Imperial College London

MEng Individual Project

Sentiment Analysis For Debates

Author:
Christina Michael

Supervisor:
Dr. Francesca Toni

Second Marker:
Dr. Krysia Broda

June 17, 2013

Abstract

The underlying trend of people using microblogging to express their thoughts on various topics, has increased the need for developing computerised techniques for automatic sentiment analysis on texts that do not exceed 200 characters. An important, but not yet much explored area, is dealing with debates extracted from social networking sites like Twitter, Facebook, LinkedIn, MySpace, etc. which incorporate the use of microblogging.

For the purposes of this project, we have investigated some of the sentiment analysis techniques that were successfully used in the past, for longer texts, and performed adjustments for making them suitable for microblogging text. We have built two classifiers that can be used for this kind of debates: a support/opposition classifier that deals with dual-sided debates, and an agreement-disagreement classifier for dealing with replies to arguments posted by users in debates. For training these classifiers, we developed two corpora based on the ones used in other studies combined with data extracted from debating websites.

The two classifiers were the outcome of a series of experiments that includes exploring the performances of different supervised learning algorithms such as Support Vector Machines, Linear Regression, Naive Bayes, etc. when triggered with different parameters on various corpora, as well as dealing with different feature sets. The proposed techniques and methodologies are written in Python using NLTK and SciKit-Learn.

Additionally, since people are sometimes too busy and in need of a quick and easy way to understand debates, without having to look through the entire history of arguments and comments, we implemented three visualisations. The visualisations make use of Chernoff Faces, streamgraphs and line charts.

Finally, we evaluated the classifiers and visualisations using an existing debating system for social networking. The debating system can be found at http://www.quaestio-it.com.

Acknowledgements

I would like to offer my special thanks to my supervisor and personal turor, Dr Francesca Toni, for accepting to supervise my project. Her guidance and continuous feedback was very helpful and valuable throughout the course of this project.

Many thanks to Dr. Krysia Broda, my second marker, for her advice and constructive feedback on the interim report.

I would like to acknowledge the assistance provided by Valentinos Evripidou for integrating the classifiers and visualisations to quaestio-it.com; and Lucas Carstens for his insight on various sentiment analysis techniques.

Contents

Chapter 1

Introduction

> *"The growing stream of content placed on the Web provides a huge collection of textual resources. People share their experiences on-line, ventilate their opinions (and frustrations), or simply talk just about anything. The large amount of available data creates opportunities for automatic mining and analysis."*[1]

1.1 Objectives

This project aims to expand on existing solutions used for automatic sentiment analysis on text in order to capture support/opposition and agreement/disagreement in debates. Adding to this, it also aims at visualising the classification results for enhancing the ease of understanding the debates and for showing underlying trends. Finally, the project aims at evaluating the proposed techniques on an existing debate system for social networking.

As the number of people making their opinion available on social media has dramatically increased in the last decade, the opportunity to use these data for understanding what people think prevails. Being able to use information technologies to do so, has therefore many applications. For example, companies can use it to their advantage for quickly gathering feedback about their products and targeting any customer complaints. Furthermore, people tend to rely on reviews they read online for deciding whether to buy a particular product, or settle on an alternative one, or which movie to rent, or even where to go for a night out. Having to go through all the comments debating for a particular product can be frustrating and very time consuming.

It was not until recently that the area of Automatic Sentiment Analysis has seen interest from the academic community. Even though there was a booming of publications which are mainly oriented to indicate whether a document (or part of it) is positive, negative or neutral, or finding out the topic of discussion, there has not yet much been done for analysing debates. This gives the opportunity to expand on the current studies and elaborate on the techniques used in other areas, for creating a robust tool for sentiment analysis on debates.

Objective 1 - Classifications

The traditional way of extracting information is by poll taking and surveys. This methodology however, is not only time consuming and costly, but also limited to the number of people participating (usually no more than a few hundreds). Everyday, millions of people use the internet to broadcast their personal opinions on flaming issues like politics, health, love etc. So, why not take advantage of this vast amount of data and try to automatically extract this information about your topic of interest?

The main focus of this project is to explore various approaches for achieving "supervised" classification on two aspects. The first aspect is ***Support/Oppositon Classification***, for indicating whether the stance a person is taking in the debate supports or opposes the initial statement; and the second aspect is ***Agreement/Disagreement Classification*** for indicating whether a reply to an argument posted on a debate, is in agreement or disagreement with it. For example, consider the statements in Figure 1.1 showing a debate on abortion. The debate's purpose is to provide the underlying opinion to the following question: *"Should abortion be allowed?"*.

