Queen Mary, University of London
Department of Electronic Engineering and Computer Science

Draft Report

Fayimora Femi-Balogun

Supervisor: Dr. Matthew Purver

Submitted in part fulfilment of the requirements for the degree of
BSc Computer Science with Industrial Experience, April 2014

Abstract

Acknowledgements

Dedication

Dedication here.

‘No amount of experimentation can ever prove me right; a single experiment can prove me wrong.’
Albert Einstein

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Organisations today continuously search for new ways to get feedback from their clients in a bid to improve customer satisfaction. Technology firms like Apple, Samsung and Google want to know if their software/hardware products meet their consumers' needs. Merchandise retailers like Walmart and Tesco are constantly trying to make sure they are serving the right products in the right quantity and at the right price. Startups continuously evaluate their products to measure the probability of the company being successful sometime in the future. Postal services like Royal Mail are very interested in how their services are doing and what their customers despise most so they can improve. Current ways of achieving this include **Surveys** (questionnaires or interviews) and **Focus Groups**.

Surveys are very easy to create and distribute. There are also a variety of tools to help with this. Some of them include SurveyMonkey[1] and Google Docs[2]. Unfortunately, Surveys also have a few unpleasant drawbacks like time consumption and labour intensity. It can also be difficult to encourage participants to respond. Nevertheless, the main drawback to using Surveys is that some questions are left unanswered while the answers given in answered questions may not reflect the truthful sentiments of the participant. **?** concurs with this and he goes on to discuss how this problem can be solved (to a certain extent) with imputation[3]. **?** also agrees with this

[1] https://www.surveymonkey.com/
[2] https://drive.google.com
[3] Imputation is the process of inferring plausible values for missing entries

point of view and suggests the use of well designed leading questions to put the participant in the right frame of mind. For instance, a leading question like "*How likely will you recommend our service to friends?*" gets the participant thinking about recommendations. While the above solutions might work, they have the same drawbacks as the original problem. Imputation can be very time consuming, labour intensive and error prone while the use of leading questions fails to solve the problem of unanswered questions.

Unfortunately, interviews and focus groups also suffer from false answers due to the fact that they are not anonymous. This means that the participants, in the face of an interviewer, try to be lenient in other not to sound too negative. This could sometimes be due to the fact that participation in the interview/focus group has been incentivised with money or desirable items.

Ideally, the next question we should be asking is "*How can we get the truthful views of our clients about our products and services?*"? We need to find a way to get this information without putting any pressure on our clients.

1.2 Aims and Objectives

The main aim of this project is to investigate other means of getting our data and also, how we can make use of Machine Learning and Natural Language Processing techniques to make sense of the data.

Fortunately, the recent surge in the use of social media makes the former relatively easy. People, more often than not, tend to post their truthful feelings about services they use on social media. For instance, Person A buys an iPhone today and realises that the Wi-Fi connectivity is faulty. He/She will most likely post something like "*New iPhone wifi not working #NotCool*" on one or more of the available social networking platforms. From this statement, we can infer that Person A is talking about *the iPhone*, *Wi-Fi* and *Connectivity*. We could also infer that the sentiment of the user, with respect to those topics, is somewhat *negative*. The process of discovering abstract topics in text is called **Topic Modelling** while the process of discovering sentiments in text is known as **Sentiment Analysis**. Chapters 4 and 5 discuss how we can automate these processes, respectively.

We will try to answer some research questions. They include:

- Can we use supervised techniques to accurately classify tweets into what is relevant and what is not?

- Can we detect themes/topics in our dataset? If yes, are these topics related to Apple Inc in any way?

- One way to know the preferences of anyone is by knowing their interests. Can we profile/group people in terms of their interests?

1.3 Why Twitter?

Twitter is a social micro-blogging platform where users can share messages in 140 characters. It also allows its users to follow each other. This means, if person A follows person B, A will see public posts from B. These messages are usually referred to as tweets.

Tweets are capped to 140 characters and can contain text, links or a combination of both. They are usually related to either an event, interests or just personal opinion. Facebook posts are mostly always well thought out and each post might include multiple topics. Tweets on the other hand are usually written at the speed of thought. This makes it a good source of data.

