Finding Microblog Posts of User Interest

by

Adam Roegiest

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2012

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

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

Abstract

Microblogging is an increasingly popular form of social media. One of the most popular microblogging services is Twitter. The number of messages posted to Twitter on a daily basis is extremely large. Accordingly, it becomes hard for users to sort through these messages and find ones that interest them. Twitter offers search mechanisms but they are relatively simple and accordingly the results can be lacklustre. Through participation in the 2011 Text Retrieval Conference's Microblog Track, this thesis examines real-time ad hoc search using standard information retrieval approaches without microblog or Twitter specific modifications. It was found that using pseudo-relevance feedback based upon a language model derived from Twitter posts, called tweets, in conjunction with standard ranking methods is able to perform competitively with advanced retrieval systems as well as microblog and Twitter specific retrieval systems. Furthermore, possible modifications both Twitter specific and otherwise are discussed that would potentially increase retrieval performance.

Twitter has also spawned an interesting phenomenon called hashtags. Hashtags are used by Twitter users to denote that their message belongs to a particular topic or conversation. Unfortunately, tweets have a 140 characters limit and accordingly all relevant hashtags cannot always be present in tweet. Thus, Twitter users cannot easily find tweets that do not contain hashtags they are interested in but should contain them. This problem is investigated in this thesis in three ways using learning methods. First, learning methods are used to determine if it is possible to discriminate between two topically different sets of a tweets. This thesis then investigates whether or not it is possible for tweets without a particular hashtag, but discusses the same topic as the hashtag, to be separated from random tweets. This case mimics the real world scenario of users having to sift through random tweets to find tweets that are related to a topic they are interested in. This investigation is performed by removing hashtags from tweets and attempting to distinguish those tweets from random tweets. Finally, this thesis investigates whether or not topically similar tweets can also be distinguished based upon a sub-topic. This was investigated in almost an identical manner to the second case.

This thesis finds that topically distinct tweets can be distinguished but more importantly that standard learning methods are able to determine that a tweet with a hashtag removed should have that hashtag. In addition, this hashtag reconstruction can be performed well with very few examples of what a tweet with and without the particular hashtag should look like. This provides evidence that it may be possible to separate tweets a user may be interested from random tweets only using hashtags they are interested in. Furthermore, the success of the hashtag reconstruction also provides evidence that users

do not misuse or abuse hashtags since hashtag presence was taken to be the ground truth in all experiments. Finally, the applicability of the hashtag reconstruction results to the TREC Microblog Track and a mobile application is presented.

Acknowledgements

Many thanks to my supervisor, Gordon Cormack, who has shown me a great many things. In particular, the utility of a certain 32 lines of code. In addition, my thanks to Charles Clarke and Mark Smucker for graciously reading this thesis.

Thanks to all members of the Programming Languages Group Lab, who have provided an amusing albeit not always productive time.

As well, thanks are owed to my family who have not seen me all that much since I began work on my Masters.

Finally, I acknowledge the support provided by the Ontario Graduate Scholarship and the University of Waterloo President's Scholarship.

Dedication

To my father.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Microblogging is a form of social media that allows the users of a microblogging system to post short messages that are available to other users of the system and in many cases the general public. Microblogging can be seen as a short blog posts but with a higher emphasis on stream of consciousness posting rather than the elaborate blog posts on subjects that interest the author, which is typically seen in blogs. There are many different microblogging services available for use, such as Twitter[1], Plurk[2], Jaiku[3], Qaiku[4], identi.ca[5] and so on. Microblogs are in many cases extremely similar to the "status updates" present in so-called social networking sites, like Facebook[6] and Google+[7]. However, microblogs differ from these status updates in that they are broadcast ("one-to-many") messages which are potentially viewable by every user on the site. Status updates, on the other hand, are generally only viewable to one's friends on social networking websites. Twitter is one of the most popular and widely used of the microblogging services with over 250 million posts per day and over 100 million active users[8]. Due to the overwhelming popularity of Twitter,

[1] `http://www.twitter.com`
[2] `http://www.plurk.com`
[3] Subsequently, this service has shutdown
[4] `http://www.qaiku.com`
[5] `http://www.identi.ca`
[6] `http://www.facebook.com`
[7] `http://plus.google.com`
[8] According to `http://business.twitter.com/en/basics/what-is-twitter/` accessed on February 6, 2012.

it will be the primary microblogging system discussed and used in this thesis.

Microblogging and Twitter have spawned an entirely new terminology with which to discuss the service being provided. Accordingly, it is useful to be familiar with the terminology so as to avoid overly long explanations later in this thesis. The following is a set of definitions that aim to provide the requisite background information into the Twitter microblogging service.

Definition. A **tweet** is a a microblog post composed of at most 140 characters (in a Unicode enconding).

Definition. A **stream** in the context of this thesis is a continuously growing set of tweets ordered with respect to recency. A **userstream** is the stream comprised of a particular user's tweets only.

Definition. A **mention** is the inclusion of another user's screen name in a tweet in order to bring the tweet to their attention. A mention is formed by prepending the desired screen name with a '@', e.g. "@claclarke Tweet more". In other works on Twitter, mentions can be further categorized into various sub-classes, however, this thesis does not make use of them and so they are not discussed further.

Definition. A **hashtag** is a string composed of some combination of alphanumeric characters and '_' which are then prepended with the hash symbol, e.g.#this_is_a_thesis, #twitter, #etc. Hashtags were originally an organic user driven concept, which according to Chang [18] comes from Internet Relay Chat tradition. However, Twitter now officially supports hashtags and links a hashtag to a search for the linked hashtag.

Definition. A **retweet** is a repost of another user's tweet. There are two types of retweet: one supported by Twitter, which causes the retweet to appear in the retweeting user's tweet stream as if the original author posted it there, and the original format, which is a copy of the original tweet's content with "RT @<author-username>"[9]. The older retweet method remains in use as it allows the retweeting user to add their own text while the Twitter supported version does not. Figure 1.1 shows the styles of tweets. For the remainder of this thesis **simple retweet** will refer to Twitter supported retweets specifically.

Definition. Suppose we have two Twitter users. @gvcormack and @claclarke, and suppose that @claclarke follows @gvcormack meaning that @claclarke sees any tweets

[9]This is the standard retweet format, however, there are other non-standard formats

that @gvcormack posts on his personalized stream. Then we say that @claclarke is @gvcormack's **follower** and that @gvcormack is @claclarke's **friend**.

Definition. The **follower graph** is the graph induced by treating each user as a node and the following relationships among users as edges.

Definition. A **protected user** is one who restricts access to their tweets to only their followers.

Definition. A **verified user** is one whom Twitter has verified to be the person that they claim to be. This is generally only used for companies, celebrities, and important public figures, e.g. @aplusk is the verified Twitter account for Ashton Kutcher.

Definition. **Trending topics** are hashtags or words that are popular on Twitter at some scale, e.g. worldwide, in North America, in Canada, etc, for some recent period of time.

Since Twitter is becoming increasingly popular[46], it is being used for both social networking and news and information dissemination[79, 40]. Furthermore, hashtags in Twitter have been found to have conversational (a hashtag is used to denote that a tweet belongs to a particular context) rather than organizational usage (labelling the tweet to aid in finding the tweet later) when compared to tag usage in other social media services [38]. Due to these factors and the large volume of content generated on a daily basis, Twitter users can become subject to information overload, which is when a person does not know how to make sense of all the information being presented to them, when trying to find tweets about topics they find interesting. The reduction of information overload using Twitter supported methods suffers from the following issues:

- While a user's friends can help to reduce this overload it is likely the case that one's friends will not talk about every topic that is being discussed on Twitter that a particular user finds interesting. This is especially true in the usage of Twitter in discussing and reporting on natural disasters[61, 65] or other important events (e.g. the Iranian election of 2009, or the Japanese nuclear reactor meltdown in early 2011) where the odds of having a friend being involved are generally low.
- Twitter allows users to view the public timeline (a stream of the most recent tweets posted by users that are not protected), however, sifting through this can be arduous and not practical.
- Twitter also provides search capabilities[10], however, the number of results are limited

[10] `search.twitter.com`

(to around 1,500) and boolean search (where all terms in the query must be present in the returned tweet) is performed. The use of boolean search means that limitations are placed on what results can be returned. For example, boolean search will not return tweets that do not contain all query terms which can leave out relevant tweets.

- The other issue with Twitter supported search is that Twitter does support some notions of tweet ranking (by how recent the tweets are, by how popular the tweets are, etc) and allows some basic filtering parameters based upon location or language. However, many of these features are black boxes that provide no insight into how Twitter implements these features. For example, how Twitter defines a popular tweet.

- If hashtags are topical in nature, say #superbowl or #earthquake, then using them in Twitter search only returns tweets containing that hashtag. This is obviously undesirable behaviour as it is likely the case (in part due to the character limit) that not every tweet that can be tagged will be and so the user will potentially miss tweets that could be interesting to them.

- Further, hashtag usage in disaster scenarios is often cluttered and confused with many different candidate hashtags [61]. Accordingly, many different hashtags are used before Twitter users throw their support behind a single unifying hashtag.

From these issues, it can be seen that there are two main problem areas with respect to Twitter. The first is that Twitter supported search is lacking and needs to be improved by moving away from boolean search and by examining different approaches for retrieving tweets that are relevant to a user's query. The second is that while hashtags are immensely useful as means of providing context for tweets they are only useful if used consistently and are used in any tweet where they are applicable (if the hashtag can be used it should be used). The former is an issue of user behaviour which is uncontrollable and the latter is impossible (due to the character limit). Thus, the overriding goal of this thesis is to examine these two problem areas and the following section outlines the objectives of this thesis.

1.2 Objectives

The two problems outlined above while related are also distinct and accordingly this thesis treats them as such. Due to the limitations of Twitter Search and the fact that many of the

(a) Original Tweet

(b) Simple Retweet

(c) Retweet

Figure 1.1: Comparison of ways to retweet

other popular microblogging services do not offer search or limit it in some way, exploring real-time ad-hoc search, which revolves around finding relevant but also recent tweets given a user's query, in an open manner is crucial to determining the approaches that do and do not work when attempting to retrieve relevant tweets. Through participation in the 2011 Text Retrieval Conference (TREC)[11] Microblog Track, which aims to study real-time ad-hoc search in Twitter, this thesis has the following objectives:

- As ad hoc search has been extensively researched both at TREC and elsewhere in the information retrieval community, an objective of this thesis is to examine how well traditional search methodologies apply to Twitter. This can be seen as an "out of the box" approach where no Twitter or microblog specific features are used to aid in retrieval of tweets.
- Through participation in the Microblog Track, a second objective is to create a competitive baseline that can be built upon and further refined both in this iteration of the Microblog Track and any additional iterations of the Microblog Track. This baseline again is meant to be "out of the box" to allow true improvement to be seen over tried and tested approaches to ad-hoc search.

The use of hashtags as a means of providing context to a tweet is a useful and beneficial practice, however, as mentioned previously hashtags can be misused or not used at all. To remedy this problem this thesis also investigates the use of binary classification in determining if a particular hashtag should be contained in a tweet without it. Binary classification separates two sets of documents into two classes of examples, a positive and a negative class, and then determines which class new documents belong to based upon some set of examples. This thesis has the following objectives with respect to this task:

- To examine the applicability of a wide range of state-of-the-art classifiers to this task and to see which performs this task well without Twitter or microblog specific modifications. This allows for the creation of baseline which can potentially be enhanced in future work.
- This is done by first examining how well these classifiers work in an idealized experiment where the two classes are topically distinct sets of tweets.
- Following this is an examination of the real world scenario where one class is a random set of tweets and the other is a topically homogenous set based upon a particular

[11]TREC is an ongoing international effort to investigate information retrieval in various domains. A detailed explanation of TREC follows in Chapter 2

hashtag. This examination uses incremental increases in the size of training examples to determine how soon the classification could be done in a real world application.

- Finally, it may be the case that a Twitter user wants to learn more about a particular sub-topic in a set of topically homogenous tweets. Accordingly, this thesis repeats the previous experiment to see how well the classifiers can perform this sub-topic classification task.

1.3 Outline

Chapter 2 contains a brief description of TREC and discussion on a selection of related works in the theory of microblog search and classification. Chapter 3 contains an outline of the 2011 TREC Microblog Track, how the tweets were collected for the track, the search methodology used during participation in said track including the retrieval methods used and the runs created for submission and comparison, the results achieved and a discussion of what those results mean with respect to the Microblog Track as well as to the methods used, and some avenues of future research. Chapter 4 contains the hashtag based classification experiments, including how tweets were collected, how the experiments were conducted, how results were calculated, brief descriptions of the classifiers used, a discussion of the results and what they mean for a real world application. In addition, Chapter 4 provides a discussion of how this hashtag based classification can be applied to the Microblog Track and to a mobile application for use in the real world. Chapter 5 contains a summary of the contributions of this thesis.

Chapter 2

Related Works

This chapter provides some brief background in the Text REtrieval Conference (TREC) before proceeding into discussions of works related to Twitter and microblog search and classification. In addition, works that are directly applicable to microblog search and classification are presented in the closing section of this chapter.

2.1 Text REtrieval Conference Overview

The Text REtrieval Conference (TREC) is an annual conference co-sponsored by the National Institute of Standards and Technology and the U.S. Department of Defense. TREC aims to further research on information retrieval systems for large text collections through increased communication between academia, industry, and government. TREC supports a "track" format where different research areas in information retrieval are represented, e.g. legal e-discovery, spam filtering, web search, microblog search, etc. A track has a corresponding task or tasks, which are generally some domain specific application of information retrieval. While the precise setup of individual tracks may vary, the organizers all follow the same basic outline. Organizers of a track assemble or find a suitable dataset, NIST (or in some cases professionals in the field) create a set of topics[1] (usually 50 but this is track dependent), participating groups find documents in the collection that are relevant to the topics and compose them into "runs", and NIST has assessors determine the relevance of returned results, usually using pooling (discussed in Chapter 3). Evaluation is performed

[1]Topics are task dependent but can be thought of as queries in a very general sense.

based upon the assessments made. NIST then hosts a conference that allows track participants to present and discuss results and determine future directions for the track. NIST at some point makes the collection and assessments available to the public to facilitate additional research. For the 2011 version of the conference, a new Microblog Track was added. The Microblog Track aims to explore real-time ad hoc search in Twitter. A longer description of the Microblog Track can be found in Chapter 3.

2.2 Classification of Tweets

Tweets on Twitter are often seen as completely separate from any surrounding context, whether it be topically related posts or the tweets from the same author. Given a stream of tweets it is desirable to be able to classify tweets in a meaningful way. This section outlines current research into the classification of tweets with respect to topic, retweetability (e.g. if a tweet will be retweeted), and sentiment.

2.2.1 Topical Classification

Investigating the ability to separate tweets that are topically similar from those that are not can be seen as an objective of this thesis. However, this thesis is not the first to look at classification based upon hashtags. Nishida et al. use data compression [56] to look at tweet classification using the standard information gain approach, where two compression models are created one for the positive class and one for the negative class and whichever can compress a candidate document the best indicates that the document should belong to that class. However, the authors do not compare the proposed method using standard binary classifiers [11, Ch. 10], like Support Vector Machines [11, Ch. 10], logistic regression [11, Ch. 10], or other compression-based classifiers. The authors use non-standard binary classifiers[11, Ch. 10] with bag-of-words token features[11, Ch. 2] but since tweets are not necessarily composed of whitespace delimited languages means that the non-standard classifiers will immediately take a performance hit. However, their results, which do not give the best comparisons to the state-of-the-art, do provide evidence that it is possible to separate tweets based upon hashtags.

Sakaki et al. show that classifying tweets based upon topics can be used for event detection [64]. The authors present a system, Torretter, that uses a Support Vector Machine (SVM), which is a standard classifier. Various features, like bag of words, number of words, position of query word, and the words before and after the query word, were used with

the SVM to determine if an earthquake has occurred in Japan (using query words such as "earthquake" and "shaking"). Tweets, that were identified as referring to an earthquake, were used as a kind of of social sensor in a proposed earthquake detection system. The proposed system was able to determine that an earthquake had occurred in Japan for the majority of earthquakes over a few months (only a small number of weak earthquakes went undetected). This is again further evidence that tweets can be classified as belonging to a particular topic. It is important to note, however, that the only two classification examples were given and that precision tended to be dominated by recall, which the authors suggest is due to ambiguity in tweet content (e.g. "Is that a truck passing by or an earthquake?").

IMASS [75] is a system developed by Weng et al. to separate tweets into conversation threads based upon different types of conversations and using different methods to summarize the thread of conversation. For example, a tweet may be interrogative, or it may share a URL, or it may involve both sharing and question asking (discussion), or it may simply be chatting. The authors use a decision tree method with a binary classifier to determine which of the four types a tweet can fall into. It is stated that this significantly improves a multi-class classification but there is no mention of any statistical testing to lend this claim additional credibility. The authors use sentiment analysis to determine whether or not sharing tweets (and any replies) exhibit positive, neutral, or negative sentiment. For chat posts, the authors use LibSVM[2], a publicly available Support Vector Machine library, to determine if one post is in response to another. Finally, for interrogative tweets the authors use LibSVM to determine the most relevant responses to a question. All their methods are reported only in terms of accuracy with respect to 10-fold cross validation, which is undesirable as LibSVM can give actual probabilistic scores that can be used for more meaningful result analysis. However, from the small set of examples the authors provide it would appear that their system performs reasonably well.

2.2.2 Predicting Retweets

Naveed et al. are of the opinion that interesting tweets are those that will be retweeted. In their paper [54], the authors outline the goal of finding interesting tweets by predicting if a tweet will be retweeted. To accomplish this the authors use a logistic regression model and train it on features such as: the presence of a URL, the presence of a hashtag, if tweet contains a mention, if the tweet has high valence (the emotional value associated with a tweet, e.g. happy tweets may have high positive valence), the arousal of a tweet (how exciting a tweet is), the topics discussed in the tweet, etc. The authors test their model by

[2]Available from `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

using the chronologically first 75% of their corpus as the training set and the last 25% as the test set. In an examination of the weightings for the proposed model and it was found that mentions are unlikely to be retweeted, messages with hashtags or URLs are likely to be retweeted, having positive words increases probability of being retweeted, tweets with negative valence (e.g. displeasing content), exciting and intense tweets are likely to be retweeted, and that the term content of the tweet has a strong influence on the probability of it being retweeted. While the authors do comparisons of how well certain features affect performance, no statistical evidence is presented to show that any feature is significantly better than another.

Predicting retweets is also explored in a paper [37] by Hong et al. where the authors go one step further and attempt to predict a rough estimate of the number of retweets a tweet will get. Like Naveed et al. the authors use a logistic regression model and train on features such as derived topic features, how many times the author has been retweeted in the past, tf-idf features, if the tweet has been retweeted previously, and degree distribution (which is not clearly elaborated on but given the context likely has to do with the number of followers and friends). The model is able to improve over baseline models, which use whether a tweet has been retweeted before and a tweet's tf-idf features. By removing features the authors state that removing previous retweet status and the degree distribution result in the biggest decreases in predicting if a tweet will be retweeted, however, these decreases are not shown to be statistically significant. Furthermore, when the authors attempt to place tweets in retweet count buckets (e.g. 0 retweets, < 100, < 1000, $> 10{,}000$) the authors find that only tweets that get no retweets or are retweeted the most are able to be predicted with any accuracy; however, the authors note that by adding in unspecified temporal features the the prediction capabilities of their algorithm for mildly retweeted tweets is able to be increased. What the authors do not mention is the fact that the likely reason that the 0 retweet category is so easy to predict results from the fact that such tweets are extremely common in Twitter. But the case for the $> 10{,}000$ retweets class is not readily apparent.

Uysal and Croft investigate personalized tweet ranking, where tweets that a user is likely to retweet appear at the top of a stream (as opposed to recency based ranking)[71]. The authors use a variety of features to determine whether or not a tweet is likely to be retweeted from four main categories: author-based (follower count, friend count, account age, tweet count, tweet rate, if the user verified, etc), tweet-based (presence of hashtags/URLs/-mentions/...), content-based (minimum cosine distance to other tweet's appearing in that particular user's userstream, minimum cosine distance to the tweets of the author), user-based (is the author a friend, has the user and author mentioned each other, did the user retweet the author, etc). Using these features the authors took a learning to rank approach using the Coordinate Ascent algorithm on a decision tree based classifier to predict the

retweetability of tweets by a user. Their results indicate that their method can generally determine retweetability. Using a "leave one out" approach the authors found that the presence of a URL, whether the tweet had been retweeted, the tf-idf score of the tweet, and the author being a user's friend were the most effective features but present no statistical evidence to verify these findings. The authors find that the presented method is able to rank tweets according to retweetability (where the rank is accurate if the user later retweets a tweet). However, the authors compare their results to a random baseline which the authors state is similar to how Twitter ranks tweets in a stream, however, in a stream the tweets are ranked by recency which is obviously not random.

2.2.3 Classifying Tweet Sentiment

Bermingham and Smeaton investigate the classification of sentiment in microblogs in an attempt to determine if the brevity of microblog posts affects classification ability [7]. The authors use three classifiers in an attempt to measure sentiment: multinomial naive Bayes (MNB)[11, Ch. 10], a Support Vector Machine, and SentiWordNet (which determines the subjectivity of words)[29] as baselines. The authors explore the utility of using unigram, bigram, trigram, and part-of-speech stop worded-bigram features. The authors ran 10-fold cross validation across microblogs, blogs, microreviews (140 character limit), and reviews as a means of comparison. The authors found that for long-form documents (blogs and reviews) SVM (with unigrams, part-of-speech stopworded-bigrams, and trigram features) was most accurate while for short-form (microblogs and microreviews) MNB (with unigram features only) achieved the highest accuracy. This trend held regardless if binary (positive/negative) or ternary (positive/negative/neutral) sentiment classification was performed. However, accuracy for both classifiers tended to be fairly close and it is not apparent that these results were not simply occurrences by chance. However, as accuracy was high for all microblogs and microreviews (72%+) then it is likely the case that the brevity of microblogs and microreviews may not be as detrimental to sentiment classification as one might think.

Barbosa and Feng approach sentiment analysis as a two step process, first the authors perform subjectivity classification ("Does a tweet have sentiment?") and then polarity classification ("Is the sentiment positive or negative?") [5]. For classification features the authors use part-of-speech tagging for tweet terms, term polarity (positive/negative/neutral), presence of a hashtag or URL, number of retweets, if the tweet is a mention, the emoticons present in the tweet, and so on. However, no reason is given for tweet based features (like hashtags or retweet) and so it is not clear as to why the authors believe these can help sentiment classification other than the fact that the features are there. The authors

create a large corpus from three different tweet sentiment analyzing websites, which determine if a tweet matching a query has positive, negative, or no sentiment, and remove only those tweets which the sites disagree on the sentiment label (e.g. one site says the tweet has positive sentiment but another says it has negative sentiment). This creates a bias towards tweets that only come from one of these sites as these tweets have no guarantee that the prescribed label is correct. The authors find that there is relatively low agreement among the websites and claim that this may be a good thing as each site provides diverse labels that may help polarity classification. This seems counter intuitive as that would mean that each site has a different idea of what positive and negative tweets look like and accordingly could hinder classification. For the purpose of comparison the authors used a Support Vector Machine with unigram features and LingPipe [12], which is a sentence level sentiment classifier. The authors found that according to error rates their approach performed the best, however, there is no statistical evidence to verify this and given the issues with their corpora it would seem that such evidence is necessary. In addition, it is not clear how fair the comparison is given that tweets cannot be comprised of more than a few short sentences or a very long sentence and so LingPipe may have had an immediate disadvantage.

Davidov et al. apply their semi-supervised sarcasm identification algorithm [26] to the problem of detecting sarcasm in tweets by looking at pattern based features (e.g. "How much of this tweet matches a learned pattern?") and punctuation features (e.g. "Is an exclamation mark present?"). Patterns are combinations of high-frequency words (like punctuation) and low-frequency content words. In the preprocessing of their Twitter data, the authors change mentions, URLs, and hashtags to general meta-tags to avoid learning specific references, e.g. the authors do not want to learn tweets to "@comicbookguy"[3] are sarcastic. However, it is not clear that converting all hashtags to a general meta-tag is beneficial as this can result in the loss of information. For example, the content of the hashtag itself can make it clear that one was being sarcastic (without use of "#sarcasm" or "#sarcastic") and could be an integral part of the pattern, such is often the case for meme hashtags (#thatswhatshesaid). In addition, the hashtag meta-tag is treated as a high frequency word when it may not be the case that all hashtags are high frequency words. Regardless, the authors find that sarcasm is hard to detect using tweets alone, however, if the authors add to the training sets examples pulled from Amazon reviews, which also tend to be short, then the authors can improve performance. It is not clear how well their dataset was created when manual annotation of results had low inter-annotator agreement (and this was after the authors reduced the problem to a binary judgement as opposed to a scaled sarcasm judgement). These results would indicate that finding sarcastic tweets

[3] "Best. Thesis. Ever."

is hard and that cross-domain training is necessary to improve sarcasm detection. It is perhaps not surprising that sarcasm detection is hard given the use of sarcasm can vary radically among individuals in real life.

Davidov et al. continue the use of hashtags as labels by examining supervised sentiment classification based on Twitter [25]. The authors extended the model above by using words, 2-5 word n-grams. Emoticons and hashtags are used as labels for their data, and result in a 51-class classification problem for hashtags and 16-class problem for emoticons. The authors find that emoticon based classification performs better than a random baseline. However, this is unsurprising since the hashtag labelled classification has far too many classes to hope to perform well. In addition, when the authors preprocess their data set the authors again convert hashtags to a general meta-tag, which as previously mentioned could result in decreased performance due to information loss as not all hashtags have sentiment content. When training for no sentiment, only tweets that did not contain hashtags or emoticons were used. In doing so an unintentional bias towards how sentiment is detected can be created, e.g. no hashtag or emoticon at all implies there is no sentiment. When the authors perform binary sentiment classification (e.g. "Should this tweet be tagged with #happy?") a considerable improvement over the multiclass equivalent is seen but this fact is not overly surprising given that it is an easier classification to make. Finally, Davidov et al. state that the pattern features used in the algorithm are most useful in finding sentimental tweets.

2.3 Microblog Search

Finding tweets that users may find interesting or relevant through the use of classification generally requires that users have some preconceived notion of their information need (by knowing a hashtag for example). Microblog search follows the traditional search approach of being given a query and finding tweets that are relevant to it. However, microblog search can make use of different factors that are specialized to the microblogging environment, like the number of times something is retweeted or the number of followers of a tweet's author. This section examines research into how traditional search is adapted for microblog search.

A comparison of Twitter search and traditional web search was conducted by Teevan et al. [70]. In an examination of the query logs and browsing history of users who use the Bing toolbar, the authors found that hashtags comprise about a fifth of the Twitter queries, however, this trend was seen after Twitter implemented the linking of hashtags to a search for that hashtag. Accordingly, the actual results of queries that are manually issued to Twitter Search through web interface may be different. How the query was issued cannot be

disambiguated since the queries were extracted from URLs. In addition, the authors found that between 3-4% of Twitter and Web queries overlap for queries that would be hashtags if a '#' was present. The most popular Twitter queries tended to contain a hashtag while very low frequency queries did not. Web queries tended to contain more terms but Twitter queries tended to contain longer terms, which is understandable considering many popular hashtags are memes that are actually sentences, e.g. #thatswhatshesaid. This idea is reinforced when top Twitter queries are memes, celebrity names, or are holiday based. While top web queries were navigational in nature (e.g. "twitter", "ebay", etc). However, celebrity queries commonly overlapped between web and Twitter search. The authors found that Twitter search sessions tended to be shorter than web search. In a study of Microsoft Research employees who use Twitter, the authors reported that employees used Twitter to search for timely information (news and events) and social information (friends, celebrities, etc) and used web search for finding factual information.

Twitter and many other microblogging services provide an interface for search and by default these search services return relevant results by most recent post time. Nagmoti et al. [53] investigate heuristic ways to rerank Twitter search results so that tweets that are "more relevant" are ranked higher. The authors look at four quality indicators with regards to tweets: tweet length, number of tweets the author has made, the ratio of followers to the sum of followers and friends, and the presence of a URL. The authors find that the number of tweets by the author performs poorly and that length of a tweet produces a better ranking. The authors find that if the followers to followers plus friends ratio, length of a tweet, and presence of a URL are all combined trivially the best re-ranking of Twitter Search results is produced. Evaluation is done by first having users issue a query and randomly selecting 2 tweets out of the top 100 and presenting them to the querying user and asking which they prefer. From these preference judgements, each measure is scored and said to be good if the measure's reranking matches the preference judgement. No evidence is presented that indicates that a definitive ranking was created for the top 100 results for each query using user preference. In addition, it could very well be by chance that the preference judgements occur in the order that they do. Thus, the measures presented by the authors are only compared on partial rankings by users and so the best performing measure may perform that well by chance. Finally, there is no mention that these preference judgements were assessed multiple times for quality assurance and it could be the case that there were adversarial users of their system that improperly judged tweets.

Massoudi et al. take a slightly different approach by finding relevant tweets from their own collection of tweets using a generative language model approach [51], which can be thought of as using the probability that query terms are contained in a tweet to generate a relevance score. The authors tweak the performance of their baseline language model

approach by adding in "quality indicators," which include the number of retweets, the author's follower count, emoticons, punctuation, and presence of a URL, and query expansion, where terms are added to the query to improve the results. Query expansion is performed by taking the top 10 terms that are scored with respect to how recently they are used, which incorporates the idea that the way an idea is expressed can change over time. The methods presented by the authors were compared to a boolean search that ranks results by recency (similar to Twitter Search) and their model with a standard query expansion model using standard information retrieval evaluation techniques. From the presented results it appears that the method of query expansion presented produces some improvement over the baseline method used as opposed to any combination of quality indicators. Combining query expansion and all quality indicators produces a competitive method of tweet retrieval. However, queries were generated from trending topics when the authors were collecting their corpus. The use of trending topics likely had a positive influence on performance since trending topics are popular keywords or hashtags and so can make retrieval easier since they are widely used in Twitter.

Efron and Golovchinsky [28] use Bayesian estimation as a base method and change how the temporal nature of tweets is used to produce three different approaches to examine how retrieval is affected by queries that are time dependent, e.g. searching about a celebrity may be different than searching about an earthquake that just happened . The three approaches are query-specific re-ranking (if a query is temporal in nature then the time of a tweet is more of a factor in score generation), temporally informed smoothing (where older tweets are smoothed as their language likely does not match the most recent language used to discuss a topic), and temporally biased query expansion (which uses the previous approach to perform query expansion). The authors find that temporally informed smoothing performs the best on their 6 month Twitter dataset as it improves results returned for recency based queries but does not affect non-recency based queries. In all of the cases, the authors are able to improve on non-temporal baselines. The interesting aspect of this work and the previous work is that it addresses an important issue with respect to Twitter. Language on Twitter is something that likely evolves much quicker than it does in other social media due to the fact that tweets are so short and can be sent in rapid bursts. Thus, any method that is able to cope with language changes with respect to topics is likely going to improve results. In addition, this work and the work of Massoudi et al. reflect that tweets closer to the query time are more likely to be relevant as the authors will use the same or similar language to discuss topics.

In an investigation of document quality and sparsity for Twitter [55], Naveed et al. find that length normalization is not necessary for tweets as the authors generally only cover a single topic and that the length of a tweet is not caused by repetition of words

(term frequency tends to 1 in their dataset). The authors theorize that in microblog search "interesting" tweets are more desirable and define the "interestingness" of a tweet as the probability of being retweeted it being retweeted. The a priori odds of a retweet are computed using Bayes' Rule and the terms of a tweet. The odds of retweet along with other features (topic, presence of URL or hashtag, punctuation, sentiment, etc) are used to train a logistic regression model to compute the probability of a tweet being retweeted. The authors use three retrieval systems all of which are Lucene[4] based (a vector space retrieval model [11, Ch. 2]) to test how well their measure of interestingness works. The authors use out of the box Lucene, Lucene with no length normalization, and Lucene with no length normalization and reranking the top 100 relevant results according to probability of retweet. From this the authors find that for extremely short queries the re-ranking method produces more interesting tweets (for some definition of interesting), however, for longer queries performance is best when Lucene with no length normalization is used. The authors believe this is because longer queries are more specific and so can introduce more marginally relevant results into the result set which are more "interesting" than the actual relevant results. This may seem counter intuitive, however, small queries are generally not that specific and so it is easier to find "interesting" tweets in this larger set of results. Instead of performing statistical testing to verify their results, the authors conduct a case study which requires participants to subjectively pick which system the authors prefer when presented with the different length queries and the results. The results of this subjective experiment agree with their objective analysis.

