Information filtering on micro-blogging services

Master Thesis

Author(s):
Güç, Barış

Publication date:
2010

Permanent link:
https://doi.org/10.3929/ethz-a-006180617

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Swiss Federal Institute of Technology Zürich

Master's Thesis

Information Filtering on Micro-blogging Services

by

Barış Güç
baris@student.ethz.ch

Supervisors:

Dr. Maria Grineva
Prof. Dr. Donald Kossmann

August, 2010

Abstract

Micro-blogging is an emerging form of communication and became very popular in recent years. Micro-blogging services allow users to publish updates as short text messages that are broadcasted to the followers of users in real-time. Twitter is currently the most popular micro-blogging service. It is a rich and real-time information source and a good way to discover interesting content or to follow recent developments. However, the service is fairly simple, and rely on the concept of following other users. With the lack of classification or filtering tools, the user receives all messages posted by the users she follows. In most cases, the user receive a noisy stream of updates. In this paper, an information filtering system for Twitter is introduced. The system focuses on one kind of feeds on Twitter: Lists which are a manually selected group of users on Twitter. List feeds tend to be focused on specific topics, however it is still noisy due to irrelevant messages. Therefore, we propose an online filtering system, which extracts the niche topics in a list, filtering out irrelevant messages. To classify messages as relevant or irrelevant, next to text-based features, we utilize the social network of Twitter and different aspects of messages such as the temporal properties and the links included in the text. We evaluate our approach on a labeled dataset of lists and with the help of these novel features, we achieve accuracies between 85% and 95%. Finally, we present the online prototype of the system.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Micro-blogging is an emerging form of communication and it became very popular in recent years. Micro-blogging services allow users to publish short text messages. These messages are broadcasted to followers of the users in real-time. Twitter, Jaiku, Pownce, Yahoo Meme are some examples of micro-blogging services.

In this paper, we will be focusing on Twitter[1] which is the most popular micro-blogging service with more than 100 million users by April 2010[8]. Next to micro-blogging features, Twitter is also a social networking site. Users may subscribe to other author's updates which is known as *following*. The users who follows a user is called *followers*, and the users a user is following is called *friends* in Twitter. This social network model is different from most social networks: the relations between users are not necessarily reciprocal, i.e. a user can follow a user without being followed. Messages posted by all friends of a user is received by the user as one feed. This feed can also be described as an information stream[17].

With the wide usage of Twitter, it quickly became a rich, real-time information source. With its large user base, Twitter can act as a real-time news source. The social network model of Twitter also helps fast diffusion of information[28]. In recent events such as US Airways jetliner crashing into Hudson River[7], the death of Michael Jackson and terrorist attacks in Mumbai[6], Twitter was faster in providing news than mainstream media. Twitter is also a good source for discovering recent and interesting content on the Internet. A good percentage of messages sent on Twitter refer to external content such as news and blogs. Furthermore, with its increasing popularity, influential and important people on a wide variety of topics is using Twitter for communication. Therefore, for any possible topic, Twitter became a medium for informal discussion, exchange of ideas or following recent developments.

While Twitter is a great source of information, retrieval of information on a specific topic is difficult. There are two main reasons for this problem. The first reason is that each user posts messages through one channel. A user can post messages with different intentions, such as conversation with other users, daily chatter, jokes, sharing or referring to information. A user may be following another user because he is posting about a specific topic, however as there is no mechanism to send messages through different channels, he has to receive all messages posted by that user. Therefore a portion of the feed he receives will be irrelevant for him. Second reason is that messages posted by friends of a user is collected in one feed. A user may follow different users as he has different interests. Receiving posts from a large group of people through one channel will lead to information overload, and will make it harder to notice posts related to a specific topic.

For the second problem described above, Twitter provides a feature called *Lists*. With this feature, a user can create a list of users which is a subset of the users he is following. Lists are created manually by Twitter users and they can contain people interested in specific topics such as databases or independent films, but can be also groups of colleagues, family, people in a specific

[1] http://www.twitter.com

location(such as people in Zurich). Lists have their own feeds, and a list feed contains all messages posted by users in that list. Lists can also be followed by other users, therefore "good" lists can be used as information sources by the Twitter community.

Lists can be used in order to follow messages about specific topics by grouping a number of users related to them. However we still identify two kinds of noise in lists. One kind of noise is the result of the problem we stated earlier: users in the list may post with different intentions and some messages can be irrelevant to the topic of the list. Second kind of noise we observe is globally popular topics: some topics can get very popular in Twitter community and all lists contains them as every user is posting about them. Posts related to these popular topics can be ubiquitous and overwhelm the users. Some examples are popular products such as iPad or iPhone or box-office movies. These topics are widely discussed by Twitter community. Another recent example is the ash cloud of Eyjafjallajokull volcano which closed the European air space and affected millions of people. During that time, posts related to ash cloud flooded any kind of lists.

As the Twitter grows in number of users and number of messages posted per day, it is getting harder for users to cope with rate of messages and the noise in feeds[13]. Therefore a information filtering mechanism is needed in Twitter.

1.2 Challenges

There are different challenges in information filtering on micro-blogging environment. Below, we list the challenges:

Short texts One challenge is the limitation on text length. In Twitter, the text of a post is limited to 140 characters. In terms of text classification, short texts contain sparse data, therefore it is a challenge to classify them.

Identifying topics It is necessary to identify topics of messages, so that posts with irrelevant topics can be filtered out. Relevant topics can be very general(e.g., *Information Technology*), less general(e.g., *Databases*) or very specific(e.g., *NoSQL databases*).

Informal language Another problem is the informal structure of the language on micro-blogging services. It is less structured, it can contain abbreviations due to text length limitation and slang; and users modify words to emphasize or show emotions. Therefore the vocabulary is very large and any external source such as Wikipedia or WordNet will not be sufficient to get more information about the text. Also it is necessary to identify which words are very common and which are keywords and useful for text classification. Due to the informal language, a simple stop word list won't be enough, common words on Twitter should be identified.

Constantly changing vocabulary The vocabulary on Twitter is constantly changing with new words and phrases. For instance, new movie or product names, event names can suddenly appear in the language and become very popular. Therefore the text classification system can not be static, it should react to the change in the language.

Different Languages Twitter is used by users from all over the world, therefore many languages are used in posts.

Therefore, an information filtering system on Twitter should be dynamic and it should track Twitter and react to changes. For text classification, it should overcome the challenge of text length limit, and extract keywords in an informal and multilingual text collection. Finally, the system should be able to accurately identify topics and filter irrelevant ones.

1.3 Our Approach

Our aim is to develop an information filtering system for Twitter. For this purpose, we are focusing on one kind of feeds: lists. Many lists on Twitter are created by selecting a group of people related to a specific topic(or topics). Hence by filtering out irrelevant messages from a list feed, it is possible to receive messages only related to the main topic of the list. In a recent study on Twitter lists, it is shown that words extracted from list feeds are representative of characteristics of users in these lists, confirming that the lists are created to gather users with similar interests[27].

First of all, we focus on extracting the main topic of each list. As users related to a topic is gathered in a list, it is expected that words related to that topic is frequent. However, some topics such as globally popular ones and some words that are common in the language can be also frequent as the words related to key topic in the list. To solve this problem, we propose a scoring function that compares the frequencies in the list and frequencies on the whole Twitter community. With this scoring system, we are able to distinguish keywords related to each list's key topic, and we can measure how relevant a post is to the a list's key topic. The frequencies are retrieved in a real-time fashion, therefore our system tracks changes on Twitter. For the problem of short texts in text classification, we utilize the links that are added in posts. With the contents retrieved from these URLs, we manage to improve the accuracy of text classification.

For the filtering task, classifiers are used to decide the relevance of each post in a feed. Classifiers are built using feedback from the users of the system. We extract different kinds of features from each post. One text-based feature is the result of the scoring function. Furthermore, we benefit from social network of Twitter users and use community based features of each post's author. The rest of the features are the temporal features and the link feature. With combination of features from different sources, we are able to build highly accurate classifiers for each individual list. Also with the dynamic nature of our system, we are able to dynamically react to changes on Twitter.

In the following section, related work will be discussed. The third section is about classification, where we discuss different types of features and how they are extracted. The fourth section is on the system architecture. We have developed a prototype of our information system which users can register any list on it and then view the feeds of lists through a web interface with filtering capabilities. The architecture of the system is described in this section, including the web interface and the backend of the system. Next, the results of experiments on the system is presented. With a manually labeled data on selected Twitter lists, we evaluate our system. We especially focus on the learning speed and the accuracy

of the system, as these are directly effective on the user experience. Further, each feature is evaluated, and their benefit on the classification task is discussed. In the last section, conclusion is given and the future work is discussed.

2 Related Work

There are two main areas of research related to our work. First one is text classification and filtering. In this section, we give an overview of text classification and filtering systems. Furthermore, research on short text classification is reviewed as short texts is an important challenge in classifying micro-blogging messages. In the second part, an overview of Twitter related research is presented.

2.1 Text Classification and Filtering

A survey on text filtering is given in [30]. In this paper, a distinction is made between two types of text filtering systems: content-based and social-filtering systems. In content-based systems, filtering is done by exploiting the information extracted from the text of documents. In social filtering systems, documents are filtered based on annotations made by prior readers of the documents. With respect to this framework, our system is closer to content-based filtering systems, however we utilize other sources of information next to the text of documents. We use social features of the users to identify the ones who are more likely to post relevant content, however it is different from the social filtering systems where other users' feedbacks are used.

Text filtering can be seen as an application of text classification where documents are classified into two disjoint categories: relevant and irrelevant. [40] gives an overview of machine learning for automated text classification and describes how classifiers for text classification can be built and evaluated.

Limitation on text size in micro-blogging services result in sparseness of data for text classification. This problem also exists in different domains such as web search snippets, forum and chat messages. Different approaches are proposed for this problem, such as using Wikipedia as an external source[11][39], or using web search engine results[1]. Another approach[31] is using topic models. In this approach, a topic model using Latent Dirichlet Allocation[15] on an universal corpus is built and then this topic model is used to classify short texts. Another work focusing on short texts is [34], where a method for measuring the semantic similarity of texts, using corpus-based and knowledge-based measures of similarity is proposed.

2.2 Twitter Related

In the last years, social networks and micro-blogging services became very popular research topics. Twitter is the most popular micro-blogging service, but it is especially interesting to researchers due to the accessibility of data. Unlike most social networks, posts by most of the users are publicly accessible and these posts can be retrieved in real-time through Twitter API. Also the social network relations can be retrieved. Therefore Twitter became an appealing platform for researchers.

[28] is an extensive study of Twitter. The entire Twitter is crawled and social network, ranking of users, trends and information diffusion on social network is studied. [25] studies why users are using Twitter and how communities are formed. In this paper, it is argued that daily chatter, conversations, sharing information/URLs and reporting news are the main intentions of users. [48] is

a similar study on the usage of Twitter, exploring its benefits and impacts on informal communication.

Most research on Twitter is focused on analyzing, summarizing and mining real-time data. [42] and [41] study the collection and evaluation of related real-time debates on Twitter during events. In [23], a system is proposed for searching the history of activity for a given query. The system identifies activity bursts for the given query string and also selects the message that best describes a burst. [10] and [12] describe systems for collection and indexing of large amounts of streaming text data such as new posts on blogs, messages on Twitter as well as news articles. These systems allow online analysis of these large amount of data such as extraction of popular trends, bursts, spacial and temporal aspects of data. [38] proposes a system for discovering news related posts on Twitter and clustering these messages based on their location.

Another line of research on Twitter is on ranking users and identifying influential ones. Unlike most social networks, Twitter relations are one way: i.e. a user can follow another user without being followed by that user. In this regard, Twitter user graph is similar to the graph of pages in World Wide Web. Therefore research on Twitter social graph follows the idea of PageRank. In [46] follow relations between users is examined and TwitterRank, an extension of PageRank algorithm for social networks, is proposed. TwitterRank takes into account the topical similarity between users as well as the follow relations between users. Another influence metric proposed is TunkRank[44]. The basic ideas of this ranking algorithm are: The amount of attention a user can give is spread among all users she is following; and the influence of a user depends on the amount of attention the user's followers can give. It is also similar to PageRank algorithm, in terms of calculating authority of a node based on the nodes referring to it. In [22] a survey of different ranking methods is done and TunkRank is evaluated against different ranking methods including TwitterRank and PageRank. Results show that TunkRank is the best ranking scheme in penalizing spammers and marketers.

As a real-time system, Twitter is a good source for fresh content. Therefore the real-time stream of Twitter can be used for discovering fresh webpages. Major search engines uses Twitter stream to provide recent webpages in their results. In [20] and [19], real-time micro-blogging is used to detect fresh URLs and improve recency ranking in search engines.

While processing and analyze of large collections of data on Twitter is a very popular topic, limited work is focusing on individual feeds that users are receiving. [43] proposes a method to classify posts in feeds into one of the following categories: news, events, opinions, deals, and private messages. Features used in classification follows the description of categories(e.g., existence of keyword "deal" for deals category). In [37], each post is examined using topic models[36][35] built on recent Twitter stream data. Each topic is labeled by one of the categories(Substance, Social, Status, or Style). The system then can evaluate each post and decide the ratio of each category in that post. In [17], a recommendation system for URLs is introduced. In this system, social voting and topical interests are used.

As a summary, the research on Twitter focuses on the big picture. The papers that focuses on individual feeds are focusing on classifying posts into fixed categories. Our approach differs from this line of research by focusing on lists that are feeds on certain set of topics and automatic extraction of topics

from each list. In this way, we are able to distinguish topics that are relevant to a user. Furthermore, we use novel features beyond text-based and authorship features.

3 Filtering as a Classification Problem

Filtering irrelevant posts from a feed can be achieved by classifying each post in a feed as *relevant* or *irrelevant* and removing the latter from the feed. For this purpose, we propose a method to identify relevant topics in a feed and a classification system which learns from user feedback. Users can view different feeds and can give feedback to individual posts by marking them as *relevant* or *irrelevant*. The classifier learns from the user feedback and makes a decision (relevant/irrelevant) on each new post.

Classification is done based on a selected set of features. For each post in the feed, we retrieve values for these features and create a feature vector. Each post is represented by its feature vector. With user feedback, classifier learns which features are good indicators of relevant posts. Therefore, extracting good features is an important step in building an accurate classifier. Luckily, Twitter provides various information about the post as well as its author and the social network of users on Twitter.

We've extracted 10 different features in total which are grouped based on their sources. The first feature is a text-based similarity score which measures the similarity of a post's text to a feed's key topics. We propose a method to remove noise and understand what are the key topics in a feed, and based on this knowledge we compute the score. Further, we use the authorship feature and additional features extracted from social network information. Another group of features are the temporal features which are extracted from the create timestamps of posts. The last feature is the link domain, which is extracted from links that are included in the text of posts. In the following sections, these different feature groups will be discussed.

3.1 Text Classification and Textual Features

As the aim of classification is to filter posts that have texts irrelevant to the topic of a list, textual features are the main features in our system. The textual features should represent how relevant the topic of the text is to the list. Each list has a different topic(or topics), therefore we need a method to automatically extract the key topics in a list feed and measure the similarity of a post's text to the topic of the feed.

In traditional approach, term frequency(tf) or term frequency-inverse document frequency(tf-idf)[2] measures are used in text classification. These measures indicate how important a word is to a document in a corpus. In tf-idf, the importance of a word to a document is higher when it is used more in the document, and it is lower when the word occurs more in the document collection. After tf-idf weights are computed for each term in a document, a term vector is obtained which each value represents the tf-idf weight of a term in the document. Similarity of two documents then can be calculated by cosine similarity between the term vectors of the documents. Also the vectors can be used in classification algorithms as a vector of features, in this case the classification algorithm will learn which attributes are useful to decide the class of an instance.

Tf-idf weighting is a popular weighting scheme and used in search engines as well as different text classification tasks. With inverse document frequency

[2]http://en.wikipedia.org/wiki/Tf-idf

measure, tf-idf gives lower weight to terms which occur more in the document collection to eliminate common terms such as "the" or "and" in English. However, in the case where the document collection is more focused on one or more topics, the frequent occurrence of a term might mean that this term is very relevant for the list. On the other hand, the tf weighting doesn't take the document collection into account at all, therefore can't distinguish common words. Therefore these weighting schemes don't utilize the existence of key topics through the documents in a collection.