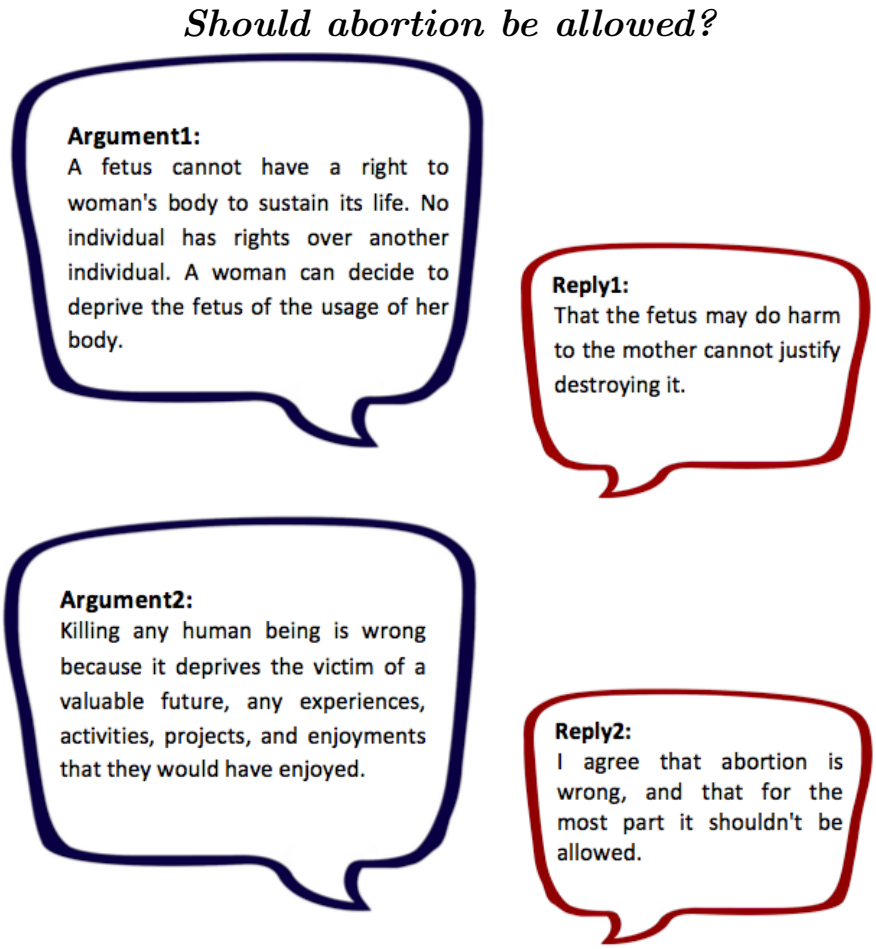


Figure 1.1: Debate Example wth a supporting argument (Argument1), an opposing argument (Argument2), a reply (Reply1) in disagreement with an argument (Argument1) and a reply (Reply2) in agreement with an argument (Argument2).

The task of the Support/Opposition Classifier, is to classify 'Argument1' as in *support*, and 'Argument2' as in *opposition* to the debate question. The Agreement/Disagreement Classifier's task is to classify 'Reply1' as being in *disagreement* with 'Argument1', and 'Reply2' as being in *agreement* with 'Argument2'.

Objective 2 - Visualisations

The second important aspect of this project is, having implemented the classifiers, to create interesting and interactive visualisations which are capable of engaging with the user, in order to provide important information that is based on the classification results. This enables the user to get an overall picture of the debate and to pick up underlying trends without having to go through the history of posts of other users on the debates.

Objective 3 - Evaluation on an existing debating system

Finally, an important aspect of this project is to evaluate the classifiers and the visualisations on a real system. Sometimes what seems to work well in theory, turns out to be rubbish when tested on actual situations. That is why, the classifiers and the visualisations have to be tested on an existing debating system, for a more thorough investigation on the applicability and usefulness of the proposed techniques for a microblogging debating system.

Compared to other studies that have already tried to target support/opposition or agreement/disagreement, this project emphasises on training data that do not exceed 200 characters each. This is very useful as the new trend of writing opinions is using extremely short messages, also known as microblogging. One example of such microblog service is Twitter[1], which poses a limit of 140 characters. Other social networking sites that tend to use microblogging as a means of "status update" are LinkedIn, Facebook and MySpace. Training a classifier for short messages requires different methodologies than when dealing with large pieces of text, allowing us to investigate on the appropriateness of various techniques for training and testing data extracted from microblogging.