According to Mashable, DOMO, a Business Intelligence company paired up with Column Five Media to create an infographic[4] about the web back in 2012. It showed that Twitter at the time received around 100,000 tweets per minute. As at 1st February 2014 Twitter claims to receive 500 million tweets a day[5]. That is roughly 350,000 tweets per minute which is 3 times the amount 2 years before. Twitter also claims to have 241 million monthly users.

Finally, Twitter's data is open compared to other social platforms like Facebook. This means developers are free to tap into this wealth of data in almost real time. This makes Twitter a perfect source for our data.

[4]See `http://mashable.com/2012/06/22/data-created-every-minute/`
[5]See https://about.twitter.com/company

1.4 Methodology

How do I plan to approach this research? What kind of machine learning/text processing methods, evaluation techniques will be used?

This study requires social data and the dataset used is gathered from Twitter over a time frame. The data collected is related to Apple Inc and their products.

With data already gathered, the first step would be to start training a classifier to help filter out as many irrelevant tweets as possible. We will briefly analyse different ways to classify text. We eventually settle with the Naïve Bayes Classifier and we look into different ways of analysing its performance.

The next step will be to identify topics/themes in the data. This means we will attempt to detect the main topics being discussed with respect to Apple Inc. We briefly look at Latent Semantic Indexing and why it might not be suitable for our needs. We then look into Latent Dirichlet Allocation, a common approach to topic modelling and use it to detect themes in our dataset.

1.5 Statement of Originality

Statement here.

Chapter 2

Background Theory

2.1 Introduction

Automatic Text Classification or Text Categorization is a rapidly growing field in Machine Learning and Natural Language Processing. This is mainly due to the amount of electronic data we currently generate. The main task is to assign one or more classes to a given text document. Applications of text classification include *Email Spam Detection* and *Language Detection*. The former involves trying to distinguish spam emails from legitimate ones while the latter involves the identification of the language a document was written in.

However, this study makes use of classification techniques for data filtration (removing irrelevant documents from a list of documents, similar to spam filtering), topic modelling (extracting topics from a list of documents) and sentiment analysis (predicting the sentiment of the author of a document). This chapter explains a few background concepts and reviews some relevant research previously done in this area.

2.2 Naïve Bayes Classifier

The Naïve Bayes classifier is the simplest classifier that can be used and this is due to the fact that it is based on simple Bayes Theorem. It is a probabilistic classifier which assumes that all features of the documents are independent of each other. This means that if a document

has features $f1$ and $f2$ (could be length of document, occurrence of words, language e.t.c), the existence of $f1$ has nothing to do with the existence of $f2$ and vice versa. This also means that it makes assumptions that may or may not be correct, hence the "Naïve" in its name.

Bayes theorem states that the probability of A given B is the probability of B given A times the probability of A divided by the probability of B. Mathematically, this is written as:

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{2.1}$$

Applying this logic to text classification, the probability that a document $d_i \in D$ belongs to a class c is denoted as:

$$p(c|d_i) = \frac{p(d_i|c)p(c)}{p(d_i)} \tag{2.2}$$

Although other techniques like Maximum Entropy, Random Forests or Support Vector Machines tend to perform better, a naive Bayes classifier will require less memory and CPU cycles. Furthermore, it is computationally less complex and simpler to implement. With regard to performance, **?** showed using multiple datasets from **?** that the naive Bayes classifier in many cases performs as good as other complex classifiers and **?** goes further to explain why it performs well. Other studies have also found Bayesian classifiers to be effective without being affected by its simple independence assumption (**??**).

The Naïve Bayes classifier has been used in many text classification problems but one of its common applications which is relevant to tweet classification is email spam[1] filtering. A spam filter is a system that takes in text and decides whether or not it is spam. **?** addressed this issue using a naive Bayes classifier. They trained the model using a predefined set of manually labelled messages. They were able to show that the naive Bayes classifier was capable of classifying messages with impressive accuracy and precision compared to the then common keyword based approach to classification. **?** also carried out a similar research and the results were equally impressive and similar.