2.4 Other Related Topics

2.4.1 User Influence

The influence of Twitter users is a common research topic with regards to microblogging. Being able to measure the influence of users has the potential to help boost search results by returning tweets from a highly influential user before tweets of less influential users. As well, influence can be used from a marketing perspective as an influential microblogger may be able to introduce a product or service to a wide range of users without requiring much effort. Influence could also be used a measure of trust, so that a more influential user is less likely to knowingly post tweets containing damaging or wrong information.

One of the first reports on the influence of Twitter users was conducted by Leavitt et al.[47]. The authors consider the influence of a user "as the potential of an action of

[4]Lucene is a text search engine available at `http://lucene.apache.org/core/`.

a user to initiate a further action by another user". In their investigation, the authors conclude that a user can be in one of three general categories: a materialist, who pushes content (e.g. news media, celebrity, etc), a conversationalist (e.g. a user who is using the service for social networking purposes), and a spammer (a user who tries to get users to click malicious links). These categories are defined based on a user's friend to a follower ratio (# of friends/# of followers), if it is high then the user may be a spammer, if it is approximately 1 then the user may be a conversationalist, and if it is large then the user is likely a materialist. The authors argue that it may be better to judge user influence by mentions as this is a better measurement of other users actions in response to the targeted user. Using mentions to judge influence may in fact be more useful as any measure based solely upon friend and follower numbers is likely to be predisposed to finding users with an extremely large number of followers as "influential", which may defeat the purpose of such measures.

Lee et al. propose an interesting method of determining user influence [49]. The authors argue that influence can be measured by a user's number of effective readers, e.g. those users exposed to a topic for the first time by the tweets of that user. However, the authors find that the majority of users have very few effective readers, which means that having a large number of followers may not result in a user being a good source for information diffusion, e.g. spreading information. News media outlets are found to generally have the largest group of effective readers and that there is little overlap with ranking users by follower or retweet count and even smaller overlap when PageRank[5][57] is used. The direct benefit of this measure is that it allows emergency alert systems, which are systems that override television and radio broadcasts to pass time sensitive information to the audience, to find key Twitter users to pass time sensitive information to so that it is spread to as many users as quickly as possible in an emergency. However, as the majority of Twitter users that have many effective readers are news media accounts it is not clear that this measure is useful beyond emergency alert situations.

The "million follower fallacy", that more followers does not equal more influence, is investigated by Cha et al. over a corpus of 1.7 billion tweets, 57 million users, and the 1.9 billion follower links among those users[16]. Given the previously discussed works, it is unsurprising that the authors find follower count is not all that useful as an influence measure since it is biased to extremely popular users who may not post useful/interesting content. The authors use retweet and mention counts as additional measures of influence. Over three topics (the Iranian Election, Swine Flu, and Michael Jackson's death), Cha et al. find that influence tends to be stable across topics, influence follows a power law, that retweet and mention based influence tend to be correlated, and that the top influentials are

[5]PageRank is the authority score that is the basis for Google search

news media, content aggregators (users who post about trending topics, see @TweetMeme), and celebrities. However, the stability across topics is suspect since only three topics are used and the authors are all news and other popular topics that would lend themselves to being reported about by celebrities and news media. It would have been interesting to see how stable influence was across less newsworthy or less popular topics. The authors found that news media generated the most retweets over time compared to celebrities who generated the most mentions. The result for news media supports the idea that the authors can be used for diffusion of information in times of emergency as news media provide the highest chance of furthering the spread of information. Cha et al. state that the high mention counts for celebrities would indicate that celebrities may be ideal for product advertisement as users apparently care about celebrity opinion due to the interaction rate that is implied by the large number of mentions generated by celebrities. After doing some manual inspection the authors come to the conclusion that while ordinary users (e.g. Leavitt et al. 's conversationalists) can increase their own influence it must generally be done with a single topic and requires a great deal of effort.

In a similar vein to Cha et al. Anger et al. investigate influence for the top 1000 Austrian Twitter users [3]. The authors investigated friend to follower ratio, follower count, retweet and mention ratio (number of retweets and mentions divided by total number of posts), and interactor ratio (number of users that retweet or mention a user divided by that user's follower count). The interactor ratio could be seen as a measurement of a user's actual readers, which are those users that likely read some if not all of a user's tweets. While the retweet and mention ratio could be seen as an interestingness indicator meaning that a user who publishes little but has many retweets or mentions likely says something interesting. However, retweet and mention counts are not readily available for a user and it is not made clear how the authors determined this information[6]. Ignoring this fact, the authors find that follower count and friend to follower ratio produce equivalent influence rankings, but retweet and mention ratio and interactor ratio produce different rankings. Subsequently, the authors propose a Social Networking Potential (SNP) measure, which is a linear combination of the retweet and mention ration and the interactor ratio. The authors believe that such a measure incorporates the interactions and content of Twitter. That is, the authors show that users who are ranked highly by this measure tend to be more active in Twitter discussions and development. However, the applicability of these results to Twitter at large is unknown as a small selection of Twitter users that are regionally located are used in the experiments conducted by the authors. This question of applicability is supported by the fact that their top users were political commentators

[6]Collecting this information would require polling a user's own tweets and the tweets of their followers. But even this may not find all mentions as it is possible to mention a user one is not following.

as opposed to the distributions as seen in previously discussed works. In addition, while SNP sounds interesting it is not something that can be easily generated on the fly and accordingly has limited usage to Twitter.

PageRank is a popular algorithm for measuring a web site's influence (or authority) and is often used as a default measure of influence in Microblog research with respect to the follower graph. Haveliwala [35] extends PageRank for web search by adding an additional 16 PageRank vectors biased towards websites that fall under the 16 top-level categories of the Open Directory Project (e.g. Entertainment, Politics, etc). Thus, for a particular PageRank vector any website that is listed under the corresponding category has higher authority than those not. Scores are combined in a weighted fashion based upon the probability of the query referring to each category (e.g. the probability acts as the weight). The author finds that the query probability generation produces reasonable weightings. In addition, the results presented indicate that users prefer topic sensitive rankings compared to generic PageRank, however, no comparisons are done to other influence algorithms, like HITS[44] (which is query sensitive similar to the presented method) or SALSA[50], both of which are popular influence measuring algorithms. Similarly, no standard corpus or evaluation measures are used even though there are many available. However, this method could very well be a useful extension of PageRank in the Twitter domain as it would not require too much tweaking to the algorithm (e.g. we can use the tweets of a user to determine the topics the authors tweet about).

TwitterRank [74] is a PageRank algorithm that is a topic sensitive but designed specifically for Twitter. Weng et al. , the designers of TwitterRank, in their investigation of homophily of Singaporean Twitter users (that users follow each other based upon topical similarity and not just randomly) found that their collection of users do exhibit homophily and from this the authors developed TwitterRank. The basic premise of TwitterRank is that if user a follows user b then the more topically similar a and b are and the more b posts then the more likely a random surfer (of PageRank notoriety), who randomly views users, would be to take the path from a to b. However, this immediately has an issue concerning tweet frequency. Tweeting more frequently does not guarantee a user is more influential than one who does not. For example, suppose a person has two friends, one who talks all the time but doesn't say anything of import and another other that does not speak a lot but when they speak it is meaningful. In such a case, it is clear that, if both are assumed to be equally topically similar, the second friend likely has more influence which is contradictory to the proposed algorithm and it is not hard to imagine this being the case on Twitter. Weng et al. briefly acknowledge this by stating that the authors would like to investigate how mentions and replies between two users affect this value rather than pure tweet count. The proposed algorithm is compared to PageRank and Topic-Sensitive

PageRank by having each algorithm recommend for a particular user new users to follow. The experimental results show that TwitterRank outperforms the other algorithms with this recommendation task. From these results, TwitterRank could be used to personalize search results by ranking tweets according to whether or not TwitterRank recommends each tweet's author as an account that should be followed by the searching user.

Designing algorithms around the Twitter (or any microblog's) follower graph is potentially problematic. The follower graph is not only extremely volatile but for some microblogging systems, like Twitter, the graph comprises so many users that it is computationally infeasible to generate the whole graph[7]. Welch et al. addressed this issue by looking at a sampling of the follower graph and a graph based upon retweets (an edge is formed from a to b if a retweets b) [72]. The authors find that their retweet graph tends to perform better than TwitterRank and Topic-Sensitive PageRank when ranking user relevance to topics. The methods were evaluated on how well assessors thought a proposed influential user's tweets fit a particular topic. However, in general the proposed method is still computationally infeasible as the number of tweets is unbounded and Twitter only ever explicitly stores at most the first 100 retweet links and so making a complete graph is not easily performed. But this method could still be used in an offline fashion. If the tweet corpus is random and representative then one could can compute the graph for this dataset without requiring extra information as would be required with use of the follower graph.

From these results, it can be seen that calculating user influence on Twitter is not clear cut and there is unlikely to be a single definitive measure of influence. User influence appears to need to be generated in different ways depending on how user influence is meant to be used. That is, a user has more than a single measure of influence and for a particular task this should be taken into account. However, it is unlikely that any follower count based measure is ever a good measure of influence unless one desires to find popular public figures.

2.4.2 Topic Detection

Being able to find or classify tweets belonging to a topic is all well and good, however, how users find these topics is a problem that needs to be solved. Twitter tracks the most popular hashtags and words being used on Twitter and displays them as so-called "Trending Topics". However, this does not help users find less popular but up and coming topics. In this section, research into finding topics in a microblogging environment will

[7]For example, Twitter itself has no notion of the follower graph.

be discussed. In addition, the work described here provides possible methods of selecting candidate topical hashtags for use in hashtag based classification.

Cataldi et al. present a method of finding emerging topics (collections of terms) using term frequency and user influence (computed through PageRank) [14]. Using these two features and the usage history for each term the authors are able to find words that are used more in the current period of time than previously, which are called hot terms. Emergent terms are found by ranking hot terms based upon their score and only those that occur prior to the critical drop (the point where the difference in score between two hot terms exceeds the average difference) are kept as emergent terms. Topics are generated using the co-occurrence of terms with the emergent terms to create a weighted graph. Strongly connected components rooted at the emergent terms are extracted and used as topic descriptions. Their results would indicate that emergent terms are often the most specific and tend to be rare in the community. While the presented results do return consistent and interesting topic descriptors for the time period it is not necessarily made clear that these were the true emerging topics and that there couldn't have been a better emerging topic reported. Finally, it would have been interesting to see how influence generation affects the results and this could have investigated done by using different influence generation measures; many of which were outlined early in this chapter. In doing so, if the topics found tended to be the same it would add further evidence that the authors are truly finding emerging topics.

The previous work implicitly takes advantage of the "burstiness" of terms; that is, terms used much more frequently in the current time period rather than in previous time periods were seen as "hot". However, Cataldi et al. confounded this with user influence which by definition will favour those terms used by highly influence Twitter users and this may not be desirable as a few highly influential users may not reflect the common topics of ordinary users. Platakis et al. present a method for finding event descriptions using the burstiness of terms in blog titles [59]. As blog titles and tweets are both relatively short it is likely that their approach could be adapted for microblogs. The authors use the state series of terms (as proposed by Kleinberg in an earlier work) to model the term frequency. For terms experiencing a period of burstiness the authors find the 5 Nearest Neighbours using a Euclidean distance based metric to judge dissimilarity between terms. Their results would seem to indicate that this method holds some promise and is resilient to terms undergoing multiple bursts of usage (e.g. a term being bursty prior to an event will not effect the event description). In addition, it does not rely on user influence which may be more fair to all users and may highlight important "grassroots" emerging topics, like a natural disaster.

The "interestingness" of hashtags is investigated by Weng et al. [73]. The main goal

of their work is to separate the Twitter follower graph (or in this case a subset of it) into strongly connected communities and then using the number of communities in which a hashtag is being discussed along with how much the hashtag is being used in those communities to determine "interestingness". The novel aspect of their follower graph is that the authors define the influence between two users (one of which is the following the other) as how soon after posting a tweet with a hashtag will the follower post a tweet using the hashtag. Thus, the goal becomes to find communities such that the edge weights inside the community are has as high as possible and those between them are as small as possible. The authors find that when compared to judging interestingness of hashtags based solely upon frequency of usage or number of unique users using the hashtag that their method performs better. However, the ground truth for the experiments is created using the search traffic generated from a query which is based upon a description (from a website that describes a hashtag's meaning) of the hashtag. While not only seeming to be overly complex it requires using a description of the hashtag that may not be accurate and a query that that may not accurately reflect that original description. As no examples of descriptions or queries and the corresponding hashtag are given it is not possible to judge how well their ground truth set is created. In addition, all tweets and ground truth data come from Singaporean users and Singaporean search data and accordingly may not extend to Twitter as a whole. In addition, the work of Weng et al. rely on the follower graph which as stated previously is too large and to volatile to compute in a reasonable manner.

Chapter 3

TREC 2011 Microblog Track

3.1 2011 Microblog Track Overview

The 2011 Microblog Track was the first occurrence of this track and was coordinated by Craig Macdonald, Iadh Ounis, Jimmy Lin, Abdur Choudhury and Ian Soboroff. The primary goal of this track was to investigate real-time ad-hoc search for Twitter. As microblogs are a relatively new phenomenon in information retrieval there are no standardized evaluation measures and so the track had a secondary goal of determining the efficacy of various evaluation methodologies.

3.1.1 Task Outline

The only task for participants at the 2011 Microblog Track was a real-time ad hoc search task. The task was outlined as follows, "given a user's information need, which is modelled by a query at a specific time, a participating system should return the most recent but relevant tweets before the query time." Thus, the results returned to the user are returned newest to oldest. It was suggested that participating systems return "interesting" but "newer" tweets relevant tweets but no guideline definition was provided for either term. This means that a participating team's definition of "interesting" could end up penalizing them. In addition, the coordinators stated that the "novelty" between tweets was not to be considered. NIST created 50 topics for this task and no narrative or description tags, which would describe what the querying user is explicitly attempting to find for a particular topic, were available to participants. However, assessors had the querying user's

information need clearly defined for each topic for their use in determining relevance. Topics were provided in the following format:

```
<top>
<num> Number: MB01 </num>
<title> Mmmmm </title>
<querytime> 14th March 2011 01:59:26 +0000 </querytime>
<querytweettime> 3141592653589793 </querytweettime>
</top>
```

where:

- <num> specifies the topic number.
- <title> specifies a representation of the user's query.
- <querytime> specifies the timestamp of the of the query in a human and machine readable ISO standard format.
- <querytweettime> specifies the timestamp of the query in terms of the chronologically nearest tweet id within in the corpus, e.g. the id of the tweet posted most recently before the query was issued.

Participating groups could submit up to four runs for official assessment. However, at least one run could not use any external or future source of evidence. External evidence was any information external to the corpus but was available before the query's timestamp. Future evidence was any evidence that would not have been available when the query was issued, e.g. tweets in the corpus after the query time. A submitted run could contain up to 1000 tweets per topic but only the 30 most recent were the target of evaluation. Runs being submitted for assessment were required to follow the standard TREC format:

```
MB01 Q0 3141592653589790 1 0.995 piRun
MB01 Q0 3141592653589783 2 0.990 piRun
MB01 Q0 3141592653589762 3 0.885 piRun
...
MB01 Q0 3141592653586793 1000 0.0001 piRun
MB02 Q0 3141592653589793 1 0.991 piRun
...
```

The fields are the topic number, a literal "Q0", a tweet id, the rank that the tweet was retrieved at, the score, and a run identifier. Since only the thirty most recent tweets were used it is the case that the highest scoring tweets may not be returned in the top 30 when tweets are ranked by how recently they were posted. As well, all retrieved tweets for a topic are required to have a a tweet id no greater than the query tweet time specified in the topic description.

3.1.2 Corpus

The corpus for the 2011 Microblog Track was the Tweets11 corpus, which was a sample of just over 16 million tweets provided by Twitter (through the coordinator Jimmy Lin) ranging from January 23rd, 2011 to February 8th, 2011. The Tweets11 corpus was specifically collected over this period as the time range covered events of varying importance, such as the Arab Spring (in particular, the Egyptian Revolution), Super Bowl XLV, the 2011 Sundance Film Festival, etc. The Tweets11 corpus was provided to participating teams in an atypical manner when compared to other TREC corpus distributions. Due to Twitter's Terms of Service, each participating team had to download the corpus themselves using either the Twitter API or an HTML crawl of the tweets. Participants were given a set of files which contained tweets in chronological blocks of ten-thousand in the following format:

```
tweet-id author-screenname MD5-hash-of-tweet
```

Using this the URL to a particular tweet can be reconstructed for the purposes of crawling the tweet page, e.g. `https://twitter.com/#\protect\kern-.1667em\relax/author-screenname/status/tweet-id`. Using the Twitter API all that is needed is to send in a request for the JSON formatted version of the tweet. Using the Twitter API method easily provides much more information, e.g. author information is returned as part of the JSON format. However, Twitter restricts the number of API calls to 350 per hour for certain features of the API, like those for retrieving specific information regarding a tweet or a Twitter user. Thus, retrieving the tweets using the API is not feasible for those who do not have legacy whitelisted[1] accounts or IP addresses, which are accounts or IP addresses that Twitter allows to make up to 20,000 API calls per hour.

The collection of tweets used in the work presented in this thesis was collected with the provided HTML crawler[2]. The provided crawler was ran from May 27th to June 3rd.

[1] Whitelisting stopped in early 2011 before the Microblog Track distributed the corpus

[2] Available at `github.com/lintool/twitter-corpus-tools`

Table 3.1: Distribution statistics for the University of Waterloo version of the Tweets11 corpus

HTTP Status	Num. of Tweets	Num. null text
200	14,313,888	14
302	1,137,183	25,571
403	138,560	138,560
404	552,181	552,181

Due to the timeouts that can occur when crawling webpages, repair code was provided to refetch any tweets that timed out in an initial run. It was determined on the Microblog Track mailing list that the HTML crawler returned tweets with null text bodies, which means that while a tweet was deemed to be fetched the text of the tweet was not present in the archived tweet. Accordingly, the supplied repair code was modified (by the author of this thesis) to re-fetch and scrape all tweets with null bodies. The modified repair script was run six times as continuing to try to fix any remaining tweets was deemed to interfere with usage of the data. Table 3.1 shows the distribution of tweets in the copy of the corpus used in this thesis. HTTP Status codes have a particular meaning: 200 are regular tweets, 403/404 denote tweets that were deleted or are otherwise inaccessible to the public at large. The 302 status code is a bit misleading, initially it was meant to denote simple retweets and the according redirect that linking to a retweet causes. However, 302 in the HTML version of the corpus also denotes tweets that redirect due to a user's change in screenname and in the JSON version of the corpus would have been tagged as having a status of 301. This issue did not become apparent until later when the corpus was being used in experiments. Accordingly, the affected tweets were not repaired as doing so would have required recrawling the affected portions of the corpus and this was not done due to time constraints.

No further attempts were made to retrieve the remaining 25,585 null text body tweets as it was not clear whether or not the tweets were still available, e.g. not protected or deleted, as the HTML crawler and the repair code only checks for the status code during the initial crawl. In addition, it was believed that the missing tweets would not severely affect the performance of the retrieval systems used.

3.2 Experimental Setup

3.2.1 Search Methodology

The search methodology used in this thesis approaches the previously outlined task by applying general use retrieval methods shown to work on other tasks to microblogs, specifically Twitter. To this end the Wumpus Search Engine[3], developed by Stefan Büttcher while at the University of Waterloo, is used as the main retrieval engine for the runs conducted in this thesis. While there are many possible search engines out there, Wumpus was designed for efficient indexing and retrieval in a large corpus setting under which the Microblog Track falls and was designed with the earlier TREC Terabyte track in mind. While Wumpus treats a file as a document for retrieval purposes, this is potentially unwieldy when dealing with 16 million files. In addition, Wumpus supports an XML-like tagging system that allows multiple documents to be stored in a single file. Accordingly, to simplify storage and management of tweets it was decided that a single file be used to store all of the tweets in chronological order. The following is an example of how a single tweet was stored using the XML-like tagging structure of Wumpus:

```
<doc><docid>tweet id</docid><text>tweet text</text></doc>
```

This compact format allows all tweets to be stored in a single file 1.6GB file. Since each topic has a specific query tweet time then it was trivial to split this file up into 50 smaller files based upon the query tweet time for each topic. This disallowed any future information to leak from tweets that appear chronologically later in Tweets11 corpus. In doing so, every run presented in this thesis uses no future information.

These sub-corpora for each topic are then able to be used by Wumpus to search for tweets relevant to the corresponding topic. However, Wumpus is a customizable search engine that allows different features to be used when processing queries. For example, words can be stemmed or tweets can be treated as a sequence of character n-grams. Using combinations of different features can result in the creation of different indexes and accordingly six different indexes were used in the runs presented in this chapter. Table 3.2 describes the features that were used to create the six different indexes.

Each topic could have a query issued against each of the six indexes created for it, however, the issue of ranking relevant tweets then becomes important. Unfortunately, there does not exist a perfect ranking method for retrieval and accordingly Wumpus allows

[3]Available at `http://www.wumpus-search.org/`

users to rank retrieved results based upon a number of different ranking methods. For the purposes of participation in the Microblog Track, 5 different ranking methods were used to rank the relevance of tweets to a topic and are outlined in Section 3.2.2.

This means that for each of the 6 indexes the results of the query are ranked 5 different ways which creates 30 sets of results for a particular query. It has been shown that a "wisdom-of-the-crowds" approach to determining relevance of an object tends to produce better results than any one method. Accordingly, these 30 sets of results are combined into a single definitive results using Reciprocal Rank Fusion (RRF) [24]. The algorithm for RRF is as follows:

$$RRFscore(t) = \Sigma_i \frac{1}{k + r_i(t)} \tag{3.1}$$

where $r_i(t)$ is the rank of the tweet t in result set i and k was set to 60 as this has been to shown to work well in web search.

RRF was chosen because it has been shown to perform well against other "meta-ranking" functions and is exceedingly simple to implement as it only requires a document's rank (e.g. it's place in the result list) and not a score. The reason combining results tends to be easier with rank as opposed to score is that scores may be generated in vastly different magnitudes and so scores need to be normalized, which may not be trivial.

Once the results for each topic are re-ranked according to RRF, the top 30 highest scoring tweets are taken (according to their RRFscore) and sorted into chronological order for most to least recent and returned as the result set for that run. In doing so, the spirit (but not the letter) of the track guidelines is being partially ignored as the intent of the task was to have the 30 most recent but relevant results returned. The intent was ignored because it was thought that a user will likely prefer more relevant but slightly order tweets to less relevant newer tweets.

This ends the general outline of how retrieval was performed for the Microblog Track task. However, there are many ways queries and the result sets can be modified to attempt to improve performance. Section 3.2.3 provides an outline of the runs submitted to TREC and runs performed after the submission deadline that implement some possible modifications. The evaluation procedures of runs will be discussed in Section 3.2.4 and the results of which will be presented and discussed in Section 3.3.

Table 3.2: Descriptions of the features used to build several indexes for Wumpus to use in the retrieval of tweets

Index	Description
1	This index includes stemmed and unstemmed versions of words present in the tweet. The stemming is accomplished using a Porter stemmer. In addition, word bigrams are included in the index.
2	Identical to Index 1 but with no word bigrams present.
3	This index contains only unstemmed versions of words and word bigrams.
4	Identical to Index 3 but contains no word bigrams.
5	Instead of words as before, character 3-grams are used in the index. Subsequently, stemming is not used as it has no practical benefit.
6	Identical to Index 5 but uses character 4-grams.

3.2.2 Ranking Methods

Okapi BM25

BM25 is a probabilistic method of determining a document's relevance to a query using a bag-of-words model, e.g. BM25 can be seen as asking "How probable is it that if one were to randomly select words from the document that a query term would be selected?". BM25 first appeared at TREC-3 [42] in 1993 and has continued to be used as a baseline in further TREC conferences as well as in information retrieval at large due to its relative simplicity and ability to perform well. Due to it's popularity in information retrieval domains only a brief outline of the formula implemented by Wumpus is provided. Given a query $\mathcal{Q} =< t_1, \ldots, t_n >$ and a document D, the BM25 score is computed by Wumpus as:

$$score(\mathcal{Q}, D) = \sum_{i=1}^{n} \log\left(\frac{|\mathcal{D}|}{|\mathcal{D}_{t_i}|}\right) \cdot \frac{f_{t_i,D}(k_1+1)}{k1((1-b)+b(|D|/l_{avg}))+f_{t_i,D}} \tag{3.2}$$

where $\mathcal{D}$ is the set of all documents, $\mathcal{D}_t$ is the set of all documents containing the query term t, $|D|$ is the number of features (words, n-grams, etc) in D, l_{avg} is the average document length, and $f_{t,D}$ is the frequency of term t in document D. The parameter b controls

the affect of document length normalization; k_1 controls the affect of term frequency within in the document, e.g. higher k_1 will give greater influence to the term frequency within the document. The default values for these parameters in Wumpus are $(k_1, b) = (1.2, 0.75)$, which are generally the default values used for BM25 in other applications. The runs presented in this thesis use the default parameter values unless otherwise stated as this reflects the "out of the box" intent of this thesis. The first part of the product represents what is referred to as the *inverse document frequency (IDF)*, of which there are several different possible formulations. A term's IDF controls the affect of how rare the term is in the collection. The second part of the product is called the *term frequency* portion of BM25.

Ponte-Croft Language Modelling

A language model for a document D is a set of probabilities for terms that says, "How likely is it term t was entered by a user to retrieve d?" The simplest such model is the maximum likelihood model based on term frequency in the document, namely:

$$\mathcal{M}_D^{ml}(t) = \frac{f_{t,D}}{l_D} \tag{3.3}$$

where $f_{t,D}$ is as usual the term frequency of t in document D and l_D is the length of document D. For terms not in D, then $\mathcal{M}_D^{ml}(t) = 0$. So for each document, the probability of a term being contained in that document can be calculated. Thus, if it is assumed that query terms are independent from each other then given a query $\mathcal{Q} =< q_1, \ldots, q_n >$ and a document D, then

$$Pr(\mathcal{Q}|D) = \prod_{i=1}^{n} p(q_i|d) \tag{3.4}$$

Then if we estimate $p(q_i|d)$ by $\mathcal{M}_D^{ml}(q_i)$, the probability is:

$$Pr(\mathcal{Q}|D) = \prod_{q \in \mathcal{Q}} \left(\mathcal{M}_D^{ml}(q)\right)^{Q_q} \tag{3.5}$$

where Q_q specifies the query-term frequency when the query vector $\mathcal{Q}$ is treated as a set. Note that such simplification is not necessary for the purposes of ranking but it allows further simplification through the use of logarithms (not presented in this thesis).

Ponte and Croft[60] find particular issues with this model. Namely, when a query term is not in a document then the entire score of that document would be 0 by Equation 3.5.

This is undesirable behaviour as missing a query term should not immediately remove a document from contention. For such terms Ponte and Croft suggest using $\mathcal{M}_{\mathcal{C}}(t) = \frac{l_t}{l_{\mathcal{C}}}$, where l_t is the term frequency of t in the entire collection $\mathcal{C}$ and $l_{\mathcal{C}}$ is the number of terms in the entire collection, which makes the probability of seeing the term in a document no better than the chance of seeing it in the entire collection.

Further, Ponte and Croft suggest that the since only a document sized sample can be retrieved for $\mathcal{M}_D$ then a more robust measure can be used. The proposed measure is $p_{avg}(t)$ which is the average probability of seeing a term t in all documents in the collection that contain t. Obviously, such a measure cannot be used to replace $\mathcal{M}_D(t)$ since it would assume the same language model for different documents. To compensate for this, Ponte and Croft propose using a risk function $R_{t,d}$ (omitted for brevity) that reflects the fact that as term frequency moves away from the normalized mean then $p_{avg}(t)$ becomes riskier to use.

Using $R_{t,d}$ as a mixing parameter, Ponte and Croft calculate $\mathcal{M}_D(t)$ as:

$$\mathcal{M}_D(t) = \begin{cases} \mathcal{M}_D^{ml}(t)^{(1.0-R_{t,D})} \times p_{avg}(t)^{R_{t,D}} & \text{if } f_t > 0 \\ \mathcal{M}_{\mathcal{C}}(t) & \text{otherwise} \end{cases} \tag{3.6}$$

Then Ponte and Croft calculate the score for a query $\mathcal{Q} =< q_1, \ldots, q_n >$ and a document D by:

$$Pr(\mathcal{Q}|D) = \prod_{q \in \mathcal{Q}} \mathcal{M}_D(q) \times \prod_{q \notin \mathcal{Q}} 1 - \mathcal{M}_D(q) \tag{3.7}$$

where the first product is the probability of generating a term in the query and the second is the probability of not generating terms not in the query.

Language Modelling with Dirichlet Smoothing (LMD)

This method is based upon the same maximum likelihood model as Ponte and Croft's method, however, Zhai and Lafferty use Bayesian smoothing with Dirichlet priors [78]. Rather than giving the full breakdown of the derivation of the formula, the intuition is presented. The idea is to pretend an extra $\mu > 0$ terms are added to each document and distributing them according to $\mathcal{M}_{\mathcal{C}}(t)$. Thus, in this pretend maximum likelihood model no document can have 0 occurrences of a term. In doing so, one of the primary concerns of Ponte and Croft is eliminated (that of 0 occurrences affecting the probability). Thus, using this pretend model a score can be generated for a query $\mathcal{Q} =< q_1, \ldots, q_n >$ and a document D by:

$$Pr(\mathcal{Q}|D) = \sum_{i=1}^{n} \left(\frac{f_{q_i,D} + \mu \mathcal{M}_{\mathcal{C}}(q_i)}{l_D + \mu} \right) \tag{3.8}$$

Cover-Density Ranking (CDR)

CDR is a form of proximity ranking proposed by Clarke et al. that relies upon the proximity of query terms in a document and not explicit term frequencies [20]. The version of CDR presented here is a modified version that is present in Wumpus. Given a set of query terms $\mathcal{Q} =< q_1, \ldots, q_n >$ create all possible combinations of subsets of $\mathcal{Q}$, e.g. $\{q_1, q_2, q_n\}$, $\{q_1, \ldots, q_n\}$, of which there are 2^n possible subsets. Weights are computed for each subset by summing the IDF values for each of the query terms in the subset and sorting the subsets according to this weight. Each document is placed into a set corresponding to the largest subset of query terms they contain. Thus, each document corresponds to exactly one set and a document from a set with a higher subset weighting will be ranked above those with a lower subset weight, where the set corresponding to all query terms is ranked the highest.

This gives an approximate ranking for how "relevant" a particular subset is considered; that is, subsets containing more and/or rarer query terms will be ranked higher. What follows is a description of how documents are ranked internally in a set. Given a document $D =< d_1, \ldots, d_n >$ and a query $\mathcal{Q} =< q_1, \ldots, q_n >$, a cover is an interval [p,q] such that all query terms present in D are occur in this interval and where there does not exist a smaller interval containing the same query terms. Scoring of a document is based upon the idea that shorter covers will be more likely to contain relevant text, and the more covers a document has the more likely it is to be relevant. Thus, given $\mathcal{C} = \{\{u_1, v_1\} \ldots \{u_n, v_n\}\}$ which is a set containing all covers in a document D for a query $\mathcal{Q}$, the score of D is calculated by:

$$score(\mathcal{C}) = \sum_{i=1}^{n} I(u_i, v_i) \tag{3.9}$$

where

$$I(u, v) = \begin{cases} \frac{\mathcal{K}}{v-u+1} & \text{if } v - u + 1 > \mathcal{K} \\ 1 & \text{otherwise} \end{cases} \tag{3.10}$$

The parameter $\mathcal{K}$ affects the influence of cover length, where covers shorter than $\mathcal{K}$ have value 1, and longer covers are assigned scores less than 1 in proportion to the inverse of their length. For the experiments presented in this chapter, the Wumpus default value of $\mathcal{K} = 16$ was used.