Automatic sentiment analysis for data extracted from networking sites can be very challenging. Communication language varies and changes depending on the age, education, and ethnic background of the speaker. Consequently, the use of explicit rules (like you would do with any programming language), that deal with all the arising groups of speakers is extremely difficult or even impossible. To make things even more complicated, is trying to figure out the emotional status of the speaker as it is expressed through the text when it is not explicitly stated but rather implied or given in an indirect manner. This is a situation that arises quite often when trying to classify a text as support/opposition or agreement/disagreement. Furthermore, as the text is extracted from microblogging websites, there is little chance to be properly structured, in contrast to previous studies that dealt with congressional debates.

An automatic sentiment analysis tool for debates is therefore extremely helpful for topics like politics, new product release, etc., since these tend to be hot topics of discussion and you

[1] `https://twitter.com/`

can find many debates on Twitter, Facebook, personal blogs, forums, etc. which constitute a representative public opinion measurement review.

Additionally, giving the opportunity to see a visualisation of the debate conversation, provides an overall picture of the debate and, as a result, useful observations can be made about the prevailing thoughts on the matter, without having to go through the history of comments of other users. The visualisations have to be clear (no ambiguities) and easy to understand without any particular previous knowledge on visualisations.

1.2 Contribution

Contribution 1

In order to meet the first objective, we have developed two classifiers. One for showing support/opposition and one for showing agreement/disagreement. For the purposes of these classifiers, we have used several corpora and adjusted them to meet the specifications required for training and testing the classifiers. The corpora used for the Support/Opposition Classifier included political and ideological debates[2] and amazon reviews[3]. For the Agreement-Disagreement Classifier, the corpus used was extracted from the debating websites convinceme.net and 4forums.com.

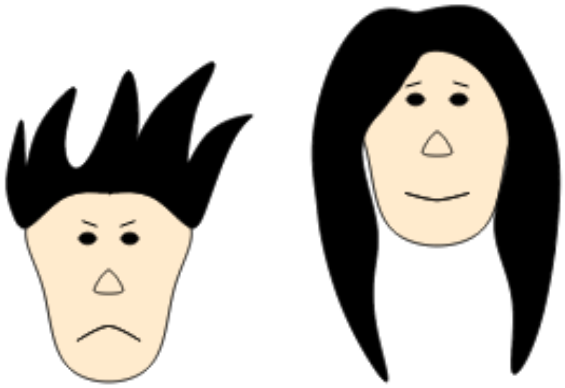
The development of the classifiers was based on some of the most successful techniques used in the past for targeting sentiment analysis. Using these techniques, we tried to find the most informative combination of features that, combined with a machine learning algorithm, will give high classification results. The resulting classifiers were therefore the outcome of a series of experiments using various machine learning techniques and machine learning algorithms when triggered using different parameters.

The proposed techniques and methodologies are written in 'Python' using the natural language processing toolkit (NLTK) [2] combined with Scikit-learn (sklearn) [3] tool.

Contribution 2

For meeting objective 2, we created three visualisations based on 'Streamgraph', 'Chernoff Faces' and 'Line Charts'. The Streamgraph visualisation is used to show topic popularity throughout the entire period of the debating system. Chernoff Faces can visualise the attitude of a user towards other users, as well as how other users interact with him/her. For example, the extend to which his/her arguments are in support or opposition, and the extent that this particular user agrees or disagrees with the arguments of other users. An example is shown in Figure 1.2.

[2] `http://www.cs.pitt.edu/mpqa`
[3] `http://archive.ics.uci.edu/ml/`



(a) Male Chernoff Face (b) Female Chernoff Face

Figure 1.2: (a) Visualisation showing the attidar of male user, (b) Visualisation showing the attidar of female user

Finally, the Line Chart is used as a means to provide a history of the most popular arguments made for a particular debate, allowing the user to find out the prevailing opinions on a matter quickly.

The underlying technologies for implementing the visualisations are 'D3.js', 'HTML5' and 'CSS3'.

Contribution 3

For evaluating the methodologies mentioned in Contribution 1 and Contribution 2, we used the debating website 'http://www.quaestio-it.com/'. The outcomes of this evaluation were extremely useful for providing a first hand experience on the effectiveness of the techniques used and their impact on debates for social networking websites.

