

6.5.1. Retweet Counts

Remember that the retweet counts of related tweets is used in the calculation of the final score. Although a normalization is made, we saw that using retweet count directly may supply high scores to microbloggers who have a single tweet retweeted by a big number of microbloggers and produce lower scores for microbloggers who has high content-based scores but tweets with low reputation.

6.6. Performance

In this thesis, our aim was not to build an end-user ready recommendation application. Rather handling performance issues, we concentrated on the model we developed to see whether our strategies work well or not. The priority of topics related to application performance would increase after developing solid insight on the recommendation approach.

A serious difficulty on dealing with real-time user data is the limitations of the provided microblogging API. We did not want to make asynchronous calls for reading and processing independent microbloggers data since Twitter would ban our application due to a high overload.

7. DISCUSSION AND CONCLUSION

In this thesis, a content-based microblogger recommendation model is designed. An approach was proposed to recommend a list of microbloggers regarding a user query. Twitter [3] was selected as the microblogging system for the implementation and a live user test was performed to evaluate the model. Results of recommendations and strategies we utilized seem promising and encouraging for improvements.

7.1. Discussion

The evaluation application developed for this study helped us a lot when analysing the results. A considerable time has been spent for developing it, but it all paid off in the long term. Its modular structure allowed us analysing results and adding new features easily. We also observed that such a user friendly application encourages people to participate the evaluation.

When we were thinking about the alternatives of evaluation scenarios, an alternative was forcing test users to evaluate fixed set of queries. However during the evaluation, we observed that it was very critical to allow users to evaluate their own queries for a number of reasons. First of all, people are much more willing to participate with their own queries. Secondly, we should be sure that the query should be about something which users are familiar with. Lastly, it was a very valuable experience for us to see users' different way of writing queries. Many improvements could be done to handle such kind of differences.

Numerous test users have expressed their gratitude for the results. They said they discovered many valuable microbloggers to follow during the evaluation. Moreover, some of people who are not Twitter users decided to start microblogging.

User tests was a tiring but also a very important process. We think that if we performed a smaller test in the early stages of the work, final implementation would be

better.

Remember that we utilized a number of properties specific to microblogging environments, named *meta characteristics*, rather than treating the problem of recommendation like a regular information retrieval problem. In our opinion if we just performed content-based algorithm, satisfaction of test users would be higher due to the higher content relevance. However, a microblogging environment is a very different platform than regular documents or web pages. If one seeks a microblogger to *follow* on a particular topic, the satisfaction would also depend on how much the microblogger interacts with others, variety of hashtags used, URLs shared, etc. Therefore, the recommendation of a microblogger needs special treatment. This is the reason of utilizing the *meta characteristics*.

As we state constant values of Micromender for the user test in Table 6.1 on page 49, the data collected for such a recommendation goal is actually very limited. For each term, only recent 10 microbloggers are identified. This leads to at most 230 microbloggers for a user query. And only recent 50 posts of that microbloggers are fetched. It will be possible to analyse much more microblogger data with a a scalable architecture, as offered in Section 7.2.6. Therefore, we believe that it would also improve the user satisfaction for recommendations.

Our prototype implementation lacks disambiguation. During the user tests, we observed that the need for a solid disambiguation module for both query expansion and post processing phases.

7.2. Future Work

During the evaluation, we observed many improvement opportunities for the model proposed.

7.2.1. Query Handling

- Currently, there is no query operators available for the user. A user will be able to give more information about his/her intention with query operators. For example, writing a list of words in quotation marks, excluding particular words, giving information about a geolocation, or even supplying positive/negative attitude in the user query will improve the satisfaction of the user.
- A spell checker should be used to fix spelling errors in writing queries.

7.2.2. Query Expansion

- Instead of using external resources such as Delicious, Youtube or any social site supporting tagging, query expansion can be performed using microblog posts. Frequently used co-occurring words or phrases with the user query might improve the quality of results since microblogs probably have much more fresh data than any other social systems.
- Finding synonyms and acronyms of the query words might help for better a query expansion.
- Proven solutions of third party query expansion libraries should be also tried [37, 38].