Divergence from Randomness (DFR)

Fundamentally, DFR differs from BM25 and LMD because it does not use unintuitive parameters (e.g. "ideal" values of b, k_1, etc, are determined through experimentation). That

is, DFR is non-parametric. DFR can be summarized by the formula $(1 - P_2 \cdot (-\log P_1))$, where P_1 is the probability that a random document D contains exactly $f_{t,D}$ occurrences of term t, and $-logP_1$ can be seen as the number of bits associated with D containing exactly $f_{t,D}$ occurrences of t. P_2 can be viewed as the probability of seeing at least one more instance of t in D after seeing $f_{t,D} - 1$ occurrences in D and accordingly as $f_{t,D}$ increases so to does P_2. Thus, $(1 - P_2)$ dampens the effect that an increasing $f_{t,d}$ will have.

Amati and van Rijsbergen[2] provide methods for estimating P_1 and P_2. The details of this estimation are left to the reader to investigate if they desire and a summary of the ideas behind the estimation of P_1 and P_2 is presented. Let n_t denote the number of occurrences in all N documents in a corpus and $f_{t,D}$ still be the number of occurrences of t in D. Then P_1 can be seen as the fraction of all possible distributions of the remaining $n_t - f_{t,D}$ occurrences of t over all possible distributions of n_t over the whole corpus. P_2 is estimated using Laplace's law of succession. From this the following formula can be derived for a term t and document D:

$$(1 - P_2) \cdot (-\log P_1) = \frac{f_{t,\mathcal{C}} + 1}{n_t \cdot (f_{t,D} + 1)} \cdot f_{t,D} \log \frac{N + 1}{f_{t,\mathcal{C}} + 0.5} \tag{3.11}$$

where $f_{t,\mathcal{C}}$ is the number of times t occurs in the collection $\mathcal{C}$. However, this does not account for differing document lengths. Amati and van Rijsbergen investigate different ways of normalizing document length but the best performing normalization is done using:

$$f'_{t,D} = f_{t,D} \cdot log(1 + l_{avg}/|D|) \tag{3.12}$$

Taking the sum over each term in a query will determine a score for a query. In doing so, this is how Wumpus generates a score for a document using DFR.

3.2.3 Runs

In this section we describe the official and unofficial runs that were conducted. Runs 1 through 4 were the runs submitted to NIST for official evaluation and so the run ID for each of these is given.

Run 1 - Baseline (waterlooa1)

The baseline run performs exactly as outlined above with no additional changes. That is, the 5 ranking methods were used on each of the 6 indexes and RRF was used on the 30 sets of results. This provides an out of the box baseline upon which further modifications were made.

Run 2 - HTTP Status Code 200 Only (waterlooa2)

This run is identical to the baseline run except with a significant difference in the indexes used. Each index was pruned to remove any tweet that did not have a status code of 200. The intent was to improve the return of highly relevant results as simple retweets could not be returned and would not influence the statistics used in the ranking methods (e.g. term frequencies, etc). It is noted here that the issue of 301/302 tweets was not known when this run was completed and submitted.

Run 3 - Query Expansion (waterlooa3)

This run uses pseudo-relevance feedback (PFR) to perform query expansion. A brief outline of PFR and the methods used follows. PFR has the following four basic steps: (1) Execute initial query; (2) Select the top scoring k documents; (3) Score the terms appearing in selected documents and select the top m terms, ignoring terms from the initial query; (4) Add new terms to the initial query, adjust weights as applicable, and execute search with the expanded query; (5) present the results to the user. For the purposes of brevity only the two methods of scoring terms in the selected documents will be discussed further.

Due to limitations of Wumpus, an external language model (e.g. term frequencies, etc) is recommended when using PFR and accordingly all values discussed are obtained from this external language model. Wumpus implements an Okapi-styled version of scoring based on work done by Robertson [62]. A term's selection score for a set of R selected documents is given by:

$$score(t) = n_{t,R} \cdot \log \frac{|D^{FB}|}{|D_t^{FB}|} \tag{3.13}$$

where $n_{t,R}$ is the number of selected documents in R containing term t, D^{FB} is the set of all documents in the feedback language model and D_t^{FB} is the number of documents in the feedback model containing t. Additionally, Wumpus implements a second method of scoring based on Kullback-Leiber Divergence (KLD)[13] and the details of the derivation are omitted for the sake of brevity. KLD-based term scoring calculates a score as follows:

$$score(t) = \frac{n_{t,R}}{|R|} \cdot \log \frac{|D^{FB}| \times n_{t,R}}{f_{t,D^{FB}} \times |R|} \tag{3.14}$$

where $|R|$ is the number of documents in R, $f_{t,D^{FB}}$ is the term frequency of t in the feedback language model, and all others are the same as for Equation 3.13.

As stated previously this run uses PRF. To perform PRF the GOV2 corpus (from the TREC 2009 Terabyte Track) was used as the feedback language model, with KLD-styled and Okapi-based term selection being performed. Wumpus has default values of k=15 and m=8 which were used whenever PFR was performed in this thesis. Due to some internal Wumpus limitations, CDR could not be performed for this run or any run performing PFR. Accordingly, this resulted in a total of 48 result sets being used rather than the standard 30. From the use of RRF, these additional 18 result sets could have improved the final results returned, however, it was thought that the impact of these additional result sets would not be overly large. In addition, since this run uses the GOV2 corpus it is considered to use external evidence by the Microblog Track standards.

Run 4 - Recency Blending(waterlooa3)

This run takes the results of the Run 1 and blends the results with respect to how recent the tweet is. The goal here to use the idea that the search request likely happens when the particular topic is (relatively) popular and so more recent tweets are more likely to be relevant to the topic. Namely, each tweet, t, is re-scored as follows:

$$recscore(t) = \left(\frac{\text{tweetid}(t)}{\text{max-tweetid}}\right) * \text{run1-score}(t) \tag{3.15}$$

where max-tweetid is the most recent tweet ID by at time of the query and run1-score is the score determined for a tweet in Run 1.

Run 5 - Conjunctive Query

This run modifies the baseline by creating a new set of query terms by taking the pairwise conjunction (read: boolean AND) of the original query terms. The intent of this run is to force tweets to be returned with more query terms present in retrieved tweets. Note that CDR was ran as normal for this run as the version in Wumpus restricts the number of query terms to 8 and this method of pairwise conjunctions can easily break that limit. Further, leaving out CDR was deemed to not be beneficial as the run would have fewer result sets and thus impact the use of RRF.

Run 6 - Stopword Removal

This run removes all stopwords (where stopwords come from the list available at `http://www.lextek.com/manuals/onix/stopwords1.html` from the query terms (excepting those

that are part of quoted terms, e.g. "The Quick Brown Fox" would have no stopwords removed) as a means to remove the possibility of Wumpus returning tweets that are relevant to stopwords. In doing so, it was believed that query results may improve.

Run 7 - Tweet-based Query Expansion

This run builds upon the theory of Run 3 by using the all the tweets up until the oldest query tweet time (e.g. the earliest query) and builds a Wumpus-style language model using those tweets. The intent here was to see how much effect the language model has on the results returned. This run was performed as a direct result of the improvement seen in the original PRF run in the official TREC Microblog results. As well, this run is effectively a "clean" run in that it contains no external or future evidence unlike the original PFR run which contained external evidence.

Run 8 - Conjunctive Query and Stopword Removal

In an effort to boost the effectiveness of Run 5 and Run 6, all stopwords were removed before taking the pairwise conjunction as this would remove silly conjunctions, like '(at^the)'[4].

Run 9 and Run 10 - Query Expansion using Okapi-style feedback only

These two runs are reproductions of Run 3 and Run 7, respectively, with the KLD-based PFR queries removed, meaning that only 24 sets of results were used in the computation of the final ranking. The purpose of this run is to see how much effect KLD-based PFR had on the original results, e.g. to determine how much affect the 18 additional queries had due to the usage of RRF.

3.2.4 Evaluation

TREC evaluation is performed by NIST assessors (generally former analysts), who use a technique, called pooling, to reduce the number of judgements needing to be made. Pooling is done by taking union of the top k documents returned (k=100 has been used in the past) by each group for judgement. Documents outside this group are considered non-relevant. For the Microblog Track, participating teams were allowed to select 2 runs which would

[4]Mountains of Madness

be used for pooling (Run 1 and Run 3 were the selected runs used when the University of Waterloo runs were submitted). Tweets were judged on a ternary scale of not relevant, relevant, and highly relevant, where highly relevant was meant to denote tweets that are extremely "interesting" for the particular topic. The pooling of tweets resulted in just over 40,000 relevance judgements. However, there was a initial pooling snafu were results were pooled by score and not recency and so the results were repooled by recency and any new tweets in the pool were judged. This resulted in 20,000 new judgements of which only 100 new relevant and highly relevant tweets were found in total. Out of the 50 topics, 49 were found to have relevant tweets and 33 were found to have highly relevant tweets. What this means is that 13 of the topics with relevant tweets had no highly relevant tweets and topic 50 had no relevant or highly relevant tweets at all. It is worth noting that out of the relevant tweets, 14 were not available in the University of Waterloo version of the Tweets11 corpus used in the runs in this chapter. What is particularly noteworthy is that only 2 of the 14 tweets fell into the group of 25,000 tweets with the 302 status code that could potentially have been retrieved had the modified repair code been run some number of additional times. The remainder were 403/404 status code tweets which means that those tweets were unavailable before the University of Waterloo version of the Tweets11 corpus was collected.

The official 2011 Microblog Track evaluation was *Precision@30* (P@30), which is calculated by taking the sum of the number of relevant results returned in the 30 most recent tweets for each topic and dividing by the potential number of relevant for each topic, e.g. 1470 for all relevant tweets and 990 for highly relevant tweets only. P@30 was reported when all relevant tweets (both relevant and highly relevant combined) and when only highly relevant tweets were considered. There was found to be very little difference in the rankings of participant systems before and after the re-pooling [5]. This thesis reports for each run the percentage of topics were the system scored above the median P@30 value, the percentage where the P@30 was equal to the median, and the average number of relevant tweets returned per topic. The results are presented in this manner to better examine trade-offs and improvements from run to run. In addition, P@30 can be recovered (with some floating point error) from the average relevant returned by multiplying the value by 49 (33) and then dividing by 1470 (990) for all relevant (only highly relevant) tweets.

[5]Ian Soboroff reports that a Kendall's tau of 0.9933 (0.9977) was found when all relevant (highly relevant) tweets were considered. Via `http://groups.google.com/group/trec-microblog/browse_thread/thread/f4b0cfe5d70f6bfd` on February 14, 2012

3.3 Results and Discussion

Table 3.3 and Table 3.4 provide summary results for topics with highly relevant tweets only. Table 3.5 and Table 3.6 provide summary results for topics with all relevant tweets (Appendix A has the per topic results for all runs for both highly relevant and all relevant tweets). Regardless of which set of assessments was used the run results presented were unchanged. From these summary results, the baseline run appears to have performed most of the time at or above median, which is a possible indicator that it is a useful baseline for comparison. The second run performed not as well as hoped, however, this is likely due in part to the 301/302 issue mentioned previously (as valid tweets are removed due to the mislabelling) as well as the fact that all ranking methods used incorporate some idea of term frequency both at a inter-document and intra-document level. Thus, by removing tweets from the corpus the term frequency of query terms is changed and this potentially affected retrieval results. The improvement in Run 3 was expected but not the magnitude of improvement as pseudo-relevance feedback can be dangerous to use due to the automatic selection of "relevant" documents. While Run 4 did not greatly improve results over the baseline, there was a slight improvement in performance. The results of Run 4 would seem to provide some evidence that relevant tweets may be more likely to occur closer to the query time as was predicted.

Using a binomial test of significance on discordant documents, the official runs were compared to the baseline to determine if any improvement was significant. Run 3 appears to yield significant improvement when both highly relevant tweets only ($p = 0.016$) and all relevant tweets ($p = 0.00004$). Since three hypotheses were tested at once, Bonferroni[6] correction [11, Ch. 12] is applied, and it is found that both improvements are still significant ($p = 0.048$ and $p = 0.00012$, respectively). The difference between the other 2 official runs and the baseline is not significant before any Bonferroni correction is used.

Using the same test and after Bonferroni correction, Run 7 and Run 10 significantly improve upon the baseline, both when highly relevant tweets only and all relevant tweets are used. In addition, Run 9 appears to improve on the baseline; however, it is up for debate whether the improvement when only highly relevant tweets are considered is significant, when corrected for multiple hypothesis testing. A strict application of Bonferroni (which is known to be conservative) would say that it is not.

The results would seem to indicate that both Run 5 and Run 6 did not significantly improve upon the baseline. This unfortunately extends to Run 8, which was the combination of methodologies for Run 5 and Run 6. Run 8 appears to have marginally improved upon

[6]Not to be confused with a particular canned pasta

Run 6 with respect to all relevant tweets but has decreased performance when compared to Run 5. While run 8 appears to have returned more highly relevant tweets than either of its related runs it would appear that it did so at the expense of returning tweets at or above the median (see Table 3.4). It is the case that Run 8 was able to find more tweets per topic but found tweets for fewer topics. This is unlikely to be useful at it means that the methodology behind Run 8 is not robust and accordingly would not be beneficial in the real world, where returning more tweets for a few topics may be less desirable than returning a minimum number of tweets to all topics.

The results of Run 3 and Run 7 would seem to indicate that the language model used for query expansion does matter. In particular, the tweet-based language model for query expansion performs quite a bit better than the baseline and still appears to give some improvement over the initial pseudo-relevance feedback run. This would indicate that a better tweet-based language model, e.g. consisting only of English tweets, may improve feedback results. It was initially thought that a language model from a larger collection of tweets would improve results, however, using the entirety of the Tweets11 corpus did not produce different results than were produced in Run 7. Furthermore, applying recency blending to Run 7 had a marginal performance boost but the boost did not warrant reporting in this thesis.

Perhaps the most interesting aspect of the results for all of the runs considered is that while the pseudo-relevance feedback runs are able to find more highly relevant tweets it is done at the expense of topic breadth as can be seen from examination of the summary tables. That is, runs without pseudo-relevance feedback were able to meet or exceed the median number of tweets returned more often than runs that used pseudo-relevance feedback. While this difference may not be significant it does give rise to the possibility that pseudo-relevance feedback may not be as robust for finding highly relevant tweets and will sacrifice finding tweets in one topic to find more in another and this may not be beneficial. On the other hand, this issue is not apparent when all relevant tweets are considered. Thus, it may the case that not using pseudo-relevance feedback may be useful if a a user wants to be able to find a minimum number of highly relevant tweets.

Furthermore, the results of Run 9 and Run 10 would seem to indicate that the additional results provided by the KLD-based pseudo-relevance feedback do not appear to significantly affect the retrieval results. There is a minor drop in the average number of relevant results returned at 30 (see Table 3.5 and Table 3.6). However, this drop is not nearly dramatic enough to indicate that the additional set of results significantly increased the effectiveness of Run and Run 7. What is perhaps more interesting is that Run 9 has a higher percentage of topics with relevant tweets returned at or above the median than does Run 3 but the opposite occurs for Run 7 and Run 10. This may indicate that the choice of language

model used can affect how tweets are returned in unforeseen ways.

During the presentation of the results for the Microblog Track at TREC 2011, it arose that a BM25 run with b and k1 both set to 0 was able to perform extremely well, outperforming all of the official runs presented in this thesis and many of the unofficial runs, except Run 7[7] and Run 10. The reasoning behind setting both parameters to 0 is that due to the brevity of tweets the term frequency is not very important as a term cannot be repeated a large number of times while still being relevant to a topic. Thus, a document's score is based upon the IDF value of query terms alone. This idea had merit and while it was not possible to replicate the results presented using Wumpus (likely to due to how Wumpus breaks ties, as the group's system broke ties based upon recency where Wumpus breaks ties arbitrarily), an increase in improvement was seen over the default BM25 and BM25 with 0 valued parameters was seen. Accordingly, Run 7 was run with the parameters set to 0 and non-trivial change can be seen in Table 3.7 (no hypothesis testing was performed as there was no prior hypothesis). This change would make Run 7 the second best performing system at the Microblog Track (with respect to all relevant tweets) with the top performing system having a very slim lead of 0.1 in P@30. However, such a change is Twitter specific and may not perform well in other contexts where more than 140 characters are permitted but there is still a character limit. The value of 0 for b and k1 in BM25 is just a guess and there is no evidence to show that it is ideal for Twitter. In fact, it may very well be the case that there are ideal values for k1 and b that lay between 0 and the corresponding default values and empirical testing would need to be done to find the ideal values. The change may also not be necessary given how well Run 7 performs by itself and so it may be the case that further attempts to improve pseudo-relevance feedback may be more beneficial and widely applicable. In addition, this changes appears to negatively affect the finding of highly relevant tweets from a breadth and depth perspective as it does not perform nearly as well in that respect to the default Run 7. Thus, the change appears to be dependent on the type of tweets one desires and finding highly relevant tweets does not appear to be a suitable task for this change in the BM25 parameters.

3.4 Limitations and Potential Improvements to the Microblog Track

The first iteration of the Microblog Track was not perfect, which is perhaps an unsurprising fact. One of the most controversial aspects of the Microblog Track was the corpus distri-

[7]Run 7 is unofficially in the top 5 of the Microblog results according to P@30

bution since Twitter is not accessible from all areas of the world and the Twitter Terms of Service prohibit redistribution. This may indicate that looking at microblogging services that are more friendly to redistribution should be examined.

There are also some problems with the 2011 format of the Microblog Track. Namely, the poor definitions of "highly relevant" and "interesting" (or perhaps lack thereof of any real definition of either term). This left participants looking at example topics and trying to guess what the organizers and assessors would deem to be relevant and highly relevant, which is never ideal. Realistically, the track didn't perform all that well as no team was able to even return 15 relevant documents per topic on average. Whether this is a failing of the corpus, or the organizers, or the participants is hard to tell but when a third of the highly relevant topics have at most 5 highly relevant tweets and just under half of the topics, when all relevant tweets are considered, have less than 30 relevant tweets then the corpus is thrown under a bus. Accordingly, it may be necessary in additional iterations of the Microblog Track to have a larger corpus so that such issues are not as prominent.

As mentioned in Section 3.2.4 12 relevant tweets were marked with HTTP status 403/404. As the corpus was collected fairly soon after it was released it was hoped that this would not happen. While it is likely that missing these 12 tweets did not terribly affect the performance of the presented runs, it is possible, however, that teams who collected the data set later would be penalized for doing so. Thus, a true comparison of system performance may require that any HTTP status 403 or 404 tweets from any group be removed from the relevant set of tweets. But this could substantially affect the number of relevant tweets, which at this time seems rather low. In any future iterations of the Microblog track, it would be desirable if a stable data set could be used or enforced.

Another issue that arose is that "novelty" ended up having some effect on the results as simple retweets (or retweets with no additional content) were deemed not relevant from the onset. This was done by the organizers with the thought that for any simple retweet the original tweet was already (potentially) seen by the user[8]. But this is not necessarily the case as a Twitter user may just be introduced to a topic and so would have no prior knowledge. Accordingly, it is likely the case that in further iterations some limited form of "novelty" should be used such that the first occurrence of a retweet is treated as the canonical form of the tweet (if the original tweet is not present) and all others are duplicates of it. In doing so, duplicate tweets can also be removed (where duplicate tweets are from Twitter users using the "Tweet this!" button on a website that create a tweet with a basic message and link to the webpage to be posted by the user) or treated in a manner similar to retweets.

[8]The thread 'Evaluation Clarification' on the Microblog Track mailing list outlines this discussion

3.5 Future Work

From the performance of Run 7, the utility of a better tweet-based language model should be investigated to see if performance can be increased further. For example, the use of a tweet-based language model composed of English tweets only may be useful as this should correspond better to the task of finding relevant English-only tweets for the provided topics. In addition, a dynamic language model should be explored for utility as run 4 indicates that recent tweets may be more likely to be relevant to a topic. There are two readily apparent choices for a dynamic language model: one that scales to the query time stamp (and thus becoming increasingly larger for more recent topics) and one that shifts to maintain a language model consisting of the k most recent tweets for each topic. The reasoning behind the second proposed dynamic language model is that the language used to describe a topic may change over time and using a large selection of older tweets may harm the term selection as the older tweets could overpower the more recent tweets. While this may not be significant if the the time range of the corpus is small, like for the Tweets11 corpus, it is potentially an issue if a larger range of tweets is considered.

Due to the increasing usage of hashtags as part of the Twitter experience, it was initially thought that hashtags may be used as a query method. However, initial testing on the example topics showed an inconsistent pattern of usage, e.g. some topics had very high usage and some had little to no usage. Thus, hashtag queries were eliminated for use in the automatic runs presented in this thesis. In examining the set of relevant tweets, a similar pattern was seen where some topics had reasonable hashtag usage and others did not. While this still prohibits an automatic search system, it may be of some use to do a manual run where hashtags are selected from initial query results. In addition, it may be possible to perform pseudo-relevance feedback styled term selection for hashtags.

There was an initial plan to crawl the URLs contained in tweets and use the first 10,000 bytes as an additional source of information, however, due to time and storage constraints this was unable to be accomplished. Since approximately 82% of the relevant tweets contain at least one URL, such an endeavour may have a non-trivial effect on the retrieval of tweets. Such extra information may help to boost the retrieval results (in fact, linking to relevant content made a tweet relevant or potentially highly relevant depending on the content). In doing so the system may make use of external and possibly future information which may be undesirable. Due to the fact that most tweets do not contain full-length URLs (they are shortened using a URL shortening website which redirects the user to the original URL), it may be sufficient to resolve shortened URLs into their full length version and use that as a source of additional information while still respecting the external and future evidence restrictions. For example, two tweets which contain a link to

Table 3.3: Comparison of the average number of highly relevant tweets returned and the percent of topics that have meet or exceed the media number returned at 30 for Official Runs

	Official Runs			
	1	2	3	4
Percent above median	54.55	51.52	66.67	54.55
Percent at median	36.36	39.39	18.18	36.36
Avg. rel. ret. 30	3	2.88	3.45	3.03
p-value (vs baseline)	-	1.00	0.016	0.5

the same URL may indicate a duplication of information (and so only 1 tweet needs to be returned if novelty is considered) or if a tweet deemed not relevant contains a URL present in a tweet deemed relevant then the tweet deemed not relevant may in fact be relevant or vice versa.

As mentioned previously, an avenue of potential future work is determining ideal values for BM25's k1 and b parameters. This is especially true for the highly relevant case where the modified Run 7 performed horribly. In fact, it may be the case that the ideal values for finding highly relevant tweets differs from those for finding relevant tweets. In addition, LMD and CDR have parameters and so the next logical step would be to determine the ideal value(s) for them. However, in doing so Twitter specific values are found and such values may not be applicable to other microblogs or other micro formats (say, microreviews). This would suggest that it would be beneficial to see how well the Run 7 method performs in other microblogging systems (or micro formats). Furthermore, it may also be beneficial to determine if the ideal values for the parameters of BM25, CDR, and LMD are consistent across different microblogging systems.

Table 3.4: Comparison of the average number of highly relevant tweets returned and the percent of topics that have meet or exceed the media number returned at 30 for Unofficial Runs

	Unofficial Runs					
	5	6	7	8	9	10
Percent above median	57.58	57.58	66.67	60.61	63.64	69.70
Percent at median	33.33	33.33	21.21	27.27	24.24	15.15
Avg. rel. ret. 30	3.09	3.09	3.82	3.30	3.45	3.73
p-value (vs baseline)	0.34	0.19	0.00005	0.055	0.013	0.003

Table 3.5: Comparison of the average number of all relevant tweets returned and the percent of topics that have meet or exceed the media number returned at 30 for Official Runs

	Official Runs			
	1	2	3	4
Percent above median	73.47	73.47	77.55	77.55
Percent at median	12.24	12.24	16.33	12.24
Avg. rel. ret. 30	11.10	10.82	12.29	11.27
p-value (vs baseline)	-	1.00	0.00004	0.1

Table 3.6: Comparison of the average number of all relevant tweets returned and the percent of topics that have meet or exceed the media number returned at 30 for Unofficial Runs

	Unofficial Runs					
	5	6	7	8	9	10
Percent above median	73.47	73.47	83.67	73.47	83.67	85.71
Percent at median	14.29	12.24	12.24	14.29	12.24	8.16
Avg. rel. ret. 30	11.18	11.10	12.65	11.16	12.22	12.55
p-value (vs baseline)	0.4	0.6	0.0000002	0.4	0.00004	0.0000006

Table 3.7: Average number of tweets returned and the percent of topics that met or exceeded the media number of relevant and highly relevant tweets at 30 for a modified Run 7

	All relevant tweets	Highly relevant tweets only
Percent above median	83.67	66.67
Percent at median	14.29	18.18
Avg. rel. ret. 30	13.33	3.15

Chapter 4

Hashtag Based Classification

This chapter provides the motivation for and outlines how hashtag based classification is performed. Furthermore, the collection of tweets for use in experiments and the motivation for each experiment will be outlined. The procedure for how each experiment was conducted and evaluated will be explained in this chapter. Following the presentation and discussion of the experimental results, a discussion of the applicability of hashtag based classification to the TREC Microblog Track and to a mobile application will close this chapter.

4.1 Task Motivation and Outline

As the popularity of Twitter continues to grow there are increasingly more tweets being posted. Accordingly, there will be tweets that do not contain a hashtag but should; whether this is due to the character limit or an author's lack of knowledge about a hashtag or the users preference not to use hashtags. Thus, it is desirable for there to be an automatic method of determining that a tweet should have a hashtag. By having such an automatic method it is then possible for users that are interested in the topic associated with a hashtag to find tweets that discuss that topic but do not contain the hashtag. For example, a user that has experienced a natural disaster (like an earthquake or tsunami) may not know what the popular hashtag(s) associated with the disaster is and so would not be able to use it. Accordingly, this user's tweets may have pertinent information concerning the disaster but these tweets are harder to retrieve and put in the context of the disaster. By having an automatic method that says this user's tweets "should" have a hashtag then it allows better retrieval and can provide context to the user's tweets about the disaster.

This problem can be formally defined as follows, "Given a hashtag and a set of tweets without the given hashtag, return all tweets which should contain the given hashtag." By defining the problem in this way, tweets with the given hashtag are ignored. Ignoring tweets that already have the given hashtag is partly a matter of convenience because those tweets can be used as examples of what tweets that have that hashtag look like. However, it is also a matter of how hashtags are used. If the problem were to consider tweets that had given hashtag then it would imply that users do not use hashtags appropriately which in turn defeats the purpose of using hashtags for search as users cannot expect a tweet's content will correspond to the hashtags they contain.

By defining the problem in this way, the problem becomes one of binary classification. Where a tweet is classified as belonging to one class (those that should contain the given hashtag) or a different class (those that should not have a hashtag) based upon previously seen evidence, called training data. Thus, a binary classifier uses training data features (e.g. words, character n-grams, etc) as a means of judging similarity between a candidate document and the two classes. Due to the fact that Twitter users produce copious numbers of tweets on a daily basis separating tweets that should have a hashtag from those that should not has a spam filtering flavor [1]. Accordingly, this thesis makes use of the spam filtering conventions where tweets that should not have a hashtag belong to the positive class (spam) and tweets that should have a hashtag belong to the negative class (ham) [23].

Using the methodology of spam filtering, three experiments were conducted to determine how well this binary classification can be performed on Twitter using multiple classifiers for each experiment as a means of comparison:

Experiment 1: This experiment uses an idealized scenario to determine what the best case performance could possibly be for hashtag-based classification. This scenario involves two sets of tweets where each set is defined by the presence of a unique hashtag (e.g. #osama and #mothers day). Using these two sets of tweets, a large quantity of training data is used to determine how well the classification task can be performed. This situation is ideal as it involves two topically different sets of tweets and more training data than would realistically be used. For this experiment, the choice of which set belongs to the positive and which set belongs to the negative class is largely arbitrary as the only concern is attempting to distinguish between the two sets.

[1] Spam filtering is the process of separating wanted e-mails/text messages/etc from unwanted ones for some definition of wanted.

Experiment 2: The goal of this experiment is to investigate the real-world scenario of differentiating random tweets from those that should have a hashtag, like #mothersday. That is, this experiment is meant to simulate the real world scenario of taking a random sample of tweets (in the order that they were posted) and automatically selecting from that sample those tweets that should have a particular hashtag (presumably provided by the user). More importantly, this experiment uses incremental training, which is when additional examples are added to the base set of examples over time, to examine the changes in classification performance. This incremental training aspect provides insight into how soon classification can begin in a real world streaming scenario as waiting for a large training set means a delay in providing classified tweets to a user.

Experiment 3: Looking at how well filtering applies to sub-topical hashtags can also be important due to a user's desire to learn more about a particular sub-topic. This experiment investigates whether tweets that share a hashtag, say #tsunami, can be distinguished based upon another hashtag, say #fukushima, shared by some portion of that original set of tweets. The motivation for this experiment is similar to that of Experiment 2, except the random sample is actually a topically related sample and the user is interested in a particular sub-topic. Due to the nature of this experiment, incremental training is also conducted because a user will likely want the classification to happen as soon as reasonable results are possible.

4.2 Experimental Methodology

4.2.1 Test Collections

Four data sets were used for these three experiments, which were collected using the Twitter Search and Streaming APIs[2]. Using Twitter Search, approximately 6,000 tweets containing the hashtag '#tsunami' were collected from March 5th, 2011 until March 10th, 2011. These tweets are a mixture of the the most recent and most popular (for Twitter's definition of popular, which is unknown) tweets for each day[3]. All tweets were transformed into lower case (as the Search and Streaming API ignore case when matching query terms). While it is possible that in general '#tsunami' could refer to any number of events, at the time of collection there was further speculation about another tsunami hitting Japan (caused by

[2]Information available at `dev.twitter.com`

[3]The Search API at the time of collection limited results for a query to 1,500 tweets

another earthquake) and the subsequent consequences (when an earthquake occurred on March 7, 2011) and so one can be reasonably certain the tweets refer to the same topic.

Two of the remaining data sets were collected using the Twitter Streaming API's filter capabilities[4]; where Twitter will return in real time tweets matching the filter query or queries. Using the Twitter filter method approximately 84,125 tweets from May 2 to May 5, 2011 with the hashtag, '#osama', were collected. These tweets are intended to correspond with the death of Al'Qaeda leader Osama bin Laden by the United States military. From May 5 to May 9, approximately 120,000 tweets with the hashtag, '#mothersday', were collected. As these data sets were intended to be used for Experiment 1 and Experiment 2, all occurrences of the corresponding hashtag were removed from both data sets to remove any explicit signal that could be used in classification. As well, any mention of "Osama Bin Laden" was removed from the #mothersday tweets and similarly any mention of "Mother's Day" was removed from the #osama tweets. This was done to ensure that there could be not be any obvious overlap between the two sets of tweets. In doing so, all tweets were converted to lower case to facilitate such removal of tweets and hashtags. The removal of tweets meeting these criteria constituted the removal of several hundred tweets and so it is reasonably certain that there is likely to be little overlap between the two sets of tweets.

The Twitter Streaming API supports a sample method that sends out a stream of tweets randomly selected from the Twitter public timeline. While Twitter does not make it clear how this random sampling is done, it appears to provide a random enough sample for the purposes of this thesis. The sample method was used to collect approximately 9.7 million tweets from April 21, 2011 to April 27, 2011. As with the other sets of tweets, all random tweets were converted to lower case and all tweets mentioning "Mother's Day" or containing the hashtag '#mothersday' was removed from this random set to avoid overlap with the #mothersday set of tweets. The total number of tweets removed from this random sample was approximately 700, which is a very small percentage of the the entire set and so the lack of overlap can be reasonably assured. It is noted here that only the tweets from the last 24 hours of collection were used in any of the experiments.

There is perhaps a very important difference between these collections and their usage in these experiments when compared to other microblog research; especially, the 2011 Microblog Track. All tweets regardless of the language contained within them are eligible for positive and negative classification. That is, classification is not restricted to tweets in English. This is due to the fact that in a real world scenario it may not be desirable to limit the language of tweets. If a user is interested in first-hand accounts of the tsunami in Japan then discarding tweets in Japanese could drastically limit the amount of information

[4]See `http://dev.twitter.com/doc/post/statuses/filter` for specific details

Hashtag	Count	Hashtag	Count
6000	#tsunami	1852	#fukushima
1769	#japan	1669	#genpatsu
718	#earthquake	542	#jishin
343	#eqjp	338	#prayforjapan
276	#nuclear	243	#japon
228	#jpquake	206	#jisin
147	#quake	140	#japn
120	#terremoto	108	#newsquake
106	#hinan	99	#radiation
91	#save	88	#anpi

Table 4.1: 20 most commonly used hashtags in our data set

they receive. Similarly, the reaction of the Arabic community to the killing of Osama bin Laden may not be expressed solely in English and by limiting the language to English then there is the possibility of missing interesting or important tweets.