1.3 Report Structure

In this report we will start by going through the background of automatic sentiment analysis and the visualisations that are to be implemented (Chapter 2), which will cover basic terminology that will be used throughout the report as well as the most frequently techniques used in this field. Furthermore, we will go through some of the related work that stands out and can be possibly adopted for the purposes of this project. Additionally there is a small section on the technologies used for implementing the classifiers and the visualisations. Finally, there will be a small discussion on the challenges that are associated with this project.

In Chapter 3, we will first cover the main points when constructing the corpora used for the classifiers. Then we will go through the implementation details for the two classifiers, the investigation process for selecting the most suitable combination of features and machine learning algorithms, and how they changed based on the accuracy on the training corpora.

In Chapter 4, we will go through the implementation process for creating the various visualisations and explain how they can be used in a debating system.

In Chapter 5, we will see how the classifiers and visualisations were evaluated on a new corpus and what conventions were used for each one of them. Additionally, there is a short discussion on their integration into the system.

Finally, in Chapter 6, we will go through what has been learnt through this project and what can be done in the future for improvement.

Chapter 2

Background

"Sentiment analysis or opinion mining refers to the application of natural language processing, computational linguistics, and text analytics to identify and extract subjective information in source materials"[5].

"**Natural Language** (also known as ordinal language) refers to any human language such as English, Greek or Italian which is casually used by humans for their everyday communication and arises in an unpremeditated fashion. They are differentiated from constructed and formal languages, such as programming languages and mathematical notations, since they have evolved throughout the centuries and as a result they cannot be formalised into specific rules." [6]

"**Computational Linguistics** deal with the statistical or rule-based modelling of natural language from a computational perspective. Some of the linguistic areas are computational semantics, pragmatics and sociolinguistics which use models for parsing and learning grammatical structure, models of communication, conversation, and dialogue and computational psycholinguistic models of human language comprehension and production." [7]

"**Text Analytics** uses a set of linguistics, statistical, and machine learning techniques that model and structure the information content of textual sources for business intelligence, exploratory data analysis, research, or investigation."[8]

Before going into detail about some of the techniques mentioned above, it is important to realise what the concept of emotion in written text is.

2.1 Deducing Emotion From Written Text

When studying emotions in a text we must ask ourselves two questions:

(a) How does the writer choose certain words and other linguistic elements to express their emotions?

(b) How does the reader interpret the emotion from a given text, and which are the linguistic methods used by the writer to transfer an emotion to the reader?

Therefore we are interested in mimicking, using computer technologies, they way humans deduce emotion in a text. Particularly, we need to investigate how particular linguistic methodologies are used to describe appraisal and action-readiness, as these two give the most clues for inferring emotion from text.

Appraisal denotes whether something is positive or negative, the significance of an event, the involvement of the own ego etc. According to Osgood and al, it has three dimensions.

1. *Positive* or *Negative* evaluation:
   The opinion of the writer for the topic of discussion is usually expressed through adjectives. It can also be expressed using specific words, and as part of a speech, as well as using discourse strategies, conversational techniques and word organisation patterns.
   e.g. "It was the most *rewarding* experience of my life".

2. *Power, Control* or *Potency* dimension:
   Identifying whether the writer associates or dissociates himself from the content of the sentence. This is sub categorised into the following:

   (a) *Proximity*:
   Linguistic elements showing whether the writer identifies himself or distances himself from the topic.
   "*Mrs Varnava/Mary* is the owner of this lovely cafe";

   (b) *Specificity*, Differentiate between an item being referred to a direct or indirect way.
   "I used *my/a* blue pan to write the note";

   (c) *Certainty*:
   Showing the certainty of the writer for the expressed opinion
   "It is *seemingly/absolutely* a nice picture".

3. *Activity, Arousal* or *Itensity* dimension, consists of emotional words for modifying the strength of the expressed emotion.
   "This is *undoubtedly* the best chocolate cake I have had in my life".

Action-Readiness refers to any mental or physical action which results to an emotion. This can be fairly obvious such as crying and laughing or more subtle like yawning during a lecture which is a sign of boredom.
"I could not stop *crying* as I watched her lying helpless on the hospital bed".

Sometimes the emotion is expressed in a more *direct* way, that is without using appraisal or action-readiness. Verbs and adjectives are typically used.
"I *was very happy* to see him again".

However, we quite often encounter emotions expressed in ways not described above like using irony or figurative language. That is why the techniques that tend to be frequently used emphasise on terms that denote judgement, appreciation or evaluation. This is particularly helpful for understanding emotion in reviews.

Following this, we will go through the fundamental techniques in natural language processing which are frequently used for classifying text.