7.2.3. Selecting Potential Microbloggers

- Referenced microbloggers in microblog posts reached by microblog search operations are not considered as potential microbloggers for recommendation. Only authors of the posts are analysed. We observed many valuable microbloggers are referenced in these posts. Therefore they also can be utilized in the recommendation process.
- Selections might be performed in a time and post frequency dependent manner.
- Relevant posts are determined to calculate meta characteristics. We observed that these posts may contain valuable hashtags and words which we couldn't discover in query expansion. Hence, these hashtags and words can be used to make new

microblog searches to obtain new microbloggers to be analysed.

7.2.4. Disambiguation

- Semantic web integration to disambiguation both for user query and the content in microblogs.
- Relevance feedback [13] can be taken from the user. The idea of relevance feedback is to involve the user in the retrieval process so as to improve the user satisfaction. For example, initial details of query expansion and sample microbloggers can be shown and feedback of the user can be requested.

7.2.5. Indexing and Similarity Measures

- In Micromender, *Cosine Similarity* is used as a measure of similarity for the microblog data indexed with *TF-IDF*. Different indexing schemes and similarity measure should be studied and tested.

7.2.6. Performance

- Queries performed on the microblog system to fetch the data can be performed using capabilities of parallel computing.
- Alternative architectures for the implementation of the model should be tested to improve the execution time. Moreover, much more real-time data (more microbloggers, more microblog entries) can be fetched and analyzed. For example, The Apache Software Foundation [39] has a number of open-source projects for reliable, scalable and distributed computing [40, 41]. A scalable machine learning library can be used to support such a distributed architecture [42]. Also, using a capable information retrieval and NLP library [43] and a natural language processing library [44] will be helpful for common text processing tasks.

7.2.7. Ranking and Scoring

- Microblog systems may provide some indicators regarding the reputation of a microblog entry other than the number of reshares (*retweets* in Twitter) of an entry. For example, in Twitter, if a microblogger likes a tweet she can show this by *favoriting* the tweet[39] . The Twitter API also supplies the favorite counts of tweets as well as their retweet counts. Using such extra indicators for reputation in scoring will improve the quality of ranking.
- Different algorithms can be evaluated for content-based scoring.

7.2.8. Spam Handling

- Although we use a blacklist for query expansion and term variation values to prevent spammers, a stable filtering method should be utilized both for query expansion and microblogs.

7.2.9. Adoption

- Adoption for other natural languages.
- Adoption for other social networks such as Facebook and LinkedIn.

7.2.10. Network Properties

In this work we only concentrated on the microblog content not the connections of the microbloggers. Connections and network properties of a microblogger might be analysed and used when recommendation.

[39] *What Are Favorites?*, 2012, `https://support.twitter.com/articles/14214-what-are-favorites`, accessed at March 2012.

APPENDIX A: STOP WORDS LIST

In computing, a stop word list consists of frequently used words in a natural language. The program dealing with natural language processing filters out stop words to concentrate remaining words having semantic potential.

Here is the full list of stop words used for the implementation of the proposed model. Note that the list is extended by a number of words which are commonly used by Twitterers.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, a's, able, about, above, according, accordingly, across, actually, add, after, afterwards, again, against, ago, ain't, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, aren't, around, as, aside, ask, asking, associated, at, available, away, awesome, awfully, b, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, believe, below, beside, besides, best, better, between, beyond, big, both, brief, but, by, c, c'mon, c's, came, can, can't, cannot, cant, cause, causes, certain, certainly, changes, check, clearly, co, com, come, comes, coming, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, couldn't, course, currently, d, definitely, described, despite, details, did, didn't, different, do, does, doesn't, doing, don't, done, down, downwards, due, during, e, each, edu, eg, eight, either, else, elsewhere, end, enough, entirely, especially, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, exactly, example, except, f, far, few, fifth, first, five, followed, following, follows, for, former, formerly, forth, four, from, fun, further, furthermore, g, get, gets, getting, given, gives, go, goes, going,

