DETECTING SPAM IN MICROBLOGS

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Chandra Shekar

August, 2011
DETECTING SPAM IN MICROBLOGS

Chandra Shekar
Thesis

Approved:

Advisor
Dr. Kathy J. Liszka

Accepted:

Department Chair
Dr. Chien-Chung Chan

Committee Member
Dr. Yingcai Xiao

Dean of the College
Dr. Chand Midha

Committee Member
Dr. Chien–Chung Chan

Dean of the Graduate School
Dr. George R. Newkome

Date
ABSTRACT

Microblogging is one of the widely used modes of expression today. People like to upload their activities, photos, opinions on Facebook, Twitter, Identi.ca, Tumblr etc instantly. It is quick, simple and easily sharable with friends on social networks. However microblogs have also become favorites among spammers as they get a wide audience to target without using e-mail harvesting techniques. All they need to do is post it on the wall.

This research deals with the detection of pharmaceutical spam and general spam in Twitter and Identi.ca. We introduce a methodology for data preprocessing and a new methodology for classification of pharmaceutical related tweets. We generate comprehensive lists of spam words for both pharmaceutical and general spam as a base for spam classification. We compare the classification results against manually labeled tweets to estimate the effectiveness of the classifiers and the comparisons shown.
ACKNOWLEDGEMENTS

I wish to express my heartfelt appreciation and gratitude to Dr. Kathy J. Liszka, a wonderful advisor and teacher. Without her constant guidance, insightful suggestions and encouragement this work would not have been possible.

I express my appreciation and gratitude to Dr. Chan for guiding me with all the data mining experiments. I appreciate my committee member Dr. Xiao for his insightful corrections. I wish to thank Mr. Chuck van Tilburg, for extending his help in the research labs at all times and maintaining a workable environment there.

Last, but not the least, I would like to thank my family and friends for their constant encouragement, support and timely help during my study at the University of Akron.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>viii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
</tbody>
</table>

## CHAPTER

### I. INTRODUCTION

1

### II. MICROBLOGS AND DATA RETRIEVAL

3

#### 2.1 Twitter

3

##### 2.1.1 The Twitter REST API

4

#### 2.2 Identi.ca

5

##### 2.2.1 Identi.ca API

5

#### 2.3 Status.Net

6

#### 2.4 Data

6

##### 2.4.1 Existing Sources

6

##### 2.4.2 Twitter Download

7
2.5 Identi.ca Messages........................................................................................................7

III. RELATED RESEARCH........................................................................................................8

IV. CURRENT SPAMMING TECHNIQUES...............................................................................11

V. PREPROCESSING.............................................................................................................15

5.1 Nature of Microblog Messages.......................................................................................15

5.2 Methodology...................................................................................................................16

5.3 Preprocessing Methodology..........................................................................................16

5.4 Wordlist Creation.........................................................................................................18

VI. DATA MINING TECHNIQUES.........................................................................................22

6.1 Overview.........................................................................................................................22

6.2 Classification..................................................................................................................22

6.3 Clustering.......................................................................................................................23

6.4 Association Rules...........................................................................................................23

6.5 Decision Trees.................................................................................................................23

6.6 Weka................................................................................................................................24

6.6.1 J48...............................................................................................................................24

6.6.2 Random Tree and Random Forests.............................................................................26

VII. EXPERIMENTS AND RESULTS FOR PHARMA SPAM............................................27

7.1 Pharma Spam..................................................................................................................27

7.1.1 Decision Logic.............................................................................................................28
7.1.2 Training and Testing Data...............................................................28
7.1.3 Overview of Performance Measures..............................................29
7.1.4 Testing Manually Labeled Tweets..................................................31

VIII. EXPERIMENTS AND RESULTS FOR GENERAL SPAM.........................33

8.1 General Spam Overview........................................................................33
8.2 Methodology............................................................................................33

IX. CONCLUSIONS AND FUTURE WORK......................................................36

REFERENCES..................................................................................................38

APPENDICES..................................................................................................40

APPENDIX A. SCRIPT TO DOWNLOAD TWEETS........................................41
# LIST OF TABLES

3.1 Corpus.................................................................................................................................7

5.1 List of primary words for pharma spam..................................................................................20

5.2 List of secondary words for pharma spam..............................................................................20

5.3 Words which needs a URL to be present..............................................................................21

5.4 Words which don’t need a URL...........................................................................................21

7.1 Tweets2006 manually labeled test set results for each wordlist........................................32

7.2 Tweets2010 manually labeled test set results for each wordlist........................................32

7.3 Identica Messages manually labeled test set results for each wordlist...............................32

8.1 Tweets2006 as Test Set........................................................................................................35

8.2 Tweets 2010 as Test Set........................................................................................................35

8.3 Identica Messages as Test Set...............................................................................................35
LIST OF FIGURES

2.1 A Twitter Tweet........................................................................................................3
2.2 An Identi.ca status message......................................................................................6
4.1 Hashtag in tweet.......................................................................................................12
4.2 Spam using hashtag...............................................................................................12
4.3 Tiny URL..................................................................................................................13
5.1 Preprocessing Framework.......................................................................................18
6.1 Decision tree generated for weather dataset.........................................................25
6.2 Random tree for weather data.................................................................................26
CHAPTER I
INTRODUCTION

Micro blogs are convenient media to express feelings and present information and opinions to your friends which can be accessed via different devices. Microblog content ranges from a single sentence to few lines which make them attractive and easily updatable. Microblog contents include topics ranging from "What i'm doing right now" to topics such as Technology, short news, sports etc. Some of the popular microblogs which are also social networks are Facebook, Twitter, Identi.ca, tumblr, jaiku etc. The first microblogs were known as tumble blogs. The term was coined by "Why the lucky stiff" in a blog post on April 12, 2005. During 2006 and 2007, the term microblogs gained much notice for the service provided by Tumblr and Twitter. In May 2007, 111 microblogging sites were counted. Notable of them were Twitter, Tumblr, Plurk, Emote.in, PingGadget, Beeing, identi.ca etc

But as with the case of e-mails micro blogs are also increasingly becoming the target for spamming. Spamming refers to unsolicited messages, something which is not anticipated by a user. E-mail spam includes pharmaceutical spam, image spam, advertisements, offers etc. In e-mail spam most of the spam messages are generated by bots and the cost of receiving the message is borne by the receiver in form of bandwidth usage. Micro blogging is a faster way to send spam to a large group of people at once.
The only effort needed is to post it on the blog and then it is sent to all the friends or followers of a user. As with e-mail spam spammers are using microblogs to spread pornography, make easy money by fake lotteries, fake inheritances, counterfeit watches, free software, and illegal or ineffective pharmaceuticals. Phishing sites and malware threats are also getting very common threat in microblogs. Though it is has not reached the aggressive levels of spamming in email, it still has a potential for phishing, and malware infection and identity theft.

Pharmaceutical or pharma spam is a type of general spam which refers to the tweets related to sale of, advertisement of pharmaceuticals such as Viagra, Levitra, and Xanax. According to Symantec [1], pharmaceutical spam currently accounts for 65% of all traditional email spam sent. This statistic motivates the study of pharmaceutical spam in the context of microblogging. Illegal sale of prescription drugs or fake drugs has not yet taken over twitter but it still makes people naïve enough to believe that they can purchase their medications cheaper through these scammers.

General spam refers to advertisements related to any commercial product. We test a methodology to identify spam in two microblogging site called Twitter and Identi.ca. We try to identify pharmaceutical spam and general spam in tweets. We identify a comprehensive word set for pharmaceutical spam and general spam and then use these to check for existence of spam in tweets. We use weka data mining tool to classify spam from non-spam tweets. We test our classification results with a set of manually classified tweets to estimate the accuracy of the classification.
CHAPTER II
MICROBLOGS AND DATA RETRIEVAL

2.1 Twitter

Twitter is a real time information network which has over 175 million registered users and about 95 million tweets in a day [2]. It allows users to communicate by posting short messages of up to 140 characters called as tweets on their walls (each user has a virtual wall post where he can post and receive messages). User’s friends can reply to the post or re-post it on their wall called retweet in twitter terminology. Apart from these there are hashtags and trending topics which are discussed in some of the later sections. Figure 2.1 shows an example of a tweet

![Figure 2.1 A Twitter Tweet](image)

Twitter lets us download the data (tweets) using its APIs. In order to request data from Twitter we use a Uniform Resource Locator (URL) [3]. No authentication is required to download tweets but rate limiting limits the number of times we can request. It allows about 150 requests per hour to the twitter API.
We use cURL library to make http calls to twitter API and retrieve the data. cURL is a tool to transfer data to or from a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP etc). The command is designed to work without user interaction. After retrieving the data we write the tweet and tweet info to files for further processing.