As these weighting schemes are not suitable for our classification task, we propose a new method which basically scores the proximity of a text to the key topics of a feed. The main challenge here is to distinguish which frequent words are really related to the key topics of the feed, and which are frequent because they are common in the language. This distinction is possible when we observe the frequencies of words in a much more general feed. When a word's frequency in a very specific feed and in a "global" feed where a large community of users post messages are compared, it can be observed whether the word is more frequently used in the feed. Comparison of these frequencies will give insight about how specific a term is to a focused feed.

For instance, let's assume there is a focused feed which consists of posts from a list of database related users in Twitter and there is a global feed which collects all posts by the users of Twitter. It is intuitive that *MySQL* word will appear frequently in the focused feed as a result of users' collective interest in this database system. Unlike the focused feed, the global feed will contain users which have variety of interests. There will be thousands of different topics in this global feed and one of them will be also database systems. Therefore the frequency of database topic in the global feed will be much lower, hence *MySQL* word will appear less frequently. If we take a very common such as *time*, *people* or *today*, it would be expected that these words are very common in both the focused feed and the global feed. *Today* can be more frequent than *MySQL* in the focused feed, however we will be able to distinguish these two words by their frequencies in the global feed. It can be argued that words like *and* can be removed by a stop words list, however words like *time* can still be a related term for some feeds (e.g., Time magazine for a journalism feed). Also the vocabulary used on Twitter is large and constantly changing, therefore this approach allow us dynamically identify common words.

By using the idea of comparing a term's frequency in a focused feed to a general reference feed, we compute a score indicating how specific(or relevant) the term is to the feed. While we call the focused feed "local", we will call the reference feed as "global". While computing term frequencies, we treat a feed as one document so we don't distinguish posts in the feed as separate documents. Therefore the text of the feed consist of texts of all posts in the feed. Then the term frequencies of words in feeds are calculated as:

$$tf_{w,f} = \frac{n_{w,f}}{\sum_k n_{k,f}} \tag{1}$$

where $n_{w,f}$ is the number of occurrences of word w in feed f while the denominator is the sum of number of occurrences of all words in feed f.

The relevance score(RS) of word w is:

$$RS_w = \frac{tf_{w,local}}{tf_{w,global}} \tag{2}$$

By using the relevance scores of individual terms in a post, we compute the text relevance score(TRS) for post p as:

$$TRS_p = \frac{\sum_{i=1}^{n} RS_{w_i}}{n} \tag{3}$$

where n is the number of words in post p and w_i is the i^{th} word in the text of the post. This value is the average relevance score of words in the post.

Text size for each post is limited to 140 characters in Twitter which makes the classification based on text more challenging. To overcome this problem, we use the links that text contains. It is very common in Twitter to share an URL by adding it to the text of the post. The contents of the URLs can be very useful in text classification. We retrieve the text contents of URLs in posts through an HTML text extraction service, and merge this text with the original text of each post. Also in calculating term frequencies in list feeds, we use these extended texts. We show the improvement by this approach in experiments section.

To show how the scoring function works on real datasets, we collected 9 user-created lists from Twitter. These lists will be also used in experiments section. Table 1 shows these lists. In each list, term frequencies and term relevance scores are computed. 30 words with highest term frequencies are listed in Table 2. 30 words with highest relevance scores are listed in Table 3. When the first table is observed, we see that some of the words with high frequencies are mostly related words to each list(such as *baby* in pregnancy-parenting and *film* in film-people-to-follow). This result indicates that lists are focused on specific topics as we have discussed. However, there are very common words which are general and not list specific such as *time*, *twitter*, *make*. These are very general words that appear in all feeds and they don't contribute to the classification of texts. On the second table, we observe more specific words. The highest scoring words are mostly very specific to each list and there are no general words. Also we start to observe words starting with a hash symbol(#). These special words are user created tags which users mark their posts to indicate a topic. On the other hand, there are some words which are made of multiple words. These words also have high scores as they rarely appear in the global feed.

Id	List name	List creator	Topics
1	pregnancy-parenting	Disc_Health	Pregnancy and parenting
2	film-people-to-follow	IndieFlix	Independent movies
3	ecotweets	RosieEmery	Ecology
4	markets	WSJMarkets	Stock markets
5	dbpeople	edward_ribeiro	Databases
6	apple-news	huffingtonpost	News related to Apple Inc.
7	comicscreators	iFanboy	Comic writers
8	mysql-co	lucaolivari	MySQL and related systems
9	database	pcdinh	Databases

Table 1: 9 lists on various topics

By our method, we manage to identify specific words in each feed and based on this information, we calculate a score measuring the relevance of a post to a feed. The score has the lower limit as 0 but has no upper limit. In each feed, the results of the scoring function may vary with the popularity of the feed's topics in the global feed. For instance, in a feed with a very niche topic would contain posts with very high text relevance scores. We use the result of the scoring function as a feature in classification. Therefore, classification algorithm learns the threshold score above which the posts are relevant for each list.

List name	Top 30 words ordered by frequencies in list
Disc_Health/ pregnancy-parenting	baby(0.00898) child(0.00806) parents(0.00675) time(0.00565) make(0.00561) read(0.00447) pregnancy(0.00429) women(0.00416) children(0.00342) find(0.00333) kids(0.00333) life(0.00329) feel(0.0032) people(0.00316) back(0.00307) parenting(0.00285) pregnant(0.0028) dr(0.00272) day(0.00267) home(0.00263) good(0.00263) im(0.00254) love(0.00245) week(0.00241) weeks(0.00237) parent(0.00232) things(0.00232) mother(0.00232) learn(0.00228) matter(0.00223)
IndieFlix/ film-people-to-follow	film(0.01583) films(0.00551) time(0.00402) movie(0.00397) cannes(0.00336) people(0.00306) festival(0.0029) director(0.00249) work(0.00249) year(0.00241) years(0.00237) make(0.00233) story(0.0022) world(0.00216) made(0.00213) im(0.00212) life(0.00212) day(0.00208) back(0.00204) good(0.00203) man(0.00191) movies(0.0019) great(0.00178) love(0.00177) filmmakers(0.00165) filmmaker(0.00162) show(0.0016) hes(0.00155) cinema(0.00146) set(0.00142)
RosieEmery/ ecotweets	de(0.00802) oil(0.00619) twitter(0.00561) para(0.0051) energy(0.00281) people(0.00281) ferramenta(0.00274) water(0.00256) time(0.00244) bp(0.00224) spill(0.00223) year(0.0022) solar(0.00219) gulf(0.00218) green(0.00218) make(0.00211) years(0.00202) day(0.00196) environmental(0.0017) world(0.00163) company(0.0015) work(0.00149) em(0.00147) climate(0.00145) site(0.00144) blog(0.00144) food(0.00138) power(0.00133) change(0.00129) long(0.00128)
WSJMarkets/ markets	market(0.00687) year(0.00516) euro(0.00429) stock(0.00413) investors(0.00388) markets(0.00379) stocks(0.00376) shares(0.00371) financial(0.00334) trading(0.00329) million(0.00328) billion(0.00324) company(0.00311) week(0.00295) bank(0.00283) debt(0.00271) day(0.00255) oil(0.00255) banks(0.00249) european(0.00249) time(0.00232) companies(0.00228) nyse(0.00225) dow(0.00218) government(0.00216) nasdaq(0.00214) sales(0.0021) quotes(0.00206) points(0.002) quarter(0.00199)
edward_ribeiro/ dbpeople	data(0.00847) sap(0.00486) time(0.00432) business(0.00335) software(0.0029) technology(0.00275) work(0.0027) database(0.00265) people(0.00254) sybase(0.00245) engineering(0.00239) cloud(0.00219) make(0.00218) company(0.002) code(0.00198) systems(0.00189) customers(0.00187) applications(0.00186) based(0.00184) system(0.00182) good(0.00179) information(0.00176) problem(0.00174) web(0.00173) world(0.00171) server(0.0017) open(0.00158) years(0.00157) back(0.00155) memory(0.00155)
huffingtonpost/ apple-news	apple(0.01564) iphone(0.01441) ipad(0.01223) app(0.00637) apples(0.0044) mac(0.00419) time(0.00393) google(0.00341) users(0.00315) store(0.0029) video(0.00289) mobile(0.00263) apps(0.00263) company(0.00259) market(0.00253) os(0.00242) year(0.00239) device(0.00234) web(0.00232) flash(0.00225) data(0.00224) free(0.00224) content(0.00221) ipod(0.0022) game(0.00216) make(0.00215) jobs(0.00198) devices(0.00191) itunes(0.00189) touch(0.00186)
iFanboy/ comicscreators	im(0.00513) time(0.00477) comics(0.00456) work(0.00389) book(0.00362) comic(0.00346) people(0.0034) man(0.00311) story(0.00302) back(0.00286) good(0.0028) day(0.0028) make(0.0024) art(0.00232) great(0.00221) world(0.00218) series(0.00215) love(0.00208) years(0.00204) read(0.0019) show(0.0019) issue(0.00185) page(0.00175) today(0.0017) life(0.00167) year(0.00163) made(0.00163) marvel(0.00162) thing(0.00161) books(0.00161)
lucaolivari/ mysql-co	mysql(0.00886) de(0.00686) oracle(0.00456) sun(0.00433) time(0.00395) data(0.00382) la(0.00327) open(0.00309) en(0.00289) netbeans(0.00283) server(0.00281) java(0.00255) source(0.00254) people(0.00239) cloud(0.00239) web(0.00236) performance(0.00235) database(0.00229) work(0.00227) software(0.00206) application(0.00205) system(0.00196) file(0.00195) make(0.00191) support(0.00191) blog(0.00188) el(0.00187) business(0.00186) code(0.00181) page(0.00177)
pcdinh/ database	server(0.01485) sql(0.01441) data(0.01202) database(0.00747) microsoft(0.0043) web(0.0042) business(0.00364) time(0.00325) software(0.00298) ms(0.00278) center(0.0027) information(0.00258) windows(0.00243) services(0.00238) system(0.00235) mysql(0.00234) work(0.00222) performance(0.00209) application(0.00207) users(0.00205) applications(0.002) people(0.002) user(0.00198) management(0.00197) development(0.00191) support(0.00189) code(0.00187) key(0.00187) access(0.00185) based(0.00184)

Table 2: Top words by term frequency

List name	Top 30 words ordered by relevance scores in list
Disc_Health/ pregnancy-parenting	petrie(6547.96) swaim(2529.89) indicating(2232.26) recurrence(2083.44) futuretweets(1785.81) maintanance(1785.81) hormonal(1636.99) particulate(1636.99) mucus(1636.99) breech(1488.17) epidural(1488.17) shebang(1488.17) pediatrician(1413.76) obstetrician(1339.36) circumcision(1339.36) lactation(1339.36) eddleman(1190.54) miscarriage(1140.93) prenatal(1140.93) chorpita(1041.72) whoopie(1041.72) misbehavior(892.9) adoptive(892.9) midwife(892.9) amniocentesis(892.9) parentingtoolbox(892.9) sensory(892.9) implantation(892.9) homosapien(892.9) 100e(892.9)
IndieFlix/ film-people-to-follow	filmmakers(5594.18) indiewire(2856.6) cinematic(1089.54) godards(915.58) kohn(819.44) mubi(778.24) cinematical(759.93) kiarostami(663.79) auteurs(604.28) binoche(567.66) apichatpong(558.5) berney(549.35) biutiful(526.46) auteur(466.94) leighs(448.63) indiewires(444.06) palme(425.74) cinematographer(425.74) croisette(416.59) #independent(398.28) gokustom(384.54) weerasethakul(384.54) indicating(384.54) liman(375.39) boonmee(363.94) directorial(361.65) araki(352.5) kiarostamis(347.92) brolin(343.34) panahis(343.34)
RosieEmery/ ecotweets	evergreen(2203.29) eslr(1306.87) beekeeping(1083.73) solfocus(1045.5) forests(988.38) ferramenta(774.85) permaculture(690.22) evergreens(617.13) everq(609.87) ecosystems(546.47) whaling(541.87) estatsticas(537.27) gerenciar(537.27) cpv(524.2) eslrs(522.75) devens(522.75) ecological(507.74) mammals(464.18) suttles(435.62) contamination(425.46) preventer(393.51) photovoltaic(379.96) deforestation(363.02) indicating(356.73) tweepml(338.33) cotweet(326.72) multiplas(322.36) possibilita(322.36) monitorar(322.36) #ocean(318.01)
WSJMarkets/ markets	comstock(4269.63) nls(1118.4) treasurys(972.22) poors(877.04) reprints(795.46) schapiro(761.46) jpm(754.66) securities(693.47) premarket(618.69) csco(605.09) nyse(588.88) cdos(530.3) monetary(526.9) deficits(503.11) tightening(496.31) djia(496.31) bocvip(489.51) cramer(475.91) deflation(469.11) kudrna(469.11) typically(469.11) comply(460.62) ibd(457.78) exporters(428.32) dows(428.32) interbank(421.52) composite(421.52) holdings(418.12) seasonally(407.93) paulson(401.13)
edward_ribeiro/ dbpeople	voltdb(4040.81) #ensw(3211.54) algorithms(1929.94) dbms(1673.62) sybase(1664.57) vertica(1568.07) hadoop(1226.31) sybases(1191.13) replication(1176.06) terrastore(1145.9) transactional(1085.59) postgres(1070.51) couchdb(1040.36) analytic(1025.28) scalability(1010.2) snabe(964.97) relational(964.97) oracles(949.89) #sybase(904.66) hbase(904.66) mcdermott(874.5) sikka(859.43) mapreduce(829.27) nanotechnology(814.19) cluster(776.5) typically(768.96) riak(678.49) hasso(678.49) oltp(678.49) bydesign(648.34)
huffingtonpost/ apple-news	ballmer(1405.27) tipb(1400.68) adobes(1070.02) toget(757.74) tuaw(743.97) widgets(740.14) modifying(734.78) affidavit(730.19) macrumors(727.89) macbooks(721) digitimes(610.79) teardown(606.19) mywi(560.27) includingturningit(546.49) butyou(546.49) decidewhetherto(546.49) foxconns(495.98) conductive(486.79) sweatshop(473.02) infrequent(468.42) shamma(463.83) frequencies(459.24) unregulated(450.05) taoviet(445.46) totunein(440.87) proprietary(436.28) munster(404.13) turnarounds(394.94) changewave(390.35) filemaker(390.35)
iFanboy/ comicscreators	frazetta(2486.96) avengers(1560.91) frazettas(722.21) dredd(710.56) hawkeye(652.32) bendis(594.08) colorist(500.89) marvels(477.59) witchblade(477.59) timegate(471.77) brubaker(460.12) nrama(454.29) tcaf(436.82) eisner(436.82) malia(425.17) newsarama(401.87) hellboy(401.87) shadowland(401.87) dazzler(396.05) jeanty(396.05) comics(387.02) heroescon(366.93) parlov(366.93) niles(349.46) colorists(331.98) thunderbolts(326.16) indicating(320.33) narrator(302.86) c2e2(297.04) palmiotti(297.04)
lucaolivari/ mysql-co	glassfish(5909.87) opensolaris(2680.76) replication(2437.06) netbeans(2398.98) innodb(1949.65) zfs(1888.72) asadmin(1705.94) helloworld(1584.09) eucalyptus(1523.16) javafx(1462.24) failfast(1462.24) cluster(1454.62) scalability(1370.85) mysql(1368.08) workbench(1309.92) oracles(1279.46) iscsi(1279.46) dtrace(1218.53) jtreg(1188.07) privileges(1005.29) maatkit(1005.29) interceptors(974.82) ejb(944.36) propertychangesupport(944.36) dbas(913.9) typically(913.9) bleonard(913.9) clustering(852.97) samevm(822.51) sendmail(822.51)
pcdinh/ database	scala(1660.49) replication(1333.63) sql(1254.51) configure(1098.28) algorithms(993.68) relational(993.68) dbcc(889.08) scalability(876.01) ssis(797.56) mdf(784.49) cursor(732.19) typically(706.04) coresite(706.04) checkdb(666.81) scalable(653.74) workloads(653.74) concurrency(653.74) db2(627.59) etl(601.44) dexterity(601.44) fabros(588.36) entities(588.36) rdbms(588.36) transactional(588.36) dbas(562.21) datacenter(536.07) lambda(522.99) unlocker(522.99) trackback(522.99) zemanta(509.92)

Table 3: Top words by relevance scores

3.2 Authorship and Social Network Features

Besides the text, the author of the post can also be a good indicator of relevance of the post. By using the author id as a feature, the classifier can learn which users are more likely to submit relevant content. However, authorship feature is not useful when a post of an author which the classifier haven't seen before will be evaluated. In case no previous posts of an author is in the training dataset, the author feature of that author's posts won't bring any information gain as there is no information about that author. This means that author feature is only useful with authors which there is training data on. In our classification task, our aim is to filter feeds that may have hundreds of authors and we expect to have good accuracy even with very few feedback from the user of the system. Therefore we benefit from the social network of users in Twitter and extract some features of users related to their positions in this social network. In this way, we aim to extract features that are common for authors that post relevant content.