[1] irrelevant or unsolicited messages. They are typically to large numbers of users

2.3 Topic Modelling

Topic Modelling is a process by which abstract topics/themes are extracted from a collection of documents. This process is usually carried out with the aid of topic models, a suite of algorithms used for topic modelling. It has been applied in a variety of fields like Software Analysis where **?** used topic modelling to find topics embedded in code and **?** used topic modelling to capture coupling among classes. **?** applied topic models on disaster related data from Twitter in an effort to determine what topics were discussed within the time span of a natural disaster. **?** introduced a new topic model that can be used to analyze videos with complex and crowded scenes in other to discover regularities in the videos. A system built on such model will be able to answer a question like "What interesting events happened in the last 5 hours". Other fields include Audio Analysis (**?**), Influence modelling (**?**), Finance (**?**), Writer Identification (**?**) and many more.

There are a number of topic models but the two main ones are ***Latent Semantic Indexing*** (LSI) and ***Latent Dirichlet Allocation*** (LDA) and we discuss them further in the following sections.

2.3.1 Latent Semantic Indexing

Latent Semantic Indexing(LSI) (**?**), sometimes referred to as *Latent Semantic Analysis*, is an indexing technique that leverages matrix-algebra computations[2] to identify any patterns in relationships between a collection of text documents. It works based on the assumption that words used in the same context tend to have homogeneous meanings (**???**). LSI, has been used mostly in Information Retrieval and Search Engine Optimisation where it tries to figure out what words in a web page are relevant to the web page even though they might not be used in that page. One of the main drawbacks the LSI model suffers from is ambiguity.

Assuming we have two documents, one talking about Microsoft Office and the other talking about actual physical office space. How can the model differentiate between the two? Unfortunately, it is unable to and a significant step forward to solve this problem was made by **?** who

[2]Specifically, it uses Singular Value Decomposition which is a factorization of a complex matrix. See `http://en.wikipedia.org/wiki/Singular_value_decomposition`

presented the probabilistic LSI model. **?** argues that while Hoffman's work is a very useful step towards using probabilistic models to model text, it is incomplete.

2.3.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation(LDA) (**?**) is a generative[3] and probabilistic model that can be used to automatically group words into topics and documents into a mixture of topics. It works based on the assumption that each document contains one or more topics. Words can also exist in multiple topics as they actually do in natural language. In other to tackle the problem of ambiguity in LSI, Latent Dirichlet Allocation takes a combination of all topics that seem relevant to a document in a corpora[4] and compares that document to the topics in an effort to determine which topic is more closer to the document.

Most of the research done on social media data, especially Twitter, has been to detect usage and communities (**?**). Nonetheless, recent research has started to look into the detection of topics in social media. **?** used LDA to extract topics/themes from a collection of disaster related tweets. **?** used LDA to compare news related tweets on Twitter with topics in The New York Times. They were also able to show that the standard LDA might not always work well on tweets and so they proposed a new model which is a slight variant of LDA. **?** proposed an algorithm that leverages LDA to find topic-sensitive influential twitter users.

[3]See `http://en.wikipedia.org/wiki/Generative_model`
[4]Corpora is simply a large collection of documents

Chapter 3

Data Classification

First step towards this project is to fetch our data from Twitter. The data is classified into two groups, relevant and irrelevant. We will be spending most of our time with the relevant data.

To carry out our experiments, we will need to filter out irrelevant tweets. Irrelevant tweets are tweets which we do not really care about. Some examples include:

- *Every day I'm levelling! And now I'm level 19 in #CSRClassics for iPhone!*
- *Yes, our apple juice and cider are both GMO-free.*
- *I just had my first carmel apple*

All three tweets could be regarded as relevant but for our use case, they are not. This is because we are only interested in tweets that contain personal opinions about Apple Incorporated. Examples of relevant tweets include: their thoughts

- *Once you get hooked to #Mac, you will definitely go back to #Windows! Lol!*
- *If Tim Cook at Apple knows anything about him, it'd be to stay away from Icahn.*

Of course we can manually classify this data but when we have millions of tweets, this becomes impracticable. This is where we employ some classification algorithms to assist us. This is a three step process and we will discuss them in the next sub sections.