2.2 Sentiment Classification Techniques

Sentiment classification techniques can be either based on a symbolic technique (also known as using manually crafted rules and lexicons), or a machine learning technique that constructs a classifier based on a training corpus.

2.2.1 Symbolic Techniques

2.2.1.1 Lexicon Based Techniques

This technique treats the text as a "bag-of-words" and does not take into account any relations between the words in the text.

Web Search

Turney's [9] approach was to use the Altavista search engine for classifying a text as positive or negative. First, he created tuples that consisted of combinations of adjectives and nouns, and combinations of adverbs and verbs, extracted from the text. Then, he issued two queries for each combination; one for returning the number of documents containing the tuple which is closest (less then ten words away) to the word 'excellent' and the other to the word 'poor'. If the first query has more results, then it is classified as positive and as negative otherwise.

WordNet

WordNet [10] database consists of nodes (words) connected by edges (synonym relations). This database is used by Kamps and Marx [11] in order to find out which is the emotional content of the word using the Osgood et al.'s dimensions (see section 2.1 Emotions in Written Context). In order to find the shortest path (least amount of edges traversed) between the word that needs classification and the positive and negative end of that dimension, they use a distance metric to compare the words found in WordNet. Depending on the evaluative dimension these words are "good/bad", "strong/weak", "active/passive", etc.

2.2.1.2 Sentiment of Sentences

When trying to classify a sentence, the relations between the words are very important and have to be taken into serious consideration. Mulder et al. [12] used an affective grammar which not only classifies a sentence as positive or negative, but also the topic this sentiment is directed at. To do so, a lexical and grammatical approach was taken.

2.2.2 Machine Learning Techniques

> *"Machine Learning is the study of computer algorithms that improve automatically through experience" [13]*

Before looking into some of the machine learning techniques that are currently used, it is important to understand the following features which are used for representing a document for classification.

2.2.2.1 Features

Feature is an "individual measurable heuristic property of a phenomenon being observed. Choosing discriminating and independent features is key to any pattern recognition algorithm being successful in classification. [..] This set of features and their values are stored in a feature vector". [14]

A common error with choosing a good feature set is trying to provide too many features based on the training set provided, which makes the resulting classifier prone to the idiosyncrasies of the training set and therefore fails to generalise well to examples that were not encountered before (i.e new to the classifier). The term used for describing this situation is *overfitting*.

Unigrams

This is known as representing documents as a feature vector with the elements indicating whether a word appears in the document. Each word has also a frequency of appearing in the text.

N-grams

A sequence of n-words from a given sequence of text or speech [15]. This allows the features to be a pair of words (bigrams), triples (trigrams) and more. This allows capturing of more context like the sequence of words "not ideal". The size of the feature vector size is usually limited by a frequency threshold number, and/or using a set of rules.

Lemmas

The lemma (basic dictionary form) of a word is used as a feature instead of the actual word from the text. For example best becomes good, dogs becomes dog and so on. The reason behind this idea is to make the features more general for easier classification of the document. However, sometimes overgeneralisation may occur and lose the details of the sentence. For example saying *"this is a good movie"* is not the same as saying *"this is the best movie"*.

Negation

This is an alternative way to n-grams for capturing negation in a sentence, proposed by [16]. When a negation appears in a sentence, e.g. "not", then a not is added to each of the following words. For example *"I don't want to go to the movies"* becomes *"I don't NOT_want NOT_to NOT_go NOT_to NOT_the NOT_movies"*.

Opinion Words

Opinion words are expressed through adverbs, adjectives, verbs or nouns that indicate positive/ negative opinion. By incorporating them into the feature vector, we can tell whether such words appear in the text or not. One technique to do so is by using a predefined lexicon. This was adopted by Wiebe and Riloff [17], who created an opinion word-list for combining this method with machine learning methods. The second technique is to identify words describing a feature of an item in a text [18].

Adjectives

Wiebe [19] observed that adjectives can express subjectivity in a document. Therefore he used only the adjectives of the document for its classification. Salvetti [20], added on this observation by using WordNet for enriching these feature vectors that include only adjectives, and found the synsets of adjectives and then used hypernym generalisation. This however had a decrease in accuracy due to the generalisation.

2.2.2.2 Machine Learning Algorithms

Learning algorithms can be either *"supervised"* or *"unsupervised"* depending on the availability of training examples. When training data are available and therefore the classes are known and finite then the algorithm is supervised. When no training data are available and the classifier relies only on test data then it is unsupervised.