Twitter users receive posts of other users by *following* them. The group of users that follows a user is called the *followers* of that users, and the users that a user follows are called *friends* of that user. A user receives all posts by her friends. Lists are on the other hand, a subset of the friends of a user. When we consider the social network as a graph, a user is a node whereas the follow relation is a directed arc. Number of followers a user has will be the number of incoming arcs to that user's node. Similarly, number of friends of that user is the number of outgoing arcs of that users node. Using that social graph properties, and a special user ranking algorithm called TunkRank[44], we propose 5 social features:

Number of followers in list Each list of users in Twitter can create a subgraph of users and user relations. While each user in a list is represented as a node, follow relations are the arcs between these nodes. This feature represents the number of incoming arcs, i.e., the number of Twitter users in a list that follows the user. For a user that is in different lists, this attribute may have a different value in each list due to difference of graphs in these lists. This is logical as a user can be more related to some lists and submitting more relevant content regarding these lists. With this feature, we try to take into account how authoritative a user is in that community. A high number of followers in a list might indicate that the user is followed by users related to the topics of the list, therefore the user is also authoritative in the topic.

Number of friends in list Similar to the previous feature, this feature represents the number of Twitter users that are followed by the user. With this feature, our aim is to understand how the user is related to the topics in this list. Our assumption is that if the user is following a large number of users from the list, he would be more interested in the topics of the list, therefore also posting relevant content to these topics.

Number of followers For this feature, we take the number of Twitter users that are following the user. We expect that a higher number of followers would indicate that the user is relevant and trustworthy. Contrary to the first two features, all Twitter users are taken into account.

Number of friends This feature represents the number of Twitter users that the user follows.

TunkRank score TunkRank is a social influence metric for users on Twitter described in related work section. TunkRank can be good metric indicating the relevance/authority of the user, therefore we use it in our social features set.

For each post, we use the id and social features of the post's author as features of the post. By author feature, the classifier learns about individual users whereas by social features it learns about the features of a relevant author. Therefore posts by unseen users can be classified utilizing social features. We will evaluate these different features in experiments section and show that social features are improving authorship feature.

3.3 Temporal Features

Another aspect of posts are their temporal properties. Each post sent on Twitter contains also its create time. We use this information to extract temporal features of posts. Our assumption is that there can be time periods where users are posting relevant content while in some periods they submit irrelevant content. For instance, a developer who is included in a software engineering list might submit relevant content during work hours, however during nights or weekends, he might submit post about food and travel. The patterns might be user specific but users in a list might share a common pattern. To benefit from a common pattern and to make the temporal features accurate, we convert the GMT time of the post to local time of the author. For this, we use the time zone of the user. This can be sometimes inaccurate as users are setting time zones manually, however this is the only information about the local time of the users on Twitter.

After we calculate the timestamp of the post adjusted to user's local time, we extract two features out of it:

Hour of the day This feature indicates the hour part of the time, i.e., 0-23.

Day of the week This feature indicates the day of the week, i.e., Monday, Tuesday, ... , Sunday.

With hour of the day feature, we try to discover patterns during the day such as work hours. On the other hand, with day of the week feature, we try to distinguish days, such as weekend vs weekdays.

3.4 Link Domain Feature

A large percentage of posts on Twitter contains links. Links are added to posts to refer to news articles, blogs, company web pages, other social networks, etc. The URL of a link can provide information about the topic of a post. If the link is referring to a website which is very related to a list, it is very likely that the post is relevant. This is especially valuable in Twitter as texts in posts are very limited in size and they may not be enough in deciding if the post is relevant.

As the number of possible URLs are infinite and it is not very likely that each URL will be given feedback to by users, we use the domain of URLs as

a feature. With the domain of a URL, it is possible to distinguish different types of web sites, for instance news portals, blog pages, social networks. With the feedback from the user, classifier learns which website domains are referred when a post is relevant.

Due to the text length limit on Twitter, mosts links are shortened through URL shortening services. These services provides short URLs which redirect to original URLs. Therefore before extracting the domain of an URL, the original URL is retrieved from the shortened version. Then the domain of the URL is extracted from retrieving the portion of URL until the end of top level domain. Therefore we also get the subdomain of an URL. Some examples are:

$$http://www.nytimes.com/pages/technology/index.html \rightarrow www.nytimes.com$$
$$http://movies.yahoo.com/ \rightarrow movies.yahoo.com$$
$$http://news.yahoo.com/ \rightarrow news.yahoo.com$$
$$http://www.techcrunch.com/2010/08/10/google-wave-death/ \rightarrow www.techcrunch.com$$

For some web domains such as Yahoo, subdomain indicates different themes such as movies, news, travel etc. In this case, our approach benefits from the subdomains and can distinguish different themes in portal sites. In case there is no link in a post, then the value is set as *NO_LINK*. If there is more than one link in a post, only the first one is used.

In text-based features, we retrieve the contents of URLs included in the post and merge it with text of the post. However, still we use domain of the URL as a separate feature. Therefore we utilize links in posts in two ways. One of the reasons is that for some webpages, such as image hosting sites or social networks which requires authentication, no text content can be retrieved from links. In this case, domain of the URL becomes the only information about the link.

4 System Architecture

In [30], a conceptual framework for information filtering is proposed. In this framework, information filtering process is divided into three subtasks: collecting the information sources, detecting useful information, and displaying the useful information. We can also describe our system with three main modules which are responsible for collection, detection and display tasks. The first module processes the feeds which are added to the system and maintains them over time. Second module stores related data about each feed as well as classification structures for filtering task. Finally the third module is responsible for presenting the filtered feeds as a webpage. In this section, each module will be explained in detail.

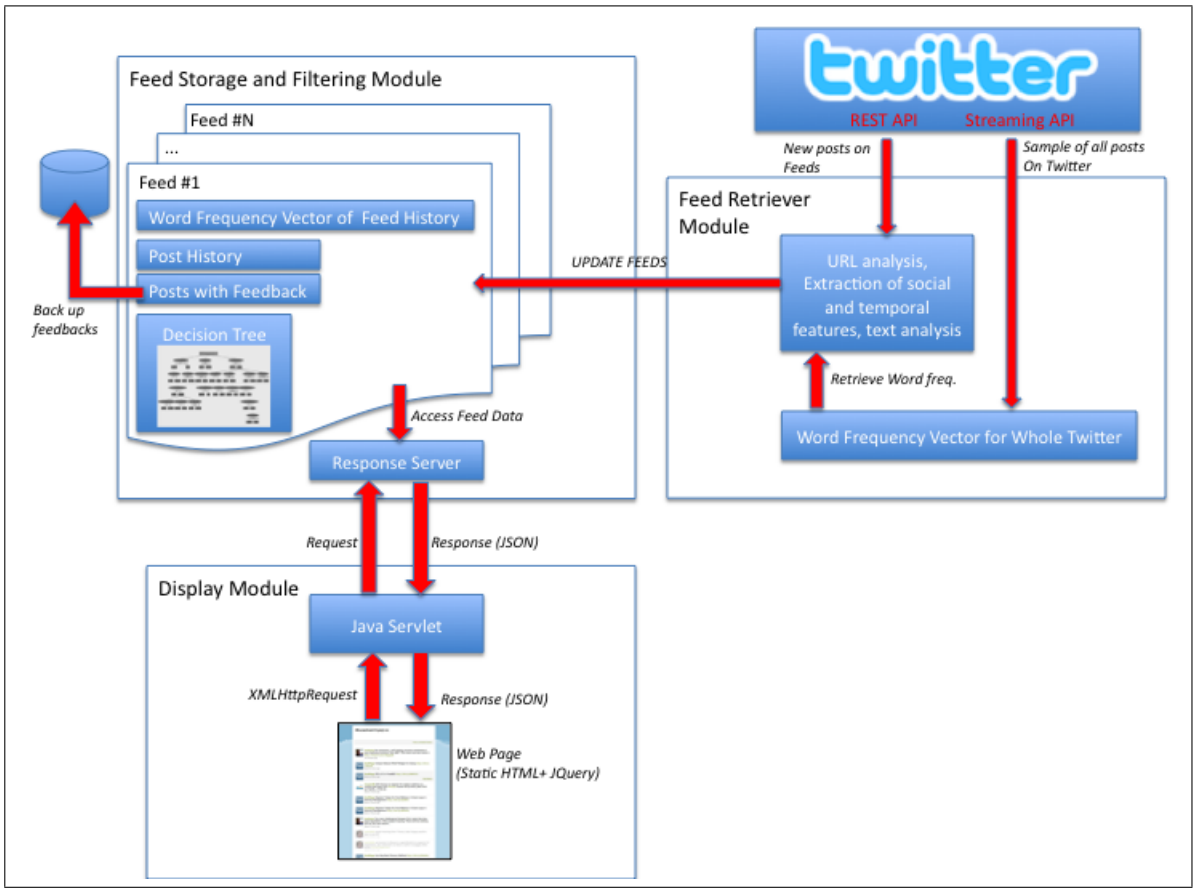

Figure 1: Overview of the system

4.1 Feed Retriever Module

This module is responsible for retrieving the feeds from Twitter and processing them.

Twitter provides data to third party services through an Application Programming Interface(API) [9]. The API consists of two parts: one is a Representational State Transfer(REST) API by which different kinds of data such as messages posted on feeds or social graph relations can be retrieved. Second part of the API is a streaming API which allows users to create consistent connections and have realtime access to various subsets of messages posted on Twitter. While requests on REST API are analogous to single queries, streaming API connections are to long-lived queries.

The feed retriever module retrieves the messages posted into feeds from Twitter by making requests to REST API. While many feeds can be registered in the system, this module checks for new messages periodically for each feed. In case a new feed is added to the system by one of its users, the feed retriever module retrieves the history of the feed, i.e., a number of latest messages posted to a feed. The history of feed is used in understanding the nature of the feed and therefore filtering it. As there are access rate limits on the API, the module watches the rate of the requests to the server. The responses are received from the API in JavaScript Object Notation(JSON) format [5] and parsed with a JSON parser.

After the latest posts in a feed is received, these posts are processed before they are stored. The posts are processed to extract different features which then will be used to decide if the post is relevant to a feed. First of all, the links in each post is analyzed. As the text size is limited to 140 characters in each post, it is very common to use URL shortening services on Twitter. While different shortened URLs might redirect to the same end URL, it is necessary to replace these URLs with the ones they redirect. After the real URLs are discovered, we use an API service [2] for HTML text extraction which returns key contents of the page in plain text. This service removes ads, links, comments and other unrelated parts of a web page and returns the main text such as the text of the article on a news portal. These two tasks are done in parallel with multiple threads as waiting for the result of HTTP request takes time. It is possible that same post exists in different feeds or different posts contains same links. Therefore cache is used for recently expanded URLs and retrieved URL contents. Least Recently Used(LRU) cache type is used and only a number of recent data is kept in this cache. Utilizing this cache, number of URL connections and API service calls for URL processing are reduced.

The next step of the processing is the text analysis. In text analysis, both the text from the post and the text extracted from the URLs that the post contains are taken into account. First the text is cleaned by removing ASCII characters, URLs, punctuation, numbers and stop words. Also the mentions of other Twitter users in posts are removed as well. Finally the text is converted to lower case and tokenized. After the text is cleaned, each post receives a score indicating its similarity to the feed history. In calculating the score, we take into account the word frequency vector of the feed, the word frequency vector of the post and frequency of these words in Twitter global stream history. The latter is retrieved from a separate process which will be explained next. After the score is calculated, the words in the post is also added to the word frequency

vector of the feed.

Besides URL and text, there are different features used, such as the user posting the message and social graph information as well as temporal features. They are also extracted from the API response. After the processing is done, the post data and its extracted features are stored in the system.

4.1.1 Word Frequency Vector for Twitter

As we also take into account the frequency of words in whole Twitter community for the text analysis, we have a separate process for keeping track of word frequencies. The Twitter API supplies a stream which returns a sample of all messages posted on Twitter by users with public accounts. Currently the sampling rate is 5%, however it is enough for maintaining a word frequency vector. Our process keeps the word frequencies for the history(e.g., 1 week) of data on Twitter global stream. Each post retrieved from the stream is processed by cleaning the text and tokenizing, and then updating the word frequency vector.

The word frequencies are kept in a sliding windowed fashion. The time window consists of a number of time intervals and for each of them a word frequency vector is kept. Also a larger frequency vector for the whole time window is kept. When the time window is to slide, the oldest word frequency vector is removed. The frequency values of words in this vector is subtracted from the frequency vector which is kept for the whole time window. This process runs continuously and keeps a real-time word frequency vector for the whole Twitter. This data gives an insight about the popularity of each word in general and is used for understanding how specific each word is to a feed. As it keeps a time window in real time, it can track bursts, therefore can identify general words that suddenly get popular, and filter them in local feeds.

The process runs standalone and it responses to word frequency lookups through a socket connection. When the feed retriever module accesses this process, it sends a list of words and the response is the word count of each word plus the number of all words in the time window. With this data, word frequencies are calculated.

4.2 Feed Storage and Filtering Module

The second module is responsible for storing the feed data, keeping classification structures for filtering and finally responding the requests from the display module. This module stores many feeds and these feeds are updated by the feed retriever module.

For each feed, various data is stored in the module. These are:

- **History of posts:** For each feed, a history of latest posts are stored. For each post, two kinds of data are kept. First group consists of data such as text, user name, timestamp which are displayed to the user of the system. The second kind of data consists of extracted features of the post which is used internally by the filtering system. The number of posts to be stored is limited in the history, after a certain threshold older posts are removed. The posts are stored time-ordered.

- **Feedback:** When a post receives feedback from a user, the post is copied to a list consisting of posts with feedback. The list is kept separately and is used as a dataset for decision tree. The feedback is also stored in database. In case the system is restarted, the feedback is restored from the database.
- **Feed information:** Each feed has some meta information such as name and description. These information are kept so that users can get detailed information about feeds registered in the system.
- **Decision tree:** The filtering in the system based on learning from user feedback on the posts by using a decision tree. Therefore a decision tree is kept for each feed in the system. The decision tree uses the posts with feedback as training dataset. When a new feedback is given to a post of a feed, the decision tree of the feed is rebuilt. For the decision tree implementation, WEKA data mining library is used[24].
- **Word frequency vector:** Each feed has a word frequency vector for the words seen in the feed history. As new posts are received from the feed retriever module this vector is updated. Also when older posts are removed from the history, the vector is updated again.

The feed storage and filtering module provides data about feeds to display module. The data is formatted in JSON and sent through socket to the display module. Below the different use cases will be described.

- **Get posts from a feed:** The posts in feeds can be accessed in pages which contain 20 posts. The request contains the feed name and the page or the maximum id of the post. The module returns the posts from the feed history. Before sending the posts, each post is classified by the decision tree of the feed and therefore gets a label which indicates if the post is relevant or not. The label is also added in the response and is used by the display module in filtering the irrelevant posts.
- **Give feedback:** After displaying the posts in a feed, user can mark individual posts as relevant or irrelevant. When a feedback is received by the system, feedback will be saved in feed data, and the decision tree will be rebuilt.
- **Add a new feed:** The users can also add feeds that are not registered in the system. In this case, the feed retriever module will fetch the data from Twitter, and the new feed will be stored in feed storage module.
- **Get the list of feeds:** The users can request a list of all feeds in the system. In this case, detailed information about the feeds is provided.