gone, got, gotten, greetings, guy, guys, h, had, hadn’t, happens,
hardly, has, hasn’t, have, haven’t, having, he, he’s, hello, help,
hence, her, here, here’s, hereafter, hereby, herein, hereupon, hers,
herself, hi, him, himself, his, hither, hopefully, hour, hours, how,
howbeit, however, i, i’d, i’ll, i’m, i’ve, ie, if, ignored, immediate,
in, inasmuch, inc, indeed, indicate, indicated, indicates, inner,
insofar, instead, into, inward, is, isn’t, it, it’d, it’ll, it’s,
its, itself, j, just, k, keep, keeps, kept, know, known, knows,
l, last, lately, later, latter, latterly, least, less, lest, let,
let’s, lie, like, liked, likely, little, live, lol, look, looking,
looks, love, low, ltd, m, mainly, make, many, may, maybe, me, mean,
meanwhile, merely, might, more, moreover, morning, most, mostly, much,
must, my, myself, n, name, namely, nd, near, nearly, necessary, need,
needs, neither, never, nevertheless, new, next, nice, night, nine,
no, nobody, non, none, noone, nor, normally, not, nothing, novel,
now, nowhere, o, obviously, of, off, often, oh, ok, okay, old, on,
once, one, ones, only, onto, or, other, others, otherwise, ought,
our, ours, ourselves, out, outside, over, overall, own, p, particular,
particularly, people, per, perhaps, placed, please, plus, possible,
pre, presumably, probably, provides, put, q, que, quite, qv, r, rather,
rd, re, really, reasonably, regarding, regardless, regards, relatively,
respectively, retweet, right, rt, s, said, same, saw, say, saying,
says, second, secondly, see, seeing, seem, seemed, seeming, seems,
seen, selves, sensible, sent, serious, seriously, seven, several,
shall, she, should, shouldn’t, since, six, so, some, somebody, somehow,
someone, something, sometime, sometimes, somewhat, somewhere, soon,
sorry, specified, specify, specifying, still, sub, such, sup, sure, t,
t’s, take, taken, tell, tends, th, than, thank, thanks, thanx, that,
that’s, thats, the, their, theirs, them, themselves, then, thence,
there, there’s, thereafter, thereby, therefore, therein, theres,
thereupon, these, they, they’d, they’ll, they’re, they’ve, think,
third, this, thorough, thoroughly, those, though, three, through,

throughout, thru, thus, to, together, tomorrow, too, took, toward, towards, tried, tries, truly, try, trying, twice, two, u, un, under, unfortunately, unless, unlikely, until, unto, up, upon, use, used, useful, uses, using, usually, uucp, v, value, various, very, via, viz, vs, w, want, wants, was, wasn't, way, we, we'd, we'll, we're, we've, welcome, well, went, were, weren't, what, what's, whatever, when, whence, whenever, where, where's, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, who's, whoever, whole, whom, whose, why, will, willing, wish, with, within, without, won't, wonder, would, wouldn't, x, y, yes, yet, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves, z, zero

APPENDIX B: SAMPLE RECOMMENDATION RESULTS

B.1. Child Development

Table B.1. Recommendation Details For "child development".

Query	child development
Start Time	2012-03-28 00:18:04
Finish Time	2012-03-28 00:54:08
Microbloggers Analyzed	222
Tweets Fetched	10734

Table B.2. TopRelatedConcepts For "child development".

tag	**weight**
psychology	11
education	10
children	9
parenting	8
kids	6
health	5
baby	3
early	3
childcare	3
reference	3

Table B.3. Avarage MetaCharacteristics For "child development".

$\mathbf{So}_s$	0.24
$\mathbf{So}_r$	0.59
$\mathbf{Fd}_s$	0.76
$\mathbf{Fd}_r$	0.62
$\mathbf{Ht}_s$	0.56
$\mathbf{Ht}_r$	0.70
$\mathbf{Re}_s$	0.30
$\mathbf{Re}_r$	5.34
$\mathbf{Tv}_s$	4.72
$\mathbf{Tv}_r$	6.76

Table B.4. Evaluation Result for “child development”.