4.2.2 Experimental Setup

Before proceeding into an actual discussion of how each experiment was setup, a small discussion regarding the limitations of the experiments is presented. For the purposes of this thesis the presence (or lack thereof) of a hashtag in a tweet is taken to be the ground truth. That is, if a tweet has a hashtag then this is the definitive ruling for that tweet. Similarly, the lack of a hashtag is a definitive ruling that the tweet should not have that hashtag. This allows an easily created ground truth set, e.g. the class a tweet falls into is determined by the presence of the corresponding hashtag. While this potentially allows for adversarial behaviour, is subject to user misuse of hashtags, and is limited by the fact that a tweet cannot contain every appropriate hashtag; it also reflects the real world scenario where verification of results is impractical due to the sheer number of tweets. However, to combat these problems only hashtags which focus on a specific topic are used. This allows a greater likelihood of homogeneous sets of tweets being created. As opposed to heterogeneous sets that can arise when looking at sentiment based hashtags (e.g. #sarcasm, #happy, etc) or meme-styled hashtags (e.g. #thatswhatshesaid).

For Experiment 1 and Experiment 2, the sets of tweets used are from different periods of time. More precisely, there is a small gap between the #osama tweets and the #mothersday

tweets as the collection of the latter started after the collection of the former. There is a larger gap (of about 8 days) between the random tweets and the #mothersday tweets. This means is that there exists the potential for the language used in tweets to change, which can affect the classification. However, these gaps are not large (especially for the #osama and #mothersday tweets) and so the impact of changes in the language used should be minimal. In addition, the #tsunami, #mothersday, and #osama tweets span multiple days which could affect classification, however, this was unavoidable due to the limitations of the Twitter Streaming API.

Experiment 1 Setup

For the idealized classification experiment, the #osama tweets and the #mothersday tweets were used. Due to the size of the #osama tweets, each set of tweets was split into 40,000 training tweets and 40,000 test tweets; where tweets are in chronological order such that training tweets from a class were posted before test tweets from the same class and tweets in the respective set are in the order that they were posted. Furthermore, the #osama tweets were treated as the positive class and the #mothersday tweets as the negative class. The distinction of positive and negative is arbitrary for this experiment as the goal is merely to test the ability to distinguish between two sets of topically different tweets.

Each classifier was trained in a one-to-one fashion (where a positive tweet was trained and then a negative tweet was trained until all training tweets were exhausted) so that at any time the same number of positive and negative examples were seen. This was done to attempt to remove any bias being learned by a classifier. Once a classifier was trained in this fashion, it then classified all 40,000 #osama test tweets and then all 40,000 #mothersday tweets. For each test tweet, the classifier outputs a score with which the class of the tweet is determined by the varying the threshold at which a tweet's score denotes belonging to a class (the default threshold is 0, such that a score > 0 corresponds to the tweet being classified as a positive tweet, and as a negative tweet otherwise).

Experiment 2 Setup

The random tweets and #mothersday tweets were used in this experiment as the mother's day tweets would allow the most tweets to be used for training data out of the topical tweet sets. The #mothersday tweets were split into 60,000 training tweets and 60,000 test tweets and as before are in chronological order (so that training tweets are older than the test tweets). The most recent 120,000 tweets from the random tweet set were split in the same

manner as the #mothersday tweets. The 120,000 most recent tweets were chosen from the random sample as this would help to minimize the differences in language used in the #mothersday tweets. The random tweets were considered to belong to the positive class and #mothersday tweets were considered to belong to the negative class. This corresponds to the similarities between this experiment and spam filtering where the user wants to be presented with as few random tweets as possible while still retaining as many #mothersday tweets as possible.

This experiment uses incremental training to mimic the streaming scenario of Twitter, where old messages need to be used to classify whether recently posted messages require a hashtag. The goal of incremental training is to determine the optimal trade-off between the number of training examples seen and classification performance. This is done so that classification can begin as soon as possible without having to wait for a lot of training data which can potentially delay providing results to users.

Incremental training for this experiment was conducted by using the 60,000 #mothersday tweets set aside for training and the 60,000 random tweets set aside for training. From this 164 training sets were created, such that the first training set created comprises the old tweets from both #mothersday and random training tweets and the last training set comprises the most recent tweets from the #mothersday and random training tweets (e.g. later training sets were comprised of increasingly more recent tweets). 164 incremental training sets were created for this experiment. The first hundred have 5 positive tweets and 5 negative tweets each, for a total of 10 tweets per training set. Thus, once a classifier is trained (in order of creation) on the first 100 training sets it will have seen 500 positive and 500 negative tweets. An additional 5 training sets of 100 positive and 100 negative tweets each were created. So after the first 104 training sets, a classifier would have seen 1,000 positive examples and 1,000 negative examples. Finally, 59 training sets composed of 1,000 positive and 1,000 negative tweets each were created. After a classifier had been trained on all 164 training sets then it would have seen 60,000 positive examples and 60,000 negative examples. Furthermore, a classifier was trained in the one-to-one fashion outlined in the description of Experiment 1.

Once a classifier was trained on a single training set then it would classify the 60,000 random tweets selected for testing (these are the most recent random tweets) and then it would classify the 60,000 #mothersday tweets selected for testing (these are the most recent tweets from the #mothersday tweets). As in Experiment 1, the classifier outputs a score for each tweet that is used to determine the class a tweet belongs in.

Experiment 3 Setup

In this third experiment, the collection of tweets used was the #tsunami tweets as they provided a reasonably specific topic given the time period that the hashtag was used. As can be seen in Table 4.1, the hashtag #fukushima provides an easy split for positive and negative tweets. In addition, Fukushima refers to a prefecture and the capitol city of that prefecture in Japan that was heavily affected by the earlier tsunamis and earthquakes. Accordingly, the #tsunami tweets were split into a negative class containing tweets containing #fukushima and a positive class of tweets that do not contain #fukushima. Similar to the previous experiments, the #fukushima hashtag was removed from all negative tweets. Each class was split into 300 training tweets and 1,500 test tweets in chronological order of Twitter post time. Thus, all tweets set aside for training that belonged to a particular class were posted before tweets selected for testing from the same class.

Similar to Experiment 2, incremental training sets of 5 positive and 5 negative examples were created so that 60 incremental training sets of 10 tweets each (5 positive and 5 negative) were created. Training sets of tweets are created such that the first training set contains the oldest tweets that were allocated for training and the last training set contains the most recent tweets from the tweets set aside for training. Further, a classifier was again trained in the same one-to-one fashion (as seen in Experiments 1 and 2) to prevent any bias from being formed. As with Experiment 2, once a classifier was trained on a training set it would classify the 1,500 positive tweets selected for testing and the 1,500 negative tweets selected for testing and would output a score to for each tweet.

4.2.3 Classifiers

A number of classifiers were selected for use as it provides a differing number of approaches. As previously mentioned, this task is very similar to spam filtering and a selected number spam filters are used. The classifiers that are commonly seen as spam filters are denoted as with the #SPAM tag.

Bogofilter - #SPAM

Bogofilter is a statistical classifier, which makes use of Bayesian statistics, and was designed for spam classification. Bogofilter very simply tests how badly the null hypothesis (that the document is a random collection of independent words with some derived probability) fails and in which direction, e.g. the document is spam (negative) or ham (positive). It

empirically derives probabilities, from the training data, for words indicating how often each word is seen in spam and ham documents. For a candidate document it computes the probability that the document is spam or ham based upon these derived probabilities and combining them using a formula based on Fisher's "inverse chi-squared function" [63]. Simply put, a document is seen as spam if these combined probabilities indicate high usage of "spammy" words and ham otherwise. Bogofilter 1.2.1 and Berkeley DB 4.8.24 were used for our experiments.

TextCat

TextCat is a "n-ary" classifier written by Gertjan van Noord using the algorithm proposed by Cavnar and Trenkle [15]. TextCat uses character n-gram (n=1,2,3,4,5) word slices as the basis for comparison. For example, for the word 'tea' TextCat will create n-grams like 'e','a','ea','Te',' Te','Tea', and so on. The frequency of usage of these n-grams for training examples in a particular class are stored in a profile in descending order of frequency. A candidate document has its own profile computed and then compared to all of the profiles that are available to TextCat. The similarity of a document and a candidate profile is computed using the "out-of-place" metric. Namely, for each n-gram in the document's profile, its corresponding n-gram is found in the candidate profile and the difference in rank gives the similarity for that n-gram; but if the n-gram doesn't exist in a category's model then some maximum value is used. For example, if "cat" is the third most used n-gram in the document but the fifth in the candidate profile then the score is 2. Thus, the sum of these "out-of-place" values can be seen as the similarity between a document and a candidate profile. For these experiments, TextCat version 1.10 was used. TextCat was used for the simplicity of its algorithm.

Logistic Regression (LR)

Logistic regression is a method of fitting the log-odds of positive classification (logit of the conditional probability that a document belongs to the positive class given the document's features) to the features present in the training data. That is, the log-odds of positive classification creates a linear scoring functions with respect to the features and logistic regression attempts to maximize the likelihood of the coefficients in front of the features in this linear scoring function. It is also important to note that contrary to Bayesian classifiers, logistic regression does not assume independence of features. For the purposes of these experiments, two logistic regression classifiers were used. One classifier used was LibLinear[30] version 1.8 's with the L2-regularized logistic regression solver (option -s

0) using byte 4-grams as features. However, as LibLinear is meant to be used with any features it was required to use additional software to generate the byte 4-gram features for LibLinear. The other classifier used was supplied by Gordon V. Cormack and also uses byte 4-grams as features. Cormack's version is used due to its simple implementation and subsequent ease of translation to a mobile setting. As well, Cormack's logistic regression filter was seen extensive use in spam filtering [23]. For the purposes of this thesis, LibLinear will denote the use of LibLinear's logistic regression classifier and GCLR will denote the use of Cormack's logistic regression classifier.

Dynamic Markov Compression (DMC) - #SPAM

DMC is a data compression model created by Cormack and Horspool [21]. The basic premise of DMC is to treat documents as sequences of bits and to create a progressively larger Markov model representing substrings of the training examples. Therefore, binary classification is done by using DMC to create two Markov models, representing the positive and negative documents. The models are then used used compresses a candidate document and the model which does so most effectively (e.g. compresses the document the most) indicates that candidate document likely belongs to the corresponding model's class. The version of DMC used is supplied by Gordon V. Cormack and has previously seen usage in the 2007 TREC Spam Track [23] as well as other spam filtering contexts[10]. The use of DMC is not essential as any compression algorithm can be used in the same manner, however, DMC was chosen due to its performance in the TREC Spam Track[23] and other spam filtering works [9].

OSBF-Lua - #SPAM

Orthogonal Sparse Bigrams with confidence Factor (OSBF) is a Bayesian classification algorithm which uses Orthogonal Spare Bigrams (OSB) [67] for feature extraction and Exponential Differential Document Count (EDDC) [4] for automatic feature selection. OSB creates a set of bigrams for a sequence of tokens, such that combining some of these bigrams would produce any other combination of the tokens (in the same order as the tokens are in the original sequence). EDDC is an attempt to better separate poor features by reducing the local probability of features inversely to their class separation power. Basically, EDDC removes features that do not help in the discrimination between documents in two different classes. OSBF-Lua is a Lua C-module for text classification designed primarily for the binary classification task of spam filtering and is written by Fidelis Assis. For these

experiments, Lua 5.1 and OSBF-Lua version 2.0.4 were used. OSBF-Lua has performed competitively in the TREC Spam Track[22].

Support Vector Machines (SVM)

Support Vector Machines are traditionally binary classifiers, however, multi-class SVMs do exist. A binary SVM attempts to create a hyper-plane(s) between two classes of training data (where the planes are defined by the features used), such that the distance from each class to the hyper-planes(s) is as large as possible. Thus, a new document is classified based upon the location determined by the document's feature in relation to the hyperplane(s). Two SVM libraries were used for the purposes of comparison. The first was LibSVM[17] version 3.11 and the other was SVM$^{\text{light}}$[41] version 6.02. Both used byte 4-grams as binary features. Both libraries use the default values everywhere except for LibSVM, which used the supplied linear kernel, had the cache size increased to 3000MB, and had probability generation enabled. The probability generation was enabled to allow comparison with the other filters, which all generate scores. Consequently, the training of the LibSVM classifier is extremely slow without the the first two parameters. To simplify formatting, SVM$^{\text{light}}$ will be referred to as SVMLight from this point onward. Finally, like LibLinear, both LibSVM and SVMLight are meant to be used with any desired features and so additional software was required to generate the byte 4-gram features used.

4.2.4 Experimental Evaluation

As stated previously, the hashtags provided by authors were taken to be the ground truth in training and test set creation as this mimics the real world where such guarantees cannot be shown to hold. Evaluation of classifiers is generally performed using some combination of precision and recall, which measure how many results returned were labelled correctly and how many correctly labelled results were returned, respectively. For the purposes of this thesis, neither measure adequately describes how well a classifier performed at separating the two classes as they are threshold dependent methods of evaluation (the threshold being where the transition from positive to negative class scores occurs).

In following the tradition of spam filtering, positive misclassification (pm%) and negative misclassification(nm%) percentages[5] are used to examine the trade-off in favouring better positive class tweet classification or better negative class tweet classification. This is done by varying the threshold at which a classifier determines the class a tweet belongs

[5]pm% is equivalent to the false negative rate and nm% the false positive rate [11, Ch. 10]

to. By design the scores returned by the classifiers used have the value 0 as their default threshold, where scores above zero denote a negative tweet and scores less than or equal to zero denote a positive tweet. By adjusting this threshold the differences in the trade-off between pm% and nm% can be seen.

By plotting these differences, a receiver operating characteristic (ROC) curve is created. By plotting ROC curves from different classifiers together the relative performance of these classifiers can be judged. However, doing so can often create a cluttered mess when comparing many different classifiers. Instead, the area under the ROC curve (AUC or ROCA) is a number between 0 and 1 that is threshold independent and can be used to compare classifiers. A classifier with a larger AUC will tend to have have a superior ROC curve. For convenience, the area above the curve, 1-AUC, is reported as a larger value would indicate poor performance.

Accordingly, evaluation of the classifiers in the aforementioned experiments is done by comparing the (1-AUC)% values of the classifiers. These values are obtained by using the TREC Spam Track's evaluation toolkit[6]. The evaluation toolkit reports, among other things, the pm%, nm%, lam (the logistic average of pm% and nm% that tends to be less sensitive to threshold values than over averaging metrics), and (1-AUC)%, all of which are reported with corresponding 95% confidence intervals. For the purposes of brevity and clarity only the (1-AUC)% values are presented for the incremental training experiments in the main body of the thesis. Appendix B reports (1-AUC)% with 95% confidence intervals.

4.3 Results and Discussion

Table 4.2 shows the results of Experiment 1. Figures 4.1, 4.2, 4.3, 4.4, 4.5 show the results of Experiment 2. The results of Experiment 3 are given in Figures 4.6 and 4.7. Appendix B contains the (1-AUC)% with 95% confidence interval for each classifier for each training set for Experiment 2 and Experiment 3. From Table 4.2, it can immediately be seen that in an idealized scenario two sets of tweets can be distinguished as the area above the curve is quite small for most classifiers. In fact, logistic regression performs very well achieving (1-AUC)% < 1, which begins to approach values commonly seen in spam filtering [22, 23]. Even more surprising is how well GCLR performs considering it is an extremely concise and relatively simple implementation of logistic regression. The other comforting fact is that all of the confidence intervals appear to be very tight and so there it is unlikely these results are by chance. By being able to separate tweets based upon two distinct and unrelated

[6]Available at the time of writing at `http://plg.uwaterloo.ca/~gvcormac/jig/`

Table 4.2: (1-AUC)% for all classifiers used in Experiment 1 with 95% Confidence Intervals

Classifier	(1-AUC)%
Bogofilter	2.4804 (2.3907 - 2.5734)
DMC	3.1286 (3.0057 - 3.2564)
GCLR	0.9805 (0.9391 - 1.0236)
LibLinear	**0.8612 (0.8209 - 0.9036)**
SVMLight	1.0394 (0.9937 - 1.0872)
LibSVM	1.0853 (1.0432 - 1.1291)
OSBF-Lua	3.1039 (3.0120 - 3.1985)
TextCat	8.8434 (8.6472 - 9.0436)

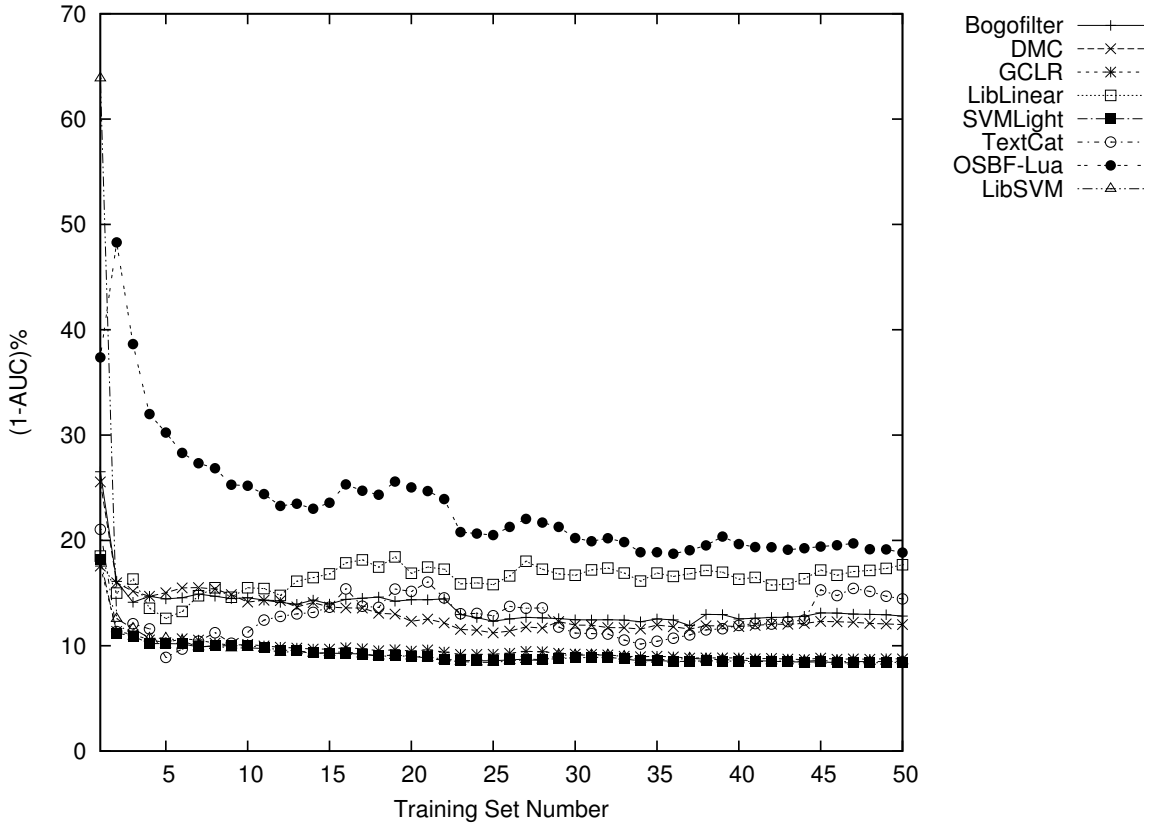

Figure 4.1: (1-AUC)% for all classifiers for the 1st to 50th incremental training set in Experiment 2

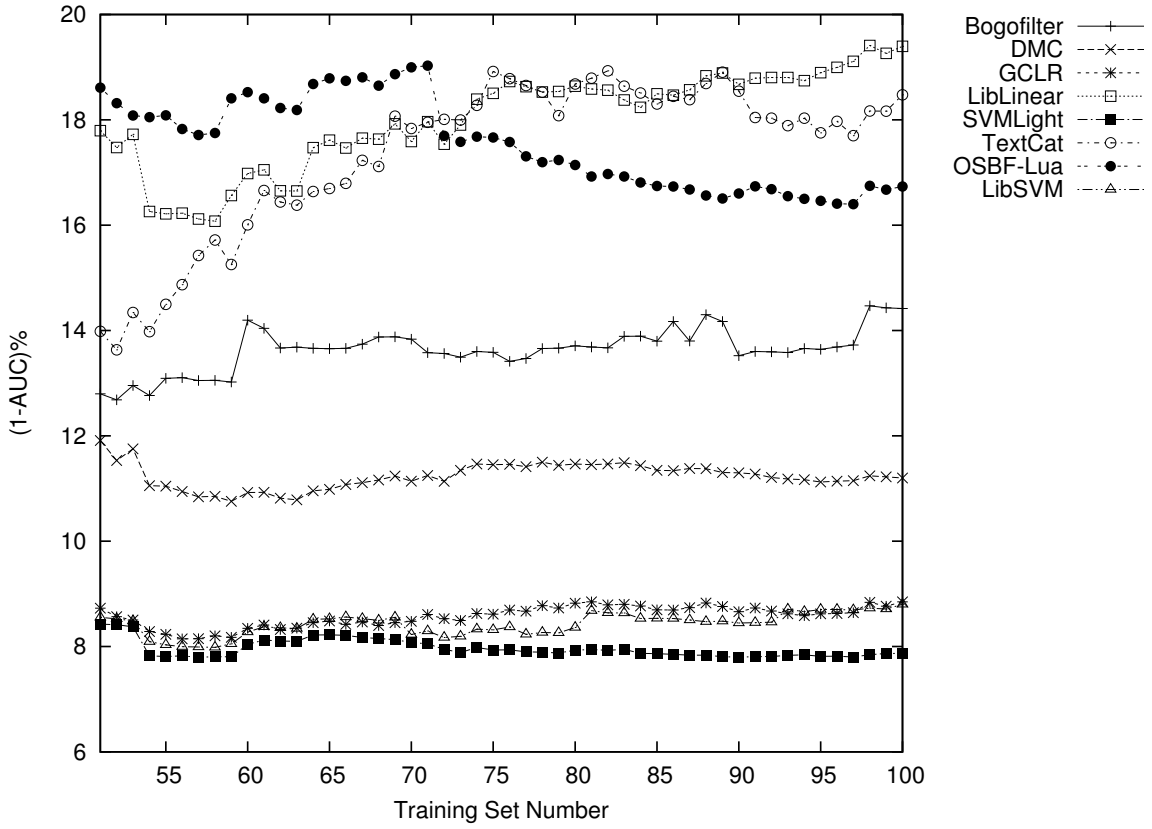

Figure 4.2: (1-AUC)% for all classifiers for the 51st to 100th incremental training set in Experiment 2

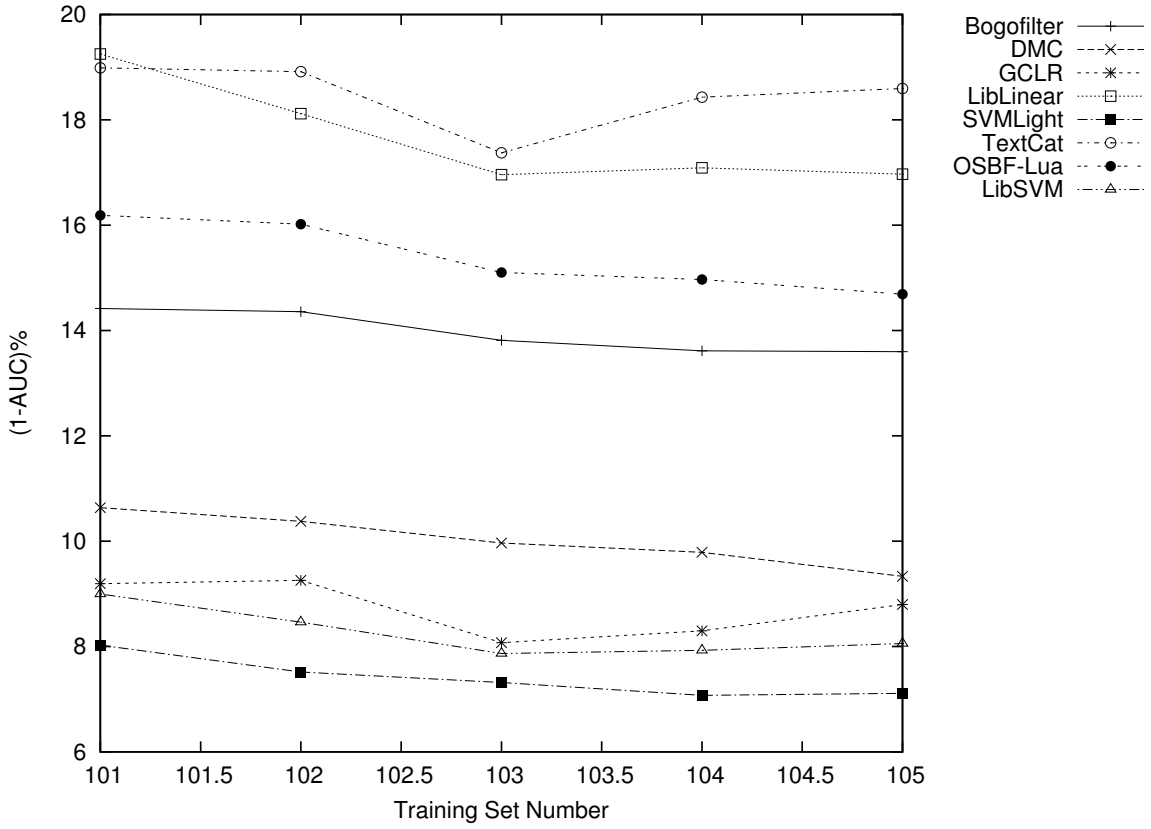

Figure 4.3: (1-AUC)% for all classifiers for the 101st to 105th incremental training set in Experiment 2

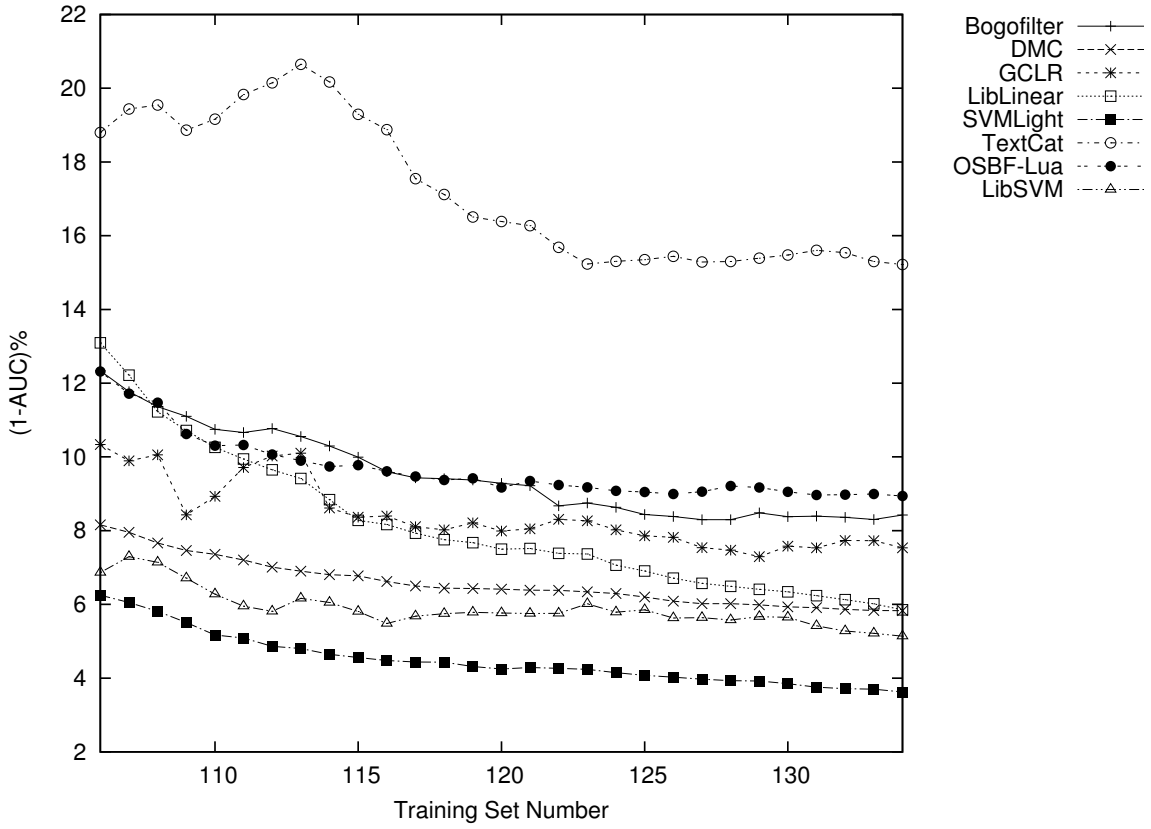

Figure 4.4: (1-AUC)% for all classifiers for the 106th to 134th incremental training set in Experiment 2

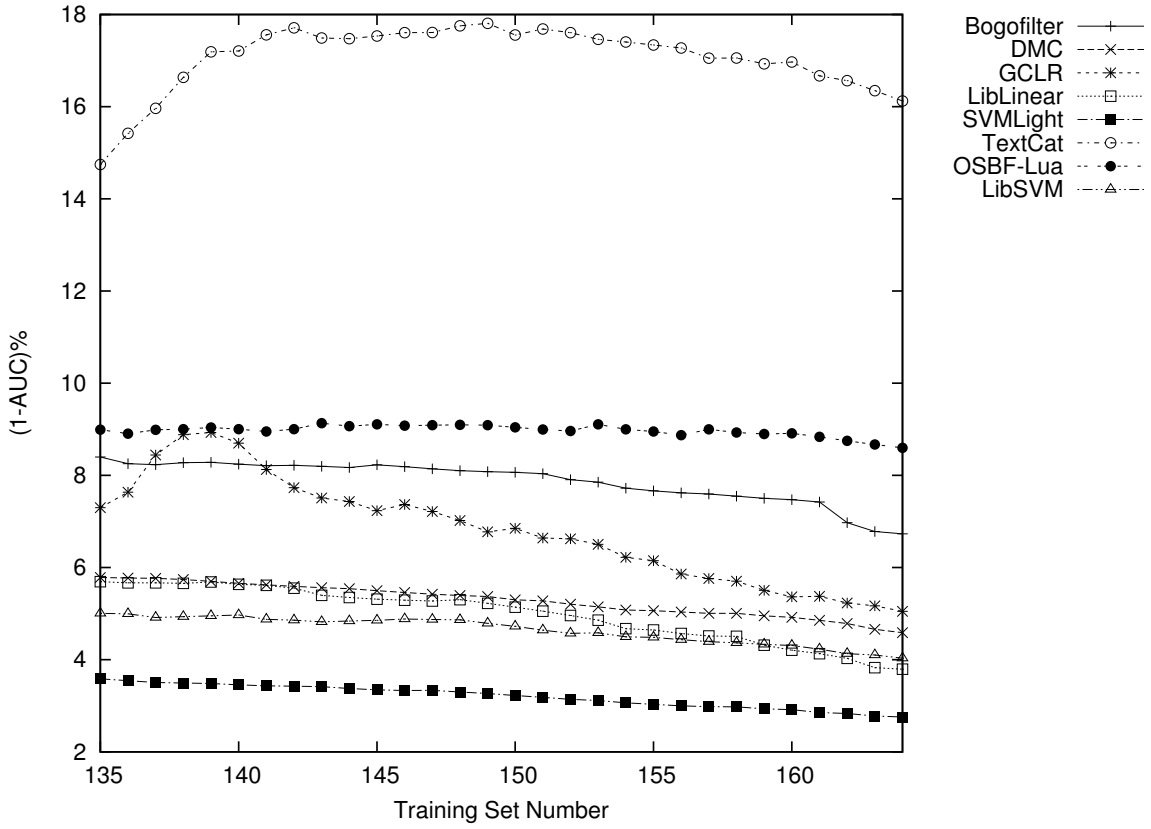

Figure 4.5: (1-AUC)% for all classifiers for the 135th to 164th incremental training set in Experiment 2