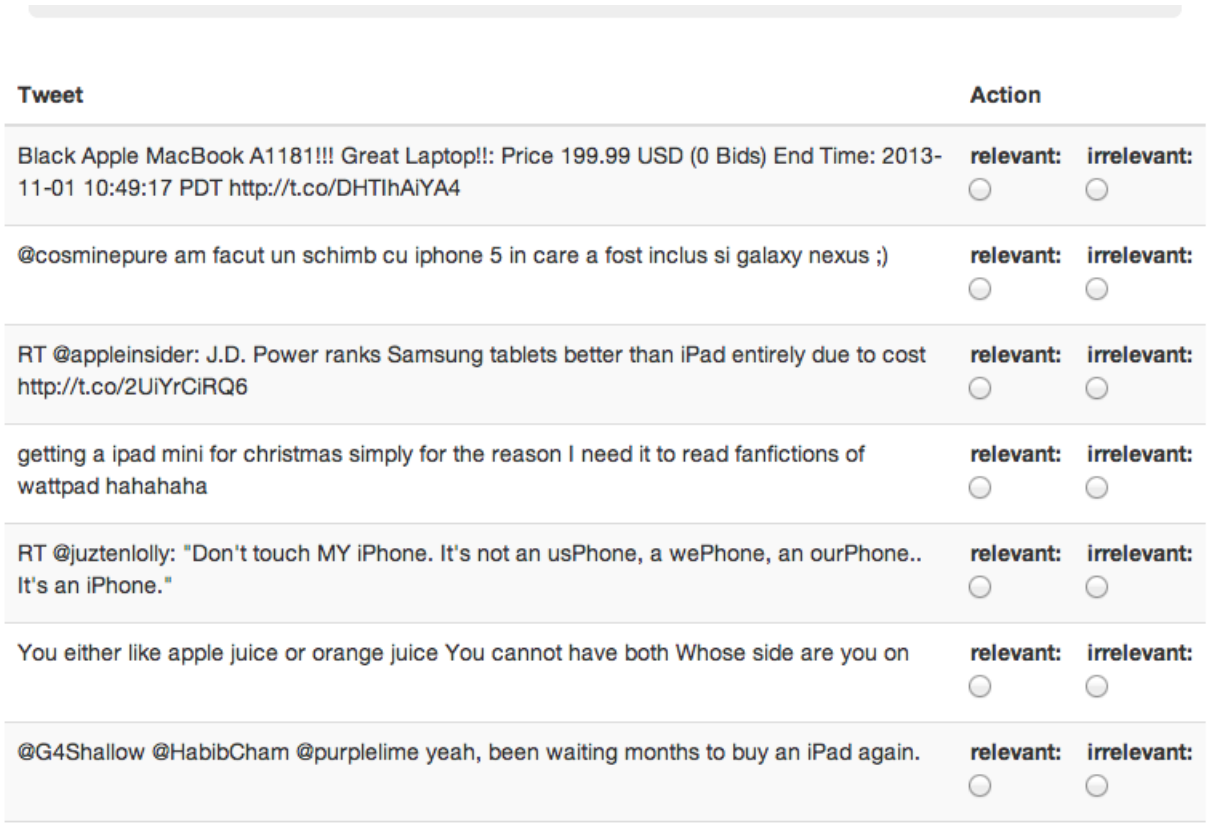


Figure 3.1: The data labelling application

3.1 Preparing train data

Train data, also known as a training set is a set of data used to train a knowledge database, in this case, a classifier. Our training set will be created by manually labelling a fraction of our dataset. People write in different ways on Twitter and trying to create a new training set to encompass all possibilities would be very time consuming and intractable. To make this process a little easier, a web application for labelling tweets was created. Figure 3.1 is a screen shot of what the application looks like.

While using the web application in Figure 3.1 makes labelling tweets easier and a little quicker, it does not change the fact the we still have to manually label a plethora of tweets. To speed up this process even further, the data labeller was made public and the labelling was crowd sourced. A list of instructions (Figure 3.2) were also given to anyone who helped label the tweets.

One problem with crowd sourcing this task is that people have different opinions about what is relevant and what is not. In an attempt to solve this problem, each tweet was classified twice.