4.3 Display Module

The last module is responsible for displaying the feeds with filtering capabilities to the users. This module consists of a Java Servlet running on Apache Tomcat server[3] and a web page using JQuery[4].

The webpage contains a directory page for the feeds in the system and a second page for showing filtered feeds. In the directory page, feeds are grouped

under categories and users can add new feeds under these categories. When a feed is selected, the feed is shown on a new page. Here posts which are labeled as *relevant* are shown less visible or can be hidden by user preference. This page is very similar to Twitter list pages except the filtering system.

Pages contains static HTML elements and JQuery code which is based on JavaScript. When user presses a button for one of the user cases described above, a request is sent to Java Servlet. Servlet page redirects the request to feed storage module, which in turn returns a response in JSON format. This data is passed back to JQuery script on the webpage. With the data, JQuery modifies the webpage without reloading. In this way, user can give feedback to posts or view the history of the feed without reloading the page. The relevant and irrelevant posts in feeds are shown with different opacity values. Therefore irrelevant posts still exist on the page so that users can give positive feedback on them. Users can also choose to hide irrelevant posts.

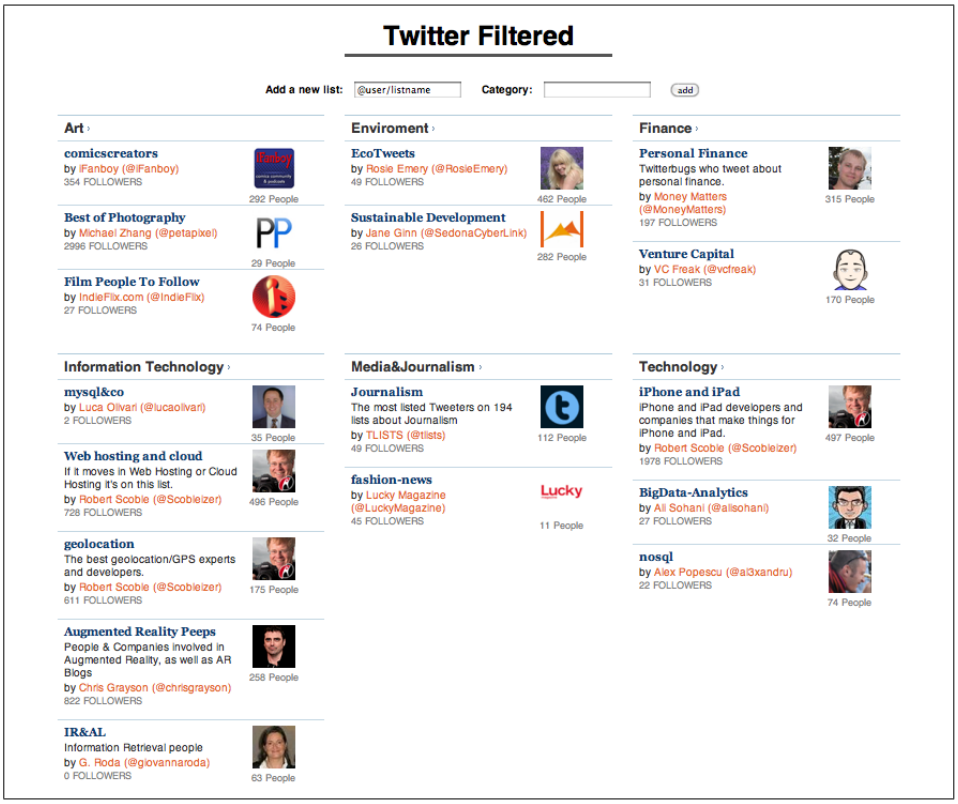

Figure 2: Directory page

Figure 3: Showing a feed with filtering

5 Experiments

In this section, we evaluate various aspects of our classification approach. As our approach requires user feedback, a dataset with user feedback has been manually created. In the next section, this process will be described. Then we will compare different classification algorithms which can be used for the classification task. In the following sections, we will evaluate how different features affect the performance of the system and compare our approach with traditional Bag-Of-Words text classification method in terms of: (1) accuracy, (2) learning speed and (3) computation time necessary for building the classifiers.

5.1 Experiment Data

For the experiments, we retrieved 9 lists on various topics from Twitter. Each list is manually created by Twitter users by selecting a group of other users. The posts sent to these lists are collected between 7th May 2010 and 31st May 2010. Also during this period, a sample portion(5%) of all posts sent on Twitter are collected from streaming API. Starting from 12th May 2010, posts on lists are labeled through an interface. The posts between 7th and 12th are stored as the history of each list which is used for calculating similarity of a post to a list. The interface(Figure 4) fetches a random unlabeled post from the database and displays the post and contents of the link in case the post contains an URL. The user then labels the post as "relevant to the list" or "irrelevant to the list" and also can pass the post in case no decision could be made (e.g., when the post is in a foreign language).

The lists we have selected vary in terms of number of users they contain, the number of posts they receive and the number of relevant posts. Also number of feedback given to each list varies. The information about these lists are shown in Table 4.

Id	List name	List creator	# of users	# positive labels	# negative labels	% of positive labels
1	pregnancy-parenting	Disc_Health	12	58	30	65%
2	film-people-to-follow	IndieFlix	74	388	322	47%
3	ecotweets	RosieEmery	461	74	36	67%
4	markets	WSJMarkets	15	115	13	89%
5	dbpeople	edward_ribeiro	75	43	47	52%
6	apple-news	huffingtonpost	29	227	78	74%
7	comicscreators	iFanboy	292	43	37	54%
8	mysql-co	lucaolivari	35	38	54	41%
9	database	pcdinh	45	54	33	62%

Table 4: Labeled datasets of each list

5.2 Evaluation Metrics

While the experiment results are presented, accuracy, precision, recall and f-score measures will be used. These measures are defined as:

$$\text{Accuracy} = \frac{\text{Number of true positives} + \text{Number of true negatives}}{\text{Number of positives} + \text{Number of negatives}} \quad (4)$$

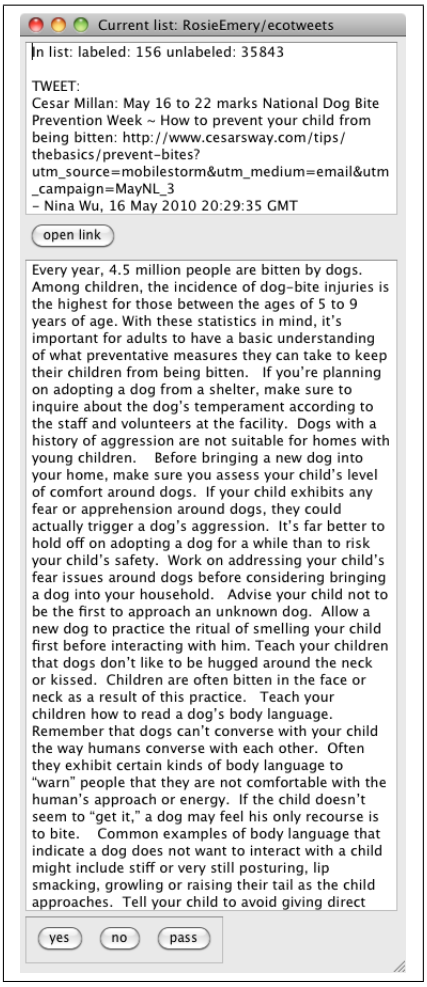

Figure 4: Manual labeling interface

$$\text{Precision} = \frac{\text{Number of true positives}}{\text{Number of true positives} + \text{Number of false positives}} \tag{5}$$

$$\text{Recall} = \frac{\text{Number of true positives}}{\text{Number of true positives} + \text{Number of false negatives}} \tag{6}$$

$$\text{F-Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

Accuracy is the ratio of correctly classified posts to all posts in our case. Precision measures the correct decision rate only in instances classified as positive(*relevant* in our system) and takes into account false positives(*irrelevant*

post classified as *relevant*), recall takes into account the false negatives(*relevant* post classified as *irrelevant*). We don't make a distinction between false positives and false negatives, therefore we mainly focus on accuracy results in the experiments. Still the precision and recall values are displayed to give additional information. Other than accuracy, precision and recall, we also use f-score. This is the harmonic mean of precision and recall, giving the two measurements equal value.

5.3 Comparison of Classification Algorithms

The selection of classification algorithm can be very effective on classification results. Therefore, we experimented with different classification algorithms to find an algorithm which performs well for the classification task. We run several classification algorithms from Weka library[3] on each list in our dataset. As our dataset contains nominal and numeric attributes, it is not possible to use some types of classifiers such as Bayes classifiers. We use the following classification algorithms: Alternating decision tree(ADTree)[21], J48 which is an implementation of C4.5 decision tree algorithm[33], Random forests[16], Lazy K* algorithm[18], Decorate[29], MultiBoostAB[45] and SMO algorithm which is sequential minimal optimization algorithm for training a support vector classifier(will be called SVM in the experiments)[32]. To describe shortly; ADTree, J48, REPTree are different kinds of decision trees. K* is an instance based classifier which decides on the class of a test instance based on the most similar training instances in the dataset. As the parameters, in ADTree the number of boosting iterations is set to 20 and in J48 pruning and binary splits are turned off. For the rest of the algorithms, default settings are used. As the feature set, text relevance score(TRS), authorship feature, social features, temporal features and link domain feature are used.

Results are shown in Figures 5 to 13. Accuracy results vary from list to list (Figure 5), and different classifiers perform the best in different lists. When we look into average accuracy scores (Figure 6), ADTree, J48 and MultiBoost AB are the top classification algorithms with the highest scores. Another important aspect is the performance of classifiers through different datasets. It is a good property that a classification algorithm performs well in different datasets. It is especially important in our case, as our system should provide decent performance for any list registered to it. Figure 7 shows the standard deviation of accuracy of classifiers through 9 lists. Accuracies of ADTree, J48, Decorate and MultiBoost AB through the set of lists have lower standard deviations which means they better suit different datasets. SVM and K* algorithms are the worst performing algorithms in general. They have low accuracy values in average and also their performance varies with different datasets. For instance, K* performs as the best algorithm in list 9 while it performs the worst in list 7 by a large difference.

In precision results (Figure 9), ADTree and Decorate have highest results while in recall results J48 and MultiBoost AB(Figure 11) have the highest. Finally in f-score (Figure 13), ADTree, j48 and MultiBoost AB are again the best 3 performing algorithms. These 3 algorithms provide good classification results through different datasets. We have chosen ADTree to use in other

[3]http://www.cs.waikato.ac.nz/ml/weka/

experiments as being one of the best algorithm in our comparison.

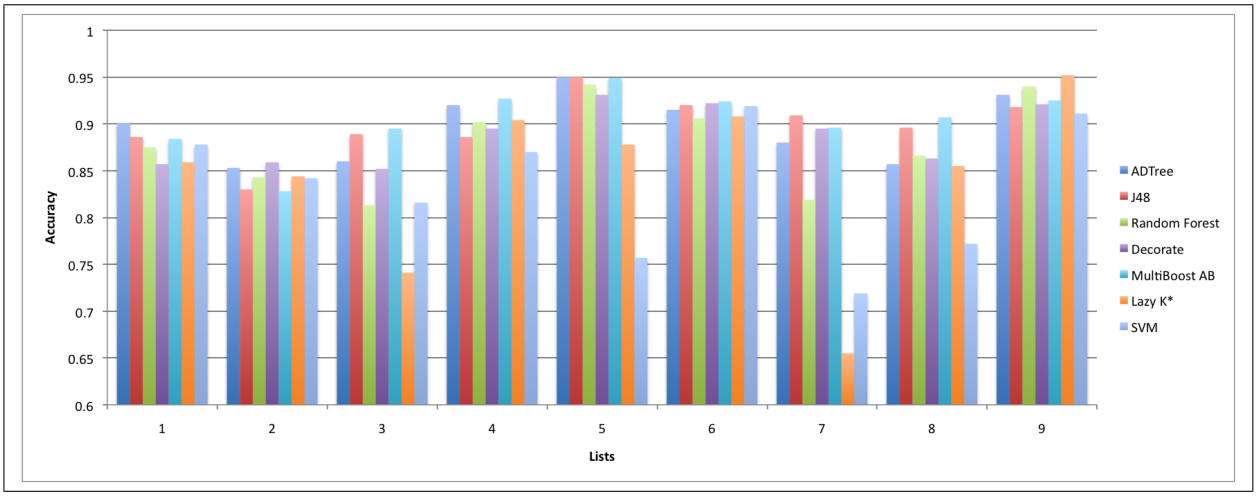

Figure 5: Accuracy results of the classification algorithms on the datasets of 9 lists

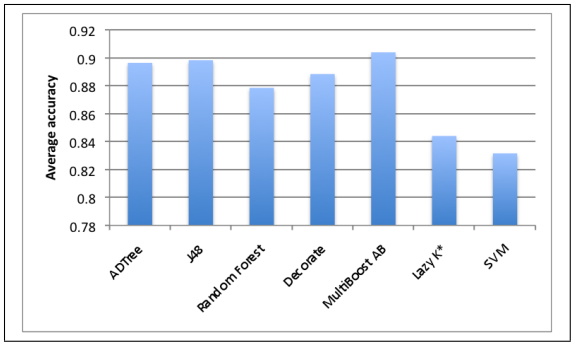

Figure 6: Average of accuracy results of 9 lists

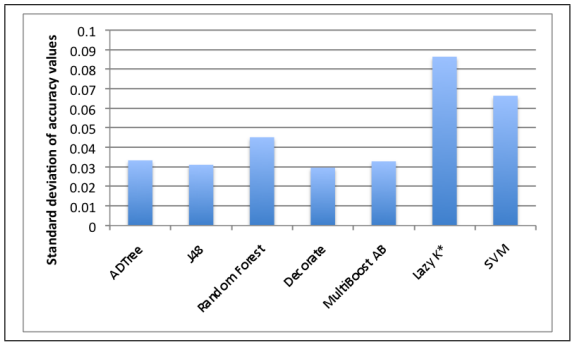

Figure 7: Standard deviation of accuracy results of 9 lists

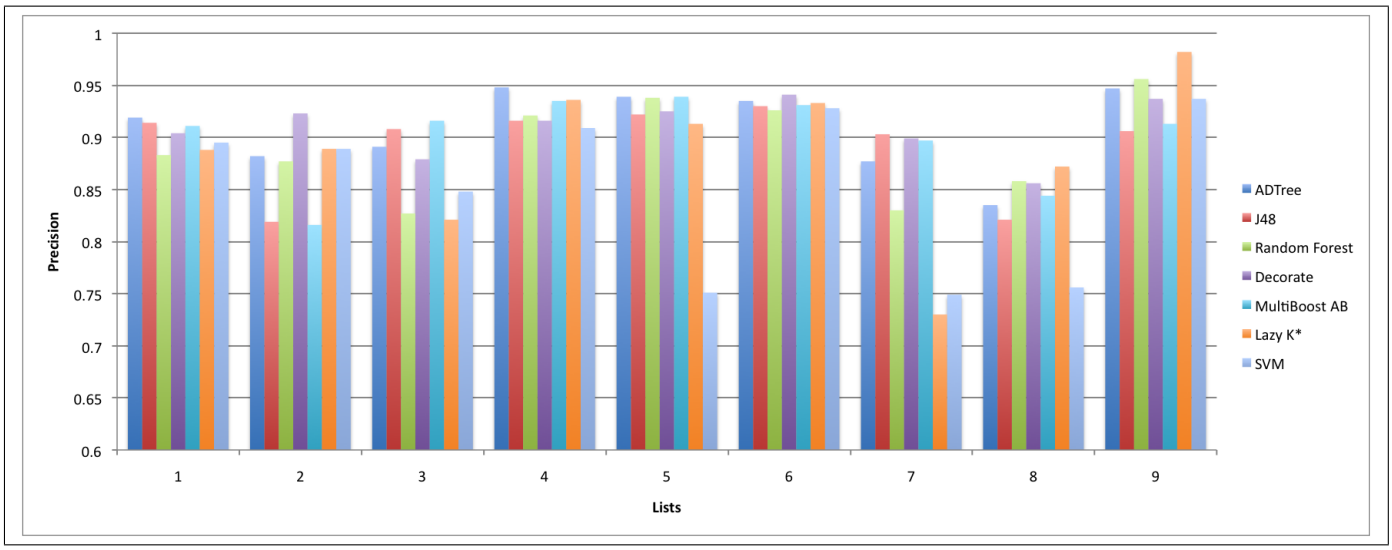

Figure 8: Precision results of the classification algorithms on the datasets of 9 lists