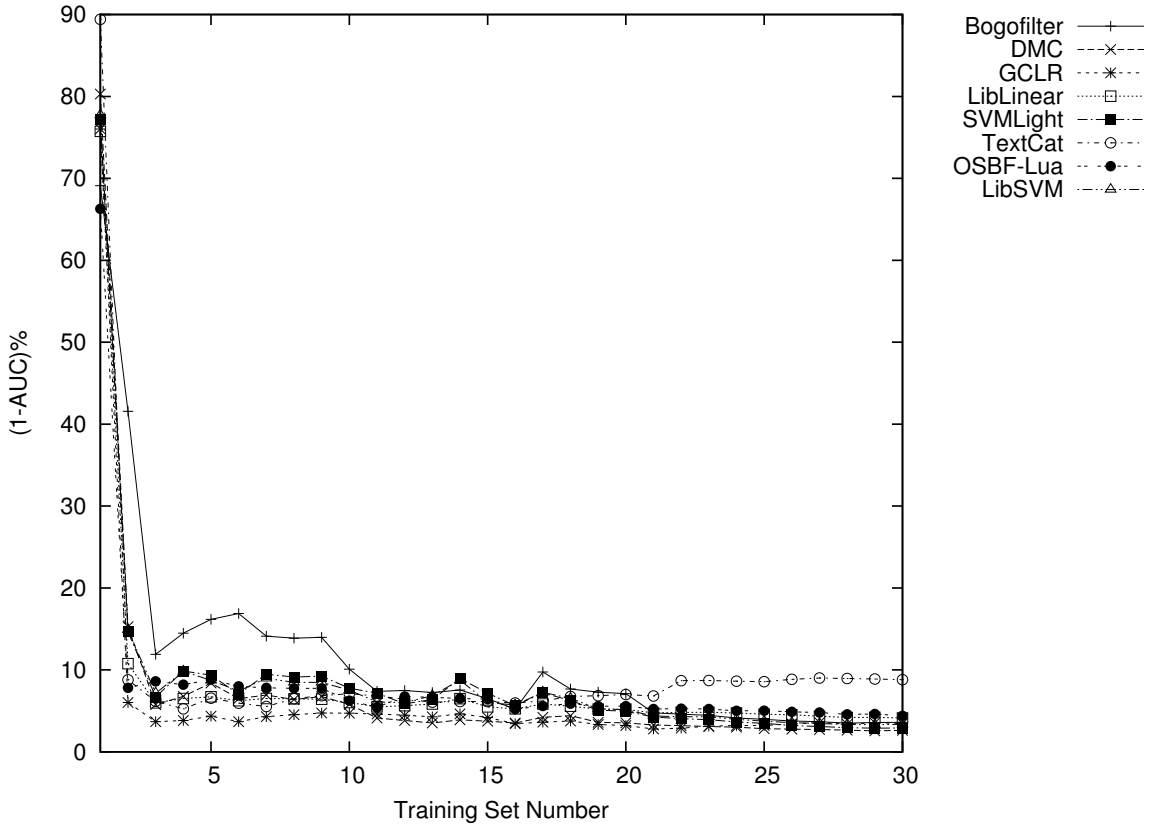

Figure 4.6: (1-AUC)% for all classifiers for the 1st to 30th incremental training set in Experiment 3

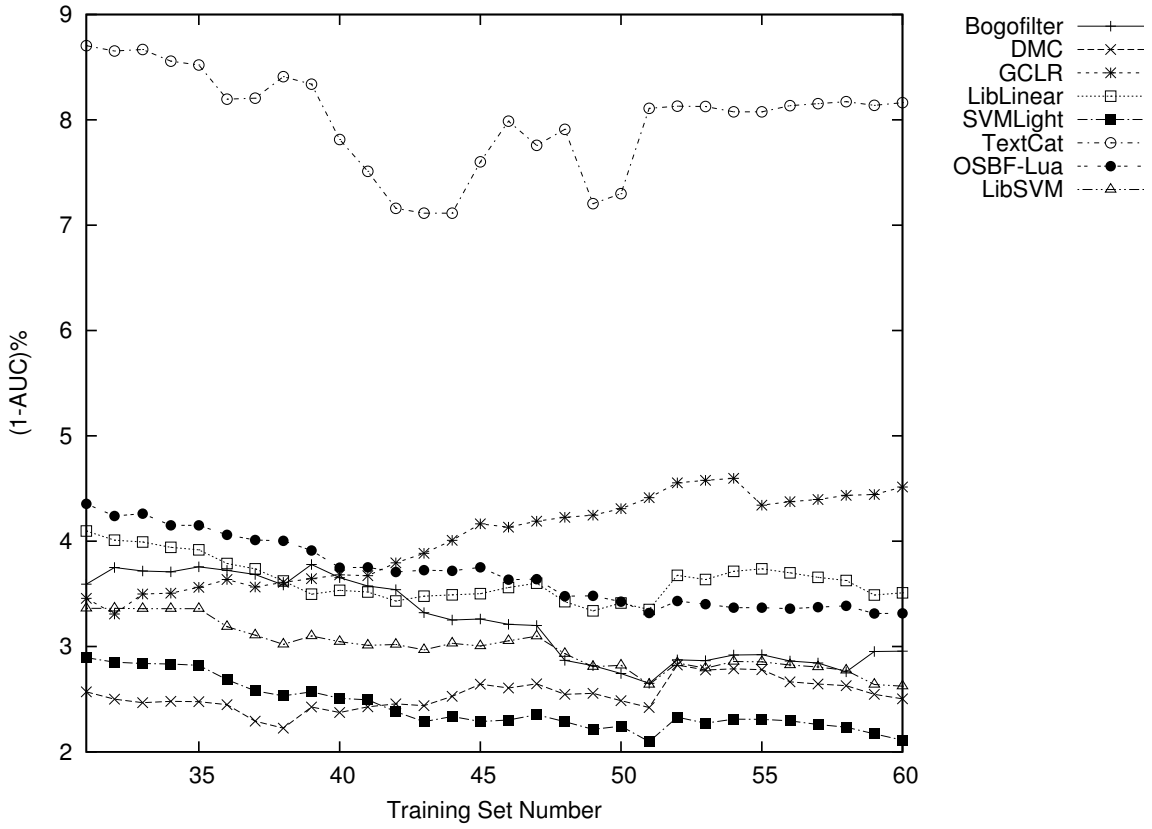

Figure 4.7: (1-AUC)% for all classifiers for the 31st to 60th incremental training set in Experiment 3

hashtags provides some evidence that Twitter users do not generally misuse hashtags. This means that generally if a topical hashtag is present in a tweet such a hashtag is likely relevant to the tweet as the training and test sets were based off of topical hashtags.

Experiment 2 is the experiment that has real world importance as it is similar to how classification using hashtags would likely be used in the real world. Initially, all the classifiers suffer from poor performance which is to be expected given that there are only 10 tweets total in the initial training set but the majority of the classifiers improve dramatically with the second incremental training set. The early performance of GCLR, SVMLight, and LibSVM, is rather startling considering how little information is contained with a tweet. In addition, the similarity in their performance for many of the small training sets is also surprising.

The results of Experiment 3 are mildly surprising as all classifiers tend to end up performing well. GCLR, again, starts out leading the pack but then falls behind many other classifiers and actually has relatively poor performance once it is trained on all the training sets. DMC is more competitive than in Experiment 2 and at times outperforms the other classifiers. However, SVMLight ends up being the best classifier at the end but it also takes more than half of the training sets for SVMLight to really lead the pack. Unfortunately, it appears that there does exist some overlap in the positive and negative training sets as there are spikes where most of the classifiers degrade in performance. Upon manual inspection of these training sets this hypothesis is confirmed; however, overall performance does not appear to be significantly degraded in the long run.

Furthermore, it appears to be the case that Experiment 3 is easier than Experiment 2 (and for some classifiers, easier than Experiment 1). In part, this could be due to the fact that the test set for Experiment 3 was smaller than that of Experiment 2 and accordingly these results could reflect that. However, even if this is the case at any point in time it is unlikely a user would be presented with more than a few hundred tweets of the same class (to limit information overload) and so this size difference is possibly not relevant in this case. The more likely case is that the Experiment 3 tweets have a higher signal to noise ratio than those found in Experiment 2. This makes sense given that the tweets in Experiment 3 are likely describing very particular aspects of the Japanese tsunami crisis; whereas a random tweet and a #mothersday tweet could both be trying to sell Twitter users something. In fact, Table 4.3 shows the top 5 hashtags used, except #tsunami, in Experiment 3 for training and test sets for both classes. What can be seen immediately is that there are hashtags which could be used to aid in classification, e.g. if a tweet has #genpatsu it is likely to belong to the negative class and the positive class if it doesn't. Thus, the performance is likely due to higher signal to noise ratio. In an examination of hashtag usage in Experiment 2, this appears to be the case as there a few "indicator"

hashtags but they are not nearly as useful as are the ones present in Experiment 3. The fact that Experiment 3 was performed so well also indicates that there may not be a lot of need to perform this task in the real world as the the two classes were able to be separated well and so there were likely very few tweets without the #fukushima tag that should have had it.

The performance of DMC is interesting in Experiment 2 as it initially improves steadily but then rebounds and decreases in performance until it starts recovering fairly quickly to become one of the better performing classifiers. There do exist some small fluctuations in DMC's performance in Experiment 3 but these appear to coincide with the training sets that have overlap between positive and negative examples. However, DMC does appear to learn the training data fairly quickly in Experiment 3 barring the small fluctuations, e.g. DMC stops readily improving around training set 25. Given the performance in all three experiments, it appears to be the case that DMC learns tweets slowly when compared to other classifiers, like SVMLight and LibSVM (even GCLR exceeds the performance of DMC for the first half of Experiment 2), which are able to continually exceed DMC's performance with the same or smaller amounts of training data.

The classifier that can likely be considered the best performing classifier for these experiments is SVMLight. SVMLight is initially competitive with GCLR and LibSVM but overcomes both to become the best performing classifier in Experiment 2. DMC is quite competitive with SVMLight for much of Experiment 3 but SVMLight eventually pulls ahead (although the difference in performance between the two may not be all that significant). In addition, SVMLight appears to be much more sensitive to the questionable training sets early in Experiment 3. SVMLight is likely the classifier of choice if a person wanted to have guaranteed classification performance, e.g. in an experimental setting.

LibSVM as stated previously is fairly competitive to SVMLight in Experiment 2 and stays somewhat competitive in Experiment 3. LibSVM ends up being one of the best performing classifiers in all experiments. However, it is unreasonably slow with the default parameters which is why a larger cache was deemed necessary. Furthermore, it was necessary to run LibSVM on a server with an abundant amount of memory for the last 30 incremental training sets of Experiment 2 as LibSVM would crash otherwise. In the author's opinion, this eliminates LibSVM from ever being considered for a real world application.

GCLR performs exceedingly well in Experiment 1 and initially performs well in both Experiments 2 and 3 but doesn't improve overly much with increased training set size. GCLR appears to be somewhat more sensitive to the training sets later in both experiments as its performance fluctuates as more training sets are seen. This is interesting given that

the other classifiers that performed exceedingly well in Experiment 1 used the same features as GCLR and so it is likely the simple nature of the algorithm that causes this sensitivity to the training data. Although, for Experiment 3 it appears to be that GCLR may get over trained which results in the degradation in its performance in the latter part of the experiment. Accordingly, it may be the case that GCLR is only useful in a real world application if the number of training examples is small where it performs competitively. This is not necessarily a mark against GCLR as requiring a large amount of training to perform well means that other classifiers may not be suited to real world use or in cases where there are not a large number of training examples available. Further, it may be necessary to provide training data that is of the highest possible quality to GCLR if one wanted to use it in a real world application (which may be non-trivial to accomplish).

Given that LibLinear is a library that has been designed for optimal performance, it takes an unfortunate amount of training data to become competitive with GCLR in Experiment 2 (the turning point is approximately around incremental training set 116). In Experiment 3, LibLinear again requires about half the training data to become competitive with and then exceed GCLR. This is also the case when LibLinear is compared to the SVM classifiers which outperform LibLinear in both Experiment 2 and 3, with the exception of LibSVM in Experiment 2 where LibLinear manages to outperform it with the last few training sets.

Bogofilter initially performs quite poorly but gradually improves to suitably perform the classification task in Experiments 2 and 3. Bogofilter ends up becoming competitive with GCLR in Experiment 2 and exceeds GCLR's performance in Experiment 3 becoming somewhat competitive with LibSVM. Bogofilter is not quite so competitive in Experiment 2 where it is only exceeds TextCat in performance when all the training sets are used. Unfortunately, the spikes in performance of Bogofilter may indicate that it is particularly sensitive to the quality of the training data used. This may manifest much more subtly in Experiment 2, which may account for why Bogofilter performs worse than the results of Experiment 1 seem to indicate it should.

OSBF-Lua, like Bogofilter, does not live up to the performance shown in Experiment 1. In Experiments 2 and 3, it appears to be very slow to learn. This is especially true in Experiment 2 where it took the majority of the training data for OSBF-Lua to match the early performance of TextCat, SVMLight, and GCLR. The slow learning of OSBF-Lua is also present in Experiment 3. Ultimately, OSBF-Lua does not appear to offer any potential utility to a real world.

The performance of Bogofilter and OSBF-Lua is interesting in light of the fact that they are word based classifiers, e.g. they use whitespace delimited words as features. This

Table 4.3: Top 5 Hashtag counts for training and test examples for Experiment 3

negative training		negative test		positive training		positive test	
233	#genpatsu	1250	#genpatsu	77	#japan	714	#japan
39	#japan	130	#japan	38	#earthquake	233	#earthquake
22	#nuclear	80	#earthquake	33	#jishin	106	#japón
13	#iwakamiyasumi	65	#nuclear	26	#japon	104	#japon
12	#earthquake	59	#radiation	25	#eqjp	91	#jishin

restriction means they are often only used in scenarios where the language(s) used are whitespace delimited, like English, French, etc. Japanese is a language that does not have whitespace delimited words and is present a non-trivial number of training and testing tweets in Experiment 3. In spite of this fact, Bogofilter and OSBF-Lua remain fairly competitive for Experiment 3 and Bogofilter is somewhat competitive with a few hundred tweets in Experiment 2. In the ideal scenario, both perform quite well with Bogofilter following behind the logistic regression methods and the SVMs. For OSBF-Lua, the difference between its score and DMC in Experiment 1 is likely not significant. Thus, the out of the box spam filters can perform hashtag-based classification well but still likely not good enough to warrant devoting extensive time to future investigation.

For Experiment 2 and Experiment 3, TextCat initially performs competitively with GCLR, SVMLight, and LibSVM, but quickly spirals out of control and ends up performing exceedingly bad. The likely cause of TextCat's behaviour is the nature of its classification algorithm and the structure of tweets. As more examples of positive and negative tweets are seen, shared n-grams become more and more common, e.g. 'http', 'com', etc. Accordingly, the "out-of-order" metric that is used becomes decreasingly useful as more similarities are seen between the two sets of tweets. As well, the fact that TextCat's algorithm ignores numbers and all punctuation except apostrophes likely does not help performance but hinders it in the informal microblog setting. In addition, TextCat was designed specifically to work best with thirty to forty thousand bytes of training data and this corresponds roughly to where TextCat performs best. Thus, TextCat could be a good classifier, due to its simplicity, to use for hashtag based classification if the amount of training data could be guaranteed to fit into its optimal performance zone.

Building on a previously mentioned result, Experiment 2 and Experiment 3 indicate that users can be trusted to use hashtags appropriately, otherwise, the results shown here would not likely be as good as they are. However, this result may only hold for topical hashtags as sentimental hashtags or meme-based hashtags are not explored. As mentioned in Chapter 2, the work on sentiment analysis using hashtags appears to only be effective in

a binary case ("Is this tweet happy?") and even then it is not effective in other cases ("Is this tweet sarcastic?"). Meme hashtag-based classification is also likely to perform poorly as the tweets using them are not necessarily homogeneous enough to clearly show they are related except by the particular hashtag. For example, one could think of many different and unique tweets for meme hashtags, like #thatswhatshesaid, that without the hashtag would appear to be random. Thus, these results are probably not applicable to hashtags that are not topical due to the potential for extremely different tweets being connected by the same hashtag.

The overall performance of SVMLight, LibLinear, and LibSVM is all together not surprising considering they are optimized to perform classification well. Whereas the competitors GCLR and DMC do not perform much, if any, optimization. However, DMC does require extensive memory to execute and this may indicate that its use in a real world application may be limited. This is also the case for LibSVM, which required a large cache to execute in a reasonable amount of time and even then it was one of the slowest classifiers to train and classify. In addition, LibLinear, SVMLight, and LibSVM, all require their feature sets to be given to them externally and so for a real-world application either the existing code would need to be modified to create features on the fly, which may not be a trivial modification, or (as was done for these experiments) an extra tool is required to format tweets into the corresponding features. Neither of these options is particularly desirable as the executable for GCLR is already much smaller than those required for LibLinear, LibSVM, and SVMLight, and adding additional code to generate features would likely increase the size of the corresponding executables. Bogofilter and OSBF-Lua require installations and additional libraries and are accordingly non-trivial in size. TextCat by far has the smallest implementation and this is unsurprising considering it is also the simplest algorithm to implement. The version of TextCat used, however, is implemented in Perl and an equivalent version in some flavour of C(++) or Java may not retain this small size.

From these results, two classifiers appear to become the most well suited for hashtag based classification (in terms of resource consumption and performance): GCLR and SVMLight. GCLR is well suited for the task as it performs well (but not always great) in all three experiments and is a simple to implement algorithm. SVMLight is best because it ends up performing the best in the two incremental training experiments. However, practical concerns tend to become more important than classification performance in the real world. SVMLight's issues with executable size and feature generation means that it is likely a poor candidate for an actual application where memory availability and execution efficiency are necessary concerns. This is especially true when the performance of GCLR and SVMLight is comparable with small training sets. Accordingly, GCLR appears to be the de facto choice for an actual application as it performs well with little training data, is

a simple and subsequently portable algorithm that does not require a great deal of memory to execute.

4.4 Future Work

There are many possible avenues of future research for this work. The most obvious is to look at what the effect would be if English only tweets were used in the negative class. Focusing solely on English is impractical, at least for Twitter, as it requires some prior classification to determine language as Twitter has no supported method to determine tweet language. It is worth noting that while the Twitter API does tell users what language the author views Twitter in, it is not the case that it restricts the language used in a tweet to that language. In addition, focusing solely on English tweets is unrealistic as the Twitter Streaming API will give out tweets in various languages and subsequently requires knowing how to handle them. Thus, it is necessary if one wishes to model the real world concerns to look at mixed language positive tweets but English only negative tweets if the goal is to present English tweets only to the user. Rather than discussing this vein of future research further, this thesis will discuss two applications of these results. The first is to the TREC Microblog Track and the second is to a potential mobile application.

4.4.1 TREC Microblog Track

The usage of hashtags in this classification task was in some sense purely arbitrary. Hashtags merely provided a convenient topical query with which to group tweets into distinct categories. Accordingly, it is entirely possible to expand these results to topical queries; namely, those of the TREC Microblog Track. Given the performance of GCLR and SVM-Light with extremely small training sets, it is likely better to choose the top 10 results for a particular query. In part this is done to maximize the likelihood of using actual relevant tweets but also because even the best performing systems at the Microblog Track had a P@30 of less than 0.5, meaning that over half the results returned were not relevant. Accordingly, using those top 30 results would on average have a lot of noise. If the goal is to find highly relevant tweets then this becomes even harder as 25 out of the 33 topics with highly relevant tweets had less than 30 highly relevant tweets and 13 had 5 or less highly relevant tweets. This situation improves when we consider relevant tweets with slightly less than half the topics having less than 30 relevant tweets. However, these numbers indicate that, at least for the Tweets11 corpus, using a smaller training set is less likely to incorporate noise in negative training examples.

Creating the positive training examples is perhaps a little simpler but also requires a little more effort. There are two possible ways that a negative training set could be created. The first is to choose 5 tweets that are not returned for a particular topic. Whether it is better to choose these tweets at random, or from the earliest possible tweet and chronologically moving forward, or starting as recently as possible (up to the query time) and moving chronologically backwards, is a matter of future investigation. The second is to create a universal positive training set such that the tweets do not appear in any of the results for each topic. Using this universal set means the tweets have a time restriction (e.g. oldest query time) for possible tweets; otherwise, the future evidence restriction is violated. It is possible that the universal training set would affect performance negatively as it would be tuned to avoid classifying all the topics as negative, which in a binary classification scenario is not desirable. Of course, whether this does affect performance is valid question and an avenue of future work.

4.4.2 Mobile Application

As smartphones (and soon so called "superphones") have become more and more prevalent in society[7], the ability to have information travel with the phone is a desirable trait and while Twitter does support some streaming and search capabilities in its application for iOS, Android, and Blackberry devices. Such support may not be enough, as stated in Chapter 1 there are tweets that do not have a hashtag for one reason or another. As mentioned in the previous section, topical queries could also be used. From this point on, it is the case that when "hashtag" is used then it can be replaced with "topical query" without loss of generality. Regardless, this classification method would allow tweets to be classified according to user interest.

Immediately there are practical concerns with respect to the classification method used. Storing large amount of training data on a mobile device is not feasible and would likely anger users of the application. In addition, users likely do not want to wait for copious amounts of training data to be found and so the sooner classification can perform well the better. Classification and training also needs to happen in an efficient and timely matter. One of the most important facts is that the algorithm needs to be able to be ported to a mobile system and for the big mobile contenders this means the classification algorithm needs to be amenable to either Objective C (for iOS) or Java (for Android). While Android currently supports C and C++ to some degree, using C/C++ is not recommended by Google unless absolutely necessary. What this means is that GCLR is the most likely

[7]Even Gordon V. Cormack has one now!

candidate method for a mobile application as it meets most, if not all, of these criteria and can likely be optimized for performance on a mobile environment.

Automatic selection of negative training tweets can easily be done by taking the $m >= 10$ most recent tweets that are relevant to a user's supplied hashtag. The value m could be set by the user, however, this may not be ideal as a user could choose extremely large values that are not practical. It is likely that the value of m would have to be set relative to the number of candidate tweets, e.g. if there are 1,000 candidate then use 100. Regardless, the value of m should have a maximum value as it would not be smart to store an arbitrary amount of information on a mobile device. There is also the option of letting the user pick the training data themselves which would allow for a more refined search. Due to performance concerns this would likely require at least 10 choices to be made to ensure reasonable results. In addition, to increase performance when user's are allowed to chose tweets the remaining tweets (e.g. that would give m negative training examples) could be chosen by their similarity to the tweets picked by the user using some measure of similarity.

Collecting positive training data is not as straightforward as collecting negative tweets. Ideally, tweets completely unrelated to the hashtag should be used. This is a hard guarantee to make as not having a hashtag does not mean a tweet is unrelated to the hashtag. Accordingly, having a user confirm that tweets are unrelated to the hashtag would be ideal. However, this is not likely to be an activity users enjoy taking part in. Thus, the method likely to be used is taking tweets from Twitter Streaming API's sample method that do not contain the hashtag. This method does lack the aforementioned guarantee but getting users to go through tweets and mark tweets as not relevant is likely to be a fruitless task. In addition, performing a similarity based method of getting more positive tweets would likely be detrimental to performance. This is because the positive tweets should be random and getting tweets similar to candidate positive tweets doesn't encourage "randomness."

For the actual classification task the Streaming API's sample method can be used. Initially it may be nice to present the user with some relatively recent negative tweets, e.g. from within the last few hours. Unfortunately, Twitter does not support any API tools that allow this kind of usage. Instead, a stand alone repository of tweets from the last few hours would be needed. The application can then request tweets from the repository to classify. This is similar to how the Twitter mobile application only shows snapshots of a user's tweet stream and they must request older tweets (and newer tweets) if they so desire. The issue with considering this method is that it revolves around creating a server and then one may get the idea that the classification should just be done on the server and the results sent to the application to reduce load on the phone. While such a model is viable, many of the concerns about which classifier to use are still very important. If the application has many users then training data still needs to be as small as possible,

classification needs to be quick and resource friendly, and so on. These restrictions still place a limit on what classifiers can be run while still maintaining a functional application.

However, there is one potentially significant reason to use a server based approach. The server based approach would have the potential to allow for automatic threshold tuning of the classifier. Namely, Experiment 2 is replicated on a small scale and the ideal threshold is picked from those examples. This removes relying on an arbitrary threshold that was picked based upon old experimental results. This ideal threshold can then be used to classify the random stream from the sample method of the Streaming API.

The issue not addressed here is that there are potentially going to still be far too many tweets for a user to view. Thus, it becomes necessary to investigate how information from tweets can be summarized. For example, it would be nice to display only information that the user hasn't seen previously. In addition, it may be useful to provide user with summarized sentiment about tweets by selecting a representative tweet and providing some knowledge of how many Twitter users agree or disagree with the information presented in the tweet (similar perhaps to likes/dislikes on Youtube, Facebook, etc). Regardless of what exactly is done, further work needs to be done regarding information management with respect to tweets, especially topical tweets.

Chapter 5

Conclusion

This thesis investigated the reduction of information overload for users of microblogging systems, specifically Twitter. This was done by examining out of the box methods for real-time ad hoc search on Twitter and the classification of tweets based upon hashtags. Real-time ad hoc search on Twitter was investigated through participation in the TREC 2011 Microblog Track. While hashtag-based classification was examined through three different experiments. Accordingly, the contributions of this thesis are as follows:

- Using the Wumpus Search Engine, Reciprocal Rank Fusion, and various ranking methods, a baseline was created that performs well by default.
- This baseline method can be significantly improved using pseudo-relevance feedback using a language model that is external to the corpus. Further improvement can be gained by using a language model that is comprised of a portion the Tweets11 corpus. This tweet-based pseudo-relevance feedback ends up performing competitively with other systems present at the Microblog Track; ranking among top 5 systems.
- This thesis showed that a minor adjustment to one of the ranking methods used can potentially increase retrieval results for all systems presented. Where this adjustment boosts the ranking of the tweet-based pseudo-relevance feedback run into an unofficial second place in Microblog Track results. However, it was argued that such an adjustment is Twitter specific and may in fact be non-optimal as it is merely based upon an untested hypothesis. In addition, such a change leads to a decreased ability to find highly relevant tweets which may not be desirable.

- The hashtag based classification results presented in this thesis indicated that Twitter users by and large use hashtags appropriately. This is indicated by the fact out of the box classifiers are able to determine whether or not a tweet should have a hashtag quite well. Given that the training and test data use the presence of a hashtag as the ground truth, the performance of the classifiers used suggest that users do not generally misuse or abuse hashtags.

- According to the results presented in this thesis, hashtag based classification appears to be able to be performed well, achieving < 11 (1-AUC)%, with small sets of training data (< 100 training examples). In addition, this performance is not achieved on tweets using a specific language but on tweets of different languages.

- Furthermore, this thesis showed that while optimized classifier libraries end up performing the best with extremely large sets of training data. Simple classifiers can be extremely competitive when very few training examples are used. Even classifiers that are traditionally word-based spam filters are able to perform this task well.

- Taking a set of tweets related by a hashtag and the classifying if they should have a different sub-topical hashtag appears to be much easier for all classifiers than classifying if a tweet should have a hashtag when compared to random tweets. In addition, the better performance of classifiers is done in spite of many questionable training examples. However, there also appears to be a higher signal to noise ratio in this case which likely contributes to the increased performance.

- Finally, this thesis proposes how hashtag based classification could be applied to real-time ad hoc search and outlines how hashtag based classification could be incorporated into an application on mobile devices while ensuring minimal resource usage.

In summary, this thesis has shown that it is possible to competitively perform real-time ad hoc search with out of the box methods and with some Twitter specific modifications can perform extremely well. In addition, hashtag based classification has been shown to be not only possible but is able to be performed well with very few training examples. Subsequently, this is indicative that users tend to not misuse hashtags in their tweets.

APPENDICES

Appendix A

Microblog Track - Per Topic Results

This appendix presents the per topic results (e.g. the number of relevant tweets returned at 30) for each of the runs conducted in Chapter 3 when all relevant and only highly relevant tweets are considered.

	Runs									
Topic	1	2	3	4	5	6	7	8	9	10
1	24	22	22	22	24	24	24	25	24	23
2	9	9	12	9	9	9	10	6	12	10
3	22	21	21	23	22	22	23	22	22	24
4	19	18	18	18	18	19	16	20	18	16
5	7	5	7	7	7	7	8	7	8	9
6	1	1	1	1	1	1	1	0	1	1
7	27	27	28	27	27	27	28	28	28	28
8	24	21	25	25	25	24	26	23	24	25
9	22	22	23	22	22	22	25	22	23	25
10	4	4	5	4	4	4	8	9	7	7
11	2	1	2	2	1	2	2	1	2	2
12	4	4	4	4	4	4	4	4	4	4
13	14	14	15	14	14	14	17	14	15	17
14	15	14	26	17	15	15	29	4	27	28
15	1	0	0	1	0	1	0	1	1	1
16	1	1	1	1	1	1	1	1	1	1
17	10	13	13	10	10	10	12	10	12	12
18	1	1	1	1	1	1	1	1	1	1
19	19	18	16	20	19	19	18	22	15	19
20	24	25	28	24	23	24	28	22	28	28
21	23	23	24	23	23	23	22	23	20	17
22	20	21	23	20	20	20	22	20	21	22
23	16	15	16	16	16	16	17	17	16	18
24	21	20	21	21	21	21	21	21	21	20
25	11	10	8	12	10	11	9	9	9	8
26	13	14	18	14	14	14	21	14	19	21
27	8	8	13	8	8	8	13	8	14	13
28	5	4	7	5	7	4	6	4	7	6
29	12	12	12	12	10	13	11	14	12	12
30	12	12	15	15	12	11	20	11	14	19
31	8	8	9	8	8	8	9	8	9	9
32	2	2	3	4	7	2	4	2	5	5
33	2	2	2	2	2	2	2	2	2	2
34	8	8	9	8	8	8	8	9	8	8
35	10	10	10	10	7	10	10	10	10	10
36	22	22	24	22	23	22	23	23	24	23
37	20	18	26	18	23	20	26	23	26	26
38	4	4	4	4	4	4	4	4	4	4
39	13	13	16	13	12	13	17	13	14	17
40	9	9	11	9	9	9	10	9	11	10
41	11	11	15	12	12	11	13	12	15	13
42	2	2	4	2	2	2	4	2	4	3
43	21	21	20	21	21	21	20	21	20	21
44	2	2	5	2	2	2	9	2	4	9
45	4	3	3	4	3	4	3	4	2	3
46	7	7	7	7	7	7	7	7	7	7
47	1	1	1	1	1	1	1	1	1	1
48	6	6	7	6	8	6	6	11	6	6
49	1	1	1	1	1	1	1	1	1	1

Table A.1: Per topic results for all runs described in Chapter 3 when all relevant tweets are considered

	Runs									
Topic	1	2	3	4	5	6	7	8	9	10
1	3	3	3	3	3	3	3	3	3	3
2	4	4	5	4	4	4	7	8	7	7
3	2	1	2	2	1	2	2	1	2	2
4	4	4	4	4	4	4	4	4	4	4
5	2	2	3	2	2	2	4	3	3	4
6	0	0	3	2	1	2	4	2	4	4
7	1	0	0	1	0	1	0	1	1	1
8	1	1	1	1	1	1	1	1	1	1
9	2	2	3	2	2	2	2	2	2	2
10	1	1	1	1	1	1	1	1	1	1
11	12	11	11	12	12	12	12	14	10	12
12	3	3	4	3	6	3	6	5	4	6
13	4	5	3	4	5	4	3	4	2	1
14	9	10	9	9	9	9	9	8	8	9
15	0	0	1	0	0	0	1	0	1	1
16	1	1	1	1	1	1	1	1	1	1
17	2	1	0	3	1	2	1	0	1	0
18	1	2	4	0	2	1	3	3	3	3
19	0	0	0	0	0	0	0	0	0	0
20	1	1	2	1	2	1	2	1	2	2
21	2	2	2	2	3	3	2	3	2	2
22	5	4	6	6	2	5	9	5	5	9
23	1	1	1	1	3	1	3	1	3	3
24	8	8	9	8	9	8	7	9	9	7
25	10	8	13	8	8	10	14	9	14	14
26	2	2	2	2	2	2	2	2	2	2
27	8	8	8	8	8	8	8	8	8	8
28	0	0	0	0	0	0	0	0	0	0
29	1	1	3	1	1	1	3	1	3	2
30	1	1	2	1	1	1	4	1	1	4
31	2	2	2	2	2	2	2	2	1	2
32	5	5	5	5	5	5	5	5	5	5
33	1	1	1	1	1	1	1	1	1	1

Table A.2: Per Topic results for all runs described in Chapter 3 when only highly relevant tweets are considered

Appendix B

Hashtag Based Classification Results

This appendix contains the (1-AUC)% with 95% confidence interval for each classifier and each training set used in Experiment 2 and Experiment 3 in Chapter 4. There contained here only for the sake of completeness and reference. Table B.1 and Table B.2 display the results for Experiment 2 as outlined above. While Table B.3 and Table B.4 display the results for Experiment 3 as outlined above.