In this project we will be working with supervised classifiers. The following Figure shows the framework used by supervised classification:

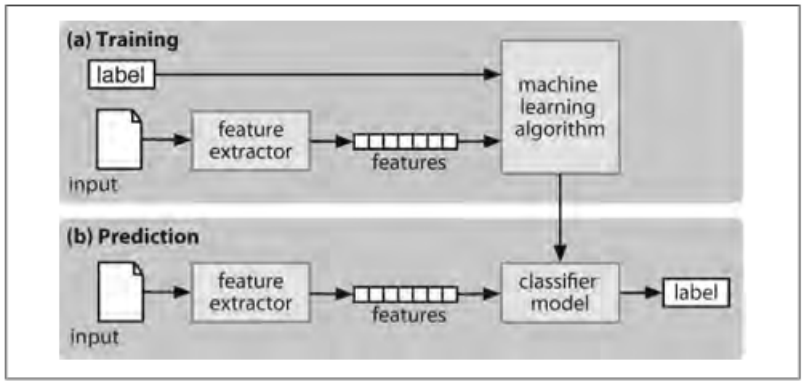


Figure 2.1: Supervised Classification (a) Training, feature extractor converts each input value to a feature set. Pairs of feature sets and labels are fed into the machine learning algorithm to generate a model. (b) Prediction, the same feature extractor is used to convert unseen inputs to feature sets which are then fed into the model which generates predicted labels. [2]

Since for this project we will be dealing with supervised algorithms let's see some of the most frequently used supervised methods.

Supervised Methods

(a) Support Vector Machines (SVM)
SVM use maximal Euclidean distance to construct a hyperplane of the distances to the closest training examples. More explicitly, this is the distance between a separating hyperplane and the parallel hyperplanes that represent the boundaries of the training examples used for each class. Having a maximal distance is believed to provide the best generalisation for the classifier. In the case where the data is not separable, the chosen hyperplane splits the data with as little error as possible. [1]

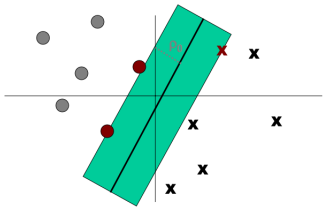


Figure 2.2: Support Vector Machines [21]

(b) Naive Bayes Multinomial (NBM)
A Naive Bayes classifier uses Bayes rule (for updating and revising believes when new evidence comes in) as its main equation, based on the naive assumption that each individual feature is independent and is an indication of the assigned class. This assumption is called conditional independence. A multinomial naive Bayes classifier constructs a model by fitting a distribution of the number of occurrences of each feature for all the documents [1].

(c) Maximum Entropy (Maxent)
This technique preserves maximum uncertainty. In order to do so, various models are constructed with each feature corresponding to different constraints. Among these modes, the one which has the maximum entropy satisfying the constraints is selected for classification. Therefore, all the assumptions are justified using available empirical evidence. [1].

(d) Decision Trees
Decision Tree Learning is a method for approximating discrete classification functions by means of a tree-based representation. It employs a top-down greedy search through the space of possible solutions. For constructing the tree it selects an attribute that has the highest *Information Gain* (i.e how informative an attribute is for classifying the training examples). This information gain is based on a measure called *Entropy*, which characterises the impurity of a collection of examples [13].

(e) Stochastic Gradient Descent (SGD)
Stochastic Gradient Descent is an under convex loss functions. When dealing with error functions we want to move in the direction opposite the gradient. Stochastic gradient descent chooses randomly a training example each time for avoiding local minima in order to minimise the squared error. [3]

(f) K-Nearest Neighbour (KNN)
K-Nearest Neighbour is an instance-based method which assumes all instances corresponding to point in the n-dimensional space. The nearest neighbours of an instance are defined in terms of the standard Euclidean distance.

(g) Nearest Centroid
In Nearest Centroid, each class is represented by its centroid, with test samples classified to the class with the nearest centroid.

(h) Logistic Regression
Logistic Regression is an approach to predicting a dichotomous outcome i.e. only two values. The goal is to determine whether each set of independent variables has a unique predictive relationship to the outcome, by converting the dependent variable to probability scores. The logistic function takes only values between 0 and 1.

2.3 Methods To Evaluate A Machine Learning Classifier

An important aspect of constructing a classifier using different techniques is finding ways to evaluate each one of them. We usually evaluate the performance of a classifier relative to the classifications that would have been made by a human expert. Since experts and impartial human judges are not usually available, we use test data of *gold standard*. This is a corpus which has been manually classified and it is used as a standard against which the classifications of the automatic classifier are assessed. The automatic classification is considered to be correct if it matches the stored value of the gold standard classifications. It is worth noting that the gold standard classifications were made by humans so it is possible that errors were made. Nevertheless, the gold standard corpus is by definition "correct".