2.1.1 The Twitter REST API

Representational State Transfer (REST) API is a architecture for distributed information retrieval systems such as world wide web [27]. The methods in REST API allow developers to access core Twitter data like update timelines, status data, and user information. We use public timeline of the REST API to download the tweets which returns 20 most recent tweets, including retweets from non-protected users. It has rate limiting of 150 requests per hour. Public timeline displays the most recent tweets or status updates from users across the world. It is cached for 60 seconds and requesting more frequent than this will not return any data. It uses a GET method to request tweets. The formats for downloading tweets include JSON, XML, RSS, and ATOM. We request the tweets to be in JSON format.

Some optional parameters for public timeline include-

- trim_user - When set to either true, t or 1, each tweet returned in a timeline will include a user object including only the status authors (person who is tweeting) numerical ID. Omitting the parameter retrieves the complete user object [3].

Example usage -

http://api.twitter.com/1/statuses/public_timeline.json?trim_user=true
• include_entities- When set to either true, t or 1, each tweet will include a node called "entities". This node offers metadata about the tweet in a discreet structure which includes user_mentions, urls, and hashtags [3].

Example usage-

http://api.twitter.com/1/statuses/public_timeline.json?include_entities=true

2.2 Identi.ca

Identi.ca [4] is an open source social networking and microblogging service. The service was started on 1-July-2008. Identi.ca is based on StatusNet [5] microblogging software package built on the open Microblogging specification. It allows users to communicate by sending text updates up to 140 characters long. It is similar to Twitter in both concept and operations. Identi.ca also provides XMPP support and personal tag clouds. It also allows users to freely export and exchange personal and friend data based on the Friend of friend specification (FOAF) standard [6].

2.2.1 Identi.ca API

The Identi.ca API is quite similar to twitter API and it is designed this way to make it easy for developers to download data with tools that exist to download twitter data and to communicate with other StatusNet sites.

• The API root for Identi.ca is at http://identi.ca/api.

• The API call used to collect the status messages from identi.ca is

http://identi.ca/api/statuses/public_timeline.json
Identi.ca doesn't enforce any rate-limiting on number of request made to collect the public status messages [7]. The features provided by identi.ca are few when compared to the features provided by twitter. There are around 32 features provided for each status message.

2.3 Status.Net

It is an open source microblogging platform which enables communities or groups to exchange short messages over the internet. It provides a similar service to sites like twitter, jaiku and plurk.

The message length is limited to a length of 140 characters. Status.net enables users to choose whom to "follow" and receive status messages from acquaintances [5]. Figure 1.2 shows a sample status message.

Figure 2.2 An Identi.ca status message

2.4 Data

2.4.1 Existing Sources

We downloaded an existing corpus from [8] which consists of 10.5 million tweets whose time period ranges from 2006-2009. The dataset has three files which contain username, time zone, location, status count, favorite count, followers and followings count and the tweets. We used only tweets file as we were only analyzing tweets.
2.4.2 Twitter Download

We downloaded another set of tweets from Twitter for a time period of four weeks in the months of November 2010-December 2010.

2.5 Identi.ca Tweets

We downloaded status messages from Identi.ca for a time period of four weeks in the month of January 2010.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>Number of tweets</th>
<th>Time period</th>
</tr>
</thead>
<tbody>
<tr>
<td>Munmun et al.</td>
<td>10.5 million</td>
<td>2006-2009</td>
</tr>
<tr>
<td>Twitter</td>
<td>0.5 million</td>
<td>November-December 2010</td>
</tr>
<tr>
<td>Indenti.ca</td>
<td>0.5 million</td>
<td>January-2010</td>
</tr>
</tbody>
</table>
CHAPTER III
RELATED RESEARCH

Research into microblogs for spam detection is still in its infancy and not much literature related to the subject exists. Some of the recent works in this area focus on identifying spammers rather than identifying tweets or messages with spam.

Machine learning techniques have been used by Benevenuto, et al., [9] to identify spammers. They identified two sets of attributes namely content attributes and user behavior attributes. The content attribute comprised of 39 attributes like maximum, minimum, average, and median of number of hashtags per number of words on each tweet, number of URLs per words, number of words of each tweet, number of characters of each tweet, number of URLs on each tweet, number of hashtags on each tweet etc.

User behavior attributes consisted of attributes like number of followers, number of followees, and other social interactions, such as age of the account, average tweets per day and so forth. They performed some preliminary experiments with SVMs to identify tweets which are spam based on spam words and the presence of a hashtag.

Yard et al., [10] tried to study a hashtag in Twitter from its inception till it is ended. The hashtag was about a robot body and started by a well known person in the field. Soon the hashtag became the trending topic and the spammers started making use of it. They studied a small collection of spammers and their behavior was. They tried to show when
and where spam occurs on Twitter and the techniques used by the spammers on the fly. They studied the number of tweets and number of followers. The structure of their small social network, how the followers are connected, was the main idea of the research. They claimed that spammers tweet and reply more often than legitimate users in their dataset. They also explained that well as use more hashtags so that tweet will show up in more searches easily that a regular tweet.

Wang [11] used a directed social graph model to explore friend and follower relationships among Twitter. Twitter’s spam policy that, if a person has a small number of followers compared to the amount of he is following the account is considered as spam was used to compute three features namely the number of friends, the number of followers, and the reputation of a user. Other features such as hyperlinks, replies and mentions and trending topics are also used as attributes. Several methods of classification like decision tree, neural network, support vector machines, and k-nearest neighbors are compared. They report that naïve Bayesian outperformed all other techniques.

In another study Moh and Murmann [12] use metrics to analyze a user’s friends and followers in addition to adding a finer-grained list of attributes to identify spammers.

Mowbray [13] discusses the abuse of automated agents using the Twitter API. They explored various mechanisms used by spammers on Twitter. Mention-spam is technique where the spammers try to reply to a tweet obtained from the public timeline API of Twitter. Another technique allows spammers to generate followees automatically in anticipation that in some cases the social link will be reciprocated. Some other techniques include fake retweets, trend abuse etc.
Blocky [14] is a website that provides a free service to block spammers in Twitter. Their methodology is a simple human voting system on whether a user is sending spam or not. Identified users are added to blacklist. This is a very effective technique in the small set of users, but impractical in a large network.

In the earlier work done on identification of spam tweets [15] we applied text mining techniques to preprocess Twitter data. We then used data-mining tools to generate classifiers for spam tweet detection. A simple method was applied for labeling a tweet as spam. Training tweets were based on 65 pharmaceutical related words selected manually. The results showed that the J48 decision tree classifier may be used as an effective tool for detecting pharmaceutical spam in tweets.
CHAPTER IV
CURRENT SPAMMING TECHNIQUES

4.1 E-mail Spam

Spam in e-mail is one of the most common forms of spam. Some of the common e-mail spamming techniques to include [16]

- Adding symbols in text. Example, free h*liday
- Adding spaces between words. Example, w e I g h t l o s s !
- Writing text vertically to avoid detection.
- Permuting words in a sentence such that humans can read them but filter cannot.
  Example, awesome soultion for hiar los

4.2 Social Network

Today, spammers use images, pdf documents and file attachments for spamming. Social networks being the fastest media to reach audiences are gaining popularity among spammers. Along with the general spamming techniques which include misspelled words, adding symbols in words, adding good words along with spam words etc, following are some techniques in twitter spamming.

- Hashtag- Hashtag is a ‘#’ symbol used before relevant keywords in tweet to categorize tweets related to the word to show up more easily in Twitter Search.
When hashtags become very popular they tend to become trending topics.

Figure 4.1 Hashtag in tweet [17]

Spammers try to attach their message or advertisement to one of the trending topics or hash tags. Figure 4.2 is an example of using such technique. The tweet has nothing to do with Iran elections

Figure 4.2 Spam using hashtag

- Mass following- Twitter lets us follow other people’s tweets and for others to follow us. Unlike in case of Facebook it is possible that someone may follow you even if you are not following them. Twitter, is not a complete bidirectional social network. Spammers try to target the network by creating fake profiles and starting to follow as many people as possible without being detected. They attempt to get users to follow them back and keep spam messages as their recent tweets and use provocative icons such that when the followed users check the followers they see the spam content.