Table B.1: (1-AUC)% with 95% confidence interval for Bogofilter, DMC, GCLR, and LibLinear for each training set used in Experiment 2

Set No.	Bogofilter	DMC	GCLR	LibLinear
1	26.5159 (26.2539 - 26.7795)	25.5498 (25.2609 - 25.8410)	17.5628 (17.3304 - 17.7977)	18.5065 (18.2421 - 18.7738)
2	16.1006 (15.8705 - 16.3333)	15.9817 (15.7540 - 16.2122)	11.3621 (11.1856 - 11.5410)	14.9945 (14.8006 - 15.1906)
3	14.1373 (13.9137 - 14.3640)	15.1710 (14.9598 - 15.3847)	11.0686 (10.8693 - 11.2712)	16.3272 (16.0872 - 16.5702)
4	14.6972 (14.4715 - 14.9259)	14.7029 (14.4774 - 14.9312)	10.5474 (10.3553 - 10.7426)	13.5551 (13.3462 - 13.7667)
5	14.4473 (14.2324 - 14.6650)	15.0380 (14.8174 - 15.2612)	10.3054 (10.1192 - 10.4946)	12.5973 (12.3791 - 12.8189)
6	14.5622 (14.3599 - 14.7669)	15.4884 (15.2864 - 15.6926)	10.6676 (10.4812 - 10.8570)	13.2600 (13.0654 - 13.4570)
7	14.9020 (14.6789 - 15.1279)	15.5295 (15.2953 - 15.7667)	10.5315 (10.3659 - 10.6994)	14.7390 (14.5162 - 14.9647)
8	14.7004 (14.5001 - 14.9030)	15.3769 (15.1788 - 15.5771)	10.2428 (10.0697 - 10.4186)	15.5103 (15.2787 - 15.7448)
9	14.4836 (14.2961 - 14.6731)	14.8544 (14.6427 - 15.0687)	10.0403 (9.8677 - 10.2155)	14.6036 (14.4231 - 14.7859)
10	14.5555 (14.3432 - 14.7704)	14.1508 (13.9563 - 14.3476)	10.1266 (9.9642 - 10.2913)	15.5014 (15.2569 - 15.7491)
11	14.3279 (14.1229 - 14.5353)	14.3152 (14.1124 - 14.5204)	9.9879 (9.8230 - 10.1553)	15.4274 (15.1942 - 15.6635)
12	14.1218 (13.9119 - 14.3343)	14.2806 (14.0670 - 14.4969)	9.8625 (9.7250 - 10.0018)	14.7848 (14.5866 - 14.9852)
13	13.9340 (13.7369 - 14.1335)	13.6901 (13.4755 - 13.9075)	9.7845 (9.6331 - 9.9380)	16.1166 (15.9081 - 16.3274)
14	14.3309 (14.1431 - 14.5207)	14.0938 (13.8968 - 14.2932)	9.6777 (9.5210 - 9.8366)	16.4688 (16.2326 - 16.7078)
15	13.9600 (13.7194 - 14.2042)	13.6738 (13.4895 - 13.8602)	9.6766 (9.4904 - 9.8660)	16.8255 (16.5916 - 17.0620)
16	14.3949 (14.1933 - 14.5990)	13.5874 (13.4064 - 13.7705)	9.8063 (9.6418 - 9.9732)	17.8657 (17.6343 - 18.0995)
17	14.5120 (14.3223 - 14.7037)	13.5773 (13.3582 - 13.7995)	9.7079 (9.5644 - 9.8533)	18.1519 (17.9278 - 18.3782)
18	14.6237 (14.4186 - 14.8312)	13.0789 (12.8970 - 13.2631)	9.5288 (9.3415 - 9.7194)	17.4697 (17.2773 - 17.6639)
19	14.2286 (14.0334 - 14.4260)	13.0136 (12.8009 - 13.2293)	9.6241 (9.4594 - 9.7914)	18.4278 (18.1793 - 18.6789)
20	14.3614 (14.1530 - 14.5724)	12.3291 (12.1394 - 12.5214)	9.4882 (9.3126 - 9.6667)	16.8903 (16.6648 - 17.1182)
21	14.3725 (14.1325 - 14.6159)	12.5250 (12.3314 - 12.7211)	9.5985 (9.4415 - 9.7579)	17.4681 (17.2394 - 17.6993)
22	14.4405 (14.2119 - 14.6720)	12.1704 (11.9814 - 12.3620)	9.3982 (9.2271 - 9.5720)	17.2619 (17.0565 - 17.4693)
23	12.9631 (12.7713 - 13.1574)	11.5460 (11.3792 - 11.7149)	9.1698 (9.0221 - 9.3196)	15.8694 (15.6760 - 16.0647)
24	12.6704 (12.4827 - 12.8605)	11.4641 (11.2863 - 11.6443)	9.1771 (9.0293 - 9.3270)	15.9631 (15.7503 - 16.1782)
25	12.3227 (12.1591 - 12.4881)	11.2139 (11.0453 - 11.3849)	9.1914 (9.0442 - 9.3408)	15.7996 (15.5871 - 16.0145)
26	12.5529 (12.3590 - 12.7494)	11.3948 (11.2080 - 11.5843)	9.2857 (9.1237 - 9.4502)	16.6331 (16.4097 - 16.8590)
27	12.6874 (12.4905 - 12.8869)	11.7951 (11.6271 - 11.9653)	9.4601 (9.2823 - 9.6409)	18.0178 (17.7792 - 18.2589)
28	12.6400 (12.4408 - 12.8419)	11.6704 (11.5137 - 11.8290)	9.4234 (9.2603 - 9.5890)	17.2571 (17.0320 - 17.4846)
29	12.5835 (12.3780 - 12.7920)	12.2895 (12.1065 - 12.4748)	9.2898 (9.1416 - 9.4401)	16.8273 (16.5952 - 17.0621)
30	12.4670 (12.2944 - 12.6416)	11.9649 (11.7718 - 12.1608)	9.2081 (9.0354 - 9.3837)	16.6917 (16.4685 - 16.9173)
31	12.4503 (12.3010 - 12.6010)	11.9433 (11.7510 - 12.1384)	9.1821 (9.0190 - 9.3478)	17.1962 (16.9509 - 17.4442)
32	12.4658 (12.2622 - 12.6724)	11.7279 (11.5349 - 11.9237)	9.1328 (8.9801 - 9.2879)	17.3710 (17.1227 - 17.6220)
33	12.4320 (12.2065 - 12.6610)	11.7048 (11.5255 - 11.8866)	8.9953 (8.8441 - 9.1487)	16.8972 (16.6500 - 17.1472)
34	12.2837 (12.0992 - 12.4705)	11.5761 (11.3949 - 11.7598)	8.9568 (8.8202 - 9.0954)	16.1470 (15.9145 - 16.3823)
35	12.5489 (12.3476 - 12.7530)	11.9547 (11.7742 - 12.1376)	9.0026 (8.8694 - 9.1377)	16.8966 (16.6703 - 17.1254)
36	12.4533 (12.2610 - 12.6482)	11.8089 (11.6156 - 12.0049)	8.9324 (8.7740 - 9.0933)	16.5809 (16.3770 - 16.7869)
37	11.9071 (11.7185 - 12.0984)	11.5790 (11.3780 - 11.7831)	8.8318 (8.6690 - 8.9974)	16.8324 (16.6017 - 17.0657)
38	12.9689 (12.7568 - 13.1839)	11.8955 (11.6841 - 12.1101)	8.8527 (8.7029 - 9.0048)	17.1515 (16.9322 - 17.3731)
39	12.9572 (12.7492 - 13.1682)	11.8863 (11.7005 - 12.0746)	8.8503 (8.6931 - 9.0100)	16.9741 (16.7295 - 17.2216)
40	12.5437 (12.3354 - 12.7550)	11.8211 (11.6229 - 12.0222)	8.8375 (8.6897 - 8.9875)	16.3198 (16.1029 - 16.5391)
41	12.6163 (12.4153 - 12.8201)	11.9251 (11.7334 - 12.1195)	8.7589 (8.6208 - 8.8989)	16.4940 (16.2750 - 16.7155)
42	12.6871 (12.4694 - 12.9081)	12.0395 (11.8443 - 12.2376)	8.7990 (8.6531 - 8.9473)	15.7673 (15.5514 - 15.9856)
43	12.7123 (12.5326 - 12.8943)	12.0444 (11.8619 - 12.2293)	8.7357 (8.5619 - 8.9127)	15.8542 (15.6342 - 16.0766)
44	12.7996 (12.5978 - 13.0041)	12.1181 (11.8905 - 12.3495)	8.6908 (8.5495 - 8.8342)	16.3539 (16.1406 - 16.5694)
45	13.1117 (12.8986 - 13.3278)	12.3126 (12.1187 - 12.5092)	8.8277 (8.6873 - 8.9700)	17.1874 (16.9810 - 17.3958)
46	13.0968 (12.8892 - 13.3072)	12.2734 (12.1015 - 12.4474)	8.7129 (8.5607 - 8.8676)	16.6917 (16.4731 - 16.9126)
47	13.0033 (12.8018 - 13.2074)	12.2309 (12.0572 - 12.4067)	8.7577 (8.5980 - 8.9200)	17.0408 (16.8223 - 17.2614)
48	12.9701 (12.7744 - 13.1683)	12.1139 (11.9337 - 12.2965)	8.7137 (8.5636 - 8.8661)	17.1555 (16.9081 - 17.4059)
49	12.9407 (12.7445 - 13.1395)	12.0655 (11.8665 - 12.2673)	8.7690 (8.6153 - 8.9252)	17.3438 (17.1130 - 17.5770)
50	12.8139 (12.6025 - 13.0284)	12.0042 (11.8219 - 12.1889)	8.7474 (8.6012 - 8.8958)	17.6968 (17.4747 - 17.9211)
51	12.7972 (12.5850 - 13.0124)	11.9127 (11.6973 - 12.1317)	8.7286 (8.5913 - 8.8680)	17.7948 (17.5931 - 17.9984)
52	12.6845 (12.4790 - 12.8929)	11.5320 (11.3407 - 11.7261)	8.5668 (8.4005 - 8.7362)	17.4759 (17.2458 - 17.7084)
53	12.9556 (12.7596 - 13.1541)	11.7543 (11.5341 - 11.9781)	8.5028 (8.3395 - 8.6690)	17.7238 (17.5067 - 17.9429)
54	12.7645 (12.5653 - 12.9664)	11.0541 (10.8643 - 11.2469)	8.2841 (8.1347 - 8.4359)	16.2596 (16.0664 - 16.4547)
55	13.0905 (12.8864 - 13.2973)	11.0436 (10.8538 - 11.2362)	8.2322 (8.0739 - 8.3934)	16.2147 (15.9874 - 16.4446)
56	13.1043 (12.9091 - 13.3020)	10.9444 (10.7804 - 11.1105)	8.1444 (7.9986 - 8.2925)	16.2266 (15.9826 - 16.4737)
57	13.0493 (12.8595 - 13.2415)	10.8402 (10.6737 - 11.0089)	8.1494 (8.0025 - 8.2988)	16.1179 (15.9032 - 16.3350)
58	13.0550 (12.8379 - 13.2753)	10.8512 (10.6770 - 11.0278)	8.2031 (8.0610 - 8.3475)	16.0760 (15.8745 - 16.2796)
59	13.0220 (12.8163 - 13.2305)	10.7558 (10.5764 - 10.9379)	8.1731 (8.0160 - 8.3331)	16.5618 (16.3452 - 16.7806)
60	14.1956 (13.9753 - 14.4189)	10.9250 (10.7533 - 11.0992)	8.3461 (8.1947 - 8.5001)	16.9824 (16.7360 - 17.2317)
61	14.0423 (13.8050 - 14.2830)	10.9261 (10.7669 - 11.0873)	8.4086 (8.2609 - 8.5588)	17.0504 (16.8270 - 17.2762)
62	13.6671 (13.4546 - 13.8824)	10.8154 (10.6309 - 11.0026)	8.3242 (8.1734 - 8.4775)	16.6505 (16.4311 - 16.8723)
63	13.6848 (13.4643 - 13.9084)	10.7853 (10.6139 - 10.9591)	8.3474 (8.1965 - 8.5008)	16.6499 (16.4193 - 16.8831)
64	13.6617 (13.4343 - 13.8922)	10.9542 (10.7876 - 11.1231)	8.4510 (8.3002 - 8.6043)	17.4728 (17.2571 - 17.6906)
65	13.6562 (13.4292 - 13.8864)	10.9846 (10.7876 - 11.1847)	8.4838 (8.3309 - 8.6392)	17.6137 (17.3977 - 17.8319)
66	13.6616 (13.4564 - 13.8693)	11.0765 (10.8732 - 11.2830)	8.4224 (8.2475 - 8.6007)	17.4656 (17.2299 - 17.7038)
67	13.7413 (13.5305 - 13.9548)	11.1103 (10.9490 - 11.2738)	8.4646 (8.3125 - 8.6193)	17.6531 (17.4357 - 17.8726)
68	13.8764 (13.6573 - 14.0985)	11.1602 (10.9774 - 11.3457)	8.3972 (8.2707 - 8.5255)	17.6320 (17.3919 - 17.8747)
69	13.8812 (13.6794 - 14.0855)	11.2373 (11.0266 - 11.4515)	8.4510 (8.2874 - 8.6175)	17.9226 (17.6947 - 18.1527)

Table B.1: (1-AUC)% with 95% confidence interval for Bogofilter, DMC, GCLR, and LibLinear for each training set used in Experiment 2

Set No.	Bogofilter	DMC	GCLR	LibLinear
70	13.8344 (13.6371 - 14.0340)	11.1402 (10.9721 - 11.3106)	8.4793 (8.3226 - 8.6386)	17.5883 (17.3818 - 17.7967)
71	13.5798 (13.3650 - 13.7975)	11.2458 (11.0502 - 11.4444)	8.6061 (8.4443 - 8.7708)	17.9644 (17.7726 - 18.1579)
72	13.5643 (13.3739 - 13.7569)	11.1370 (10.9501 - 11.3268)	8.5276 (8.3659 - 8.6920)	17.5373 (17.3023 - 17.7748)
73	13.4939 (13.2674 - 13.7237)	11.3463 (11.1474 - 11.5482)	8.4926 (8.3480 - 8.6395)	17.9024 (17.6827 - 18.1242)
74	13.6004 (13.4030 - 13.8003)	11.4667 (11.2578 - 11.6789)	8.6259 (8.4866 - 8.7672)	18.3910 (18.1578 - 18.6265)
75	13.5866 (13.3590 - 13.8175)	11.4525 (11.2710 - 11.6367)	8.6146 (8.4476 - 8.7846)	18.5022 (18.2737 - 18.7329)
76	13.4138 (13.2217 - 13.6084)	11.4580 (11.2867 - 11.6315)	8.6937 (8.5348 - 8.8552)	18.7264 (18.4855 - 18.9696)
77	13.4683 (13.2671 - 13.6720)	11.4161 (11.2354 - 11.5993)	8.6716 (8.5016 - 8.8446)	18.6305 (18.4189 - 18.8440)
78	13.6569 (13.4706 - 13.8455)	11.5013 (11.2941 - 11.7118)	8.7762 (8.5973 - 8.9583)	18.5276 (18.2838 - 18.7738)
79	13.6638 (13.4474 - 13.8831)	11.4403 (11.2668 - 11.6160)	8.7303 (8.5682 - 8.8951)	18.5379 (18.3148 - 18.7630)
80	13.7102 (13.4888 - 13.9347)	11.4667 (11.2749 - 11.6614)	8.8221 (8.6488 - 8.9986)	18.6394 (18.4172 - 18.8636)
81	13.6856 (13.4539 - 13.9207)	11.4512 (11.2654 - 11.6397)	8.8474 (8.6854 - 9.0120)	18.5855 (18.3619 - 18.8111)
82	13.6717 (13.4492 - 13.8973)	11.4643 (11.2817 - 11.6495)	8.7909 (8.6576 - 8.9260)	18.5643 (18.3376 - 18.7932)
83	13.8898 (13.6751 - 14.1072)	11.4906 (11.2963 - 11.6878)	8.7993 (8.6201 - 8.9818)	18.3781 (18.1720 - 18.5859)
84	13.8939 (13.6836 - 14.1068)	11.4346 (11.2340 - 11.6383)	8.7701 (8.6174 - 8.9252)	18.2359 (18.0283 - 18.4454)
85	13.7973 (13.6024 - 13.9945)	11.3454 (11.1561 - 11.5374)	8.6967 (8.5521 - 8.8435)	18.4943 (18.2528 - 18.7382)
86	14.1704 (13.9479 - 14.3959)	11.3397 (11.1469 - 11.5354)	8.6937 (8.5314 - 8.8588)	18.4682 (18.2416 - 18.6970)
87	13.8016 (13.6062 - 13.9993)	11.3789 (11.1721 - 11.5890)	8.7382 (8.5817 - 8.8972)	18.5702 (18.3451 - 18.7974)
88	14.2996 (14.0798 - 14.5222)	11.3772 (11.1528 - 11.6056)	8.8253 (8.6636 - 8.9897)	18.8355 (18.6099 - 19.0632)
89	14.1701 (13.9698 - 14.3728)	11.3032 (11.1049 - 11.5045)	8.7597 (8.6170 - 8.9046)	18.8900 (18.6694 - 19.1127)
90	13.5236 (13.3103 - 13.7397)	11.2966 (11.0974 - 11.4989)	8.6611 (8.5001 - 8.8248)	18.6754 (18.4532 - 18.8996)
91	13.6021 (13.4137 - 13.7927)	11.2754 (11.0838 - 11.4699)	8.7321 (8.5876 - 8.8787)	18.7907 (18.5885 - 18.9946)
92	13.5949 (13.3930 - 13.7994)	11.2088 (11.0066 - 11.4144)	8.6756 (8.5318 - 8.8216)	18.8042 (18.5842 - 19.0262)
93	13.5827 (13.3946 - 13.7730)	11.1795 (10.9962 - 11.3654)	8.6217 (8.4458 - 8.8009)	18.8045 (18.5864 - 19.0245)
94	13.6577 (13.4611 - 13.8568)	11.1679 (11.0058 - 11.3320)	8.5846 (8.4409 - 8.7306)	18.7431 (18.4717 - 19.0175)
95	13.6410 (13.4274 - 13.8575)	11.1219 (10.9290 - 11.3177)	8.6216 (8.4873 - 8.7578)	18.8947 (18.6893 - 19.1018)
96	13.6863 (13.4718 - 13.9037)	11.1377 (10.9695 - 11.3082)	8.6256 (8.4633 - 8.7907)	18.9973 (18.7502 - 19.2468)
97	13.7244 (13.5013 - 13.9505)	11.1503 (10.9714 - 11.3318)	8.6428 (8.5001 - 8.7877)	19.1118 (18.8295 - 19.3973)
98	14.4698 (14.2737 - 14.6682)	11.2396 (11.0434 - 11.4389)	8.8380 (8.6998 - 8.9783)	19.4125 (19.1722 - 19.6552)
99	14.4319 (14.2050 - 14.6618)	11.2215 (11.0239 - 11.4223)	8.7661 (8.6119 - 8.9228)	19.2634 (19.0182 - 19.5110)
100	14.4171 (14.2019 - 14.6351)	11.2000 (11.0049 - 11.3981)	8.8466 (8.6940 - 9.0016)	19.3946 (19.1716 - 19.6196)
101	14.4188 (14.2113 - 14.6288)	10.6363 (10.4639 - 10.8112)	9.1922 (9.0270 - 9.3600)	19.2502 (19.0171 - 19.4856)
102	14.3583 (14.1415 - 14.5779)	10.3778 (10.1904 - 10.5683)	9.2557 (9.1056 - 9.4081)	18.1180 (17.8959 - 18.3421)
103	13.8119 (13.5894 - 14.0375)	9.9665 (9.7882 - 10.1477)	8.0742 (7.9180 - 8.2332)	16.9589 (16.7101 - 17.2106)
104	13.6151 (13.4077 - 13.8253)	9.7881 (9.5883 - 9.9917)	8.2970 (8.1465 - 8.4500)	17.0875 (16.8994 - 17.2773)
105	13.5972 (13.3839 - 13.8134)	9.3329 (9.1500 - 9.5191)	8.7967 (8.6487 - 8.9470)	16.9673 (16.7411 - 17.1960)
106	12.3329 (12.1605 - 12.5074)	8.1566 (7.9949 - 8.3211)	10.3359 (10.1585 - 10.5161)	13.0944 (12.8771 - 13.3148)
107	11.7689 (11.5673 - 11.9735)	7.9545 (7.7897 - 8.1225)	9.8927 (9.7104 - 10.0780)	12.2188 (12.0156 - 12.4248)
108	11.3621 (11.1828 - 11.5438)	7.6629 (7.5117 - 7.8169)	10.0580 (9.8713 - 10.2478)	11.2232 (11.0172 - 11.4327)
109	11.1004 (10.9331 - 11.2698)	7.4631 (7.3277 - 7.6009)	8.4290 (8.2856 - 8.5746)	10.7176 (10.5288 - 10.9094)
110	10.7469 (10.5891 - 10.9069)	7.3562 (7.2085 - 7.5067)	8.9326 (8.7747 - 9.0930)	10.2578 (10.0791 - 10.4394)
111	10.6614 (10.4710 - 10.8548)	7.2021 (7.0527 - 7.3545)	9.7108 (9.5489 - 9.8752)	9.9420 (9.7787 - 10.1078)
112	10.7725 (10.5954 - 10.9522)	7.0106 (6.8566 - 7.1678)	10.0243 (9.8628 - 10.1883)	9.6509 (9.4713 - 9.8335)
113	10.5516 (10.3714 - 10.7346)	6.9021 (6.7550 - 7.0522)	10.1034 (9.9343 - 10.2750)	9.4128 (9.2627 - 9.5650)
114	10.2942 (10.1174 - 10.4737)	6.8045 (6.6514 - 6.9608)	8.6153 (8.4536 - 8.7798)	8.8404 (8.6777 - 9.0059)
115	9.9911 (9.8195 - 10.1653)	6.7725 (6.6162 - 6.9323)	8.3650 (8.2320 - 8.4998)	8.2813 (8.1348 - 8.4302)
116	9.6017 (9.4602 - 9.7450)	6.6186 (6.4792 - 6.7608)	8.3975 (8.2601 - 8.5370)	8.1653 (8.0184 - 8.3145)
117	9.4312 (9.2789 - 9.5857)	6.4965 (6.3409 - 6.6557)	8.1061 (7.9474 - 8.2678)	7.9305 (7.7679 - 8.0962)
118	9.4114 (9.2537 - 9.5716)	6.4398 (6.3092 - 6.5730)	8.0197 (7.8560 - 8.1865)	7.7550 (7.6149 - 7.8974)
119	9.3740 (9.2146 - 9.5360)	6.4322 (6.3044 - 6.5625)	8.2136 (8.0480 - 8.3824)	7.6720 (7.5214 - 7.8254)
120	9.2825 (9.1218 - 9.4456)	6.4113 (6.2782 - 6.5470)	7.9890 (7.8347 - 8.1461)	7.4966 (7.3644 - 7.6309)
121	9.2214 (9.0714 - 9.3736)	6.3871 (6.2609 - 6.5157)	8.0493 (7.8759 - 8.2261)	7.5169 (7.3457 - 7.6918)
122	8.6727 (8.5373 - 8.8101)	6.3787 (6.2624 - 6.4970)	8.3062 (8.1571 - 8.4578)	7.3833 (7.2331 - 7.5363)
123	8.7505 (8.6196 - 8.8832)	6.3414 (6.2044 - 6.4812)	8.2663 (8.1062 - 8.4292)	7.3657 (7.2114 - 7.5231)
124	8.6303 (8.4648 - 8.7986)	6.2951 (6.1548 - 6.4385)	8.0236 (7.8855 - 8.1640)	7.0639 (6.9250 - 7.2054)
125	8.4394 (8.2932 - 8.5880)	6.2004 (6.0678 - 6.3358)	7.8602 (7.7178 - 8.0050)	6.9071 (6.7825 - 7.0338)
126	8.3839 (8.2337 - 8.5366)	6.0865 (5.9619 - 6.2137)	7.8195 (7.6931 - 7.9477)	6.7092 (6.5649 - 6.8565)
127	8.2980 (8.1353 - 8.4637)	6.0191 (5.8914 - 6.1494)	7.5394 (7.3770 - 7.7051)	6.5722 (6.4393 - 6.7078)
128	8.2962 (8.1436 - 8.4514)	6.0173 (5.8947 - 6.1423)	7.4720 (7.3347 - 7.6116)	6.4881 (6.3635 - 6.6149)
129	8.4808 (8.3281 - 8.6361)	5.9888 (5.8658 - 6.1141)	7.2919 (7.1430 - 7.4436)	6.4066 (6.2896 - 6.5257)
130	8.3804 (8.2469 - 8.5159)	5.9351 (5.7849 - 6.0890)	7.5769 (7.4508 - 7.7049)	6.3397 (6.2014 - 6.4808)
131	8.3924 (8.2362 - 8.5513)	5.9110 (5.7792 - 6.0455)	7.5355 (7.3893 - 7.6844)	6.2339 (6.1185 - 6.3513)
132	8.3587 (8.2033 - 8.5167)	5.8673 (5.7259 - 6.0120)	7.7304 (7.5728 - 7.8910)	6.1260 (6.0003 - 6.2542)
133	8.3036 (8.1629 - 8.4464)	5.8324 (5.7115 - 5.9557)	7.7273 (7.5845 - 7.8725)	6.0153 (5.8864 - 6.1468)
134	8.4256 (8.2758 - 8.5779)	5.8279 (5.6971 - 5.9614)	7.5351 (7.3874 - 7.6855)	5.8524 (5.7324 - 5.9748)
135	8.3971 (8.2438 - 8.5530)	5.7817 (5.6578 - 5.9081)	7.2972 (7.1608 - 7.4360)	5.6910 (5.5551 - 5.8301)
136	8.2527 (8.1028 - 8.4051)	5.7713 (5.6403 - 5.9051)	7.6343 (7.5011 - 7.7696)	5.6664 (5.5307 - 5.8052)
137	8.2331 (8.0905 - 8.3780)	5.7638 (5.6349 - 5.8956)	8.4415 (8.2654 - 8.6210)	5.6678 (5.5444 - 5.7939)
138	8.2731 (8.1293 - 8.4192)	5.7411 (5.6217 - 5.8628)	8.8837 (8.7329 - 9.0367)	5.6577 (5.5248 - 5.7936)
139	8.2813 (8.1298 - 8.4354)	5.7011 (5.5616 - 5.8438)	8.9306 (8.7670 - 9.0969)	5.6867 (5.5535 - 5.8230)

Table B.1: (1-AUC)% with 95% confidence interval for Bogofilter, DMC, GCLR, and LibLinear for each training set used in Experiment 2

Set No.	Bogofilter	DMC	GCLR	LibLinear
140	8.2416 (8.0861 - 8.3997)	5.6555 (5.5253 - 5.7885)	8.6996 (8.5452 - 8.8566)	5.6331 (5.5088 - 5.7599)
141	8.2097 (8.0587 - 8.3632)	5.6225 (5.5182 - 5.7288)	8.1260 (7.9805 - 8.2739)	5.6126 (5.4825 - 5.7456)
142	8.2182 (8.0761 - 8.3626)	5.5905 (5.4781 - 5.7050)	7.7310 (7.5888 - 7.8756)	5.5511 (5.4277 - 5.6771)
143	8.1957 (8.0355 - 8.3589)	5.5628 (5.4405 - 5.6877)	7.5099 (7.3535 - 7.6695)	5.3972 (5.2692 - 5.5282)
144	8.1695 (8.0045 - 8.3376)	5.5402 (5.4188 - 5.6642)	7.4317 (7.2872 - 7.5788)	5.3500 (5.2266 - 5.4763)
145	8.2286 (8.0665 - 8.3936)	5.4955 (5.3668 - 5.6272)	7.2348 (7.0850 - 7.3876)	5.3132 (5.1979 - 5.4309)
146	8.1866 (8.0553 - 8.3199)	5.4565 (5.3359 - 5.5797)	7.3672 (7.2228 - 7.5143)	5.2894 (5.1697 - 5.4117)
147	8.1412 (8.0043 - 8.2802)	5.4201 (5.2845 - 5.5590)	7.2137 (7.0751 - 7.3548)	5.2722 (5.1555 - 5.3914)
148	8.1015 (7.9774 - 8.2273)	5.3935 (5.2780 - 5.5113)	7.0205 (6.8790 - 7.1646)	5.3010 (5.1865 - 5.4180)
149	8.0806 (7.9312 - 8.2325)	5.3698 (5.2436 - 5.4989)	6.7749 (6.6332 - 6.9195)	5.2239 (5.1166 - 5.3333)
150	8.0667 (7.9419 - 8.1932)	5.3000 (5.1716 - 5.4315)	6.8513 (6.7096 - 6.9959)	5.1402 (5.0248 - 5.2581)
151	8.0363 (7.8981 - 8.1766)	5.2779 (5.1793 - 5.3783)	6.6402 (6.5030 - 6.7800)	5.0541 (4.9418 - 5.1689)
152	7.9063 (7.7552 - 8.0600)	5.2034 (5.0787 - 5.3311)	6.6236 (6.4925 - 6.7571)	4.9601 (4.8446 - 5.0782)
153	7.8508 (7.6962 - 8.0083)	5.1476 (5.0409 - 5.2563)	6.5012 (6.3711 - 6.6336)	4.8557 (4.7461 - 4.9678)
154	7.7214 (7.5826 - 7.8625)	5.0770 (4.9613 - 5.1952)	6.2193 (6.0737 - 6.3681)	4.6731 (4.5648 - 4.7839)
155	7.6619 (7.5099 - 7.8167)	5.0629 (4.9548 - 5.1733)	6.1497 (6.0181 - 6.2841)	4.6438 (4.5481 - 4.7415)
156	7.6209 (7.4741 - 7.7703)	5.0351 (4.9205 - 5.1522)	5.8581 (5.7281 - 5.9909)	4.5672 (4.4785 - 4.6575)
157	7.5935 (7.4567 - 7.7325)	5.0067 (4.8941 - 5.1217)	5.7629 (5.6267 - 5.9022)	4.5126 (4.3975 - 4.6305)
158	7.5473 (7.3929 - 7.7047)	5.0056 (4.8852 - 5.1289)	5.7031 (5.5689 - 5.8403)	4.5066 (4.4061 - 4.6093)
159	7.5014 (7.3489 - 7.6568)	4.9528 (4.8418 - 5.0662)	5.5057 (5.3881 - 5.6258)	4.3152 (4.2002 - 4.4332)
160	7.4706 (7.3326 - 7.6111)	4.9188 (4.8039 - 5.0364)	5.3641 (5.2501 - 5.4804)	4.2069 (4.1141 - 4.3017)
161	7.4210 (7.2617 - 7.5835)	4.8552 (4.7471 - 4.9656)	5.3736 (5.2471 - 5.5030)	4.1311 (4.0275 - 4.2373)
162	6.9742 (6.8469 - 7.1038)	4.7861 (4.6861 - 4.8881)	5.2283 (5.1037 - 5.3557)	4.0291 (3.9312 - 4.1294)
163	6.7827 (6.6305 - 6.9381)	4.6572 (4.5566 - 4.7600)	5.1673 (5.0483 - 5.2890)	3.8274 (3.7294 - 3.9279)
164	6.7311 (6.6053 - 6.8591)	4.5853 (4.4662 - 4.7073)	5.0494 (4.9237 - 5.1781)	3.7926 (3.6929 - 3.8948)