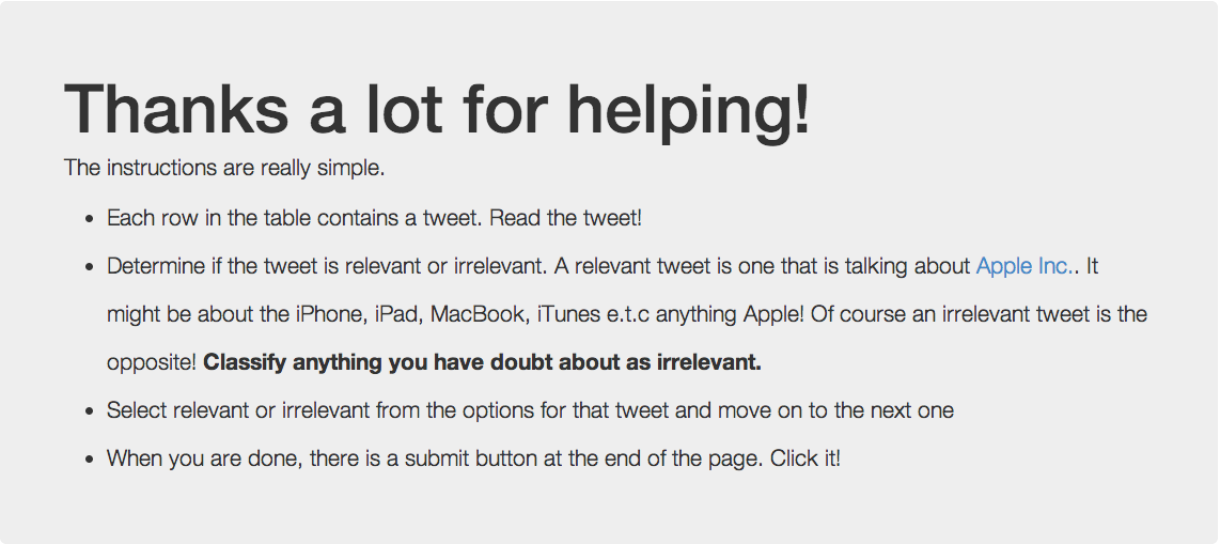


Figure 3.2: Instructions on how to label the tweets

A tweet classified as relevant gets a score of 1 and an irrelevant tweet gets 0. This means that if a tweet was classified twice as relevant, it should have a score of 2 and a tweet classified as irrelevant twice should have a score of 0. Tweets that have been classified twice and have a total score of 1 are tweets that have been classified as both relevant and irrelevant. These are tweets that we have to classify ourselves into a group. While this is not an assured way of getting the best training set, it gives us a certain level of confidence about our training set. It is also arguably much better than single handedly creating the training set.

3.2 Training a classifier

As discussed in Section 2.2, a Naïve Bayes Classifier is a probabilistic classifier which is based on the Bayes Theorem. We will train one and use it to classify the tweets into relevant and irrelevant groups.

Unfortunately, the classifier takes as input a vector space representation of our tweets and not the actual text. This means we have to convert our tweets into a vector representation of some sort. We will be using the **bag of words model** in this study but before we transform the tweets, we have to pre-process the tweets.

3.2.1 Preprocessing

Preprocessing are the tasks we have to carry out before the main transformation of the tweets to a vector space model. Firstly, we will peruse through our tweets to remove new line characters, links and stop words. We then take each tweet and convert it into a list of *n-grams*.

Some tweets have special characters like new lines, excess spaces and Unicode characters and these characters are irrelevant for our use-case. Every programming language has a function to strip off newlines and whitespace and it can be easily done in one line of code. Removing the links from the text is a little more complex and the "easiest" way to do this would be to use a regular expression. **?** in his book *Mastering Regular Expressions* describes regular expressions as a very flexible mini language that is used for text processing. The regular expression we will be using to find links in our text is

```
{(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \.-]*)*\/?}
```

Unfortunately, all a regular expression can do is search for patterns in text. Luckily, most programming languages provide support for regular expressions so all we have to do is search for the pattern in each tweet and use the language features to replace the matched pattern with nothing(an empty string preferably).