- Tiny URLs- Including hyperlinks or URLs is common in tweets. Many legitimate tweets contain URLs as people like to share information. Hence if the URL is too long and has many characters in it due to the 140 characters limit it will not be possible to post URls. Several URL “shortening” services are available for free
online [18, 19]. Phishing and spam advertisements try to lure users to fake sites or sell fake products and try to take advantage of this service. By using an obfuscated URL it becomes impossible for a user to determine the authenticity of the site before clicking on it.

Figure 4.3 shows two tweets using bit.ly to obfuscate a URL. Both of these ads for Viagra are posted by the same user (Abriannella) and are luring people to two different web sites and the posts are made only six minutes apart.

![Tweets with Tiny URLs](image)

**Figure 4.3 Tiny URL**


Expands to

http://www.mahalo.com/buy-cheap-viagra/


Expands to

http://meds-store.org/

- **Tweetjacking** - Twitter lets you post tweets of other users by a technique called retweeting. In simple terms it means reusing other user’s tweets and it is done by inserting the tag:

  RT @username

This is a convenient way to share information without actually retyping it. The username indicates the original author of the tweet. Similar to this feature are replies and mentions. Placing @username at the beginning of a tweet followed by
a message is called a reply. Placing it anywhere else in a tweet is called a mention. Twitter collects these and sends them to the user so they know what is being discussed. Spammers take advantage of these to get attention without having to be followed. It also tends to give the spammer credibility, albeit misleadingly. In tweetjacking, spammers reply to or retweet the message replacing original hyperlinks in a message with hyperlinks that lead to porn sites, phishing, or malware traps.

- Tweets that are advertisements posted from accounts operated by bots
- Helpful tips on health, beauty, weight loss etc which are actually links to fraud websites.
- Self promotional posts which invite people to join communities, organizations and get free dining offers etc. However not all promotional messages or hyperlinks are spam. Many of these contain genuine information but sites like Twitter and Facebook are growing so fast, it is difficult to determine which message or post is a spam and which is not without actually following the link.
CHAPTER V
PREPROCESSING

5.1 Nature of Microblog Messages

Microblogging is about short length messages which make them quick and hence popular. Most of the times people just type abbreviated form of words, for example how becomes hw, are becomes r and fine becomes f9. This is also a common lingo in chats and e-mail but more prominent in microblogs due to their message size limits. This kind of data is however difficult to process for the following reasons-

- Words cannot be found in dictionary but can convey meaning to reader. Example, lol, rofl, tc, etc
- Use of emoticons to express feelings like I am happy can be said just a 😊, both mean the same but emoticons convey it in a shorter way
- Use of foreign language words- For example, the following tweets has English alphabets but is written in a different language.

  esquerdo olha Dizem que o dia da errado quando a gente levanta com o pÃ© Eu sÃ³ devo ter o pÃ© esquerdo o direito deve ter sumido porque nÃ©h n

- Words like the, have, then, is, was, being etc which are necessary for sentence formation but really do not convey any message as an individual words are called as stop words.
• Use of symbols and control characters makes data more noisy and difficult to process.

5.2 Methodology

We developed a methodology to clean and process tweets or status messages after trying various combination of methods to remove symbols, stopwords etc. The messages contain a lot of noise unlike content in a book or newspaper which is clean and well constructed. There are few existing techniques available that can be used for text processing, some of which are-

• Stemming is a process of reducing a word to its root form. For example, words read, reading, reader, read will all reduce to read, the verb form. Porters stemming algorithm is one such algorithm, Weka has stemmers too. We have not used stemming in our research as it would lose some words. For example, consider two words free and freedom. In our experiments word free might be related to spam as free coupons, free Viagra etc. But stemming would reduce the word freedom to free and this would an incorrect result.

• Stop word removal -Stop words are words like been, have, is, being, he, she, etc which are mainly prepositions and pronouns which really do not convey any meaning. There are many stop word lists available on the internet.

5.3 Preprocessing Methodology

• Identify and remove any emoticons from message file but keep them in a separate file. This is to ensure we do not lose any information.
Then identify any hyperlinks and replace them with a word URL. This reduces the length of message and makes it easier to interpret. We keep all extracted URL in a separate file.

We then tokenize each message i.e. split them into tokens instead of keeping them as sentences. The reason to tokenize is that it makes preprocessing easier as we are looking at words one at a time.

After tokenizing we then start identifying if there is any punctuation in tokens and clean it. Punctuation list we used includes the symbols {, }, [ , ] , , ”, !

Next we check if there are any control characters in the tokens. We eliminate these too.

Finally we combine all the clean tokens into one token file which is the base file for our work.

All of the above steps have been implemented using Java programs and SQL queries in Microsoft SQL server. Figure 5.1 describes the above process in detail. At each step we try to preserve as much meaningful information as possible.
One of the most important part of the experiment involved generating good set of words which would capture most of the spam (pharmaceutical and general). Generating a good enough wordlist is very challenging without knowing what words are actually being used in the tweets and messages. From observation, we decided to work with two sets of words that we label primary and secondary. We identified 86 primary words and 45

Figure 5.1 Preprocessing Framework

5.4 Wordlist Creation
secondary words. We hypothesize that while we can get good results working with just primary words, additional intelligence including the presence of one or more secondary words, and in particular with a URL, will help us differentiate true pharmaceutical spam from tweets that are only relaying a joke. For example:

   Non-pharma: I can mingle with the stars and have a party on mars i am a prisoner locked up behind xanax bars

   Pharma: Order Online Buy low cost Xanax (Alprazolam) medication You can buy Xanax online at very... Buy Alprazolam -&gt;http://bit.ly/grCsej

The words in the primary list are of more significance and the words in the secondary list are supplementary. If a tweet/message contains a primary word it is labeled as spam. Secondary words are necessary in case when primary words are not present but one or more secondary words are present.

In general spam we have two lists of words, one list has words which if present mark tweet as spam and other set which needs a presence of URL along with the word in the message to be marked as spam. Tables 6.1–6.2 shows the lists of words for pharma spam
In general spam we started with an initially constructed wordlist of 323 words. The list was generated by manual observation of tweets and from internet resources. Since this was a really large list of words to compute a frequency matrix, we used Microsoft
Decision Tree algorithms to select the most useful words. The number of words was reduced to 150. Tables 5.3-5.4 show the list of words for general spam.

Table 5.3 Words which needs a URL to be present

<table>
<thead>
<tr>
<th>business</th>
<th>fat</th>
<th>lose</th>
<th>rich</th>
<th>cheap</th>
<th>free</th>
<th>loss</th>
<th>registration</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheaper</td>
<td>gold</td>
<td>lower</td>
<td>shoes</td>
<td>cheapest</td>
<td>health</td>
<td>money</td>
<td>vacation</td>
</tr>
<tr>
<td>credit</td>
<td>holiday</td>
<td>offer</td>
<td>wedding</td>
<td>deal</td>
<td>interest</td>
<td>price</td>
<td>win</td>
</tr>
<tr>
<td>diet</td>
<td>joke</td>
<td>prices</td>
<td>won</td>
<td>earn</td>
<td>live</td>
<td>rates</td>
<td>url</td>
</tr>
</tbody>
</table>