Table B.2: (1-AUC)% with 95% confidence interval for OSBF-Lua, TextCat, SVMLight, and LibLinear, for each training set used in Experiment 2

Set No.	OSBF-Lua	TextCat	SVMLight	LibSVM
1	37.3685 (37.0472 - 37.6909)	21.0482 (20.7894 - 21.3094)	18.1842 (17.9166 - 18.4550)	63.8746 (63.5813 - 64.1669)
2	48.2900 (47.9601 - 48.6201)	11.3523 (11.1921 - 11.5145)	11.1970 (11.0280 - 11.3683)	12.5587 (12.3754 - 12.7443)
3	38.6457 (38.3303 - 38.9621)	12.0855 (11.8938 - 12.2799)	10.9192 (10.7290 - 11.1123)	11.6487 (11.4571 - 11.8430)
4	31.9870 (31.7305 - 32.2445)	11.6007 (11.4135 - 11.7905)	10.2136 (10.0342 - 10.3958)	10.7082 (10.5385 - 10.8804)
5	30.2400 (29.9394 - 30.5422)	8.8987 (8.7394 - 9.0606)	10.2005 (10.0173 - 10.3867)	10.7115 (10.5558 - 10.8692)
6	28.3076 (28.0234 - 28.5935)	9.6849 (9.5159 - 9.8565)	10.1748 (9.9896 - 10.3632)	10.4780 (10.3091 - 10.6493)
7	27.3304 (27.0299 - 27.6329)	10.4254 (10.2446 - 10.6091)	9.8867 (9.7385 - 10.0369)	9.8798 (9.7259 - 10.0359)
8	26.8455 (26.5776 - 27.1150)	11.2375 (11.0748 - 11.4024)	9.9923 (9.8163 - 10.1712)	9.9866 (9.8063 - 10.1699)
9	25.2841 (24.9775 - 25.5931)	10.2349 (10.0664 - 10.4059)	10.0035 (9.8484 - 10.1608)	9.7549 (9.5668 - 9.9463)
10	25.1862 (24.9332 - 25.4409)	11.3028 (11.1288 - 11.4792)	10.0041 (9.8187 - 10.1926)	9.8038 (9.6340 - 9.9763)
11	24.4018 (24.1447 - 24.6608)	12.4425 (12.2454 - 12.6424)	9.8373 (9.6740 - 10.0031)	9.7281 (9.5605 - 9.8983)
12	23.2822 (22.9968 - 23.5700)	12.7913 (12.6003 - 12.9849)	9.5897 (9.4258 - 9.7561)	9.3583 (9.2001 - 9.5190)
13	23.4867 (23.2301 - 23.7452)	13.0184 (12.8480 - 13.1908)	9.5378 (9.3842 - 9.6936)	9.4995 (9.3253 - 9.6765)
14	23.0055 (22.7076 - 23.3062)	13.1636 (12.9661 - 13.3636)	9.3179 (9.1545 - 9.4839)	9.2627 (9.0965 - 9.4318)
15	23.5834 (23.2891 - 23.8803)	13.6589 (13.4526 - 13.8679)	9.2801 (9.1130 - 9.4498)	9.3476 (9.1861 - 9.5116)
16	25.3154 (25.0474 - 25.5853)	15.3967 (15.1594 - 15.6370)	9.2785 (9.1310 - 9.4280)	9.4246 (9.2554 - 9.5965)
17	24.7217 (24.4495 - 24.9961)	13.7443 (13.5271 - 13.9645)	9.2123 (9.0456 - 9.3818)	9.1967 (9.0620 - 9.3332)
18	24.3432 (24.0793 - 24.6090)	13.6375 (13.4377 - 13.8398)	9.0622 (8.9149 - 9.2117)	9.0070 (8.8454 - 9.1713)
19	25.5858 (25.2737 - 25.9004)	15.3823 (15.1812 - 15.5855)	9.0938 (8.9344 - 9.2557)	9.0283 (8.8737 - 9.1853)
20	25.0383 (24.7691 - 25.3094)	15.1584 (14.9418 - 15.3776)	9.0196 (8.8516 - 9.1905)	9.0089 (8.8574 - 9.1628)
21	24.6858 (24.4099 - 24.9637)	16.0404 (15.8524 - 16.2301)	8.9436 (8.7755 - 9.1146)	8.8336 (8.6863 - 8.9832)
22	23.9202 (23.6511 - 24.1914)	14.5489 (14.3442 - 14.7559)	8.7142 (8.5665 - 8.8641)	8.6009 (8.4465 - 8.7579)
23	20.7962 (20.5466 - 21.0481)	13.0455 (12.8408 - 13.2531)	8.6080 (8.4684 - 8.7496)	8.5017 (8.3555 - 8.6502)
24	20.6432 (20.3984 - 20.8902)	13.0648 (12.8734 - 13.2585)	8.5699 (8.4053 - 8.7374)	8.4367 (8.2759 - 8.6004)
25	20.5122 (20.2306 - 20.7967)	12.8408 (12.6487 - 13.0354)	8.5756 (8.4242 - 8.7295)	8.4232 (8.2632 - 8.5861)
26	21.2789 (21.0269 - 21.5330)	13.7350 (13.5350 - 13.9376)	8.6512 (8.4963 - 8.8087)	8.5634 (8.4012 - 8.7284)
27	22.0340 (21.8008 - 22.2690)	13.5583 (13.3532 - 13.7661)	8.7338 (8.5584 - 8.9125)	8.6539 (8.5039 - 8.8063)
28	21.6965 (21.4477 - 21.9473)	13.5860 (13.3866 - 13.7878)	8.6903 (8.5234 - 8.8602)	8.6155 (8.4701 - 8.7633)
29	21.2862 (21.0444 - 21.5301)	11.7800 (11.6087 - 11.9535)	8.7866 (8.6374 - 8.9382)	9.1273 (8.9561 - 9.3014)
30	20.2256 (19.9781 - 20.4753)	11.1963 (10.9934 - 11.4025)	8.8566 (8.6842 - 9.0321)	9.1034 (8.9359 - 9.2737)
31	19.9195 (19.6425 - 20.1995)	11.1495 (10.9555 - 11.3466)	8.8906 (8.7116 - 9.0730)	9.1230 (8.9639 - 9.2846)
32	20.2103 (19.9762 - 20.4465)	11.1112 (10.9459 - 11.2788)	8.9084 (8.7495 - 9.0700)	9.1212 (8.9654 - 9.2793)
33	19.8490 (19.6080 - 20.0922)	10.5413 (10.3475 - 10.7384)	8.7547 (8.5839 - 8.9286)	8.9925 (8.8338 - 9.1537)
34	18.8827 (18.6383 - 19.1296)	10.1660 (9.9877 - 10.3472)	8.5668 (8.3982 - 8.7384)	8.6568 (8.5028 - 8.8133)
35	18.8769 (18.6196 - 19.1368)	10.4331 (10.2754 - 10.5929)	8.6019 (8.4364 - 8.7705)	8.6891 (8.5330 - 8.8477)
36	18.7307 (18.4982 - 18.9655)	10.7166 (10.5526 - 10.8828)	8.5201 (8.3818 - 8.6604)	8.6817 (8.5221 - 8.8440)
37	19.0615 (18.8327 - 19.2923)	11.0239 (10.8568 - 11.1933)	8.5173 (8.3610 - 8.6763)	8.8175 (8.6316 - 9.0069)
38	19.5211 (19.2603 - 19.7847)	11.4971 (11.3270 - 11.6695)	8.5807 (8.4201 - 8.7442)	8.7256 (8.5667 - 8.8872)
39	20.3602 (20.0849 - 20.6383)	11.6179 (11.4447 - 11.7935)	8.5449 (8.3852 - 8.7074)	8.6800 (8.5422 - 8.8197)
40	19.6540 (19.4078 - 19.9025)	11.9230 (11.7371 - 12.1115)	8.5379 (8.3751 - 8.7036)	8.6420 (8.4862 - 8.8003)
41	19.3700 (19.1472 - 19.5948)	12.1000 (11.9263 - 12.2758)	8.4884 (8.3224 - 8.6574)	8.5831 (8.4248 - 8.7441)
42	19.3436 (19.1163 - 19.5729)	12.0510 (11.8681 - 12.2363)	8.5311 (8.3838 - 8.6809)	8.5379 (8.3575 - 8.7218)
43	19.1173 (18.8432 - 19.3944)	12.2989 (12.1241 - 12.4759)	8.4787 (8.3274 - 8.6324)	8.5378 (8.3636 - 8.7154)
44	19.2561 (19.0157 - 19.4988)	12.4416 (12.2559 - 12.6297)	8.4565 (8.2798 - 8.6366)	8.5363 (8.3637 - 8.7122)
45	19.4113 (19.1590 - 19.6661)	15.3026 (15.1030 - 15.5045)	8.4708 (8.3195 - 8.6245)	8.6002 (8.4445 - 8.7584)
46	19.5462 (19.2847 - 19.8104)	14.7702 (14.5788 - 14.9637)	8.4011 (8.2376 - 8.5676)	8.5204 (8.3698 - 8.6735)
47	19.7124 (19.4611 - 19.9661)	15.4436 (15.2573 - 15.6318)	8.3757 (8.2086 - 8.5459)	8.4992 (8.3389 - 8.6624)
48	19.1561 (18.8996 - 19.4153)	15.1650 (14.9473 - 15.3853)	8.3843 (8.2286 - 8.5428)	8.5041 (8.3460 - 8.6649)
49	19.1460 (18.9355 - 19.3583)	14.7016 (14.5188 - 14.8864)	8.3972 (8.2181 - 8.5798)	8.5025 (8.3475 - 8.6602)
50	18.8418 (18.6021 - 19.0838)	14.4484 (14.2622 - 14.6365)	8.4361 (8.2835 - 8.5912)	8.5407 (8.4029 - 8.6805)
51	18.6107 (18.3788 - 18.8448)	13.9830 (13.7752 - 14.1934)	8.4240 (8.2471 - 8.6044)	8.5548 (8.3907 - 8.7218)
52	18.3174 (18.0897 - 18.5473)	13.6326 (13.4458 - 13.8216)	8.4151 (8.2606 - 8.5723)	8.5252 (8.3746 - 8.6782)
53	18.0812 (17.8227 - 18.3426)	14.3420 (14.1387 - 14.5478)	8.3742 (8.2005 - 8.5512)	8.4876 (8.3187 - 8.6596)
54	18.0506 (17.8141 - 18.2896)	13.9778 (13.7734 - 14.1849)	7.8222 (7.6747 - 7.9722)	8.0906 (7.9295 - 8.2546)
55	18.0845 (17.8681 - 18.3030)	14.4973 (14.3010 - 14.6959)	7.8111 (7.6563 - 7.9687)	8.0332 (7.8795 - 8.1896)
56	17.8244 (17.5746 - 18.0769)	14.8713 (14.6357 - 15.1100)	7.8234 (7.6874 - 7.9617)	7.9974 (7.8561 - 8.1409)
57	17.7112 (17.4693 - 17.9558)	15.4237 (15.2234 - 15.6261)	7.7949 (7.6443 - 7.9483)	7.9890 (7.8445 - 8.1359)
58	17.7493 (17.5283 - 17.9725)	15.7178 (15.5234 - 15.9142)	7.8055 (7.6502 - 7.9636)	7.9846 (7.8513 - 8.1200)
59	18.4095 (18.1619 - 18.6598)	15.2546 (15.0439 - 15.4676)	7.8043 (7.6316 - 7.9806)	8.0498 (7.9050 - 8.1970)
60	18.5256 (18.3157 - 18.7374)	16.0073 (15.8010 - 16.2158)	8.0406 (7.8920 - 8.1917)	8.2750 (8.1179 - 8.4349)
61	18.4087 (18.1969 - 18.6225)	16.6575 (16.4286 - 16.8889)	8.1222 (7.9797 - 8.2669)	8.3732 (8.2131 - 8.5362)
62	18.2252 (17.9886 - 18.4642)	16.4409 (16.2217 - 16.6626)	8.1037 (7.9670 - 8.2426)	8.3533 (8.2097 - 8.4990)
63	18.1865 (17.9622 - 18.4129)	16.3809 (16.1604 - 16.6037)	8.0994 (7.9411 - 8.2606)	8.3426 (8.1893 - 8.4986)
64	18.6799 (18.4631 - 18.8986)	16.6399 (16.4403 - 16.8416)	8.2184 (8.0497 - 8.3902)	8.5171 (8.3507 - 8.6865)
65	18.7895 (18.5500 - 19.0314)	16.6973 (16.4885 - 16.9082)	8.2219 (8.0583 - 8.3886)	8.5243 (8.3802 - 8.6706)
66	18.7404 (18.5137 - 18.9693)	16.7957 (16.5537 - 17.0406)	8.2110 (8.0446 - 8.3804)	8.5613 (8.4038 - 8.7215)
67	18.8054 (18.5662 - 19.0470)	17.2307 (17.0011 - 17.4627)	8.1705 (8.0189 - 8.3247)	8.5289 (8.3587 - 8.7021)
68	18.6483 (18.4306 - 18.8680)	17.1127 (16.9131 - 17.3141)	8.1515 (7.9964 - 8.3093)	8.4975 (8.3337 - 8.6642)
69	18.8678 (18.6408 - 19.0970)	18.0683 (17.8214 - 18.3178)	8.1313 (7.9782 - 8.2871)	8.5540 (8.3840 - 8.7272)

Table B.2: (1-AUC)% with 95% confidence interval for OSBF-Lua, TextCat, SVMLight, and LibLinear, for each training set used in Experiment 2

Set No.	OSBF-Lua	TextCat	SVMLight	LibSVM
70	18.9976 (18.7554 - 19.2421)	17.8364 (17.5829 - 18.0928)	8.0760 (7.9210 - 8.2337)	8.2270 (8.0820 - 8.3744)
71	19.0293 (18.8021 - 19.2585)	17.9521 (17.7186 - 18.1880)	8.0501 (7.9047 - 8.1980)	8.2945 (8.1492 - 8.4422)
72	17.6959 (17.4879 - 17.9058)	18.0100 (17.7922 - 18.2300)	7.9448 (7.8006 - 8.0914)	8.1700 (8.0243 - 8.3180)
73	17.5837 (17.3401 - 17.8299)	17.9959 (17.7993 - 18.1943)	7.8865 (7.7386 - 8.0370)	8.1933 (8.0364 - 8.3530)
74	17.6815 (17.4566 - 17.9088)	18.2747 (18.0714 - 18.4797)	7.9801 (7.8533 - 8.1089)	8.3285 (8.1751 - 8.4845)
75	17.6644 (17.4543 - 17.8765)	18.9188 (18.6977 - 19.1420)	7.9340 (7.7808 - 8.0900)	8.3171 (8.1341 - 8.5037)
76	17.5802 (17.3446 - 17.8184)	18.7830 (18.5664 - 19.0016)	7.9362 (7.7801 - 8.0952)	8.3688 (8.2190 - 8.5210)
77	17.3075 (17.0930 - 17.5241)	18.6505 (18.4247 - 18.8784)	7.8998 (7.7478 - 8.0546)	8.2305 (8.0863 - 8.3769)
78	17.1971 (16.9860 - 17.4103)	18.5255 (18.3002 - 18.7529)	7.8922 (7.7386 - 8.0486)	8.2661 (8.1022 - 8.4330)
79	17.2382 (17.0207 - 17.4579)	18.0803 (17.8442 - 18.3188)	7.8770 (7.7008 - 8.0569)	8.2596 (8.0964 - 8.4259)
80	17.1411 (16.9155 - 17.3691)	18.6789 (18.4884 - 18.8709)	7.9317 (7.7613 - 8.1054)	8.3600 (8.1983 - 8.5246)
81	16.9252 (16.7167 - 17.1356)	18.7898 (18.5898 - 18.9915)	7.9417 (7.7686 - 8.1183)	8.6816 (8.5112 - 8.8551)
82	16.9704 (16.7772 - 17.1654)	18.9289 (18.6888 - 19.1714)	7.9274 (7.7836 - 8.0736)	8.6400 (8.4741 - 8.8089)
83	16.9228 (16.7239 - 17.1237)	18.6385 (18.4167 - 18.8624)	7.9455 (7.7917 - 8.1020)	8.6373 (8.4665 - 8.8113)
84	16.8100 (16.5930 - 17.0292)	18.5103 (18.2938 - 18.7288)	7.8707 (7.7263 - 8.0176)	8.5324 (8.3782 - 8.6891)
85	16.7441 (16.5142 - 16.9764)	18.3015 (18.0657 - 18.5398)	7.8637 (7.7032 - 8.0272)	8.5361 (8.3819 - 8.6929)
86	16.7337 (16.5129 - 16.9569)	18.4461 (18.2312 - 18.6629)	7.8527 (7.7126 - 7.9951)	8.5322 (8.3837 - 8.6830)
87	16.6785 (16.4526 - 16.9069)	18.3822 (18.1727 - 18.5935)	7.8346 (7.6710 - 8.0013)	8.5029 (8.3376 - 8.6712)
88	16.5644 (16.3490 - 16.7821)	18.6901 (18.4363 - 18.9465)	7.8357 (7.6755 - 7.9989)	8.4703 (8.3080 - 8.6355)
89	16.5057 (16.2747 - 16.7393)	18.9062 (18.6745 - 19.1402)	7.8056 (7.6511 - 7.9629)	8.4811 (8.3245 - 8.6404)
90	16.6015 (16.3783 - 16.8271)	18.5452 (18.3432 - 18.7489)	7.7989 (7.6546 - 7.9458)	8.4453 (8.2783 - 8.6154)
91	16.7372 (16.4821 - 16.9954)	18.0453 (17.8156 - 18.2773)	7.8076 (7.6527 - 7.9655)	8.4510 (8.2649 - 8.6409)
92	16.6842 (16.4712 - 16.8994)	18.0302 (17.8260 - 18.2362)	7.8102 (7.6575 - 7.9656)	8.4619 (8.3129 - 8.6133)
93	16.5491 (16.3430 - 16.7572)	17.8882 (17.6799 - 18.0984)	7.8318 (7.6782 - 7.9883)	8.7068 (8.5611 - 8.8549)
94	16.5008 (16.3015 - 16.7020)	18.0323 (17.8156 - 18.2511)	7.8408 (7.6793 - 8.0054)	8.6622 (8.5072 - 8.8197)
95	16.4609 (16.2445 - 16.6797)	17.7544 (17.5218 - 17.9895)	7.8133 (7.6535 - 7.9761)	8.7041 (8.5503 - 8.8604)
96	16.4087 (16.1844 - 16.6355)	17.9728 (17.7154 - 18.2331)	7.8170 (7.6730 - 7.9634)	8.7011 (8.5286 - 8.8767)
97	16.3990 (16.1789 - 16.6215)	17.6989 (17.4964 - 17.9034)	7.8011 (7.6576 - 7.9471)	8.6935 (8.5150 - 8.8754)
98	16.7458 (16.5366 - 16.9572)	18.1668 (17.9490 - 18.3866)	7.8444 (7.6823 - 8.0096)	8.7290 (8.5735 - 8.8869)
99	16.6728 (16.4509 - 16.8972)	18.1643 (17.9599 - 18.3705)	7.8675 (7.7141 - 8.0237)	8.7107 (8.5466 - 8.8777)
100	16.7353 (16.5079 - 16.9652)	18.4746 (18.2602 - 18.6909)	7.8699 (7.6974 - 8.0458)	8.8022 (8.6332 - 8.9741)
101	16.1859 (15.9776 - 16.3964)	18.9859 (18.7704 - 19.2033)	8.0256 (7.8978 - 8.1553)	8.9969 (8.8395 - 9.1567)
102	16.0204 (15.7885 - 16.2550)	18.9157 (18.6877 - 19.1458)	7.5171 (7.3711 - 7.6657)	8.4623 (8.3119 - 8.6152)
103	15.1010 (14.8600 - 15.3453)	17.3749 (17.1568 - 17.5952)	7.3153 (7.1699 - 7.4634)	7.8683 (7.7216 - 8.0176)
104	14.9678 (14.7289 - 15.2098)	18.4292 (18.2210 - 18.6393)	7.0739 (6.9098 - 7.2416)	7.9274 (7.7894 - 8.0675)
105	14.6886 (14.4920 - 14.8874)	18.5945 (18.3590 - 18.8323)	7.1120 (6.9626 - 7.2644)	8.0561 (7.9025 - 8.2124)
106	12.3176 (12.1278 - 12.5099)	18.7998 (18.5713 - 19.0305)	6.2485 (6.1368 - 6.3621)	6.8648 (6.7178 - 7.0148)
107	11.7132 (11.5165 - 11.9128)	19.4373 (19.2063 - 19.6703)	6.0494 (5.9344 - 6.1664)	7.2949 (7.1514 - 7.4412)
108	11.4720 (11.2926 - 11.6538)	19.5435 (19.2856 - 19.8040)	5.8116 (5.6978 - 5.9275)	7.1401 (6.9910 - 7.2922)
109	10.6206 (10.4610 - 10.7824)	18.8646 (18.6478 - 19.0834)	5.5063 (5.3904 - 5.6245)	6.7089 (6.5776 - 6.8426)
110	10.3034 (10.1377 - 10.4714)	19.1628 (18.9389 - 19.3888)	5.1671 (5.0365 - 5.3010)	6.2778 (6.1407 - 6.4177)
111	10.3226 (10.1442 - 10.5037)	19.8281 (19.5970 - 20.0612)	5.0894 (4.9850 - 5.1958)	5.9526 (5.8277 - 6.0800)
112	10.0661 (9.8997 - 10.2350)	20.1457 (19.9132 - 20.3803)	4.8630 (4.7367 - 4.9925)	5.8125 (5.6784 - 5.9496)
113	9.8968 (9.7328 - 10.0632)	20.6465 (20.4002 - 20.8949)	4.8014 (4.6844 - 4.9212)	6.1657 (6.0306 - 6.3036)
114	9.7392 (9.5637 - 9.9176)	20.1669 (19.9283 - 20.4077)	4.6454 (4.5346 - 4.7588)	6.0504 (5.8938 - 6.2108)
115	9.7774 (9.5891 - 9.9690)	19.2918 (19.0698 - 19.5157)	4.5604 (4.4503 - 4.6730)	5.8047 (5.6835 - 5.9284)
116	9.6073 (9.4414 - 9.7758)	18.8765 (18.6412 - 19.1141)	4.4790 (4.3867 - 4.5732)	5.4816 (5.3433 - 5.6232)
117	9.4688 (9.3136 - 9.6262)	17.5450 (17.3234 - 17.7689)	4.4357 (4.3375 - 4.5360)	5.6818 (5.5447 - 5.8221)
118	9.3729 (9.2157 - 9.5325)	17.1189 (16.8710 - 17.3697)	4.4323 (4.3193 - 4.5481)	5.7492 (5.6179 - 5.8833)
119	9.4190 (9.2518 - 9.5890)	16.5080 (16.3240 - 16.6937)	4.3136 (4.1965 - 4.4340)	5.7822 (5.6580 - 5.9090)
120	9.1690 (8.9846 - 9.3569)	16.3869 (16.1712 - 16.6050)	4.2487 (4.1339 - 4.3665)	5.7669 (5.6331 - 5.9036)
121	9.3444 (9.1882 - 9.5030)	16.2704 (16.0713 - 16.4715)	4.2891 (4.1768 - 4.4043)	5.7512 (5.6068 - 5.8991)
122	9.2386 (9.0807 - 9.3989)	15.6847 (15.4535 - 15.9187)	4.2646 (4.1555 - 4.3764)	5.7556 (5.6268 - 5.8872)
123	9.1732 (9.0296 - 9.3189)	15.2336 (15.0107 - 15.4593)	4.2383 (4.1262 - 4.3533)	6.0147 (5.8732 - 6.1594)
124	9.0816 (8.9112 - 9.2550)	15.3025 (15.1067 - 15.5003)	4.1488 (4.0490 - 4.2509)	5.7881 (5.6537 - 5.9255)
125	9.0472 (8.8791 - 9.2181)	15.3485 (15.1341 - 15.5653)	4.0768 (3.9689 - 4.1875)	5.8525 (5.7301 - 5.9774)
126	8.9959 (8.8369 - 9.1574)	15.4386 (15.2447 - 15.6345)	4.0265 (3.9158 - 4.1401)	5.6308 (5.5143 - 5.7496)
127	9.0560 (8.8799 - 9.2352)	15.2858 (15.0734 - 15.5006)	3.9714 (3.8637 - 4.0820)	5.6408 (5.5274 - 5.7563)
128	9.2127 (9.0369 - 9.3915)	15.3003 (15.0952 - 15.5077)	3.9305 (3.8320 - 4.0315)	5.5768 (5.4431 - 5.7136)
129	9.1687 (8.9911 - 9.3495)	15.3903 (15.2049 - 15.5774)	3.9207 (3.8151 - 4.0292)	5.6679 (5.5532 - 5.7848)
130	9.0526 (8.8918 - 9.2160)	15.4763 (15.2678 - 15.6871)	3.8479 (3.7466 - 3.9519)	5.6465 (5.5203 - 5.7754)
131	8.9646 (8.8085 - 9.1232)	15.6028 (15.3743 - 15.8340)	3.7552 (3.6515 - 3.8618)	5.4181 (5.2808 - 5.5587)
132	8.9767 (8.8222 - 9.1336)	15.5388 (15.3118 - 15.7684)	3.7144 (3.6260 - 3.8049)	5.2789 (5.1502 - 5.4106)
133	8.9927 (8.8411 - 9.1467)	15.3001 (15.1120 - 15.4901)	3.7007 (3.5975 - 3.8068)	5.2137 (5.0894 - 5.3410)
134	8.9374 (8.7811 - 9.0961)	15.2175 (15.0049 - 15.4325)	3.6236 (3.5346 - 3.7148)	5.1393 (5.0202 - 5.2611)
135	8.9921 (8.8316 - 9.1553)	14.7448 (14.5130 - 14.9797)	3.5837 (3.4942 - 3.6754)	5.0065 (4.8938 - 5.1217)
136	8.9029 (8.7385 - 9.0701)	15.4215 (15.2209 - 15.6243)	3.5445 (3.4385 - 3.6538)	4.9944 (4.8718 - 5.1199)
137	8.9886 (8.8246 - 9.1554)	15.9655 (15.7808 - 16.1520)	3.5031 (3.4153 - 3.5930)	4.9135 (4.8075 - 5.0217)
138	9.0031 (8.8361 - 9.1730)	16.6376 (16.4118 - 16.8659)	3.4951 (3.3851 - 3.6086)	4.9329 (4.8040 - 5.0652)
139	9.0366 (8.9042 - 9.1708)	17.1881 (17.0081 - 17.3695)	3.4829 (3.4035 - 3.5640)	4.9516 (4.8270 - 5.0793)

Table B.2: (1-AUC)% with 95% confidence interval for OSBF-Lua, TextCat, SVMLight, and LibLinear, for each training set used in Experiment 2

Set No.	OSBF-Lua	TextCat	SVMLight	LibSVM
140	9.0021 (8.8380 - 9.1690)	17.2084 (16.9917 - 17.4272)	3.4535 (3.3423 - 3.5682)	4.9681 (4.8399 - 5.0994)
141	8.9502 (8.8028 - 9.0999)	17.5603 (17.3680 - 17.7543)	3.4330 (3.3268 - 3.5426)	4.8759 (4.7589 - 4.9957)
142	9.0036 (8.8619 - 9.1473)	17.7094 (17.4892 - 17.9319)	3.4235 (3.3261 - 3.5237)	4.8565 (4.7527 - 4.9625)
143	9.1319 (8.9745 - 9.2918)	17.4916 (17.3009 - 17.6839)	3.4127 (3.3150 - 3.5133)	4.8221 (4.7055 - 4.9415)
144	9.0657 (8.9301 - 9.2032)	17.4769 (17.2602 - 17.6956)	3.3726 (3.2717 - 3.4765)	4.8400 (4.7238 - 4.9589)
145	9.1066 (8.9449 - 9.2710)	17.5339 (17.3295 - 17.7401)	3.3457 (3.2596 - 3.4340)	4.8530 (4.7378 - 4.9708)
146	9.0767 (8.9011 - 9.2553)	17.6049 (17.3829 - 17.8291)	3.3310 (3.2343 - 3.4305)	4.8819 (4.7618 - 5.0048)
147	9.0869 (8.9393 - 9.2366)	17.6095 (17.4100 - 17.8107)	3.3337 (3.2422 - 3.4276)	4.8717 (4.7478 - 4.9987)
148	9.0950 (8.9454 - 9.2469)	17.7546 (17.5272 - 17.9843)	3.2967 (3.2101 - 3.3856)	4.8632 (4.7477 - 4.9813)
149	9.0893 (8.9221 - 9.2592)	17.8079 (17.6059 - 18.0116)	3.2676 (3.1685 - 3.3697)	4.7875 (4.6719 - 4.9058)
150	9.0406 (8.8878 - 9.1957)	17.5558 (17.3200 - 17.7942)	3.2226 (3.1323 - 3.3154)	4.7248 (4.6138 - 4.8384)
151	8.9934 (8.8406 - 9.1485)	17.6848 (17.4439 - 17.9283)	3.1842 (3.1011 - 3.2696)	4.6409 (4.5332 - 4.7510)
152	8.9623 (8.8026 - 9.1245)	17.6057 (17.3920 - 17.8214)	3.1383 (3.0361 - 3.2437)	4.5725 (4.4762 - 4.6706)
153	9.1052 (8.9709 - 9.2414)	17.4619 (17.2050 - 17.7219)	3.1157 (3.0325 - 3.2011)	4.5833 (4.4620 - 4.7077)
154	8.9986 (8.8547 - 9.1447)	17.4036 (17.1854 - 17.6240)	3.0632 (2.9812 - 3.1473)	4.4941 (4.3832 - 4.6077)
155	8.9523 (8.7978 - 9.1093)	17.3391 (17.1251 - 17.5553)	3.0322 (2.9396 - 3.1277)	4.4830 (4.3550 - 4.6146)
156	8.8714 (8.7062 - 9.0393)	17.2728 (17.0529 - 17.4950)	2.9987 (2.9155 - 3.0842)	4.4330 (4.3279 - 4.5404)
157	8.9965 (8.8439 - 9.1514)	17.0536 (16.8448 - 17.2645)	2.9815 (2.8938 - 3.0718)	4.3922 (4.2869 - 4.5000)
158	8.9280 (8.7696 - 9.0889)	17.0557 (16.8626 - 17.2506)	2.9756 (2.8961 - 3.0573)	4.3637 (4.2512 - 4.4791)
159	8.8979 (8.7309 - 9.0678)	16.9298 (16.7105 - 17.1514)	2.9315 (2.8419 - 3.0240)	4.3316 (4.2295 - 4.4361)
160	8.9128 (8.7485 - 9.0799)	16.9698 (16.7537 - 17.1882)	2.9142 (2.8312 - 2.9997)	4.3081 (4.2063 - 4.4122)
161	8.8359 (8.6832 - 8.9911)	16.6673 (16.4464 - 16.8904)	2.8536 (2.7686 - 2.9411)	4.2267 (4.1262 - 4.3296)
162	8.7477 (8.5888 - 8.9093)	16.5659 (16.3643 - 16.7694)	2.8312 (2.7437 - 2.9213)	4.1297 (4.0175 - 4.2450)
163	8.6688 (8.5125 - 8.8278)	16.3479 (16.1341 - 16.5639)	2.7784 (2.6877 - 2.8720)	4.0996 (3.9818 - 4.2208)
164	8.5961 (8.4492 - 8.7453)	16.1224 (15.8959 - 16.3514)	2.7576 (2.6761 - 2.8415)	4.0331 (3.9282 - 4.1406)

Table B.3: (1-AUC)% with 95% confidence interval for Bogofilter, DMC, GCLR, and LibLinear, for each training set used in Experiment 3