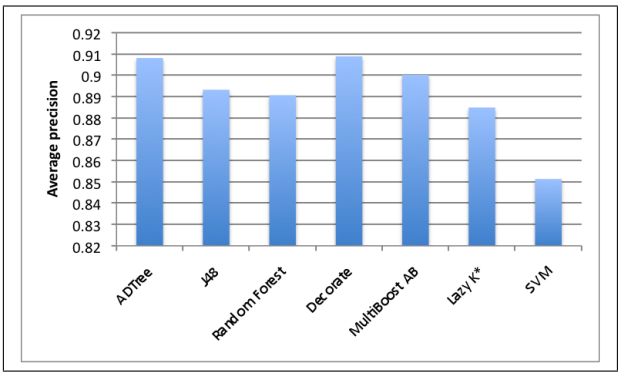

Figure 9: Average of precision results of 9 lists

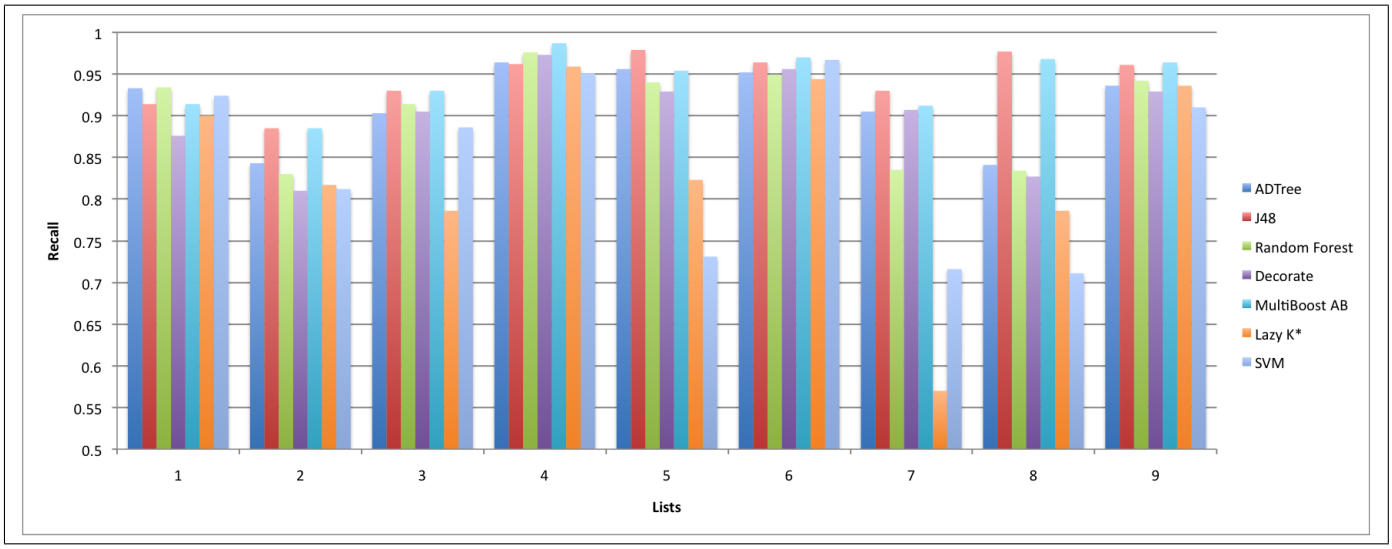

Figure 10: Recall results of the classification algorithms on the datasets of 9 lists

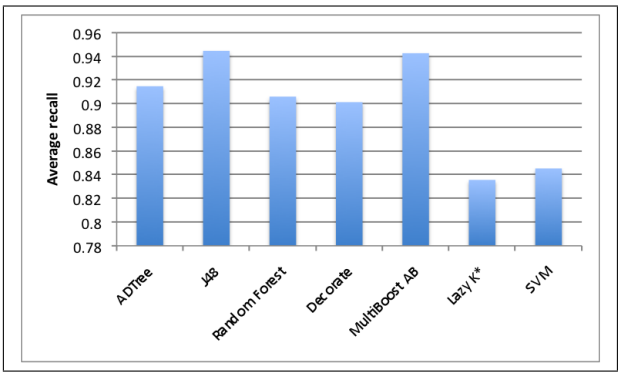

Figure 11: Average of recall results of 9 lists

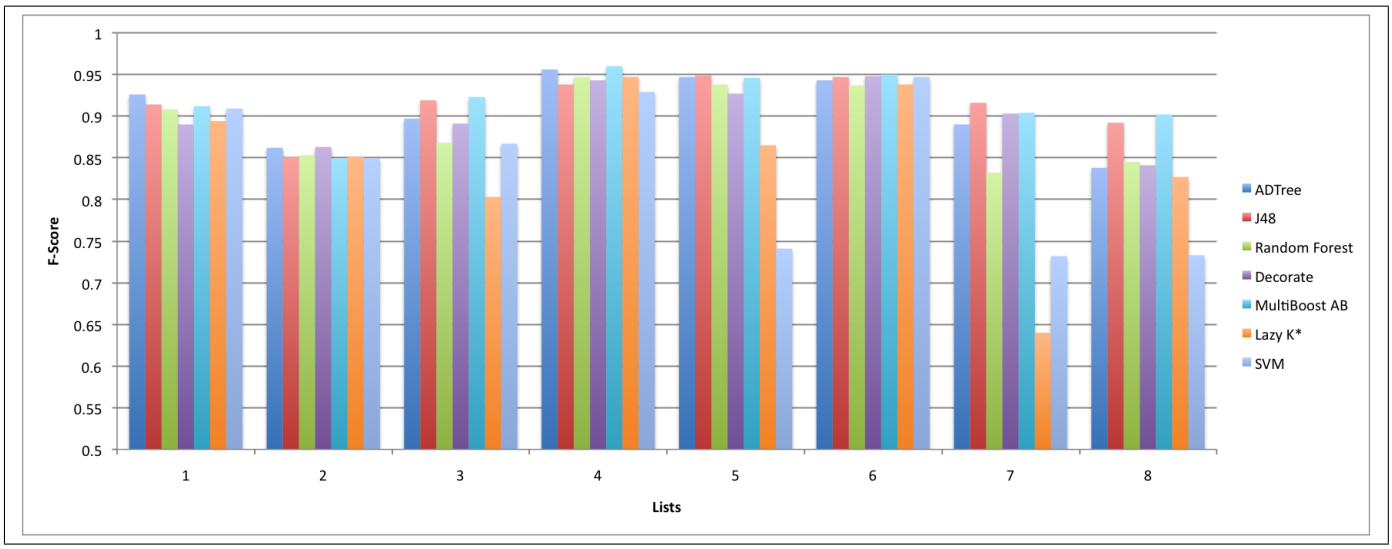

Figure 12: F-Score results of the classification algorithms on the datasets of 9 lists

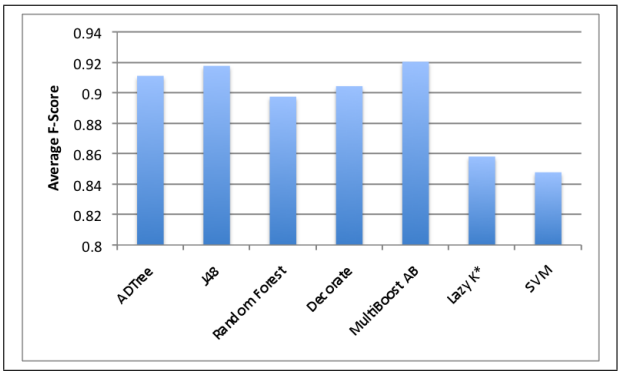

Figure 13: Average of F-Score results of 9 lists

5.4 Feature Evaluation and Comparison with Traditional Approach

In this section, the benefit from different features will be evaluated. Also we will compare two different text-based features: Text Relevance Score(TRS) and Bag-Of-Words(BOW). Bag-Of-Words approach is a traditional text classification method. In this method, tf-idf(term frequency-inverse document frequency) weight for each word of a post is calculated and used as a feature. Therefore BOW is basically a feature set rather than one feature. As a result, BOW approach leads to a high dimensional feature vector as each word in the document set represented by a feature. To deal with this high dimensional dataset, we use Support Vector Machines as the classification algorithm when we use the BOW approach as SVMs are well suited for this problem[26].

Next to text-based features, we evaluate 4 different groups of features: Author feature, social features, temporal features and link domain feature. We grouped some features together to easily combine different types of features. Social features contains 5 different features: followers count in list, friends count in list, total followers count, total friends count and TunkRank score of the user. Temporal features group contains two features: day of the week and hour of the day. Therefore we have two text-based features and 9 other features grouped into 4 groups. For each list, different feature groups combinations are evaluated in terms of accuracy, precision, recall and f-score.

Results for each list are shown in Tables 5 to 13. In terms of accuracy, feature combinations including *TRS* feature have the highest accuracy values in 7 out of 9 lists when compared to feature combinations including BOW.

An important result is that using all features may lead to a lower accuracy. The experiments show that *TRS + Author + Soc + Lnk + Temp* and *BOW + Author + Soc + Lnk + Temp* feature combinations are providing the highest accuracy values in only 3 lists. This means that the datasets of some lists are noisy for some features and using these features decreases the accuracy. This is expected as properties of each list is different and each feature works differently in each list. Using feature selection methods on the dataset can be beneficial to remove badly performing features before building the classifier. An overview of different feature selection for text classification is given in [47]. However, we are not using feature selection in our implementation, as finding the best performing features subset is costly in an online setting.

Our experiments show that, besides text-based features, other features can also be very discriminating. In result tables, it is shown how these features are performing alone and also in combination with other features. Link domain(Lnk) feature is one of the best performing features. Except *pcdinh/-database* and *lucaolivari/mysql-co* lists, this feature is the most discriminating feature among non text-based features. This feature is especially helpful in filtering automated messages, such as check-in messages automatically submitted to Twitter by location based services like foursquare[4].

Author feature and social features(Soc) are also discriminative features. In almost all lists, both features are improving the accuracy when they are used with text-based features compared to only using text-based features. In list *Disc Health/pregnancy-parenting* (Table 5), accuracy is increased by 12.3% and

[4]http://foursquare.com/

14.9% respectively. These results show that author information and social properties of authors can be good features in classifying posts in social networks. It is also notable that social features are sometimes performing better than author feature and the results are improved when they are used together. In lists *Disc Health/pregnancy-parenting*, *iFanboy/comicscreators* and *lucaolivari/mysql-co* social features are improving the results(*Author* vs. *Author+Soc*). This result is interesting due to the fact that social features are also properties of a user which is indicated by the author feature. The improvement comes from the fact that with these additional properties of the user, classifier can better learn if a user is more likely to submit a relevant post. For instance, classifier can learn that authors which have more than a certain number of followers are very likely to submit relevant posts. Suppose a post is going to be evaluated by the classifier but in the training set of the classifier there were no posts of the author that submitted that post. In this case, author feature won't provide any information. However, the classifier may have learnt from the set of authors in the training set that which social features indicates if the author is more likely to submit relevant posts. When author feature and social features are used alone, author feature works better than social features. The reason is that the author feature is more granular. This feature can discriminate individual authors whereas social features tend to group users(e.g., users with number of followers $<$ k and users with number of followers $\geq$ k). Therefore author feature works well in case there is enough amount of posts from different authors in training data.

Temporal features(Temp) are not very discriminative when they are used alone as expected. However when they are used together with author feature or social features, they are useful. In lists *Disc Health/pregnancy-parenting*, *huffingtonpost/ apple-news*, *lucaolivari/mysql-co*, it can be observed that temporal features increases the accuracy of author and social features. The increase show that users may be more likely to post relevant content in some periods, and the time and the date of a post can be a good indicator of its relevance. It is also interesting that temporal features can improve the results of social features. This might suggest that there can be patterns shared by a group of users, for instance submitting relevant content mostly during work hours or only in weekdays. We will evaluate social and temporal features separately to have a better understanding of these features in the following sections.

In summary, although the text-based features are often the most discriminating ones, other features consisting of authorship, social features, link domain and temporal features are also discriminating on our dataset and greatly improve the performance of the classification task when used next to text-based features. The experiments also indicates that each feature can perform differently in each list. While a feature can be very discriminating in a list, it can perform poorly in one other. This result supports the claim that each list should be classified separately. Furthermore, results also indicate that removing poorly performing features can improve the performance of the classifier.

Features	Accuracy	Precision	Recall	F-Score
Author	.855	.895	.883	.889
Author+Soc	.869	.906	.895	.9
Author+Temp	.911	.917	.952	.934
Soc	.88	.913	.903	.908
Temp	.741	.795	.819	.806
Soc+Temp	.933	**.942**	**.957**	**.949**
Lnk	.709	.719	.917	.806
Author+Soc+Lnk+Temp	**.919**	.935	.943	.939
TRS	.659	.722	.784	.752
TRS+Author	.782	.828	.845	.836
TRS+Soc	.808	.855	.853	.854
TRS+Lnk	.655	.729	.759	.743
TRS+Temp	.715	.763	.822	.792
TRS+Author+Soc	.8	.851	.845	.848
TRS+Author+Soc+Lnk+Temp	.901	.919	.933	.926
BOW	.818	.821	.926	.87
BOW+Author	.883	.911	.912	.911
BOW+Soc	.891	.908	.929	.918
BOW+Lnk	.808	.805	.936	.865
BOW+Temp	.832	.829	.938	.88
BOW+Author+Soc	.89	.904	.933	.918
BOW+Author+Soc+Lnk+Temp	.915	.929	.943	.936

Table 5: List Disc_Health/pregnancy-parenting

Features	Accuracy	Precision	Recall	F-Score
Author	.792	.89	.709	.789
Author+Soc	.822	.888	.771	.825
Author+Temp	.76	.817	.724	.767
Soc	.822	.885	.776	.827
Temp	.558	.597	.596	.595
Soc+Temp	.82	.868	.791	.828
Lnk	.841	**.896**	.801	.846
Author+Soc+Lnk+Temp	.834	.868	.822	.844
TRS	.82	.811	**.875**	.842
TRS+Author	.853	.864	.868	.866
TRS+Soc	**.858**	.884	.853	**.868**
TRS+Lnk	.846	.879	.834	.856
TRS+Temp	.816	.821	.849	.835
TRS+Author+Soc	.853	.874	.854	.864
TRS+Author+Soc+Lnk+Temp	.853	.882	.843	.862
BOW	.831	.877	.804	.839
BOW+Author	.831	.875	.806	.839
BOW+Soc	.832	.879	.803	.839
BOW+Lnk	.832	.878	.803	.839
BOW+Temp	.823	.865	.802	.832
BOW+Author+Soc	.83	.874	.806	.838
BOW+Author+Soc+Lnk+Temp	.827	.863	.813	.837

Table 6: List IndieFlix/film-people-to-follow

Features	Accuracy	Precision	Recall	F-Score
Author	.792	.787	**.947**	.86
Author+Soc	.762	.809	.846	.827
Author+Temp	.711	.745	.868	.801
Soc	.745	.81	.812	.811
Temp	.574	.658	.761	.705
Soc+Temp	.703	.768	.8	.784
Lnk	.815	.818	.932	.871
Author+Soc+Lnk+Temp	.757	.814	.828	.821
TRS	.852	.895	.884	.889
TRS+Author	**.875**	**.904**	.912	**.908**
TRS+Soc	.844	.877	.893	.885
TRS+Lnk	.848	.888	.886	.887
TRS+Temp	.844	.886	.881	.883
TRS+Author+Soc	.862	.896	.899	.897
TRS+Author+Soc+Lnk+Temp	.86	.891	.903	.897
BOW	.768	.806	.864	.834
BOW+Author	.821	.845	.899	.871
BOW+Soc	.776	.82	.855	.837
BOW+Lnk	.773	.795	.892	.841
BOW+Temp	.73	.779	.835	.806
BOW+Author+Soc	.818	.848	.889	.868
BOW+Author+Soc+Lnk+Temp	.785	.811	.888	.848