Table 5.4 Words which don’t need a URL

<table>
<thead>
<tr>
<th>#jobs</th>
<th>coupon</th>
<th>harvest</th>
<th>millionaire</th>
<th>relationships</th>
<th>symptoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>course</td>
<td>herbal</td>
<td>mlm</td>
<td>remedies</td>
<td>tabs</td>
</tr>
<tr>
<td>advertising</td>
<td>dating</td>
<td>horoscope</td>
<td>mortgage</td>
<td>ringtones</td>
<td>tattoo</td>
</tr>
<tr>
<td>affiliate</td>
<td>dealers</td>
<td>income</td>
<td>online</td>
<td>romance</td>
<td>tips</td>
</tr>
<tr>
<td>aging</td>
<td>debt</td>
<td>ink</td>
<td>opportunity</td>
<td>sale</td>
<td>trading</td>
</tr>
<tr>
<td>amazing</td>
<td>degree</td>
<td>insurance</td>
<td>passwords</td>
<td>sales</td>
<td>training</td>
</tr>
<tr>
<td>approve</td>
<td>discount</td>
<td>inventory</td>
<td>pharma</td>
<td>sample</td>
<td>trial</td>
</tr>
<tr>
<td>arrest</td>
<td>discounts</td>
<td>invest</td>
<td>pharmacy</td>
<td>scam</td>
<td>unlimited</td>
</tr>
<tr>
<td>attractive</td>
<td>dollars</td>
<td>investment</td>
<td>pill</td>
<td>selling</td>
<td>urgent</td>
</tr>
<tr>
<td>babes</td>
<td>drug</td>
<td>jobs</td>
<td>pills</td>
<td>services</td>
<td>vaccine</td>
</tr>
<tr>
<td>bank</td>
<td>drugs</td>
<td>loan</td>
<td>poker</td>
<td>sex</td>
<td>visa</td>
</tr>
<tr>
<td>bankruptcy</td>
<td>dvd</td>
<td>loans</td>
<td>porn</td>
<td>signup</td>
<td>visit</td>
</tr>
<tr>
<td>biz</td>
<td>employment</td>
<td>marketers</td>
<td>prescription</td>
<td>singles</td>
<td>wallpapers</td>
</tr>
<tr>
<td>bonus</td>
<td>equity</td>
<td>marketing</td>
<td>product</td>
<td>smoking</td>
<td>watches</td>
</tr>
<tr>
<td>broadcast</td>
<td>freedom</td>
<td>med</td>
<td>profit</td>
<td>spam</td>
<td>wealth</td>
</tr>
<tr>
<td>cash</td>
<td>gambling</td>
<td>medical</td>
<td>profits</td>
<td>special</td>
<td>webcam</td>
</tr>
<tr>
<td>casino</td>
<td>getaway</td>
<td>medication</td>
<td>promo</td>
<td>stock</td>
<td>weight</td>
</tr>
<tr>
<td>claim</td>
<td>giveaway</td>
<td>medicine</td>
<td>purchase</td>
<td>subscribe</td>
<td>wrinkles</td>
</tr>
<tr>
<td>click</td>
<td>guarantee</td>
<td>meds</td>
<td>refund</td>
<td>subscribed</td>
<td></td>
</tr>
<tr>
<td>congratulations</td>
<td>guaranteed</td>
<td>merchant</td>
<td>relationship</td>
<td>subscription</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER VI
DATA MINING TECHNIQUES

6.1 Overview

Data mining is a process of extracting useful patterns, information from large data sets. Data mining is different from querying a database as we can only retrieve information from existing data by querying, but data mining can help detect patterns in data which is important for data warehousing. When we use Google search we are prompted with search strings which we might be looking for. This is an example of data mining technique. Another example is Amazon’s “frequently bought together” information on Amazon.com. It tells what people buy frequently together. Data mining activities are broadly classified into three different tasks namely, classification, clustering and association all of which are preceded by some preprocessing, transformation on the data.

6.2 Classification

The process of learning a function that classifies data into one of the predefined class labels or categories [20] is called as classification. In simpler terms a given dataset has attributes or features which describe the data. These are called as conditional attributes. Based on these attributes the instances in data are assigned a label of some predefined class. Classification is also called as supervised learning as the class labels are predefined and the algorithm works under supervision of these values. The success of classification learning can be estimated by providing an independent test data set for which actual class
labels are known but are not given to the classifier. The accuracy of the classifier tells how well the concept has been learnt.

6.3 Clustering

Clustering refers to a process of grouping similar instances into clusters based on a similarity or dissimilarity measure. It is an unsupervised learning as the class labels are not predefined and we do not use class labels to cluster instances, it is based on the similarity matrix.

6.4 Association rules

These are similar to classification except that they can be used to predict any attribute or combination of attributes. These rules are not meant to be used a set unlike the classification rules. For example, a rule {milk, sugar} -> {coffee} would mean that people who buy milk and sugar are most likely to buy coffee also.

6.5 Decision Trees

The output of the learning can be represented in different ways, decision trees is one of them. Decision trees are a divide and conquer approach to learning from independent instances and representing it in a form of decision tree. The nodes of the tree test a particular attribute value; the test involves comparing the value of the attribute with a constant. Some trees also compare attributes with each other. The leaf nodes of the tree are the classification labels. So, when an unknown instance is tested it is routed down till it reaches one of the leaf nodes by testing attribute values at each node.
In order to construct a decision tree a root node is selected first based on a criterion called information gain. It is associated to each node and in simple terms it represents the amount of information that is required to specify whether a new instance should be assigned a yes or no if it reaches the node. Information gain is measured in bits. An attribute with the highest information gain is chosen as the root of the tree. The daughter nodes are selected by selecting the attribute with next highest information gain and so on. Some examples of decision tree algorithms are ID3, J48, Random Tree, Random forest etc.

6.6 Weka

Weka is an open source data mining tool implemented in Java which we have used in this research. It is a collection of machine learning algorithms and preprocessing tools which can be accessed via command line or using the GUI.

6.6.1 J48

J48 is Weka’s implementation of the 8th revision of C4.5 algorithm. It is derived from Quinlan’s [] ID3 algorithm. The general process followed by J48 is as follows [21] -

Generate a decision tree based on the attributes of training data.

It identifies the attribute that discriminates the various instances most clearly from a training set i.e. has the highest information gain.

For all the possible values of this feature, if there is any value for which the data instances falling within its category have the same value for the target variable, we terminate that branch of the tree and assign to it the target value that was obtained.
For remaining cases, we look for another attribute with highest information gain. We repeat the process until we identify a combination of attributes that gives us a particular target value, or the attributes are exhausted in which case if we cannot get an unambiguous result from the available information, we assign this branch value possessed by most of the attributes in the branch.

Once the decision tree is generated, we follow the order of attribute selection we obtained for the tree. We check all the attributes and their values with those in the decision tree model and assign a target label to the new instance.

The figure 6.1 below shows an example of decision tree generated by J48 algorithm in weka for the weather dataset. The data has three conditional attributes of temperature, outlook, windy and a decision or class attribute play. The decision tree generated shows that outlook is the attribute with maximum information gain, next are humidity and windy. So, for a test sample first attribute to be tested is outlook.

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Humidity</th>
<th>windy</th>
<th>play</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>85</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>80</td>
<td>90</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>83</td>
<td>86</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>70</td>
<td>96</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>68</td>
<td>80</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>65</td>
<td>70</td>
<td>TRUE</td>
<td>no</td>
</tr>
<tr>
<td>64</td>
<td>65</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td>95</td>
<td>FALSE</td>
<td>no</td>
</tr>
<tr>
<td>69</td>
<td>70</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>75</td>
<td>80</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>75</td>
<td>70</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>72</td>
<td>90</td>
<td>TRUE</td>
<td>yes</td>
</tr>
<tr>
<td>81</td>
<td>75</td>
<td>FALSE</td>
<td>yes</td>
</tr>
<tr>
<td>71</td>
<td>91</td>
<td>TRUE</td>
<td>no</td>
</tr>
</tbody>
</table>

Figure 6.1 Decision tree generated for weather dataset.
6.6.2 Random Tree and Random Forests

Random tree or random forest in Weka is an implementation of random forest introduced by Leo Breiman. Random forest is an combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [22].

The algorithm uses an ensemble of classifiers. It trains each of the trees with a training set generated from the original training set by bootstrapping. The training set for each tree consists of same number of instances [23].

While constructing a tree it considers a subset of attributes to find the best split. Once the trees are generated it uses all of these trees to classify a new sample. The test sample is given to each of trees to provide it with a class label. The final class label for the test sample is decided on the maximum votes received by a class label. For example, if there are 10 trees and 7 of them classify an instance as ‘Yes’ and 3 classify as ‘No’, the final class label is ‘Yes’. None of the trees which are generated are pruned. Figure 6.2 shows the output of Random Tree for the weather data in figure 6.1.

![Random Tree Diagram](image)

**Figure 6.2 Random tree for weather data**
CHAPTER VII

EXPERIMENTS AND RESULTS FOR PHARMA SPAM

We had three datasets for the experiments, one from munmun et al., another one from twitter which we downloaded and one from identi.ca. Henceforth we shall call these as Tweets2006, Tweets2010 and IdenticaMessages respectively for convenience. For each of these datasets we performed two experiments namely identifying pharma spam and identifying general spam. In order to apply data mining tools to generate a classifier, we need to determine a list of features to represent tweets. The list of features is a list of words which we need to identify for pharma and general spam. We describe both of these in detail in the following sections.

7.1 Pharma Spam

It is important to generate a good list of words or attributes in order to achieve good results from classification. We started with a list of common terms related to this type of spam derived from several sources [24, 25, 26]. Then we added words from our email spam and finally, we manually analyzed over 5000 tweets, looking for the common terms used to sell or advertise prescription drugs.
7.1.1 Decision Logic

From manual observation, we decided to work with two sets of words which we label as primary and secondary. We identify 86 primary words including, \{Viagra, xanax, pharma, prescription, pill, medication\} and 45 secondary words including \{generic, refill, shipping, wholesale\}. This is the basis for our decision logic.