The next step is to remove stop words in each tweet. **?** defines a stop word as "*a word which may be identified as a word that has the same likelihood of occurring in those documents not relevant to a query as in those documents relevant to the query.*" In other words, stop words occur in every document irrespective of the document's relevance. Stop words are usually the most common word in a language, English in this case. Some examples include *and*, *or*, *the* etc. Removal of stop words from text usually results in better model performance as shown in Figure 3.3 on page 14.

Finally, we convert each tweet to a list of *n-grams*. An n-gram "*is a contiguous sequence of n items from a given sequence of text*"[1]. The easiest way to understand n-grams is with an example. Assuming we have a document with the text "machine learning rocks". All unigrams(n-grams where n is 1) that can be extracted from that text are *"machine"*, *"learning"*

[1]See http://en.wikipedia.org/wiki/N-gram

	today	what	it	is	a	sunny	day
A	1	0	0	1	1	1	1
B	0	0	2	1	1	1	1
C	0	1	0	0	1	1	1

Table 3.1: A bag-of-words representation

and *"rocks"*. Also, all bigrams(n-grams where n is 2) in the document are *"machine learning"* and *"learning rocks"*. In this study, we will be using a combination of unigrams and bigrams.

We have discussed different preprocessing tasks that we have to apply to our documents before transforming them into the bag of words matrix representation. In the next section, we will look into how the bag of words model works and then transform our tweets into this model.

3.2.2 Transforming tweets to bag-of-words

The bag of words model is a common representation for text that involves representing a document as a multiset of its words. It is a very common way to represent documents and it has also been used recently in computer vision (**?**). All sets are combined to form a document-term matrix of the corpora. The rows represent each document while the columns represent the occurrence/frequency of a word in that document. To show how this works, let us assume we have the following documents:

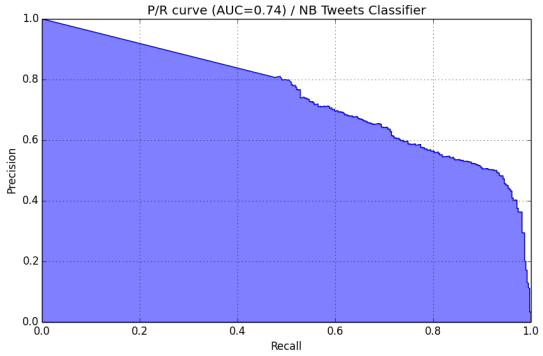
A today is a sunny day.

B it is a sunny day isn't it?

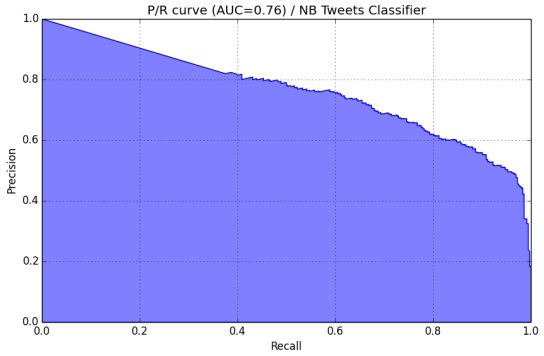
C what a sunny day!

By the above definition, Table 3.1 will be an accurate representation of our sentences using the bag of words model. Note that in our example, each sentence is a document and all sentences form the corpora.

Now that we have converted our corpora into a bag of words representation, we will now use the resulting matrix to train our classifier.



(a) AUC=74 with stopwords



(b) AUC=76 without stopwords

Figure 3.3: AUC curves with and without stopwords

accuracy	std(σ)	AUC	std(σ)
0.9875	0.0000	0.7387	0.0000
0.9877	0.0002	0.7381	0.0000
0.9878	0.0002	0.7312	0.0090
0.9877	0.0002	0.7412	0.0190
0.9878	0.0002	0.7454	0.0190
0.9876	0.0005	0.7394	0.0220
0.9875	0.0005	0.7431	0.0220
0.9874	0.0005	0.7427	0.0200
0.9874	0.0005	**0.7455**	0.0210
0.9873	0.0005	0.7454	0.0200