SN	**Rec**	**Ev**	**Score**	**CBS**	**So**$_s$	**So**$_r$	**Fd**$_s$	**Fd**$_r$	**Ht**$_s$	**Ht**$_r$	**Re**$_s$	**Re**$_r$	**Tv**$_s$	**Tv**$_r$	**ReR**
ChildCareON	R	S	4.6	0.12	3.1	1.7	0.4	0.7	1.0	1.0	8.3	0.8	1.7	1.1	0.5
DrPriceMitchell	R	S	3.9	0.11	5.2	0.0	1.1	1.6	1.4	1.4	2.2	3.6	0.8	1.6	0.0
pollyoy	R	S	3.4	0.18	7.8	0.9	0.0	0.9	0.4	0.9	1.7	0.4	1.0	1.0	0.7
tfleadership	R	S	3.2	0.14	2.1	0.8	1.1	0.4	0.3	0.9	5.0	0.4	1.7	0.9	0.8
TrentChildCare	R	S	2.9	0.19	0.0	0.9	0.3	1.4	0.0	1.0	5.3	0.9	2.2	0.9	0.8
MCRCHalton	R	S	2.8	0.17	1.9	0.0	0.7	1.6	0.9	0.0	4.0	0.4	1.3	1.5	0.0
Rickackerly	R	S	1.6	0.16	0.7	1.7	1.0	1.6	1.0	2.9	0.3	0.2	1.1	1.3	0.1
RickCHSA	R	PS	4.6	0.16	0.8	0.8	1.2	1.6	1.8	1.1	2.3	10.2	1.7	1.4	0.2
DrMerylAin	R	PS	2.6	0.15	3.5	2.0	1.2	1.3	3.3	2.0	0.4	0.3	1.3	0.7	0.1
ParentNetAssn	R	PS	1.6	0.19	1.9	0.0	1.3	0.0	1.7	0.0	1.7	0.0	0.8	0.0	0.0

B.2. Reconfigurable Computing

Table B.5. Recommendation Details For "reconfigurable computing".

Query	reconfigurable computing
Start Time	2012-03-24 15:53:31
Finish Time	2012-03-24 16:13:10
Microbloggers Analyzed	134
Tweets Fetched	6469

Table B.6. TopRelatedConcepts For "reconfigurable computing".

tag	**weight**
fpga	16
hardware	12
research	4
conferences	3
hpc	3
nanotech	3
chip	2
computer	2
config	2
programming	2

Table B.7. Avarage MetaCharacteristics For "reconfigurable computing".

$\mathbf{So}_s$	0.36
$\mathbf{So}_r$	0.00
$\mathbf{Fd}_s$	0.84
$\mathbf{Fd}_r$	0.93
$\mathbf{Ht}_s$	0.53
$\mathbf{Ht}_r$	1.13
$\mathbf{Re}_s$	0.52
$\mathbf{Re}_r$	4.40
$\mathbf{Tv}_s$	6.81
$\mathbf{Tv}_r$	8.47

Table B.8. Evaluation Result for "reconfigurable computing".

SN	**Rec**	**Ev**	**Score**	**CBS**	**So**$_s$	**So**$_r$	**Fd**$_s$	**Fd**$_r$	**Ht**$_s$	**Ht**$_r$	**Re**$_s$	**Re**$_r$	**Tv**$_s$	**Tv**$_r$	**ReR**
XilinxInc	R	S	2.2	0.11	0.8	-	1.0	1.1	0.1	0.9	2.5	1.4	1.2	1.2	0.2
geschema	R	S	1.5	0.22	1.2	-	0.6	1.1	0.5	0.9	1.2	0.6	1.0	0.7	0.2
jimwu88	R	S	1.5	0.17	0.5	-	1.3	0.9	0.6	1.2	0.6	0.9	1.1	0.9	0.2
SKing49	R	PS	0.7	0.12	0.0	-	1.2	0.0	0.0	0.0	0.0	0.0	1.3	0.0	0.0
DDAndrews	$\neg R_C$	PS	-	0.08	-	-	-	-	-	-	-	-	-	-	0.0
FAETimDuffy	$\neg R_C$	PS	-	0.08	-	-	-	-	-	-	-	-	-	-	0.3
HPC_Guru	R	NS	2.7	0.11	2.5	-	1.1	1.1	3.1	0.9	1.9	1.7	1.0	1.5	0.0
ICL_UTK	$\neg R_C$	NS	-	0.08	-	-	-	-	-	-	-	-	-	-	0.0
biomap	$\neg R_C$	NS	-	0.07	-	-	-	-	-	-	-	-	-	-	0.9
dustinsanbornbe	$\neg R_{TV}$	NS	0.3	0.22	0.0	-	0.6	0.0	0.0	0.0	0.0	0.0	0.4	0.0	0.0