In order to classify the tweets we need to transform the data into a Term Frequency Document Matrix (TDM). It is a matrix that has all the words in the wordlist as columns and each row lists the frequencies of these words. Number of columns is equal to number of words in the wordlist and number of rows equals the number of tweets. Given a tweet if the words in a tweet are present in wordlist then the frequency of the word in the matrix for the given tweet is incremented. In our experiments we had total of 131 columns in the matrix and number of rows equal number of tweets in each dataset. These 131 words are the attributes which describe the data in this case tweets or messages.

7.1.2 Training and Testing Data

Social networks are becoming popular targets for spam but in real the number of spam to non-spam is fairly low. This is an example of imbalanced dataset and in order to generate an effective training set we need to create a balanced dataset so that the learning algorithm learns well. In order to create a balanced training set we use under-sampling of non spam tweets. The Tweets2006 dataset consists of 48,133 tweets from the collection of 10.5 millions, Tweets2010 dataset consists of 1,079 tweets from the 0.5 million collection and the IdenticaMessages consists of 2594 tweets out of 0.5 million. In order to test we had independent test set for each of the three dataset which we had labeled...
manually. Manual labels help us in testing how accurate the classification is in comparison to manual classification performed by domain-expert.

Once the training set is generated we split it into training and testing in a ratio of 80(training)-20(testing). At this point we are not using our manually labeled test set. We use the J48 and RandomTree algorithms in weka and generate decision trees. The decision tree gives us a list of attributes that help in making a decision in other words the list of words that matter when deciding whether a tweet is a pharma spam or not. We select these set of words and make two new lists of words one for J48 results and one for RandomTree. We call the wordlist from J48 as newJ48list and wordlist from RandomTree as newRTlist for ease of use.

We ran the J48 and RandomTree for each of Tweets2006, Tweets2010 and IdenticaMessages datasets and all of them generate separate wordlists. So totally we have six different wordlists, two for each dataset. We now use the manually labeled tweets to test the classifier results.

7.1.3 Overview of Performance Measures

We perform a binary classification as to whether a given tweets is spam or not. In case of a highly imbalanced data accuracy alone is not sufficient measure to estimate the performance of classifier. For example, if had only 1% of positive instances in data we can achieve an accuracy of 99% but simply classifying all instances as false. In such situations the following measure are more useful and informative for evaluating the performance of such binary classifiers.
- Recall = \( \frac{TP}{TP + FN} \)

In simple terms this means the number of correctly classified positive instances out of the total instances which are positive. For example, if we had 10 fruits of which 5 are apples then recall will be how many of the 5 apples are correctly classified over number of apples. Recall is the same as sensitivity.

- Precision = \( \frac{TP}{TP + FP} \)

From the above fruits example, it would mean ratio of how many correctly classified apples over total number of fruits classified as apples.

- Specificity = \( \frac{TN}{FP + TN} \)

This is the same as recall but here we are looking at correctly classified negative instances.

- F-measure = \( \frac{2 \times recall \times precision}{recall + precision} \)

F-measure combines precision and recall and gives a single performance measure.

- Sensitivity \* Specificity another single performance measure.

These measures are computed for all the classification tests in this work. In general we can say that if the above measures have higher values the classifier performs well and accuracy is higher. However a high accuracy but lower values for above measures may indicate a poor classifier performance.
7.1.4 Testing Manually Labeled Tweets

For each tweet in the independent set we perform four classifications-

- **J48** - We check for each tweet if it contains a word from the newJ48list, if it does we label it as pharma spam else non-spam. This means that if we used J48 to classify these tweets it would mark a tweet as pharma spam if it contains any word from newJ48list. We accomplished this using a simple Java program.

- **RT** – As in case of J48 above we check if a tweet contains a word from newRTlist we mark it as pharma spam else we mark it as non-spam.

- **DL(J48)** - We use the decision logic here. We check if a tweet has a word from newJ48list. We need to note that the newJ48list can contain words from both primary list (86 words) and secondary list (45 words). We check if a tweet has a word from newJ48list and then check if it is a primary or secondary word. We count the frequency of both primary and secondary words in a tweet. A tweet is flagged as a pharma spam tweet if it has one or more primary words or if it has one primary word and one secondary word or two or more secondary words.

- **DL(RT)** - The same decision logic is used as in DL(J48) except the wordlist being considered is newRTlist.

Table 7.1, 7.2 and 7.3 below are the results for each of the three independent testing set for Tweets2006, Tweets2010 and IdenticaMessages datasets.
<table>
<thead>
<tr>
<th>Measures</th>
<th>J48</th>
<th>RT</th>
<th>DL(J48) and DL(RT)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.9552</td>
<td>0.9566</td>
<td>0.9665</td>
</tr>
<tr>
<td>recall</td>
<td>0.4157</td>
<td>0.2191</td>
<td>0.1797</td>
</tr>
<tr>
<td>precision</td>
<td>0.3854</td>
<td>0.3391</td>
<td>0.6153</td>
</tr>
<tr>
<td>F-measure = 2<em>TP/(2</em>TP+FP+FN)</td>
<td>0.4</td>
<td>0.2662</td>
<td>0.2782</td>
</tr>
<tr>
<td>sensitivity = tp</td>
<td>0.4157</td>
<td>0.2191</td>
<td>0.1797</td>
</tr>
<tr>
<td>specificity = 1-fp</td>
<td>0.9753</td>
<td>0.984</td>
<td>0.9958</td>
</tr>
<tr>
<td>sensitivity*specificity</td>
<td>0.4306</td>
<td>0.2156</td>
<td>0.179</td>
</tr>
<tr>
<td># of nodes in resulting tree</td>
<td>69</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td># of primary spam words selected</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td># of secondary spam words selected</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.2 Tweets2010 manually labeled test set results for each wordlist

<table>
<thead>
<tr>
<th>Measures</th>
<th>J48</th>
<th>RT</th>
<th>DL(J48) and DL(RT)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.9953</td>
<td>0.9945</td>
<td>0.9971</td>
</tr>
<tr>
<td>recall</td>
<td>0.2307</td>
<td>0.1538</td>
<td>0.0769</td>
</tr>
<tr>
<td>precision</td>
<td>0.1875</td>
<td>0.1111</td>
<td>0.3333</td>
</tr>
<tr>
<td>F-measure = 2<em>TP/(2</em>TP+FP+FN)</td>
<td>0.2068</td>
<td>0.129</td>
<td>0.125</td>
</tr>
<tr>
<td>sensitivity = tp</td>
<td>0.2307</td>
<td>0.1538</td>
<td>0.0769</td>
</tr>
<tr>
<td>specificity = 1-fp</td>
<td>0.9973</td>
<td>0.9967</td>
<td>0.9995</td>
</tr>
<tr>
<td>sensitivity*specificity</td>
<td>0.2301</td>
<td>0.1533</td>
<td>0.0768</td>
</tr>
<tr>
<td># of nodes in resulting tree</td>
<td>69</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td># of primary spam words selected</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td># of secondary spam words selected</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7.3 IdenticaMessages manually labeled test set results for each wordlist

<table>
<thead>
<tr>
<th>Measures</th>
<th>J48</th>
<th>RT</th>
<th>DL(J48) and DL(RT)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
<td>D3</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.9964</td>
<td>0.9952</td>
<td>0.9966</td>
</tr>
<tr>
<td>recall</td>
<td>0.1818</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>precision</td>
<td>0.1818</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F-measure = 2<em>TP/(2</em>TP+FP+FN)</td>
<td>0.1818</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>sensitivity = tp</td>
<td>0.1818</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>specificity = 1-fp</td>
<td>0.9982</td>
<td>0.9974</td>
<td>0.9988</td>
</tr>
<tr>
<td>sensitivity*specificity</td>
<td>0.1815</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td># of nodes in resulting tree</td>
<td>69</td>
<td>19</td>
<td>11</td>
</tr>
<tr>
<td># of primary spam words selected</td>
<td>31</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td># of secondary spam words selected</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
CHAPTER VIII

EXPERIMENTS AND RESULTS FOR GENERAL SPAM

8.1 General Spam Overview

General spam refers to various categories of spam like advertisements, pornography, beauty tips, cosmetics, chain messages etc. We use the same three datasets and classify tweets as general spam or non-spam.

8.2 Methodology

The method is very similar to the one used in pharma spam in section 7.1 but with slight modifications. Firstly unlike pharma spam there is just one wordlist containing words related to general spam. We started with a list of common terms related to spam in general and then added words from spam e-mails in our mailboxes and through manual inspection of 5000 manually labeled tweets.