Table 7: List RosieEmery/ecotweets

Features	Accuracy	Precision	Recall	F-Score
Author	.878	.896	.977	.935
Author+Soc	.872	.896	.97	.931
Author+Temp	.854	.903	.938	.92
Soc	.876	.896	.975	.934
Temp	.866	.895	.964	.928
Soc+Temp	.871	.921	.937	.929
Lnk	.902	.926	.968	.946
Author+Soc+Lnk+Temp	.862	.915	.934	.924
TRS	.925	.962	.954	.958
TRS+Author	**.953**	**.967**	.981	**.974**
TRS+Soc	.942	.965	.97	.968
TRS+Lnk	.935	.95	.979	.964
TRS+Temp	.928	.955	.966	.96
TRS+Author+Soc	.948	.966	.977	.971
TRS+Author+Soc+Lnk+Temp	.92	.948	.964	.956
BOW	.912	.912	**.999**	.954
BOW+Author	.93	.937	.989	.962
BOW+Soc	.905	.917	.983	.949
BOW+Lnk	.927	.927	.998	.961
BOW+Temp	.902	.92	.976	.947
BOW+Author+Soc	.915	.931	.978	.954
BOW+Author+Soc+Lnk+Temp	.928	.933	.991	.961

Table 8: List WSJMarkets/markets

Features	Accuracy	Precision	Recall	F-Score
Author	.695	.722	.585	.646
Author+Soc	.672	.663	.635	.648
Author+Temp	.627	.609	.61	.608
Soc	.671	.659	.642	.65
Temp	.578	.549	.629	.585
Soc+Temp	.693	.675	.685	.68
Lnk	.737	.72	.729	.725
Author+Soc+Lnk+Temp	.699	.681	.692	.685
TRS	.922	.902	.938	.919
TRS+Author	.938	.937	.931	.934
TRS+Soc	.935	.924	.94	.932
TRS+Lnk	**.95**	.939	**.956**	**.947**
TRS+Temp	.941	.937	.938	.937
TRS+Author+Soc	.939	.937	.933	.935
TRS+Author+Soc+Lnk+Temp	**.95**	.939	**.956**	**.947**
BOW	.846	.95	.712	.814
BOW+Author	.868	.961	.754	.845
BOW+Soc	.847	.943	.721	.817
BOW+Lnk	.836	.943	.696	.801
BOW+Temp	.841	**.968**	.688	.804
BOW+Author+Soc	.874	.959	.769	.853
BOW+Author+Soc+Lnk+Temp	.845	.92	.738	.818

Table 9: List edward_ribeiro/dbpeople

Features	Accuracy	Precision	Recall	F-Score
Author	.791	.844	.882	.863
Author+Soc	.782	.835	.881	.857
Author+Temp	.809	.853	.898	.875
Soc	.782	.831	.888	.858
Temp	.721	.762	.909	.829
Soc+Temp	.803	.857	.883	.87
Lnk	.892	.906	.954	.929
Author+Soc+Lnk+Temp	.89	.913	.942	.928
TRS	.913	.928	.957	.942
TRS+Author	.928	.936	**.97**	.952
TRS+Soc	.922	.943	.953	.948
TRS+Lnk	.911	.928	.955	.941
TRS+Temp	.916	.933	.956	.944
TRS+Author+Soc	.926	.944	.956	.95
TRS+Author+Soc+Lnk+Temp	.915	.935	.952	.943
BOW	.929	.958	.946	.952
BOW+Author	.939	**.968**	.95	**.959**
BOW+Soc	.928	.961	.941	.951
BOW+Lnk	.931	.955	.952	.953
BOW+Temp	**.941**	.964	.957	.96
BOW+Author+Soc	.938	**.968**	.949	.958
BOW+Author+Soc+Lnk+Temp	**.941**	.959	.962	.96

Table 10: List huffingtonpost/apple-news

Features	Accuracy	Precision	Recall	F-Score
Author	.462	.498	.572	.531
Author+Soc	.494	.53	.523	.526
Author+Temp	.472	.509	.519	.513
Soc	.495	.53	.523	.525
Temp	.501	.534	.56	.546
Soc+Temp	.46	.499	.493	.495
Lnk	.732	.856	.605	.709
Author+Soc+Lnk+Temp	.566	.595	.605	.599
TRS	.841	.852	.853	.852
TRS+Author	**.902**	**.93**	.886	.907
TRS+Soc	.884	.893	.891	.892
TRS+Lnk	**.902**	.909	**.909**	**.909**
TRS+Temp	.889	.909	.881	.895
TRS+Author+Soc	.886	.896	.893	.894
TRS+Author+Soc+Lnk+Temp	.88	.877	.905	.89
BOW	.781	.91	.658	.764
BOW+Author	.762	.887	.64	.743
BOW+Soc	.776	.917	.642	.755
BOW+Lnk	.794	.873	.721	.79
BOW+Temp	.77	.842	.705	.767
BOW+Author+Soc	.786	.906	.672	.771
BOW+Author+Soc+Lnk+Temp	.8	.837	.781	.807

Table 11: List iFanboy/comicscreators

Features	Accuracy	Precision	Recall	F-Score
Author	.694	.684	.57	.621
Author+Soc	.702	.7	.57	.628
Author+Temp	.75	.758	.636	.691
Soc	.704	.701	.575	.631
Temp	.586	.536	.473	.499
Soc+Temp	.748	.727	.686	.705
Lnk	.724	.655	.786	.715
Author+Soc+Lnk+Temp	.722	.692	.666	.678
TRS	.847	.826	.827	.826
TRS+Author	.884	.854	.889	.871
TRS+Soc	.87	.848	.859	.853
TRS+Lnk	**.905**	.859	**.939**	**.897**
TRS+Temp	.862	.827	.868	.847
TRS+Author+Soc	.884	.854	.889	.871
TRS+Author+Soc+Lnk+Temp	.857	.835	.841	.838
BOW	.866	**.903**	.78	.837
BOW+Author	.836	.826	.795	.81
BOW+Soc	.864	.896	.782	.835
BOW+Lnk	.87	.877	.82	.847
BOW+Temp	.866	.861	.83	.845
BOW+Author+Soc	.852	.837	.825	.831
BOW+Author+Soc+Lnk+Temp	.888	.882	.861	.871

Table 12: List lucaolivari/mysql-co

Features	Accuracy	Precision	Recall	F-Score
Author	.882	.923	.873	.897
Author+Soc	.872	.906	.875	.89
Author+Temp	.852	.875	.875	.875
Soc	.865	.896	.873	.884
Temp	.502	.573	.61	.59
Soc+Temp	.849	.874	.869	.872
Lnk	.78	.75	.941	.835
Author+Soc+Lnk+Temp	.877	.905	.885	.894
TRS	.899	.9	.932	.916
TRS+Author	.922	.921	**.949**	.935
TRS+Soc	.928	.938	.941	.939
TRS+Lnk	.914	.917	.939	.928
TRS+Temp	.916	.917	.942	.93
TRS+Author+Soc	.927	.936	.941	.938
TRS+Author+Soc+Lnk+Temp	**.931**	.947	.936	**.941**
BOW	.92	.961	.902	.93
BOW+Author	.921	**.964**	.9	.931
BOW+Soc	.92	.961	.902	.93
BOW+Lnk	.915	.931	.925	.928
BOW+Temp	.911	.941	.907	.923
BOW+Author+Soc	.915	.962	.892	.925
BOW+Author+Soc+Lnk+Temp	.917	.952	.905	.928

Table 13: List pcdinh/database

5.5 Comparison of Social Features

In the previous experiment, the features are grouped under social and temporal feature groups. In this section, different social features we used will be compared. For this purpose, we use *Information Gain* metric. Information gain measures how much more organized the class values become when we divide them up using a given feature[14]. This value is between 0 and 1, where 1 means the highest information gain. We have 5 different social features: number of followers of the author who are also in the list, number of friends who are also in the list, number of total followers on Twitter, number of total friends on Twitter and TunkRank of the author. The results are displayed in Figure 14 and Figure 15. While Figure 14 shows the results for each list separately, Figure 15 shows the average information gain of each feature in 9 lists.

The results indicates that social features are performing differently in each list. While social features are very discriminating in some lists(e.g., lists 1, 2 and 9), in some lists they provide almost no information gain(e.g., lists 4, 5 and 7). This means that lists have different properties: while in some lists more popular users are more likely to submit related content, in others this is not the case.

Out of the 5 features, TunkRank is the most discriminative feature. When the other features are compared, it can be observed that number of followers is a better indication than number of friends that author is more likely to submit relevant content(# followers in list vs # friends in list and # total followers vs # total friends).

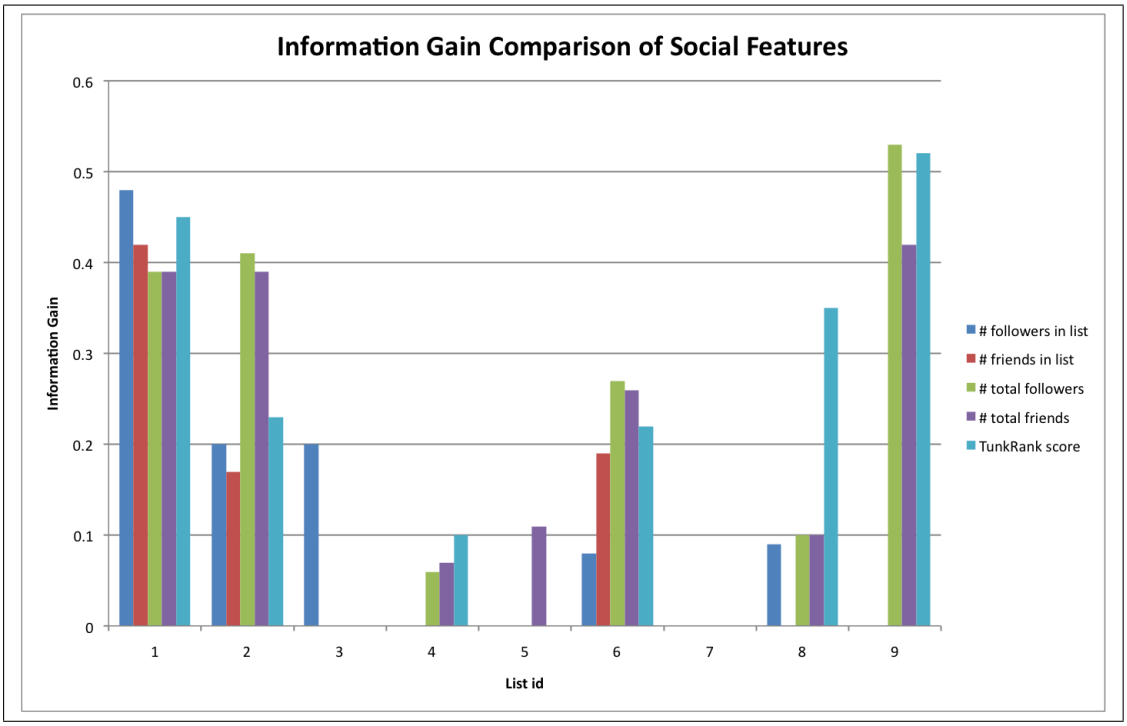

Figure 14: Comparison of information gain of social features for each list

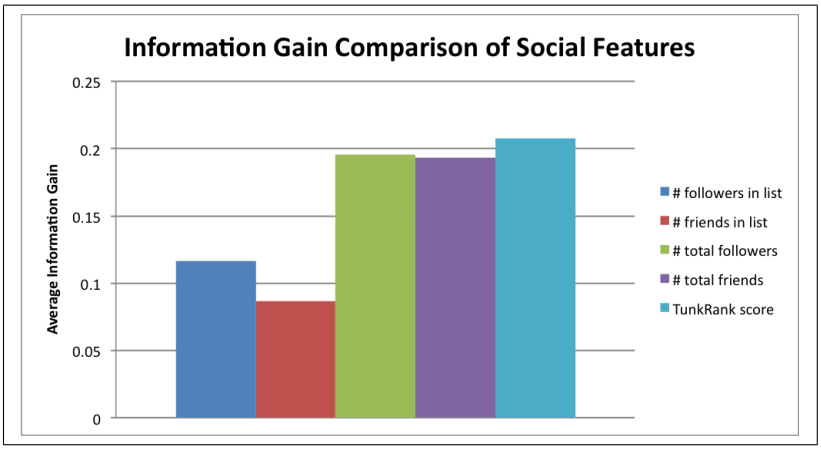

Figure 15: Comparison of information gain of social features (Average of all lists)

5.6 Comparison of Temporal Features

We have two different features in temporal features group: day of the week and hour of the day. The results of the comparison is shown in Figures 16 and 17. In all lists, hour of the day provides higher information gain then day of the week feature. In average(Figure 17), hour of the day feature provides twice the information gain then the day of the week feature brings.

In figures 18 and 19, the ratio of relevant posts to the all posts with respect to hour of the day and day of the week are plotted. For this graphs, all labeled dataset is used. During the week, the graph is more smoother, while ratio of relevant posts are low at the beginning and at the end of the week, the ratio gets high in wednesday and thursday. The graph indicates a pattern throughout the week. On the other hand, the relevant posts ratio changes sharply during the day and there is not a clear pattern. However, during some intervals(e.g., 2AM-3AM and 4AM-5AM) the ratio of relevant posts to all posts goes up to 0.8. In this case, the time of the day feature becomes highly discriminative. Therefore these graphs explain why the time of the day feature has higher information gain.

Temporal features were mostly the weakest features in our experiments. However, the way they were retrieved can also have an effect in the result. We calculated the time of the post in local time by retrieving the GMT of the post and adjusting it to the time zone of the author of the post. However, users in Twitter set their time zone manually and this may be set wrong or not set at all by the user. Furthermore when the author travels to an another timezone, local time will be inaccurate again. Therefore unlike other features, temporal features may not be retrieved accurately. Although it is possible in Twitter to add location information to messages, currently it is used by very few users. In the future, location data can be used as a more accurate retrieval method if it is widely adopted.