REFERENCES

1. O'Reilly, T., *What is Web 2.0*, 2004, `http://oreilly.com/web2/archive/what-is-web-20.html`, accessed at March 2012.

2. *Delicious*, 2012, `http://delicious.com`, accessed at March 2012.

3. *Twitter*, 2012, `http://www.twitter.com`, accessed at March 2012.

4. *Google+*, 2012, `http://plus.google.com`, accessed at March 2012.

5. *Friendfeed*, 2012, `http://www.friendfeed.com`, accessed at March 2012.

6. *Plurk*, 2012, `http://www.plurk.com`, accessed at March 2012.

7. *StatusNet*, 2012, `http://www.status.net`, accessed at March 2012.

8. *Identi.ca*, 2012, `http://www.identi.ca`, accessed at March 2012.

9. *Facebook*, 2012, `http://www.facebook.com`, accessed at March 2012.

10. *LinkedIn*, 2012, `http://www.linkedin.com`, accessed at March 2012.

11. Peters, I., *Folksonomies: Indexing and Retrieval in Web 2.0*, De Gruyter, Berlin, 2009.

12. Salton, G. and M. J. McGill, "The SMART and SIRE Experimental Retrieval Systems", *Readings in Information Retrieval*, pp. 381–399, Morgan Kaufmann, San Francisco, CA, USA, 1997.

13. Manning, C. D., P. Raghavan and H. Schütze, *An Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA, 2008.

14. Salton, G. and C. Buckley, "Readings in Information Retrieval", chap. Term-weighting Approaches in Automatic Text Retrieval, pp. 323–328, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

15. Ushiama, T. and T. Eguchi, "An Information Recommendation Agent on Microblogging Service", *Proceedings of the 5th KES International Conference on Agent and Multi-agent Systems: Technologies and Applications*, KES-AMSTA'11, pp. 573–582, Springer-Verlag, Berlin, Heidelberg, 2011.

16. Lee, C.-H., C.-H. Wu and T.-F. Chien, "BursT: A Dynamic Term Weighting

Scheme for Mining Microblogging Messages", *Proceedings of the 8th International Conference on Advances in Neural Networks - Volume Part III*, ISNN'11, pp. 548–557, Springer-Verlag, Berlin, Heidelberg, 2011.

17. Waltinger, U., I. Cramer and T. Wandmacher, "From Social Networks to Distributional Properties: A Comparative Study on Computing Semantic Relatedness", N. Taatgen and H. van Rijn (Editors), *Proceedings of the 31th Annual Conference of the Cognitive Science Society*, pp. 3016–3021, Cognitive Science Society, Austin, TX, 2009.

18. Yang, J. and J. Leskovec, "Patterns of Temporal Variation in Online Media", *Proceedings of the Fourth ACM International Conference on Web search and Data Mining*, WSDM '11, pp. 177–186, ACM, New York, NY, USA, 2011.

19. Hannon, J., M. Bennett and B. Smyth, "Recommending Twitter Users to Follow Using Content and Collaborative Filtering Approaches", *Proceedings of The Fourth ACM Conference on Recommender Systems*, RecSys '10, pp. 199–206, ACM, New York, NY, USA, 2010.