Initially we had 323 words in general spam wordlist which was quite large to generate a term frequency matrix. In order to reduce the number of words we chose frequency as a criterion and chose top 150 high frequency words which were used against each dataset. We divided the 150 word set into two categories, one which has words that need a presence of a URL along with them to label a tweet as spam and another which has words whose presence is enough to flag a tweet as spam.
We generated training set in similar way as pharma spam by under sampling of non-spam tweets. However we did not divide them into training and testing set to run through J48 and RandomTree to generate new wordlists. We just used the wordlist of 150 words to generate decision tree for J48 and RandomTree and classified the manually labeled tweets using the generated classifier.

Tweets2006 dataset contains 1.42 million spam tweets out of 10 million tweets according to the term document matrix. We used 50k spam and non spam tweets each from the dataset. The Tweets 2010 dataset had 30587 spam tweets in half million so we used 30587 spam and non spam tweets to create the training set. Identi.ca had 34669 spam messages in half million messages, so we chose about 34669 messages from spam and non spam each to create the training set. We used the manually labeled 5k tweets/messages as test set.

Finally we generated confusion matrix to check the results against the manual labels and compute accuracy of classification. The tables 8.1- 8.3 list the accuracy of classifier for each of the three datasets, Tweets2006, Tweets2010 and Identi.ca dataset.
Table 8.1 Tweets2006 as Test Set

<table>
<thead>
<tr>
<th>Measures</th>
<th>J48</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.7363</td>
<td>0.7389</td>
</tr>
<tr>
<td>recall</td>
<td>0.8439</td>
<td>0.7851</td>
</tr>
<tr>
<td>precision</td>
<td>0.7769</td>
<td>0.8138</td>
</tr>
<tr>
<td>F-measure = 2<em>TP/(2</em>TP+FP+FN)</td>
<td>0.8090</td>
<td>0.7992</td>
</tr>
<tr>
<td>sensitivity = tp</td>
<td>0.8439</td>
<td>0.7851</td>
</tr>
<tr>
<td>specificity = 1-fp</td>
<td>0.5257</td>
<td>0.6486</td>
</tr>
<tr>
<td>sensitivity*specificity</td>
<td>0.4436</td>
<td>0.5092</td>
</tr>
<tr>
<td># of nodes in resulting tree</td>
<td>315</td>
<td>297</td>
</tr>
</tbody>
</table>

Table 8.2 Tweets 2010 as Test Set

<table>
<thead>
<tr>
<th>Measures</th>
<th>J48</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.9252</td>
<td>0.9286</td>
</tr>
<tr>
<td>recall</td>
<td>0.4513</td>
<td>0.4457</td>
</tr>
<tr>
<td>precision</td>
<td>0.4807</td>
<td>0.5063</td>
</tr>
<tr>
<td>F-measure = 2<em>TP/(2</em>TP+FP+FN)</td>
<td>0.4655</td>
<td>0.4741</td>
</tr>
<tr>
<td>sensitivity = tp</td>
<td>0.4513</td>
<td>0.4457</td>
</tr>
<tr>
<td>specificity = 1-fp</td>
<td>0.9621</td>
<td>0.9662</td>
</tr>
<tr>
<td>sensitivity*specificity</td>
<td>0.4341</td>
<td>0.4306</td>
</tr>
<tr>
<td># of nodes in resulting tree</td>
<td>315</td>
<td>297</td>
</tr>
</tbody>
</table>

Table 8.3 IdenticaMessages as Test Set

<table>
<thead>
<tr>
<th>Measures</th>
<th>J48</th>
<th>RT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D1</td>
<td>D2</td>
</tr>
<tr>
<td>accuracy</td>
<td>0.9223</td>
<td>0.9265</td>
</tr>
<tr>
<td>recall</td>
<td>0.4242</td>
<td>0.4007</td>
</tr>
<tr>
<td>precision</td>
<td>0.3684</td>
<td>0.3876</td>
</tr>
<tr>
<td>F-measure = 2<em>TP/(2</em>TP+FP+FN)</td>
<td>0.3944</td>
<td>0.3940</td>
</tr>
<tr>
<td>sensitivity = tp</td>
<td>0.4242</td>
<td>0.4007</td>
</tr>
<tr>
<td>specificity = 1-fp</td>
<td>0.9539</td>
<td>0.9599</td>
</tr>
<tr>
<td>sensitivity*specificity</td>
<td>0.4047</td>
<td>0.3846</td>
</tr>
<tr>
<td># of nodes in resulting tree</td>
<td>315</td>
<td>297</td>
</tr>
</tbody>
</table>
CONCLUSIONS AND FUTURE WORK

We created a substantial corpus for the study from two micro blogging sites, Twitter and Identi.ca. We introduced a framework for preprocessing microblogs messages which can be applied to data from various microblogs. We have introduced new decision logic for identifying pharmaceutical related spam tweets using four different classifiers. The classifiers J48 and Random Tree (RT) are generated by Weka tools, and classifiers DL(J48) and DL(RT) are based on the combination of J48 and RT with the decision matrix. Though the data is highly imbalanced the results are very promising with accuracy above 95% over all the three datasets and a good recall rate.

In order to classify general spam we have generated a comprehensive list of 150 words which are strongly associated with spam. The results indicate that the Tweets2006 testset has a lower accuracy when compared to the other two testset. The recall and precision values are however high with Tweets2006 test set when compared to the other two testsets. The reason could be that since the logic is mostly based on string matching and that general spam is a very vast area wordlist alone is not sufficient. In future techniques like n-gram might give better and stable results.

The approach described for pharmaceutical spam gives better results as pharma spam contains very specific set of words and we add additional intelligence to the classification process apart from just the decision tree results.
We have used a manually labeled test set in classification so the efficiency is compared to manual labeling of tweet as spam or non-spam. The study has also provided us with some interesting observations about how newer spamming techniques are being employed by spammers to get to a larger audience. Hashtags, URLs, trending topics were actually created for users to use these microblogs services more efficiently but these are becoming popular mechanisms for spamming.
REFERENCES


[22] RANDOM FORESTS, Leo Breiman, Statistics Department, University of California, Berkeley, CA 94720 January 2001


APPENDIX A

SCRIPT TO DOWNLOAD TWEETS

/*
Program to download tweets and relevant information about the tweets from twitter using twitter API.
Program uses cURL library to make requests to twitter API and to retrieve data.
The information is written to different text files based on the type of information.
we need to set the variable "no_of_minutes" to number of calls to make.
Since twitter API has rate limiting(no of times a user can make calls to retrieve the tweets)
of 150 calls per hour,
this program makes a call and waits for 25 seconds before making the next call. so we make approx 150 calls per hour.
If the rate limit is exceeded, Twitter will block any further calls from respective IP address indefinitely.
*/

<?php

$endpoint = sprintf('http://api.twitter.com/1/statuses/public_timeline.json?include_entities=true');

$fp_tweetinfo = fopen("tweetinfo.txt",'w') or die("Can't open a file");
$fp_userinfo = fopen("userinfo.txt",'w') or die("Can't open a file");
$fp_entities = fopen("entitiesinfo.txt",'w') or die("Can't open a file");
$fp_tweet = fopen("tweet.txt",'w') or die("Can't open a file");

$no_of_minutes= 41520; // 10760; //720 for 12 hours , 1440 for one whole day
$id_counter = 1;

    fwrite($fp_tweetinfo, "TID">
    fwrite($fp_tweetinfo, "coordinates_type">
    fwrite($fp_tweetinfo, "coordinates_xaxis">
    fwrite($fp_tweetinfo, "coordinates_yaxis">
    fwrite($fp_tweetinfo, "retweet_count">
    fwrite($fp_tweetinfo, "in_reply_to_status_id">
    fwrite($fp_tweetinfo, "place_url">
    fwrite($fp_tweetinfo, "place_street_address">
    fwrite($fp_tweetinfo, "place_full_name">
    fwrite($fp_tweetinfo, "place_name">
    fwrite($fp_tweetinfo, "place_country">
    fwrite($fp_tweetinfo, "geo_type">