(a) With stopwords

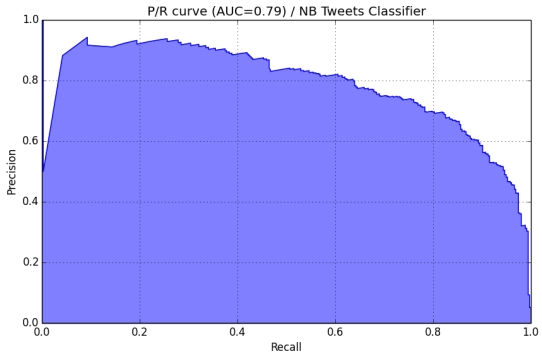
accuracy	std(σ)	AUC	std(σ)
0.9908	0.0000	0.7526	0.0000
0.9904	0.0004	0.7589	0.0060
0.9903	0.0003	0.7485	0.0150
0.9901	0.0004	0.7535	0.0160
0.9901	0.0003	0.7572	0.0160
0.9901	0.0003	0.7582	0.0150
0.9902	0.0004	**0.7618**	0.0160
0.9901	0.0004	0.7587	0.0170
0.9901	0.0004	0.7592	0.0160
0.9900	0.0004	0.7572	0.0160

(b) Without stopwords

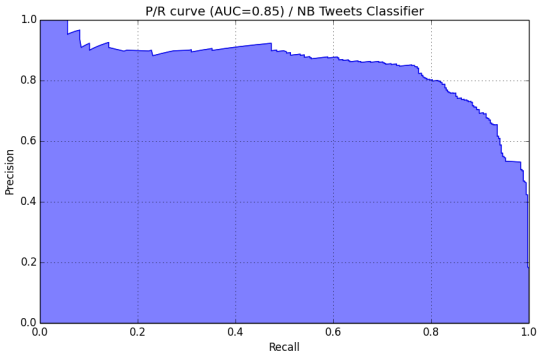
Table 3.2: Tables showing accuracy and AUC for 10-fold cross validation

3.2.3 Training the initial classifier

3.2.4 Improving the classifier



(a) AUC curve for tf-idf weighted corpora



(b) AUC curve for best found model

Figure 3.4: AUC curves for tf-idf weighted and best found models, resectively

accuracy	std(σ)	AUC	std(σ)
0.9939	0.0000	0.8068	0.0000
0.9939	0.0000	0.8006	0.0062
0.9949	0.0001	0.7893	0.0167
0.9939	0.0003	0.7960	0.0186
0.9939	0.0003	0.7892	0.0215
0.9939	0.0004	0.7920	0.0206
0.9939	0.0004	0.7902	0.0196
0.9939	0.0004	0.7900	0.0183
0.9939	0.0004	0.7890	0.0175
0.9939	0.0004	0.7876	0.0171

(a) with tfidf weighted model

accuracy	std(σ)	AUC	std(σ)
0.9961	0.0000	0.8415	0.0000
0.9960	0.0001	0.8610	0.0195
0.9955	0.0007	0.8445	0.0282
0.9955	0.0006	0.8491	0.0257
0.9954	0.0005	0.8441	0.0251
0.9954	0.0005	0.8447	0.0229
0.9954	0.0005	0.8465	0.0217
0.9954	0.0004	0.8470	0.0203
0.9954	0.0004	0.8468	0.0192
0.9954	0.0004	0.8487	0.0191

(b) best model

Table 3.3: Tables showing accuracy and AUC for Tf-idf weighte model and best searched model

Chapter 4

Topic Modelling

In this chapter, we will try to find themes/topics that exist in our dataset. Our input dataset will be a set of relevant tweets as determined by the classifier in the previous chapter. We will use Latent Dirichlet Allocation as discussed in Section 2.3 on page 7.

4.1 Analysing 10 topics

4.2 Analysing 20 topics

4.3 Analysing 50 topics

Chapter 5

Further Analysis

5.1 Sentiment Analysis

5.2 General Sentiments

5.3 Topic Based Sentiments

Chapter 6

Conclusion

6.1 Summary of Report Achievements

Summary.

6.2 Applications

Applications.

6.3 Future Work

Future Work.

Appendices

Appendix A

Sample Appendix

The content of the appendix