A classifier is trained with training examples and tested on a new set of data, test examples. The reason for testing on new data set is that the model can memorise its inputs and would incorrectly give high classification rate. Training examples are usually further divided into training data and validation data. The validation data set is used for optimising the parameters of classifiers by performing error analysis which will indicate how to adjust the feature set.

In this section we will go through some of the basic evaluation concepts that will be used for evaluating the classifiers.

(a) **K-fold Cross Validation**
This is used when the amount of data for training and testing is limited and therefore we reserve, usually, 10% of the data for testing. To guarantee that the part retained for testing is representative, one may employ K-fold cross-validation. The data is split into K folds (parts) and only one is used for testing while the remaining k-1 folds are used for training. This process is repeated K times, using a different testing fold in each case. The total error estimate is the arithmetic mean of Error(D) obtained for each of K times of testing.

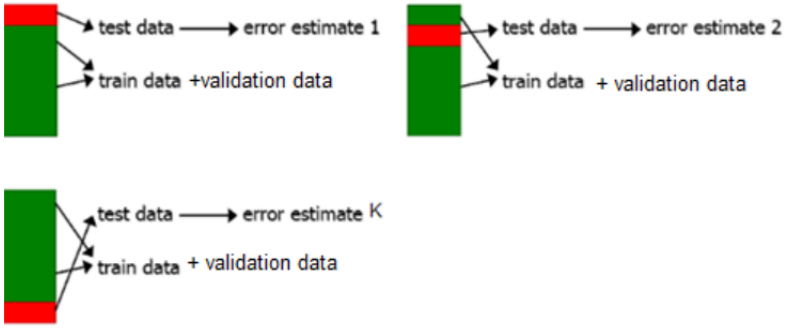


Figure 2.3: K-fold cross-validation process [24]

(b) **Confusion Matrix**
A confusion matrix is typically used as a visualisation tool to present the results attained by a learner. The columns of the matrix represent instances in a predicted class, and the rows represent instances in an actual class. For example, in the following confusion matrix, of the 12 actual samples of anger, the system falsely predicted 1 fear and 1 sadness emotion.

	anger	disgust	fear	happiness	sadness	surprise
anger	10	0	1	0	1	0
disgust	1	21	0	0	0	0
fear	1	0	5	0	1	0
happiness	1	0	0	23	0	0
sadness	1	0	0	0	10	1
surprise	0	0	0	0	0	23

Table 2.1: Confusion Matrix

(c) **Recall and Precision Rates**
For comparing two classifiers we calculate the Recall and Precision rates which measure the quality of an information retrieval process, e.g. a classification process.
Recall Rate describes the completeness of the retrieval. It is defined as the portion of true positives (TPs), i.e. the examples that were classified correctly by the process versus the total number of existing positive examples including the false negatives (FNs) not retrieved by the process i.e the examples that were classified incorrectly as members of the negative classes.
Precision Rate describes the actual accuracy of the retrieval, and is defined as the portion of the true positives (TPs) that exist versus the total of true positives (TPs) and false positives (FPs) i.e the examples classified incorrectly as members of the positive class.
Based on the recall and precision rates, we can justify if a classifier is better than another, i.e. if its recall and precision rates are significantly better.

$$\textbf{Recall Rate} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} \times \mathbf{100\%}$$

$$\textbf{Precision Rate} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}} \times \textbf{100}\%$$

(d) **$\mathbf{F_a}$ Measure**
F_a Measure combines the recall and precision rates in a single equation:

$$\mathbf{F_a} = (\mathbf{1} + \alpha) \times \frac{\textbf{precision} * \textbf{recall}}{\alpha * \textbf{precision} + \textbf{recall}}$$

where α defines how precision and recall will be weighted. If recall and precision are evenly distributed then $\alpha = 1$.

2.4 Classification Results When Using Different Methodolgies

An analysis was made by Boiy, Hens, Deschacht and Moens [1] for the techniques mentioned in the previous section.

Here are some of the classification results when using symbolic techniques:
Turney's method of using Altavista web search engine achieved an accuracy of 65.83% with a movie review corpus and 84.0% with an automobile review corpus. Kamps and Marx by manually constructing a list of the General Inquirer along the Osgood et al.'s dimensions, achieved an accuracy of 76.72% using WordNet. Pang et al. method of trying to manually construct an emotion lexicon had an accuracy of 64% (with 39% ties) compared to the 69% accuracy (with 16% ties) achieved using an automatically created lexicon.