41
fwrite($fp_tweetinfo,"geo_coordinates_x"."\t"."geo_coordinates_y"."\t"."id"."\t"."retweeted"."\t"."truncated"."\t"."created_at"."\n");

fwrite($fp_userinfo, "TID"."\t");
fwrite($fp_userinfo,"statuses_count"."\t"."profile_background_tile"."\t"."profile_background_color"."\t");
fwrite($fp_userinfo, "contributors_enabled"."\t"."following"."\t"."geo_enabled"."\t"."favourites_count"."\t"."profile_text_color"."\t");
fwrite($fp_userinfo,"followers_count"."\t"."time_zone"."\t"."friends_count"."\t"."profile_link_color"."\t"."follow_request_sent"."\t");
fwrite($fp_userinfo,"verified"."\t"."notifications"."\t"."profile_sidebar_fill_color"."\t"."protected"."\t"."listed_count"."\t");
fwrite($fp_userinfo,"profile_use_background_image"."\t"."url"."\t"."profile_image_url"."\t"."name"."\t");
fwrite($fp_userinfo,"profile_sidebar_border_color"."\t"."id"."\t"."show_all_inline_media"."\t"."lang"."\t");
fwrite($fp_userinfo,"profile_background_image_url"."\t"."utc_offset"."\t"."created_at"."\t"."location"."\t"."screen_name"."\t"."description"."\n");

for($counter=0;$counter <$no_of_minutes;$counter++)
{
  $ch = curl_init($endpoint);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  $data = curl_exec($ch);
  $info = curl_getinfo($ch);
  curl_close($ch);
  if($info['http_code']==200)
  {
    $tweets = json_decode($data, true);
    echo $counter;
    print date('D, d M Y H:i:s T');
  }
  else if($info['http_code']==401)
  {
    die('Invalid credentials');
  }
  else if ($info['http_code']== 502)
  {
    sleep(900);
  }
  else if($info['http_code']== 403)
```php
{  
    sleep(60);
}
else
{
    echo "Network Error";
    sleep(120);
}

foreach($tweets as $tweet)
{
    if($tweet['user']['lang'] == 'en')
    {
        fwrite($fp_tweet, $id_counter."t") ;
        fwrite($fp_tweetinfo, $id_counter."t") ;
        fwrite($fp_userinfo, $id_counter."t") ;
        fwrite($fp_entities, $id_counter."n") ;
        tweet($fp_tweet, $tweet);
        tweetinfo($fp_tweetinfo, $tweet);
        userinfo($fp_userinfo, $tweet);
        entities($fp_entities, $tweet);
        $id_counter++;
        echo var_dump($tweet);
    }
}
sleep(25);
}
/*This function writes the id and tweet to a text file*/
function tweet($fp_tweet, $tweet)
{
    if($tweet['text'] == null)
    {
        fwrite($fp_tweet,"null"."n") ;
    }
    else
    {
        fwrite($fp_tweet, $tweet['text']."n") ;
    }
}
/*This function extracts the information about the tweets and writes to a file*/
function tweetinfo($fp_tweetinfo, $tweet)
{
    if($tweet['coordinates'][type] == null)
```
fwrite($fp_tweetinfo,"null"."\t");
}
else
{
fwrite($fp_tweetinfo,$tweet['coordinates']['type']."\t");
}
if($tweet['coordinates']['coordinates'][0] == null)
{
fwrite($fp_tweetinfo,0);
fwrite($fp_tweetinfo,"\t");
}
else
{
fwrite($fp_tweetinfo,$tweet['coordinates']['coordinates'][0]."\t");
}
if($tweet['coordinates']['coordinates'][1] == null)
{
fwrite($fp_tweetinfo,0);
fwrite($fp_tweetinfo,"\t");
}
else
{
fwrite($fp_tweetinfo,$tweet['coordinates']['coordinates'][1]."\t");
}
if($tweet['retweet_count'] == null)
{
fwrite($fp_tweetinfo,"0"."\t");
}
else
{
fwrite($fp_tweetinfo,$tweet['retweet_count']."\t");
}
if($tweet['in_reply_to_status_id'] == null)
{
fwrite($fp_tweetinfo,"null"."\t");
}
else
{
fwrite($fp_tweetinfo,$tweet['in_reply_to_status_id']."\t");
}
if($tweet['place'] == null)
{
for($i=0;$i <8;$i++)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
 if($tweet['place']['url'] == null)
 { 
 fwrite($fp_tweetinfo,"null"."\t");
 }
 else
 { 
 fwrite($fp_tweetinfo,$tweet['place']['url']."\t");
 }
 if($tweet['place']['attributes'] == null)
 { 
 fwrite($fp_tweetinfo,"null"."\t");
 }
 else
 { 
 if($tweet['place']['attributes']['street_address'] ==
 null)
 { 
 fwrite($fp_tweetinfo,"null"."\t");
 }
 else
 { 
 fwrite($fp_tweetinfo,$tweet['place']['attributes']['street_address']."\t");
 }
 }
 if($tweet['place']['full_name'] == null)
 { 
 fwrite($fp_tweetinfo,"null"."\t");
 }
 else
 { 
 fwrite($fp_tweetinfo,$tweet['place']['full_name']."\t");
 }
 if($tweet['place']['name'] == null)
 { 
 fwrite($fp_tweetinfo,"null"."\t");
 }
}
else
{
fwrite($fp_tweetinfo,$tweet['place']['name'])."\t";
}

if($tweet['place']['country_code'] == null)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
fwrite($fp_tweetinfo,$tweet['place']['country_code']."\t");
}

if($tweet['place']['id'] == null)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
    fwrite($fp_tweetinfo,$tweet['place']['id'])."\t";
}

if($tweet['place']['country'] == null)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
    fwrite($fp_tweetinfo,$tweet['place']['country']."\t");
}

if($tweet['place']['place_type'] == null)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
    fwrite($fp_tweetinfo,$tweet['place']['place_type']."\t");
}
if($tweet['favorited'] == null)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
    fwrite($fp_tweetinfo,$tweet['favorited']."\t");
}

if($tweet['source'] == null)
{
    fwrite($fp_tweetinfo,"null"."\t");
}
else
{
    fwrite($fp_tweetinfo, $tweet['source']."\t");
}

if(sizeof($tweet['geo'])> 0)
{
    fwrite($fp_tweetinfo, $tweet['geo']['type']."\t");
    fwrite($fp_tweetinfo,
    $tweet['geo']['coordinates'][0]."\t");
    fwrite($fp_tweetinfo,
    $tweet['geo']['coordinates'][1]."\t");
}
else
{
    fwrite($fp_tweetinfo,"null"."\t");
    fwrite($fp_tweetinfo,"null"."\t");
    fwrite($fp_tweetinfo,"null"."\t");
}

if($tweet['id'] == null)
{
    fwrite($fp_tweetinfo, "null"."\t");
}
else
{
    fwrite($fp_tweetinfo, $tweet['id']."\t");
}

if($tweet['retweeted'] == TRUE)
{
    fwrite($fp_tweetinfo, $tweet['retweeted']."\t");
}
else
{
    fwrite($fp_tweetinfo, $tweet['id']."\t");
}
fwrite($fp_tweetinfo, "0");
}
if($tweet['truncated'] == TRUE)
{
    fwrite($fp_tweetinfo, $tweet['truncated']."\t");
} else
{
    fwrite($fp_tweetinfo, "0"."\t");
}
if($tweet['created_at'] == null)
{
    fwrite($fp_tweetinfo, "null"."\n");
} else
{
    fwrite($fp_tweetinfo, $tweet['created_at']."\n");
}

/*This function extracts the user information and writes to a file*/
function userinfo($fp_userinfo, $tweet)
{
    if(sizeof($tweet['user'])>0)
    {
        fwrite($fp_userinfo,$tweet['user']['statuses_count']."\t");
        if($tweet['user']['profile_background_tile'] == TRUE)
        {
            fwrite($fp_userinfo, $tweet['user']['profile_background_tile']."\t");
        } else
        {
            fwrite($fp_userinfo, 0);
            fwrite($fp_userinfo, "\t");
        }

        if($tweet['user']['profile_background_color'] == null)
        {
            fwrite($fp_userinfo,"none"."\t");
        } else
        {
            fwrite($fp_userinfo,"\n");
        }
    } else
    {
        fwrite($fp_userinfo, "$\n");
    }
}
fwrite($fpuserinfo,$tweet['user']['profile_background_color']."\t");

if($tweet['user']['contributors_enabled'] == TRUE) {
    fwrite($fpuserinfo,$tweet['user']['contributors_enabled']."\t");
} else {
    fwrite($fpuserinfo, 0);
    fwrite($fpuserinfo, "\t");
}

if($tweet['user']['following'] == TRUE ) {
    fwrite($fpuserinfo,$tweet['user']['following']."\t");
} else {
    fwrite($fpuserinfo, 0);
    fwrite($fpuserinfo, "\t");
}

if($tweet['user']['geo_enabled'] == TRUE) {
    fwrite($fpuserinfo,$tweet['user']['geo_enabled']."\t");
} else {
    fwrite($fpuserinfo, 0);
    fwrite($fpuserinfo, "\t");
}