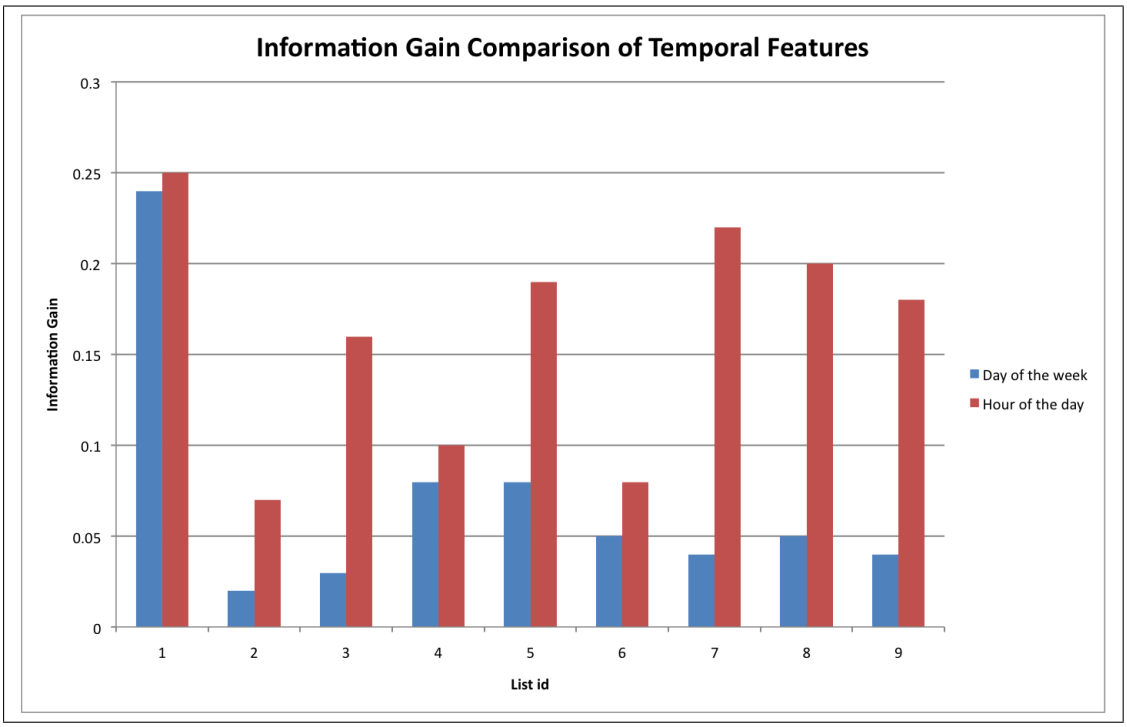

Figure 16: Comparison of information gain of temporal features for each list

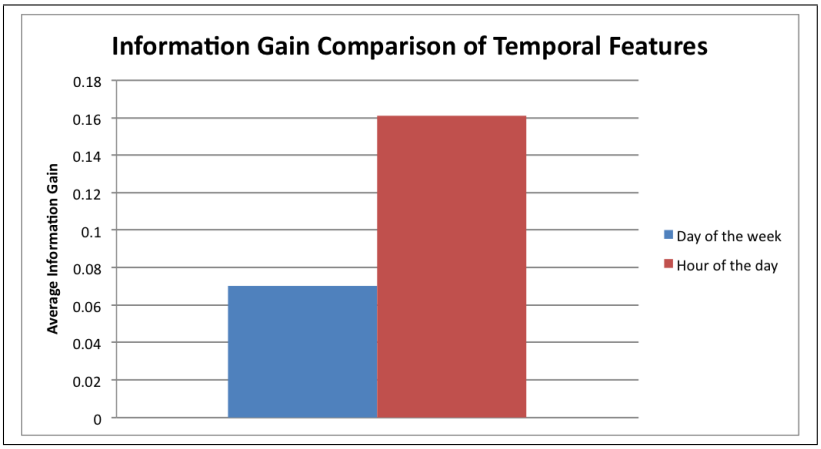

Figure 17: Comparison of information gain of temporal features (Average of all lists)

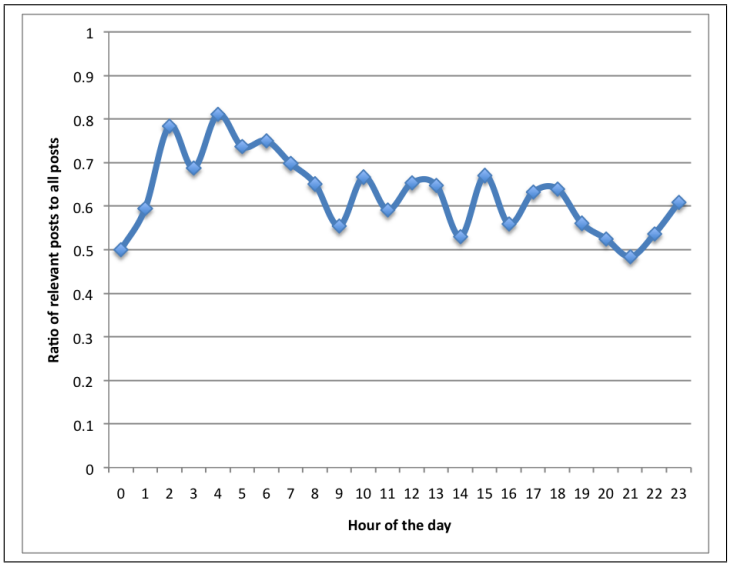

Figure 18: Ratio of relevant posts to all posts during the day

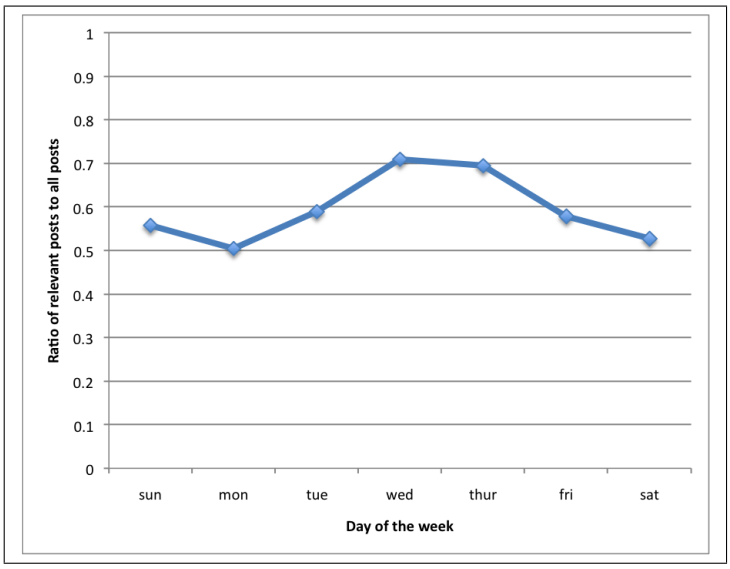

Figure 19: Ratio of relevant posts to all posts during the week

5.7 Learning Curves

In an online filtering system where the user explicitly gives feedback, it should be expected that the user will give very few feedbacks. Therefore the system should perform well in filtering task even with very few number of feedbacks, otherwise the user would stop using the system before it starts performing decently. In this experiment, the learning performance of our classification system will be evaluated by displaying accuracy of the classification by increasing the number of training instances.

For this experiment, each list's dataset is divided into 10 folds. While 9 folds are used for training, 1 fold is used for testing. The test is run 10 times and in each run, a different fold is used for testing. Learning curve is produced by using a portion of the training data, starting from 5 instances and incrementing by 5 instances until all data is used. For each training data size, accuracy is calculated by the average of 10 runs.

In the experiment, two approaches are compared: TRS and BOW as text-based features. In Figure 20, learning curves of only text-based features are compared. In Figure 21, the rest of the features are included and *TRS + Author + Soc + Lnk + Temp* feature set is compared against *BOW + Author + Soc + Lnk + Temp* feature set. For the tests with *TRS* feature, we use ADTree and for *BOW* feature, we use Support Vector Machine as classifier due to high dimensionality of the data.

In Figure 20, we observe the learning performance with two different text-based features. The charts show that generally *TRS* approach provides a straight learning curve whereas the accuracy of *BOW* approach improves over time. This result shows that even with very few training instances, *TRS* approach provides a decent performance. On the other hand, *BOW* approach gets better after some amount of training data. This is due to the fact that *TRS* is one nominal feature and classifier only needs to learn the threshold where posts with values above is relevant. On the other hand *BOW* is a set of features where each term in the dataset is represented by a feature. In 4 out of 9 lists, *TRS* performs better in every training dataset size. In *RosieEmery/ecotweets*, the accuracy of *TRS* is dramatically higher than *BOW* approach. On the other hand, in list *Disc Health/pregnancy-parenting*, TRS approach is performing poorly compared to BOW. The reason is that *TRS* is a frequency based approach. In case there is a frequent topic in the list which is not frequent in the whole Twitter, then posts related to this topic gets higher text relevance score. But in case this topic is not related to the topic the user is interested in, *TRS* approach can perform worse. In this list, topic *health* is also frequent next to *pregnancy* and *parenting* topics mainly due to its proximity to these topics. Posts related to *health* are marked as not relevant by judges but they still get high text relevance score by our scoring function. This situation where a non-relevant topic is highly frequent is only observed in this list.

In Figure 21, the learning performance of *TRS + Author + Soc + Lnk + Temp* feature set and *BOW + Author + Soc + Lnk + Temp* feature set are compared. In *Disc Health/pregnancy-parenting*, now two approaches perform similar as the added features are very discriminative. Also in *RosieEmery/ecotweets*, the difference got smaller. In this experiment, *TRS* outperforms *BOW* in 7 lists in learning performance aspect. In 5 lists, accuracy of TRS approach is around 90% even with a training dataset of 5 instances.

Learning curves show clearly that *TRS* approach learns faster. Furthermore, the experiment shows that in some lists even 5 instances can be enough to provide high accuracy values. Therefore we can conclude that *TRS* approach can provide the learning performance that an online filtering system requires.

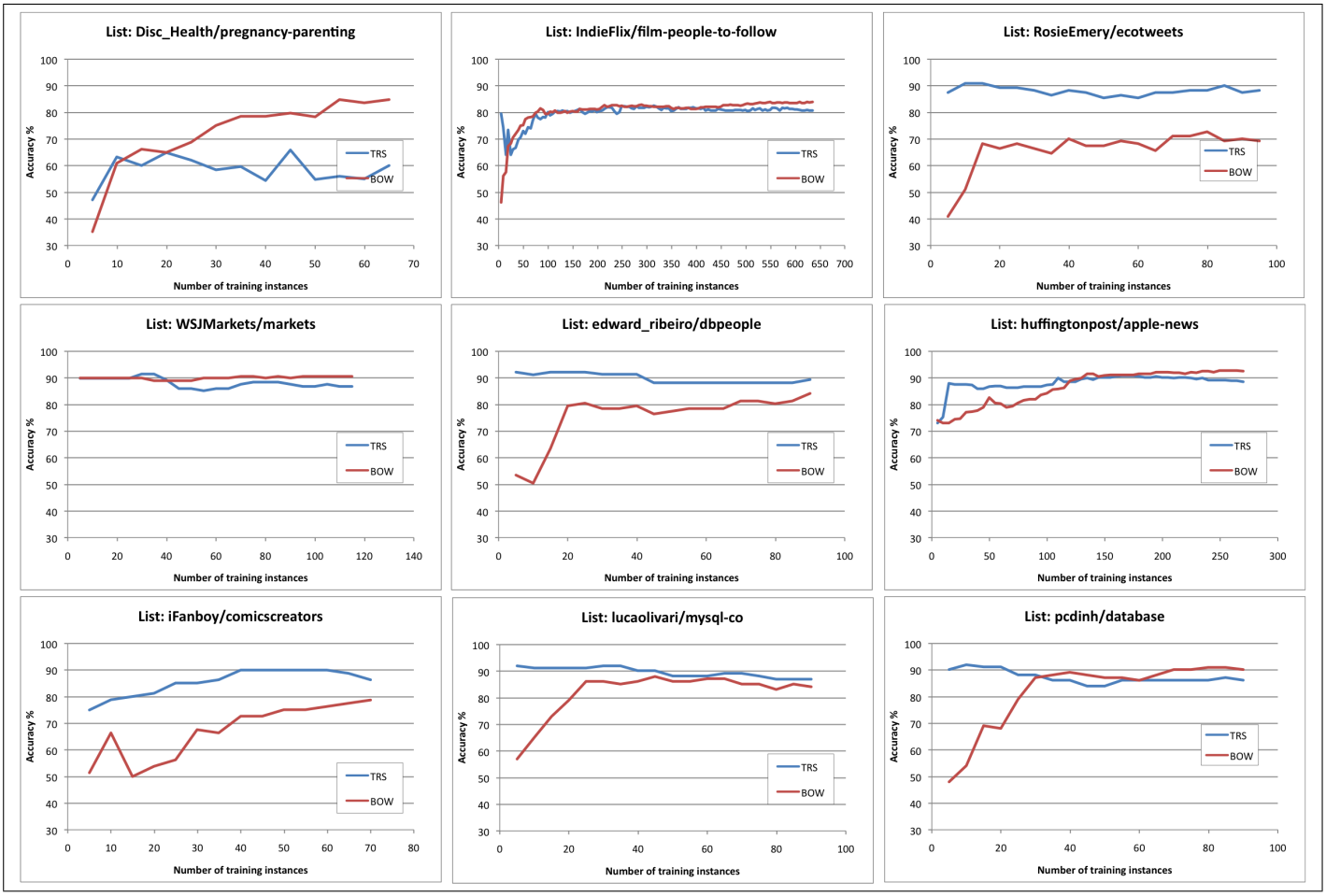

Figure 20: Learning curves for 9 lists. Only BOW and TRS features are used

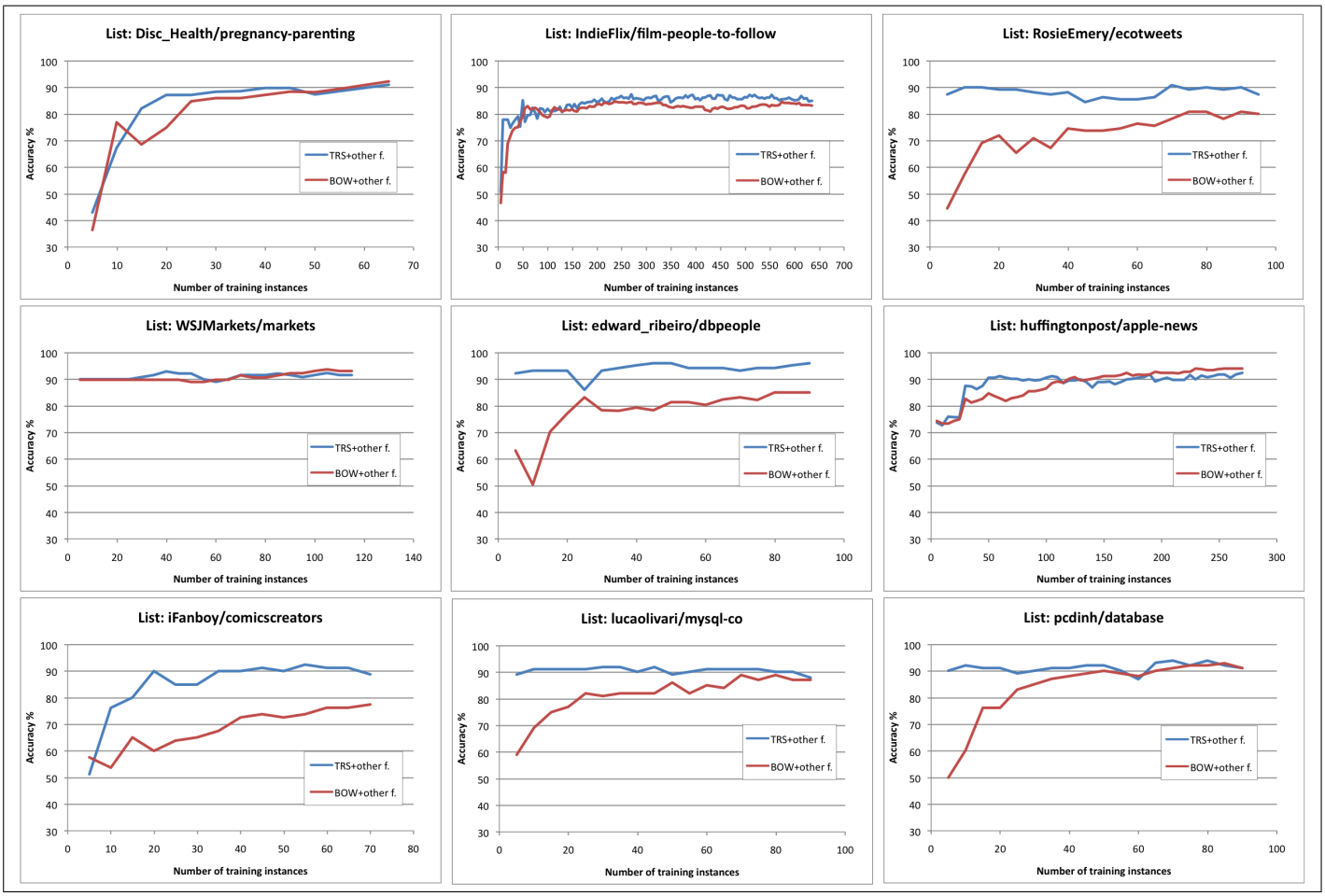

Figure 21: Learning curves for 9 lists. BOW and TRS are used with other features

5.8 Using Link Contents

The limitation of text sizes on Twitter is a drawback for text classification, therefore we use an external API to fetch contents of the URLs which are included in posts and add this text to the original text of the post. This approach may work in two ways. The text of the posts are very short but this may also mean that they are very condense. Webpages however include longer text without length limit. Using only the text of the post can be better if the text is mostly consist of keywords. On the other hand, in most posts, the text of the post and the link are complementary. To get the topic or the meaning of a post, the user should mostly follow the link. Therefore adding the contents of an URL to text classification can have positive or negative effect.