Set No.	Bogofilter	DMC	GCLR	LibLinear
1	69.1126 (67.2112 - 70.9515)	80.2947 (78.3481 - 82.1063)	76.0313 (74.0997 - 77.8619)	75.7396 (74.0651 - 77.3390)
2	41.5614 (39.3269 - 43.8312)	15.2764 (14.1438 - 16.4822)	6.0200 (5.1867 - 6.9772)	10.7543 (9.6297 - 11.9929)
3	11.9110 (10.6525 - 13.2961)	5.7476 (4.8395 - 6.8139)	3.6722 (2.9944 - 4.4963)	5.9885 (5.1023 - 7.0172)
4	14.4752 (13.0785 - 15.9936)	6.7720 (5.7688 - 7.9349)	3.8262 (3.1623 - 4.6229)	6.5002 (5.5805 - 7.5594)
5	16.1831 (14.5504 - 17.9604)	8.4090 (7.3093 - 9.6569)	4.3577 (3.6507 - 5.1942)	6.7098 (5.6159 - 7.9987)
6	16.8808 (15.4826 - 18.3778)	6.4931 (5.6126 - 7.5007)	3.6916 (3.0016 - 4.5329)	6.2400 (5.2208 - 7.4425)
7	14.1264 (12.6014 - 15.8027)	6.9383 (6.0155 - 7.9907)	4.2864 (3.5262 - 5.2016)	6.5161 (5.5052 - 7.6976)
8	13.8822 (12.4086 - 15.4998)	6.3940 (5.5855 - 7.3104)	4.5344 (3.8121 - 5.3858)	6.4781 (5.4728 - 7.6531)
9	13.9668 (12.7622 - 15.2653)	6.7596 (5.7352 - 7.9516)	4.7420 (3.9956 - 5.6197)	6.4298 (5.4883 - 7.5199)
10	10.0940 (8.9692 - 11.3423)	5.7908 (4.9464 - 6.7690)	4.7270 (3.9134 - 5.6998)	6.0288 (5.0814 - 7.1396)
11	7.3946 (6.4362 - 8.4828)	4.1350 (3.4877 - 4.8964)	4.6250 (3.8782 - 5.5073)	5.4954 (4.6319 - 6.5090)
12	7.4920 (6.4866 - 8.6390)	3.8247 (3.1476 - 4.6404)	4.4709 (3.6863 - 5.4130)	5.6395 (4.8034 - 6.6111)
13	7.2432 (6.3823 - 8.2099)	3.5114 (2.9334 - 4.1984)	4.2755 (3.4760 - 5.2489)	5.8348 (4.8616 - 6.9884)
14	7.5484 (6.4894 - 8.7640)	3.9060 (3.2551 - 4.6808)	4.6336 (3.8431 - 5.5773)	6.4891 (5.4585 - 7.6984)
15	6.4918 (5.5656 - 7.5597)	3.7386 (3.1389 - 4.4477)	4.1972 (3.5674 - 4.9326)	5.4950 (4.6227 - 6.5207)
16	5.2037 (4.4074 - 6.1348)	3.5242 (2.9298 - 4.2339)	3.4453 (2.8027 - 4.2288)	5.2542 (4.3223 - 6.3737)
17	9.7541 (8.4879 - 11.1861)	4.2212 (3.5780 - 4.9741)	3.6248 (3.0497 - 4.3035)	5.8450 (4.9983 - 6.8247)
18	7.6771 (6.7629 - 8.7033)	4.4158 (3.7768 - 5.1571)	3.7574 (3.0701 - 4.5914)	5.5823 (4.7654 - 6.5297)
19	7.2790 (6.3854 - 8.2867)	3.6045 (2.9750 - 4.3613)	3.3336 (2.6451 - 4.1938)	5.2028 (4.3794 - 6.1712)
20	7.1093 (6.2142 - 8.1221)	3.5360 (2.9224 - 4.2726)	3.2106 (2.5535 - 4.0299)	5.2108 (4.3606 - 6.2161)
21	4.7648 (4.0651 - 5.5778)	3.2826 (2.7514 - 3.9122)	2.7794 (2.2594 - 3.4148)	4.7408 (3.9513 - 5.6787)
22	4.5890 (3.8238 - 5.4985)	3.1868 (2.6506 - 3.8274)	2.8788 (2.2932 - 3.6084)	4.8238 (4.0369 - 5.7548)
23	4.4524 (3.7710 - 5.2503)	3.0922 (2.5441 - 3.7538)	3.1238 (2.5382 - 3.8392)	4.8305 (4.0310 - 5.7791)
24	4.1351 (3.5063 - 4.8710)	3.0124 (2.5237 - 3.5924)	3.2204 (2.6165 - 3.9580)	4.7302 (3.9540 - 5.6498)
25	3.9707 (3.3497 - 4.7013)	2.8249 (2.3706 - 3.3633)	3.2144 (2.6915 - 3.8348)	4.6058 (3.8812 - 5.4581)
26	3.7617 (3.1527 - 4.4830)	2.7729 (2.2826 - 3.3650)	3.4139 (2.8407 - 4.0979)	4.5177 (3.8057 - 5.3555)
27	3.6730 (3.0723 - 4.3859)	2.7211 (2.2061 - 3.3522)	3.5794 (2.9372 - 4.3557)	4.3528 (3.6869 - 5.1326)
28	3.4915 (2.9487 - 4.1299)	2.6449 (2.1632 - 3.2303)	3.3728 (2.8054 - 4.0502)	4.1943 (3.5491 - 4.9507)
29	3.5760 (3.0213 - 4.2283)	2.5655 (2.1046 - 3.1240)	3.5299 (2.9269 - 4.2515)	4.2018 (3.5425 - 4.9775)
30	3.6056 (3.0404 - 4.2712)	2.6146 (2.1485 - 3.1786)	3.4582 (2.9017 - 4.1168)	4.1249 (3.5111 - 4.8407)
31	3.5912 (3.0356 - 4.2440)	2.5689 (2.0907 - 3.1530)	3.4580 (2.8619 - 4.1731)	4.0963 (3.3878 - 4.9453)
32	3.7504 (3.1670 - 4.4365)	2.5000 (2.0292 - 3.0766)	3.3082 (2.7462 - 3.9804)	4.0090 (3.3941 - 4.7298)
33	3.7166 (3.1352 - 4.4009)	2.4670 (2.0347 - 2.9885)	3.4977 (2.8474 - 4.2899)	3.9923 (3.2995 - 4.8233)
34	3.7096 (3.2002 - 4.2966)	2.4797 (2.0201 - 3.0406)	3.5058 (2.9033 - 4.2279)	3.9420 (3.2643 - 4.7534)
35	3.7556 (3.2067 - 4.3943)	2.4770 (2.0446 - 2.9980)	3.5620 (3.0097 - 4.2112)	3.9170 (3.2808 - 4.6706)
36	3.7237 (3.1216 - 4.4367)	2.4476 (1.9432 - 3.0789)	3.6361 (3.0975 - 4.2641)	3.7887 (3.1563 - 4.5419)
37	3.6841 (3.0746 - 4.4091)	2.2911 (1.8668 - 2.8090)	3.5650 (3.0001 - 4.2316)	3.7379 (3.1313 - 4.4567)
38	3.5834 (2.9521 - 4.3436)	2.2244 (1.8005 - 2.7452)	3.6089 (3.0318 - 4.2909)	3.6222 (3.0250 - 4.3320)
39	3.7789 (3.1419 - 4.5390)	2.4262 (2.0043 - 2.9343)	3.6442 (3.0722 - 4.3179)	3.4970 (2.9464 - 4.1462)
40	3.6532 (3.0518 - 4.3678)	2.3740 (1.9803 - 2.8435)	3.6808 (3.1174 - 4.3415)	3.5330 (2.9633 - 4.2074)
41	3.5740 (2.9994 - 4.2540)	2.4268 (2.0218 - 2.9106)	3.6721 (3.0788 - 4.3747)	3.5178 (2.9597 - 4.1766)
42	3.5398 (2.9665 - 4.2191)	2.4553 (1.9915 - 3.0239)	3.7944 (3.2010 - 4.4926)	3.4318 (2.8284 - 4.1584)
43	3.3205 (2.8277 - 3.8959)	2.4379 (1.9830 - 2.9940)	3.8846 (3.2794 - 4.5962)	3.4775 (2.9082 - 4.1534)
44	3.2523 (2.7464 - 3.8476)	2.5268 (2.0538 - 3.1052)	4.0087 (3.4358 - 4.6724)	3.4899 (2.8536 - 4.2618)
45	3.2617 (2.7131 - 3.9168)	2.6421 (2.2345 - 3.1217)	4.1638 (3.5978 - 4.8145)	3.5017 (2.9134 - 4.2037)
46	3.2099 (2.6282 - 3.9151)	2.6067 (2.1640 - 3.1370)	4.1331 (3.5578 - 4.7968)	3.5604 (3.0341 - 4.1742)
47	3.1986 (2.6299 - 3.8854)	2.6452 (2.2333 - 3.1307)	4.1904 (3.6028 - 4.8689)	3.6008 (2.9615 - 4.3719)
48	2.8684 (2.4029 - 3.4210)	2.5466 (2.1063 - 3.0760)	4.2260 (3.6177 - 4.9314)	3.4258 (2.8907 - 4.0557)
49	2.8175 (2.3190 - 3.4194)	2.5556 (2.0747 - 3.1444)	4.2461 (3.5865 - 5.0207)	3.3384 (2.8275 - 3.9378)
50	2.7430 (2.2576 - 3.3292)	2.4852 (2.0651 - 2.9883)	4.3086 (3.6654 - 5.0587)	3.4132 (2.8785 - 4.0430)
51	2.6508 (2.1867 - 3.2101)	2.4214 (1.9900 - 2.9435)	4.4128 (3.8139 - 5.1007)	3.3511 (2.8194 - 3.9788)
52	2.8735 (2.3188 - 3.5560)	2.8236 (2.3497 - 3.3898)	4.5543 (3.9382 - 5.2615)	3.6760 (3.0653 - 4.4027)
53	2.8656 (2.4113 - 3.4026)	2.7774 (2.2333 - 3.4493)	4.5770 (3.9067 - 5.3558)	3.6360 (3.0294 - 4.3586)
54	2.9200 (2.4152 - 3.5264)	2.7885 (2.2743 - 3.4149)	4.5963 (4.0044 - 5.2708)	3.7132 (3.0972 - 4.4461)
55	2.9219 (2.4141 - 3.5325)	2.7807 (2.3123 - 3.3407)	4.3418 (3.7074 - 5.0790)	3.7371 (3.0583 - 4.5595)
56	2.8626 (2.3587 - 3.4702)	2.6650 (2.1925 - 3.2359)	4.3748 (3.7751 - 5.0646)	3.7000 (3.1159 - 4.3886)
57	2.8440 (2.3256 - 3.4738)	2.6444 (2.1490 - 3.2503)	4.3961 (3.6970 - 5.2204)	3.6566 (3.0924 - 4.3192)
58	2.7589 (2.2780 - 3.3380)	2.6285 (2.1208 - 3.2537)	4.4352 (3.7280 - 5.2691)	3.6267 (3.0699 - 4.2800)
59	2.9518 (2.4336 - 3.5764)	2.5458 (2.1275 - 3.0439)	4.4424 (3.8156 - 5.1667)	3.4883 (2.8995 - 4.1914)
60	2.9556 (2.5084 - 3.4796)	2.5024 (2.0581 - 3.0397)	4.5143 (3.8531 - 5.2827)	3.5102 (2.8983 - 4.2456)

Table B.4: (1-AUC)% with 95% confidence interval for OSBF-Lua, TextCat, SVMLight, and LibLinear, for each training set used in Experiment 3

Set No.	OSBF-Lua	TextCat	SVMLight	LibSVM
1	66.2867 (64.1263 - 68.3812)	89.4030 (88.0106 - 90.6509)	77.1992 (75.4493 - 78.8594)	77.7412 (75.8856 - 79.4925)
2	7.8137 (6.7936 - 8.9723)	8.7893 (7.6562 - 10.0718)	14.6524 (13.3471 - 16.0617)	14.8626 (13.5495 - 16.2789)
3	8.6055 (7.5116 - 9.8416)	6.1655 (5.2680 - 7.2042)	6.6907 (5.6839 - 7.8609)	7.1568 (6.2925 - 8.1295)
4	8.1925 (7.0295 - 9.5283)	5.2576 (4.4164 - 6.2484)	9.8661 (8.8364 - 11.0013)	9.8270 (8.8867 - 10.8550)
5	8.8092 (7.6189 - 10.1650)	6.5515 (5.6642 - 7.5665)	9.3523 (8.2652 - 10.5658)	8.7184 (7.5758 - 10.0148)
6	8.0019 (7.0109 - 9.1193)	5.9327 (5.0138 - 7.0076)	7.0326 (6.0529 - 8.1570)	7.4180 (6.4585 - 8.5070)
7	7.8168 (6.7169 - 9.0791)	5.5323 (4.7084 - 6.4906)	9.4896 (8.4360 - 10.6593)	8.9694 (7.8412 - 10.2419)
8	7.7589 (6.6895 - 8.9829)	6.4927 (5.4633 - 7.7001)	9.1398 (8.0931 - 10.3067)	8.4854 (7.5570 - 9.5162)
9	7.7491 (6.8153 - 8.7986)	6.8533 (5.9053 - 7.9406)	9.2276 (8.0747 - 10.5262)	8.4790 (7.4878 - 9.5878)
10	6.2420 (5.3737 - 7.2397)	7.0908 (6.0518 - 8.2923)	7.7763 (6.8893 - 8.7668)	7.5197 (6.5439 - 8.6277)
11	5.4475 (4.6680 - 6.3485)	6.8244 (5.7750 - 8.0482)	7.1602 (6.1553 - 8.3146)	6.0739 (5.1961 - 7.0889)
12	6.7595 (5.8279 - 7.8276)	6.4263 (5.4811 - 7.5215)	5.8646 (5.0099 - 6.8546)	6.0280 (5.1628 - 7.0276)
13	6.6444 (5.7100 - 7.7191)	6.0696 (5.0651 - 7.2582)	6.3852 (5.4869 - 7.4191)	6.8562 (5.9974 - 7.8278)
14	6.5120 (5.5967 - 7.5651)	6.1427 (5.0625 - 7.4354)	8.9453 (8.0129 - 9.9745)	8.7687 (7.7297 - 9.9324)
15	6.4267 (5.6008 - 7.3649)	6.1370 (5.2472 - 7.1663)	7.1592 (6.3504 - 8.0622)	6.0206 (5.2060 - 6.9532)
16	5.1995 (4.3705 - 6.1755)	5.9653 (5.1088 - 6.9549)	5.6391 (4.8759 - 6.5135)	5.7119 (5.0231 - 6.4887)
17	5.5961 (4.8019 - 6.5127)	6.2168 (5.2678 - 7.3236)	7.2007 (6.3546 - 8.1497)	7.2552 (6.4057 - 8.2074)
18	5.8735 (5.1086 - 6.7447)	6.7533 (5.8208 - 7.8228)	6.2679 (5.3966 - 7.2691)	6.5300 (5.6690 - 7.5113)
19	5.5813 (4.6987 - 6.6183)	6.8532 (5.8638 - 7.9953)	5.0938 (4.4495 - 5.8256)	5.2527 (4.5563 - 6.0488)
20	5.5752 (4.7858 - 6.4858)	6.9998 (5.9706 - 8.1910)	4.9542 (4.2397 - 5.7819)	5.1408 (4.4630 - 5.9151)
21	5.2293 (4.4836 - 6.0911)	6.8112 (5.9219 - 7.8229)	4.2453 (3.6359 - 4.9517)	4.3718 (3.6924 - 5.1694)
22	5.2684 (4.5109 - 6.1449)	8.6961 (7.5241 - 10.0309)	4.0460 (3.4212 - 4.7792)	4.2942 (3.6282 - 5.0761)
23	5.2343 (4.4090 - 6.2039)	8.7107 (7.4973 - 10.0990)	3.9344 (3.2760 - 4.7186)	4.2671 (3.7027 - 4.9132)
24	5.0140 (4.3366 - 5.7909)	8.6237 (7.5847 - 9.7901)	3.6345 (3.1168 - 4.2344)	3.9612 (3.2930 - 4.7582)
25	5.0180 (4.3130 - 5.8313)	8.5720 (7.4486 - 9.8468)	3.4717 (2.9070 - 4.1415)	3.7395 (3.0904 - 4.5186)
26	4.8833 (4.2401 - 5.6184)	8.8394 (7.5714 - 10.2961)	3.2242 (2.6903 - 3.8597)	3.5520 (2.9446 - 4.2792)
27	4.7935 (4.0640 - 5.6463)	9.0076 (7.8960 - 10.2584)	3.0884 (2.5657 - 3.7135)	3.4672 (2.9210 - 4.1112)
28	4.5836 (3.8759 - 5.4132)	8.9539 (7.8584 - 10.1852)	2.9424 (2.4652 - 3.5087)	3.3432 (2.7992 - 3.9884)
29	4.6084 (3.9843 - 5.3248)	8.8874 (7.7381 - 10.1885)	2.8946 (2.4217 - 3.4566)	3.3168 (2.7428 - 4.0058)
30	4.4003 (3.7902 - 5.1033)	8.8171 (7.6187 - 10.1833)	2.9015 (2.4529 - 3.4292)	3.3620 (2.8266 - 3.9945)
31	4.3549 (3.6179 - 5.2338)	8.7049 (7.6380 - 9.9048)	2.8936 (2.4386 - 3.4305)	3.3627 (2.7956 - 4.0401)
32	4.2384 (3.6463 - 4.9219)	8.6531 (7.4768 - 9.9945)	2.8515 (2.4174 - 3.3608)	3.3624 (2.8465 - 3.9680)
33	4.2615 (3.5930 - 5.0478)	8.6685 (7.5708 - 9.9084)	2.8388 (2.3875 - 3.3726)	3.3584 (2.8391 - 3.9686)
34	4.1505 (3.5609 - 4.8328)	8.5562 (7.2878 - 10.0215)	2.8332 (2.3667 - 3.3884)	3.3592 (2.8217 - 3.9949)
35	4.1508 (3.5590 - 4.8359)	8.5207 (7.3315 - 9.8822)	2.8216 (2.3231 - 3.4235)	3.3589 (2.8219 - 3.9939)
36	4.0608 (3.3826 - 4.8680)	8.1958 (7.0693 - 9.4834)	2.6853 (2.2572 - 3.1920)	3.1883 (2.7051 - 3.7545)
37	4.0105 (3.3919 - 4.7364)	8.2056 (7.1704 - 9.3752)	2.5793 (2.1328 - 3.1164)	3.1069 (2.5798 - 3.7375)
38	4.0032 (3.3446 - 4.7849)	8.4092 (7.3613 - 9.5909)	2.5324 (2.0976 - 3.0546)	3.0208 (2.5053 - 3.6384)
39	3.9109 (3.3549 - 4.5547)	8.3382 (7.2612 - 9.5585)	2.5704 (2.1002 - 3.1426)	3.0994 (2.6037 - 3.6858)
40	3.7484 (3.1574 - 4.4449)	7.8139 (6.7550 - 9.0227)	2.5086 (2.0635 - 3.0468)	3.0456 (2.5455 - 3.6401)
41	3.7519 (3.1689 - 4.4372)	7.5109 (6.4086 - 8.7851)	2.4930 (2.1083 - 2.9458)	3.0113 (2.4792 - 3.6533)
42	3.7075 (3.1753 - 4.3249)	7.1601 (6.1179 - 8.3641)	2.3808 (1.9693 - 2.8756)	3.0194 (2.5340 - 3.5944)
43	3.7254 (3.1765 - 4.3649)	7.1133 (6.2487 - 8.0872)	2.2858 (1.8686 - 2.7935)	2.9678 (2.4697 - 3.5626)
44	3.7176 (3.1436 - 4.3915)	7.1137 (6.2184 - 8.1267)	2.3347 (1.9103 - 2.8507)	3.0299 (2.5336 - 3.6197)
45	3.7508 (3.1585 - 4.4491)	7.6006 (6.5516 - 8.8018)	2.2887 (1.8629 - 2.8091)	3.0054 (2.5009 - 3.6078)
46	3.6339 (3.0096 - 4.3817)	7.9871 (6.8966 - 9.2330)	2.3022 (1.8824 - 2.8130)	3.0540 (2.5434 - 3.6634)
47	3.6395 (3.0609 - 4.3225)	7.7579 (6.7671 - 8.8798)	2.3534 (1.9043 - 2.9053)	3.0986 (2.6448 - 3.6274)
48	3.4784 (2.9174 - 4.1428)	7.9098 (6.9721 - 8.9615)	2.2870 (1.8438 - 2.8336)	2.9320 (2.4007 - 3.5766)
49	3.4803 (2.9223 - 4.1403)	7.2041 (6.2782 - 8.2546)	2.2132 (1.8257 - 2.6806)	2.8084 (2.3538 - 3.3477)
50	3.4251 (2.8536 - 4.1062)	7.2993 (6.2724 - 8.4790)	2.2438 (1.7843 - 2.8182)	2.8209 (2.3404 - 3.3967)
51	3.3178 (2.8067 - 3.9181)	8.1095 (6.9869 - 9.3943)	2.0972 (1.6910 - 2.5985)	2.6394 (2.1936 - 3.1729)
52	3.4316 (2.8771 - 4.0885)	8.1292 (7.0266 - 9.3873)	2.3286 (1.8945 - 2.8593)	2.8432 (2.3500 - 3.4364)
53	3.4026 (2.8943 - 3.9966)	8.1260 (7.0069 - 9.4059)	2.2721 (1.8746 - 2.7517)	2.7949 (2.3371 - 3.3393)
54	3.3688 (2.8591 - 3.9655)	8.0762 (7.0456 - 9.2426)	2.3108 (1.9001 - 2.8079)	2.8576 (2.3686 - 3.4440)
55	3.3694 (2.8592 - 3.9669)	8.0765 (7.0185 - 9.2782)	2.3079 (1.8889 - 2.8172)	2.8547 (2.3712 - 3.4334)
56	3.3597 (2.8047 - 4.0200)	8.1344 (6.9999 - 9.4342)	2.2944 (1.8607 - 2.8264)	2.8251 (2.3701 - 3.3645)
57	3.3741 (2.8333 - 4.0140)	8.1527 (6.9775 - 9.5056)	2.2594 (1.8855 - 2.7053)	2.8055 (2.3148 - 3.3965)
58	3.3859 (2.8309 - 4.0452)	8.1730 (7.0958 - 9.3973)	2.2341 (1.7888 - 2.7871)	2.7745 (2.2914 - 3.3561)
59	3.3132 (2.7713 - 3.9568)	8.1371 (7.0754 - 9.3421)	2.1707 (1.7274 - 2.7246)	2.6396 (2.1805 - 3.1922)
60	3.3144 (2.7537 - 3.9846)	8.1630 (7.1652 - 9.2859)	2.1109 (1.7294 - 2.5744)	2.6228 (2.1676 - 3.1705)

References

[1] Satyen Abrol and Latifur Khan. Twinner: understanding news queries with geo-content using twitter. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*, GIR '10, pages 10:1–10:8, 2010.

[2] Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20:357–389, October 2002.

[3] Isabel Anger and Christian Kittl. Measuring influence on twitter. In *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*, i-KNOW '11, pages 31:1–31:4, New York, NY, USA, 2011. ACM.

[4] Fidelis Assis, William Yerazunis, Christian Siefkes, and Shalendra Chhabra. Exponential differential document count a feature selection factor for improving bayesian filters accuracy, 2006.

[5] Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 36–44, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[6] Hila Becker, Mor Naaman, and Luis Gravano. Learning similarity metrics for event identification in social media. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 291–300, 2010.

[7] Adam Bermingham and Alan F. Smeaton. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1833–1836, 2010.

[8] Bodo Billerbeck and Justin Zobel. Questioning query expansion: an examination of behaviour and parameters. In *Proceedings of the 15th Australasian database conference - Volume 27*, ADC '04, pages 69–76, 2004.

[9] Andrej Bratko and Gordon V. Cormack. Batch and on-line spam filter evaluation. In *Proceedings of CEAS 2006 - Third Conference on Email and Anti-Spam*, Mountain View, July 2006.

[10] Andrej Bratko, Gordon V. Cormack, Bogdan Filipic, Thomas R. Lynam, and Blaz Zupan. Spam filtering using statistical data compression models. *Journal of Machine Learning Research*, 6:2673–2698, 2006.

[11] Stefan B́uttcher, Charles L. A. Clarke, and Gordon V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines*. MIT Press, 2010.

[12] Bob Carpenter and Breck Baldwin. *Natural Language Processing with LingPipe 4 Draft 0.5*. June 2011.

[13] Claudio Carpineto, Renato de Mori, Giovanni Romano, and Brigitte Bigi. An information-theoretic approach to automatic query expansion. *ACM Trans. Inf. Syst.*, 19:1–27, January 2001.

[14] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pages 4:1–4:10, 2010.

[15] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[16] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, and Krishna P. Gummadi. Measuring User Influence in Twitter: The Million Follower Fallacy. In *In Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM)*, Washington DC, USA, May 2010.

[17] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[18] Hsia-Ching Chang. A new perspective on twitter hashtag use: diffusion of innovation theory. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem - Volume 47*, ASIS&T '10, pages 85:1–85:4, Silver Springs, MD, USA, 2010. American Society for Information Science.

[19] Hsia-Ching Chang. A new perspective on twitter hashtag use: diffusion of innovation theory. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem - Volume 47*, ASIS&T '10, pages 85:1–85:4, 2010.

[20] Charles L. A. Clarke, Gordon V. Cormack, and Elizabeth A. Tudhope. Relevance ranking for one to three term queries, 2000.

[21] G. V. Cormack and R. N. S. Horspool. Data compression using dynamic markov modelling. *The Computer Journal*, 30:541–550, November 1987.

[22] Gordon V. Cormack. Trec 2006 spam track overview. In *Proceedings of the 15th Text REtrieval Conference*, Gaithersburg, Maryland, 2006.

[23] Gordon V. Cormack. Trec 2007 spam track overview. In *Proceedings of the 16th Text REtrieval Conference*, Gaithersburg, Maryland, 2007.

[24] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '09, pages 758–759, 2009.

[25] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 241–249, 2010.

[26] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 107–116, 2010.

[27] Miles Efron. Hashtag retrieval in a microblogging environment. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 787–788, 2010.

[28] Miles Efron and Gene Golovchinsky. Estimation methods for ranking recent information. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, SIGIR '11, pages 495–504, New York, NY, USA, 2011. ACM.

[29] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422, 2006.

[30] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, 2008.

[31] Ao Feng and James Allan. Finding and linking incidents in news. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 821–830, 2007.

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

[33] Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 661–670, 2009.

[34] Manish Gupta, Rui Li, Zhijun Yin, and Jiawei Han. Survey on social tagging techniques. *SIGKDD Explor. Newsl.*, 12:58–72, November 2010.

[35] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA, 2002. ACM.

[36] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 517–526, 2002.

[37] Liangjie Hong, Ovidiu Dan, and Brian D. Davison. Predicting popular messages in twitter. In *Proceedings of the 20th international conference companion on World wide web*, WWW '11, pages 57–58, New York, NY, USA, 2011. ACM.

[38] Jeff Huang, Katherine M. Thornton, and Efthimis N. Efthimiadis. Conversational tagging in twitter. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, HT '10, pages 173–178, 2010.

[39] Bernard J. Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. Twitter power: Tweets as electronic word of mouth. *J. Am. Soc. Inf. Sci. Technol.*, 60:2169–2188, 2009.

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

[41] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers, 2002.

[42] K. Sparck Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments. In *Information Processing and Management*, pages 779–840, 2000.

[43] Jon Kleinberg. Bursty and hierarchical structure in streams. *Data Min. Knowl. Discov.*, 7:373–397, October 2003.

[44] Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, 31(4es), December 1999.

[45] Giridhar Kumaran and James Allan. Using names and topics for new event detection. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 121–128, 2005.

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

[47] Evan Leavitt, A. with Burchard, D. Fisher, and S. Gilbert. The influentials: New approaches for analyzing influence on twitter. a publication of the web ecology project. `http://www.webecologyproject.org/wp-content/uploads/2009/influence-report-final.pdf`, Sept. 2009.

[48] Evan Leavitt, A. with Burchard, D. Fisher, and S. Gilbert. The influentials: New approaches for analyzing influence on twitter. a publication of the web ecology project. http://www.webecologyproject.org/wp-content/uploads/2009/influence-report-final.pdf, Sept. 2009.

[49] Changhyun Lee, Haewoon Kwak, Hosung Park, and Sue Moon. Finding influentials based on the temporal order of information adoption in twitter. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1137–1138, 2010.

[50] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, April 2001.

[51] Kamran Massoudi, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 362–367, Berlin, Heidelberg, 2011. Springer-Verlag.

[52] Michael Mathioudakis, Nick Koudas, and Peter Marbach. Early online identification of attention gathering items in social media. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 301–310, 2010.

[53] Rinkesh Nagmoti, Ankur Teredesai, and Martine De Cock. Ranking approaches for microblog search. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '10, pages 153–157, Washington, DC, USA, 2010. IEEE Computer Society.

[54] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. Searching microblogs: coping with sparsity and document quality. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 183–188, New York, NY, USA, 2011. ACM.

[55] Nasir Naveed, Thomas Gottron, Jérôme Kunegis, and Arifah Che Alhadi. Searching microblogs: coping with sparsity and document quality. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 183–188, New York, NY, USA, 2011. ACM.

[56] Kyosuke Nishida, Ryohei Banno, Ko Fujimura, and Takahide Hoshide. Tweet classification by data compression. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, DETECT '11, pages 29–34, New York, NY, USA, 2011. ACM.

[57] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999.

[58] Saša Petrović, Miles Osborne, and Victor Lavrenko. Streaming first story detection with application to twitter. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 181–189, 2010.

[59] Manolis Platakis, Dimitrios Kotsakos, and Dimitrios Gunopulos. Searching for events in the blogosphere. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 1225–1226, 2009.

[60] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '98, pages 275–281, 1998.

[61] Liza Potts, Joyce Seitzinger, Dave Jones, and Angela Harrison. Tweeting disaster: hashtag constructions and collisions. In *Proceedings of the 29th ACM international conference on Design of communication*, SIGDOC '11, pages 235–240, New York, NY, USA, 2011. ACM.

[62] S. E. Robertson. On term selection for query expansion. *Journal of Documentation*, 46:359–364, January 1990.

[63] Gary Robinson. A statistical approach to the spam problem. *Linux Journal*, 2003(107), March 2003.

[64] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, 2010.

[65] Takeshi Sakaki, Fujio Toriumi, and Yutaka Matsuo. Tweet trend analysis in an emergency situation. In *Proceedings of the Special Workshop on Internet and Disasters*, SWID '11, pages 3:1–3:8, New York, NY, USA, 2011. ACM.

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

[67] Christian Siefkes, Fidelis Assis, Shalendra Chhabra, William S. Yerazunis, and Empresa Brasileira De Telecomunicaesembratel. Combining winnow and orthogonal sparse bigrams for incremental spam filtering. In *In Proceedings of ECML/PKDD 2004, LNCS*, pages 410–421, 2004.

[68] Daniel Sousa, Luís Sarmento, and Eduarda Mendes Rodrigues. Characterization of the twitter @replies network: are user ties social or topical? In *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, SMUC '10, pages 63–70, 2010.

[69] Fabian M. Suchanek, Milan Vojnovic, and Dinan Gunawardena. Social tags: meaning and suggestions. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 223–232, 2008.

[70] Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. #twittersearch: a comparison of microblog search and web search. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 35–44, 2011.

[71] Ibrahim Uysal and W. Bruce Croft. User oriented tweet ranking: a filtering approach to microblogs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 2261–2264, New York, NY, USA, 2011. ACM.

[72] Michael J. Welch, Uri Schonfeld, Dan He, and Junghoo Cho. Topical semantics of twitter links. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 327–336, New York, NY, USA, 2011. ACM.

[73] Jianshu Weng, Ee-Peng Lim, Qi He, and C.W.-K. Leung. What do people want in microblogs? measuring interestingness of hashtags in twitter. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1121 –1126, Dec. 2010.

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

[75] Jui-Yu Weng, Cheng-Lun Yang, Bo-Nian Chen, Yen-Kai Wang, and Shou-De Lin. Imass: an intelligent microblog analysis and summarization system. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations*, HLT '11, pages 133–138, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[76] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. In *Mining Social Data (MSoDa) Workshop Proceedings*, pages 26–30. ECAI 2008, July 2008.

[77] Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *Proceedings of Collaborative Web Tagging Workshop at 15th International World Wide Web Conference*, 2006.

[78] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual*

international ACM SIGIR conference on Research and development in information retrieval, SIGIR '01, pages 334–342, 2001.

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