20. Chen, J., R. Nairn, L. Nelson, M. Bernstein and E. Chi, "Short and Tweet: Experiments on Recommending Content from Information Streams", *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, CHI '10, pp. 1185–1194, ACM, New York, NY, USA, 2010.

21. Tao, K., F. Abel, Q. Gao and G.-J. Houben, "TUMS: Twitter-Based User Modeling Service", R. Garcia-Castro, D. Fensel and G. Antoniou (Editors), *ESWC Workshops*, Vol. 7117 of *Lecture Notes in Computer Science*, pp. 269–283, Springer, 2011.

22. Abel, F., Q. Gao, G.-J. Houben and K. Tao, "Semantic Enrichment Of Twitter Posts for User Profile Construction on the Social Web", *Proceedings of the 8th Extended Semantic Web Conference on the Semanic Web: Research And Applications - Volume Part II*, ESWC'11, pp. 375–389, Springer-Verlag, Berlin, Heidelberg, 2011.

23. Uysal, I. and W. B. Croft, "User Oriented Tweet Ranking: A Filtering Approach to Microblogs", *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pp. 2261–2264, ACM, New York, NY, USA, 2011.

24. Robertson, S., "Understanding Inverse Document Frequency: On Theoretical Arguments for IDF", *Journal of Documentation*, Vol. 60, p. 2004, 2004.

25. Alhadi, A. C., T. Gottron, J. Kunegis and N. Naveed, "LiveTweet: Microblog Retrieval Based on Interestingess and an Adaptation of the Vector Space Model", *Proceedings of the Text Retrieval Conference (TREC)*, 2011.

26. Massoudi, K., M. Tsagkias, M. de Rijke and W. Weerkamp, "Incorporating Query Expansion and Quality Indicators in Searching Microblog Posts", *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pp. 362–367, Springer-Verlag, Berlin, Heidelberg, 2011.

27. Nagmoti, R., A. Teredesai and M. De Cock, "Ranking Approaches for Microblog Search", *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pp. 153–157, IEEE Computer Society, Washington, DC, USA, 2010.

28. Akman, D. S., *Revealing Microblogger Interests by Analyzing Contributions*, M.S. Thesis, Bogazici University, 2010.

29. Yurtsever, E., *Sweettweet: A Semantic Analysis for Microblogging Environments*, M.S. Thesis, Bogazici University, 2010.

30. Degirmencioglu, E. A., *Exploring Area-Specific Microblogger Social Networks*, M.S. Thesis, Bogazici University, 2010.

31. Aslan, O., *An Analysis of News on Microblogging Systems*, M.S. Thesis, Bogazici University, 2010.

32. *WeFollow*, 2012, `http://www.wefollow.com`, accessed at March 2012.

33. *Digg*, 2012, `http://www.digg.com`, accessed at March 2012.

34. *Klout*, 2012, `http://www.klout.com`, accessed at March 2012.

35. *Twiends*, 2012, `http://www.twiends.com`, accessed at March 2012.

36. *Freebase*, 2012, `http://www.freebase.com`, accessed at March 2012.

37. Rubens, N., *Lucene Query Expansion Module*, 2012, `http://lucene-qe.sourceforge.net`, accessed at March 2012.

38. Agüera, J. R. P., *Query Expansion Module for Lucene Based on BM25 Ranking Function and an Extension for Query Clauses.*, 2012, `http://grasia.fdi.ucm.es/jose/query-expansion`, accessed at March 2012.

39. *The Apache Software Foundation*, 2012, `http://www.apache.org`, accessed at March 2012.

40. *Apache Hadoop*, 2012, `http://hadoop.apache.org`, accessed at March 2012.

41. *Apache Cassandra*, 2012, `http://cassandra.apache.org`, accessed at March 2012.

42. *Apache Mahout*, 2012, `http://mahout.apache.org`, accessed at March 2012.

43. *Apache Lucene*, 2012, `http://lucene.apache.org`, accessed at March 2012.

44. *Apache OpenNLP*, 2012, `http://opennlp.apache.org`, accessed at March 2012.