In this experiment, we show how adding URL contents affects the text classification. The experiment is run using TRS feature and ADTree classifier. Accuracy values are shown in Figure 22. Results indicate clearly that adding link contents to text improves accuracy dramatically, more than 15% in some cases.

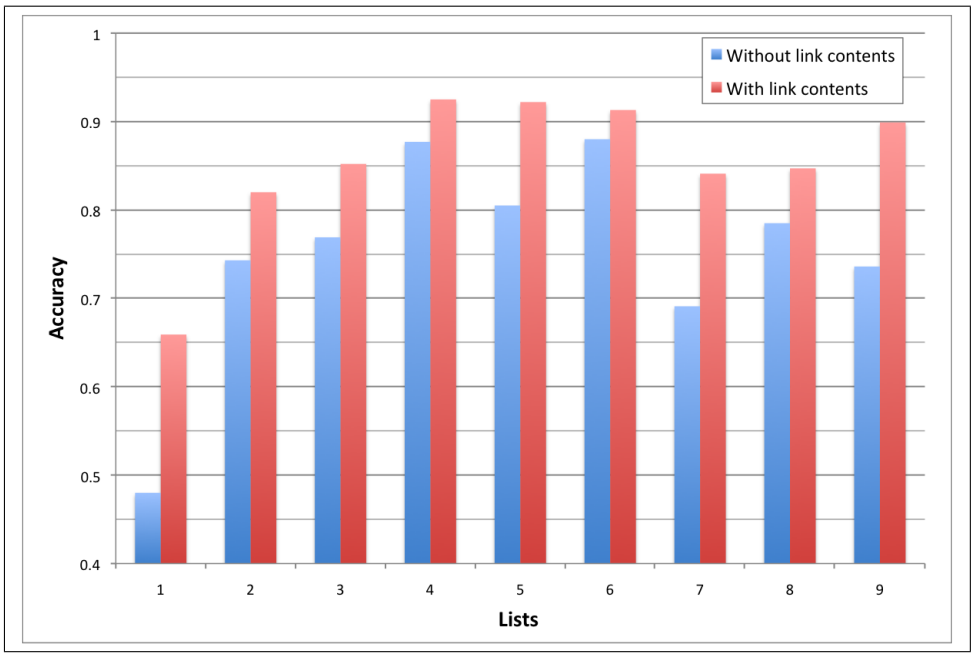

Figure 22: Evaluation of using link contents in text relevance score

5.9 Comparison of Computation Time to Build Classifiers

In this section, we compare the computation time needed to build classifiers for two approaches. ADTree decision tree is built with *TRS + Author + Soc + Lnk + Temp* feature set and Support Vector Machine is built with *BOW + Author + Soc + Lnk + Temp* feature set. We use ADTree and SVM implementations of Weka library[24]. The whole labeled dataset is used for this experiment, and computation time in milliseconds is displayed with respect to increasing number

of instances in the training dataset(Figure 23). Both classification algorithms are not incremental, i.e., for each size of dataset the classifiers are built from scratch. The performance may change in case other implementations are used or incremental versions of the classifiers are used. Therefore the results are shown to give an overall idea about the performance of two approaches.

The figure shows that ADTree+TRS approach is built a couple of times faster than the SVM+BOW approach. This is mainly due to high number of features in BOW approach while in TRS approach the number of features is only 10. Also as the number of instances increases, number of unique words seen in the dataset increases, which makes the feature vector of BOW approach larger. While TRS approach's built time increases only due to number of instances, BOW approach's built time increases both due to increasing number of features and number of instances. Therefore we observe that BOW approach's built time increases in a higher rate. The result indicates that with the advantage of having lower number of features, TRS approach has lower complexity and more suitable for online systems.

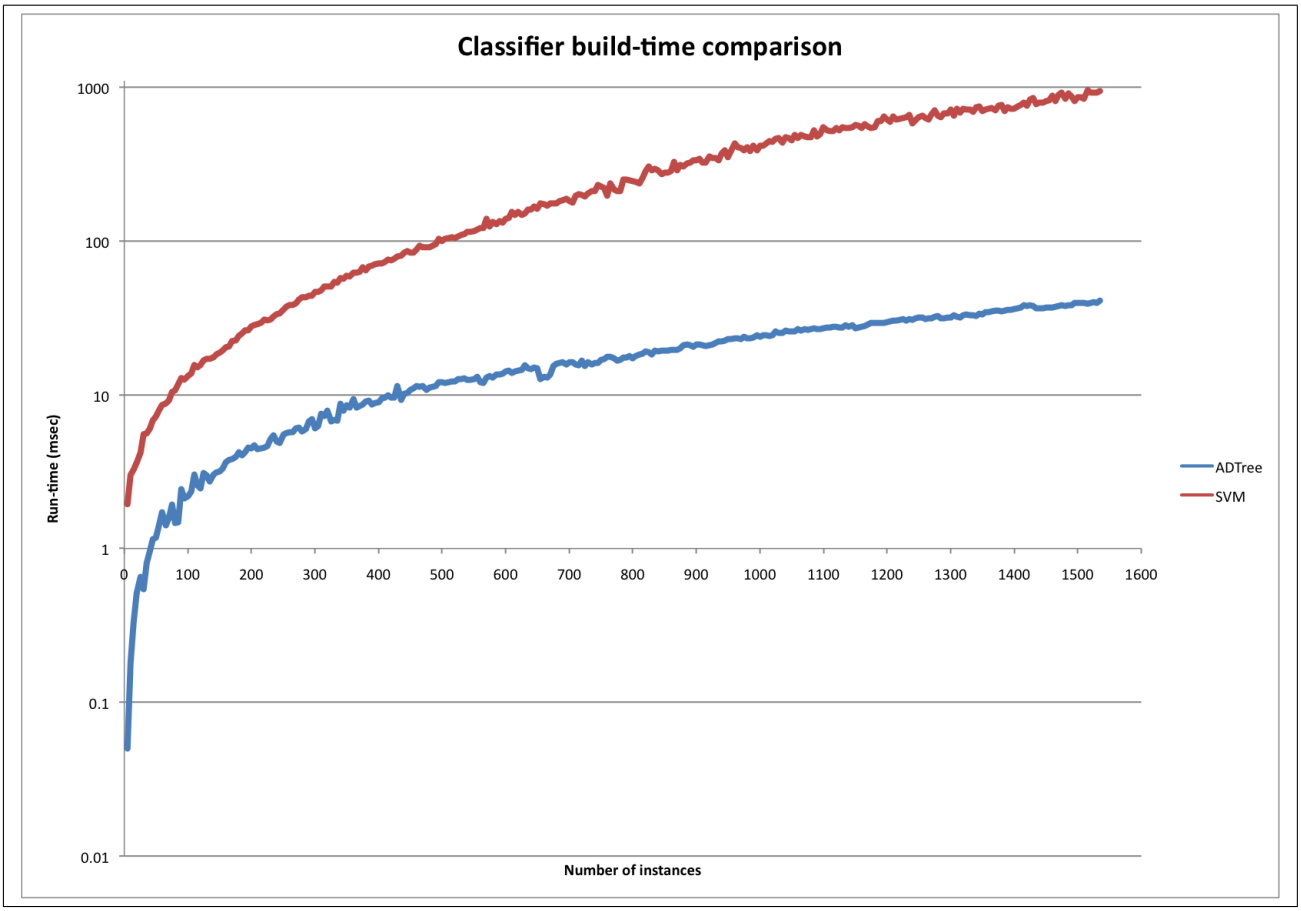

Figure 23: Computation time needed to build classifiers with respect to increasing number of instances.

6 Conclusion and Future Work

In this work, we have developed an information filtering system for Twitter. We focused on list feeds which are collection of messages from a manually created list of users and tend to be more focused on specific topics. Our system removes irrelevant content from these feeds and provide clean information sources for these topics. Although we focused on Twitter, our approach can be also applied to other micro-blogging services.

As a contribution, we proposed a scoring function that uses the word frequencies in lists and word frequencies in Twitter community as reference for scoring the relevance of a post to a feed. In the classification section, we have shown on a group of selected list that this score can accurately identify words that are specific to each list and penalize common words.

We also used novel features extracted from different aspects of micro-blogging messages. We used our scoring function as a text-based feature to identify relevant posts in a feed. Additionally, we used authorship, social network features, temporal features and link domain feature to increase the accuracy of classifiers. The experiments have shown that each group of features improved the accuracy of classification. Especially text relevance score, authorship and link domain features are very discriminative in filtering out irrelevant messages. Social features are useful in improving authorship feature, as they extract properties of users that post relevant content. From social features, we found out that TunkRank score is the best performing one. This result indicates that instead of numbers of followers and friends, more complex scoring schemes such as TunkRank is better at identifying authoritative users.

We have also shown that links included in messages are good sources of information. We exploited links in two ways: we retrieved the contents of a referenced URL and merged it with message text, and also we extracted the domain of URLs as a separate feature. Experiments indicate that both methods are improving the accuracy of the system even though most of the messages don't contain links.

With the experiments on 9 selected lists from Twitter, we reached accuracies between 85% and 95%. Using only one text-based feature and a relatively small number feature list, the classifiers were able to learn faster, for some lists even 5 training instances were enough for good accuracy results. Furthermore, we have observed that each list has different properties and each feature works differently for each list, therefore feature selection might be useful in improving results.

Finally, an online prototype is developed for the filtering system. By tracking the stream of public messages on Twitter, the system is able to keep an up-to-date word frequency vector. In this way, the system dynamically score a post's textual relevance to a feed and react to changes.

As a possible improvement, classifiers used in the system can be replaced by incremental classifiers. We used classifiers which can't be updated with a new training instance due to unavailability of incremental ones. Therefore with each new feedback, classifiers needs to be built from scratch. However, as we don't expect to have a high number of feedback from the user and the system works well with limited number of feedback, building from scratch is also affordable.

In our prototype, all users can give feedback to any list. However, each user may have different preferences on which messages to be filtered out. Therefore the system can be extended to have different classifiers to each system. In

this scenario, one feed might have different classifiers for each user that has subscribed to it while there is one copy of feed data. Therefore for each user, separate feedback histories and classifiers should be kept.

Our system is developed to work on one machine. Therefore the system can only handle a limited amount of feeds and user requests. To be able to scale with large numbers of feeds registered in the system and users, the system should be extended to work on a cluster. In this case, it can be studied how to place feeds on different machines.

Bibliography

[1] Measuring semantic similarity between words using web search engines. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 757–766, New York, NY, USA, 2007. ACM.

[2] Alchemy api. http://www.alchemyapi.com/, July 2010.

[3] Apache tomcat. http://tomcat.apache.org/, August 2010.

[4] Jquery: The write less, do more, javascript library. http://www.jquery.com/, July 2010.

[5] Json. http://www.json.org, July 2010.

[6] Mumbai attacks: Twitter and flickr used to break news. http://www.telegraph.co.uk/news/worldnews/asia/india/3530640/Mumbai-attacks-Twitter-and-Flickr-used-to-break-news-Bombay-India.html, August 2010.

[7] New york plane crash: Twitter breaks the news, again. http://www.telegraph.co.uk/technology/twitter/4269765/New-York-plane-crash-Twitter-breaks-the-news-again.html, August 2010.

[8] Twitter. http://en.wikipedia.org/wiki/Twitter, August 2010.

[9] Twitter api. http://apiwiki.twitter.com/, July 2010.

[10] Albert Angel, Nick Koudas, Nikos Sarkas, and Divesh Srivastava. What's on the grapevine? In *SIGMOD '09: Proceedings of the 35th SIGMOD international conference on Management of data*, pages 1047–1050, New York, NY, USA, 2009. ACM.

[11] Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. Clustering short texts using wikipedia. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788, New York, NY, USA, 2007. ACM.

[12] Nilesh Bansal and Nick Koudas. Blogscope: a system for online analysis of high volume text streams. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 1410–1413. VLDB Endowment, 2007.

[13] Nick Bilton. Twitter needs more filters. http://bits.blogs.nytimes.com/2010/04/07/twitter-needs-more-filters/, August 2010.

[14] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009.

[15] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[16] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[17] Jilin Chen, Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and H." Chi. Short and tweet: Experiments on recommending content from information streams.

[18] John G. Cleary and Leonard E. Trigg. K*: An instance-based learner using an entropic distance measure. In *In Proceedings of the 12th International Conference on Machine Learning*, pages 108–114. Morgan Kaufmann, 1995.

[19] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. Towards recency ranking in web search. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20, New York, NY, USA, 2010. ACM.

[20] Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. Time is of the essence: improving recency ranking using twitter data. In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 331–340, New York, NY, USA, 2010. ACM.

[21] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 124–133, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[22] Daniel Gayo-Avello. Nepotistic relationships in twitter and their impact on rank prestige algorithms. 04 2010.

[23] Maxim Grinev, Maria Grineva, Alexander Boldakov, Leonid Novak, Andrey Syssoev, and Dmitry Lizorkin. Sifting micro-blogging stream for events of user interest. In *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 837–837, New York, NY, USA, 2009. ACM.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[25] Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65, New York, NY, USA, 2007. ACM.

[26] Thorsten Joachims, Thorsten Joachims, Fachbereich Informatik, Fachbereich Informatik, Fachbereich Informatik, Fachbereich Informatik, and Lehrstuhl" Viii. Text categorization with support vector machines: Learning with many relevant features. 1997.

[27] Dongwoo Kim, Yohan Jo, Il-Chul Moon, and Alice Oh. Analysis of twitter lists as a potential source for discovering latent characteristics of users. In *Workshop on Microblogging at the ACM Conference on Human Factors in Computer Systems. (CHI 2010)*, 2010.

[28] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *WWW '10: Proceedings of the 19th international conference on World wide web*, pages 591–600, New York, NY, USA, 2010. ACM.

[29] Prem Melville and Raymond J. Mooney. Constructing diverse classifier ensembles using artificial training examples. In *IJCAI'03: Proceedings of the 18th international joint conference on Artificial intelligence*, pages 505–510, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[30] Douglas W. Oard. The state of the art in text filtering. *UMUAI*, 7:141–178, 1997.

[31] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 91–100, New York, NY, USA, 2008. ACM.

[32] John C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.

[33] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[34] Courtney Corley Rada Mihalcea. Corpus-based and knowledge-based measures of text semantic similarity. *IN AAAI'06*, pages 775–780, 2006.

[35] Daniel Ramage, Susan Dumais, and Dan Liebling. Characterizing microblogs with topic models. 2010.

[36] Daniel Ramage, Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora.

[37] Microsoft Research. Twahpic. http://twahpic.cloudapp.net/About.aspx, August 2010.

[38] Jagan Sankaranarayanan, Hanan Samet, Benjamin E. Teitler, Michael D. Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *GIS '09: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 42–51, New York, NY, USA, 2009. ACM.

[39] Peter Schönhofen. Identifying document topics using the wikipedia category network. *Web Intelli. and Agent Sys.*, 7(2):195–207, 2009.

[40] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, 2002.

[41] D.A. Shamma, L. Kennedy, and E.F. Churchill. Tweetgeist: Can the twitter timeline reveal the structure of broadcast events? In *CSCW 2010*, 2010.

[42] David A. Shamma, Lyndon Kennedy, and Elizabeth F. Churchill. Tweet the debates: understanding community annotation of uncollected sources. In *WSM '09: Proceedings of the first SIGMM workshop on Social media*, pages 3–10, New York, NY, USA, 2009. ACM.

[43] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *SIGIR '10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842, New York, NY, USA, 2010. ACM.

[44] Daniel Tunkelang. http://thenoisychannel.com/2009/01/13/a-twitter-analog-to-pagerank/, August 2010.

[45] Geoffrey I. Webb. Multiboosting: A technique for combining boosting and wagging. *Mach. Learn.*, 40(2):159–196, 2000.

[46] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. Twitterrank: finding topic-sensitive influential twitterers. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 261–270, New York, NY, USA, 2010. ACM.

[47] Yiming Yang, Yiming Yang, and Jan O. Pedersen. A comparative study on feature selection in text categorization. pages 412–420, 1997.

[48] Dejin Zhao and Mary Beth Rosson. How and why people twitter: the role that micro-blogging plays in informal communication at work. In *GROUP '09: Proceedings of the ACM 2009 international conference on Supporting group work*, pages 243–252, New York, NY, USA, 2009. ACM.