fwrite($fpuserinfo,$tweet['user']['favourites_count']."\t");

if($tweet['user']['profile_text_color'] == null) {
    fwrite($fpuserinfo,"none"."\t");
} else {
    fwrite($fpuserinfo,$tweet['user']['profile_text_color']."\t");
}
fwrite($fp_userinfo, $tweet['user']['followers_count'] . "\t");

if($tweet['user']['time_zone'] == null)
{
    fwrite($fp_userinfo, "null\t");
} else
{
    fwrite($fp_userinfo, $tweet['user']['time_zone'] . "\t");
}

fwrite($fp_userinfo, $tweet['user']['friends_count'] . "\t");

if($tweet['user']['profile_link_color'] == null)
{
    fwrite($fp_userinfo, "null\t");
} else
{
    fwrite($fp_userinfo, $tweet['user']['profile_link_color'] . "\t");
}

if($tweet['user']['follow_request_sent'] == TRUE)
{
    fwrite($fp_userinfo, $tweet['user']['follow_request_sent'] . "\t");
} else
{
    fwrite($fp_userinfo, 0);
    fwrite($fp_userinfo, "\t");
}

if($tweet['user']['verified'] == TRUE)
{
    fwrite($fp_userinfo, $tweet['user']['verified'] . "\t");
} else
{
    fwrite($fp_userinfo, 0);
    fwrite($fp_userinfo, "\t");
}
if($tweet['user']['notifications'] == TRUE)
{
    fwrite($fp_userinfo, "$tweet['user']['notifications']."\t";)
} else
{
    fwrite($fp_userinfo, 0);
    fwrite($fp_userinfo, "\t";)
}

if($tweet['user']['profile_sidebar_fill_color'] == null)
{
    fwrite($fp_userinfo,"null"."\t";)
} else
{
    fwrite($fp_userinfo, "$tweet['user']['profile_sidebar_fill_color']."\t";)
}

if($tweet['user']['protected'] == TRUE)
{
    fwrite($fp_userinfo, "$tweet['user']['protected']."\t";)
} else
{
    fwrite($fp_userinfo, 0);
    fwrite($fp_userinfo, "\t";)
}

fwrite($fp_userinfo, "$tweet['user']['listed_count']."\t";)

if($tweet['user']['profile_use_background_image'] == TRUE)
{
    fwrite($fp_userinfo, "$tweet['user']['profile_use_background_image']."\t";)
} else
{
    fwrite($fp_userinfo, 0);
    fwrite($fp_userinfo, "\t";)
}

if($tweet['user']['url'] == null)
{
    fwrite($fp_userinfo,"null"."\t";)
}
$tweet['user']['url']."\t";

fwrite($fpuserinfo,

if($tweet['user']['profile_image_url'] == null) {
    fwrite($fpuserinfo,"null"."\t");
} else {
    fwrite($fpuserinfo,

fwrite($fpuserinfo,$tweet['user']['profile_image_url']."\t");

if($tweet['user']['name'] == null) {
    fwrite($fpuserinfo,"null"."\t");
} else {
    fwrite($fpuserinfo,

fwrite($fpuserinfo,$tweet['user']['name']."\t");

if($tweet['user']['profile_sidebar_border_color'] == null) {
    fwrite($fpuserinfo,"null"."\t");
} else {
    fwrite($fpuserinfo,$tweet['user']['profile_sidebar_border_color']."\t");

fwrite($fpuserinfo,$tweet['user']['id']."\t");

if($tweet['user']['show_all_inline_media'] == TRUE) {
    fwrite($fpuserinfo,

fwrite($fpuserinfo,$tweet['user']['show_all_inline_media']."\t");
} else {
    fwrite($fpuserinfo, 0);
    fwrite($fpuserinfo, "\t");

52
if($tweet['user']['lang'] == null)
{
    fwrite($fp_userinfo,"null\"t");
} else 
{
    fwrite($fp_userinfo,
    $tweet['user']['lang']."t");
}

if($tweet['user']['profile_background_image_url'] == null)
{
    fwrite($fp_userinfo,"null\"t");
} else 
{
    fwrite($fp_userinfo,
    $tweet['user']['profile_background_image_url']."t");
}

if($tweet['user']['utc_offset'] == null)
{
    fwrite($fp_userinfo,"null\"t");
} else 
{
    fwrite($fp_userinfo,
    $tweet['user']['utc_offset']."t");
}

if($tweet['user']['created_at'] == null)
{
    fwrite($fp_userinfo,"null\"t");
} else 
{
    fwrite($fp_userinfo,
    $tweet['user']['created_at']."t");
}

if($tweet['user']['location'] == null)
{
    fwrite($fp_userinfo,"null\"t");
} else 
{
    fwrite($fp_userinfo,
    $tweet['user']['location']."t");
}
if($tweet['user']['screen_name'] == null)
{
    fwrite($fp_userinfo,"null"."\t");
} else
{
    fwrite($fp_userinfo,$tweet['user']['screen_name']."\t");
}

if($tweet['user']['description'] == null)
{
    fwrite($fp_userinfo,"null"."\n");
} else
{
    fwrite($fp_userinfo,$tweet['user']['description']."\n");
}

/* This function extracts the entities information about the tweets and writes to a text file.*/
function entities($fp_entities, $tweet)
{
    if(sizeof($tweet['entities'])>0)
    {
        fwrite($fp_entities,sizeof($tweet['entities']['hashtags'])."\t".sizeof($tweet['entities']['user_mentions'])."\t".sizeof($tweet['entities']['urls'])."\n");

        if(sizeof($tweet['entities']['hashtags'])>0)
        {
            for($counter=0;$counter<sizeof($tweet['entities']['hashtags']);$counter++)
            {
                fwrite($fp_entities,$tweet['entities']['hashtags'][$counter]['indices'][0]."\t");

                fwrite($fp_entities,$tweet['entities']['hashtags'][$counter]['indices'][1]."\t");

                fwrite($fp_entities,$tweet['entities']['hashtags'][$counter]['text']."\n");
            }
        }

        if(sizeof($tweet['entities']['user_mentions'])>0)
        {
            for($counter=0;$counter<sizeof($tweet['entities']['user_mentions']);$counter++)
            {
                fwrite($fp_entities,$tweet['entities']['user_mentions'][$counter]['indices'][0]."\t");

                fwrite($fp_entities,$tweet['entities']['user_mentions'][$counter]['indices'][1]."\t");

                fwrite($fp_entities,$tweet['entities']['user_mentions'][$counter]['screen_name']."\t");

                fwrite($fp_entities,$tweet['entities']['user_mentions'][$counter]['name']."\n");
            }
        }
    }
}
for($counter=0;$counter < sizeof($tweet['entities']['user_mentions']);$counter++)
{
    fwrite($fp_entities, $tweet['entities']['user_mentions'][$counter]['indices'][0]."\t");
    fwrite($fp_entities, $tweet['entities']['user_mentions'][$counter]['indices'][1]."\t");

    if($tweet['entities']['user_mentions'][$counter]['screen_name'] == null)
    {
        fwrite($fp_entities,"null"."\t");
    }
    else
    {
        fwrite($fp_entities,$tweet['entities']['user_mentions'][$counter]['screen_name']."\t" );
    }

    if($tweet['entities']['user_mentions'][$counter]['name'] == null)
    {
        fwrite($fp_entities, "null"."\t");
    }
    else
    {
        fwrite($fp_entities, $tweet['entities']['user_mentions'][$counter]['name']."\t" );
    }

    if($tweet['entities']['user_mentions'][$counter]['id'] == null)
    {
        fwrite($fp_entities,"null"."\n");
    }
    else
    {
        fwrite($fp_entities,$tweet['entities']['user_mentions'][$counter]['id']."\n" );
    }

    if(sizeof($tweet['entities']['urls']) > 0)
for($counter=0;$counter < sizeof($tweet['entities']['urls']);$counter++)
{
    fwrite($fp_entities,$tweet['entities']['urls'][$counter]['indices'][0]."\t");
    fwrite($fp_entities,$tweet['entities']['urls'][$counter]['indices'][1]."\t");

    if($tweet['entities']['urls'][$counter]['expanded_url'] == null)
    {
        fwrite($fp_entities,"null"."\n");
    } else
    {
        fwrite($fp_entities,$tweet['entities']['urls'][$counter]['expanded_url']."\t");
    }

    fwrite($fp_entities,$tweet['entities']['urls'][$counter]['url']."\n");
}
}
fclose($fp_tweet);
fclose($fp_tweetinfo);
fclose($fp_userinfo);
fclose($fp_entities);
?>