Their research focused on performing experiments with machine learning techniques using a movie review corpus of 1000 positive and 1000 negative reviews and a corpus gathered from discussion boards, blogs and various websites containing 759 positive, 205 negative, 3527 neutral/junk examples.

For evaluating Support Vector Machine (SVM) and Naive Bayes Multinomial (NBM) methods, they used the WEKA [25] software, and for Maxent the OpenNLP [26] package was used. Also, different techniques for creating the feature set were used. Their results on the movie corpus are shown in the following table:

Features	***SVM***	***NBM***	***Maxent***
Unigrams	85.45%	81.45%	84.80%
Unigrams & subjectivity analysis	86.35%	83.95%	87.40%
Bigrams	85.35%	83.15%	85.40%
Adjectives	75.85%	82.00%	80.30%

Figure 2.4: Classification results for different learning methods and feature sets using a movie review corpus [1]

When training the classifier on a corpus extracted from blog, they used the most successful method derived from the first corpus. Their results were the following:

	Baseline NBM
accuracy %	84.25
precision/recall % for positive	64.52/49.93
precision/recall % for negative	88.48/72.96

Figure 2.5: Classification results on the blog coprus [1]

Another study performed by Pang, Lee and Vaithyanathan [16], for evaluating these machine learning techniques on the same movie review corpus had the following results:

	Features	# of features	frequency or presence?	NB	ME	SVM
(1)	unigrams	16165	freq.	**78.7**	N/A	72.8
(2)	unigrams	"	pres.	81.0	80.4	**82.9**
(3)	unigrams+bigrams	32330	pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	pres.	77.3	**77.4**	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	pres.	77.0	**77.7**	75.1
(7)	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
(8)	unigrams+position	22430	pres.	81.0	80.1	**81.6**

Figure 2.6: Average three-fold cross-validation accuracies [16]

The results from the studies show that machine learning techniques perform significantly better than symbolic techniques. Furthermore, when using a corpus that is extracted from blogs, which is also the case for the corpus used in this project, we observe a decline in the classification accuracy. This means that dealing with this kind of corpus is more challenging and therefore makes the classification process more difficult.

2.5 Related Studies For Sentiment Analysis

The following studies are directly or indirectly related to analysing text in debates:

2.5.1 Recognising Textual Entailment

Recognising Textual Entailment (RTE) deals with determining whether a given *hypothesis H* is entailed by a given *text T*. The following examples were taken from [2] and show a positive and a negative textual entailment:

Text: "Eating lots of foods that are a good source of fiber may keep your blood glucose from rising too fast after you eat."
Hypothesis: "Fiber improves blood sugar control."
(TRUE)

Text: "All genetically modified food, including soya or maize oil produced from GM soya and maize, and food ingredients, must be labelled."
Hypothesis: "Companies selling genetically modified foods don't need labels."
(FALSE)

One approach for recognising textual entailment described in the article "Natural Language Processing with Python" [2], which compares the text and the hypothesis treating them as two bags-of-words. Ideally, they expect that textual entailment exists when all the words contained in the bag-of-words for the hypothesis, can be found in the bag-of-words for text. Their feature set measures the word overlap as well as the amount of difference in the two bag-of-words. First, they filter out high frequency words such as *the, to, etc* and then they calculate the overlap and difference for regular words and named entities. Named entities are names of people, organisations and places and tend to be more important and therefore are weighted more than regular words. This approach has a classification rate of 58%.

2.5.2 Classifying A Text According To Its Polarity

Classifying a text according to its polarity is determining whether it expresses a positive, negative or neutral opinion on the topic discussed.

In the article *"From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series"* [27], a twitter corpus of 1 billion messages was used. A transparent, deterministic approach based on prior linguistic knowledge was adopted, counting instances of positive-sentiment and negative words in the context of a topic keyword using OpinionFinder. This however, had many falsely classified messages. One of the reasons was that a part-of-speech tagger was not used, resulting to words like *will* to be considered as positive using their noun form even when it was used as a verb. A better approach was to use a web-derived lexicon based on an informal social media dialect of English. The research showed that their technique of measuring sentiment was reasonably correlated to the measurement made by Gallup Daily and Michigan ICS and was able to peak up any downward or upward trends in emotions.
This is shown in Figure 2.7:

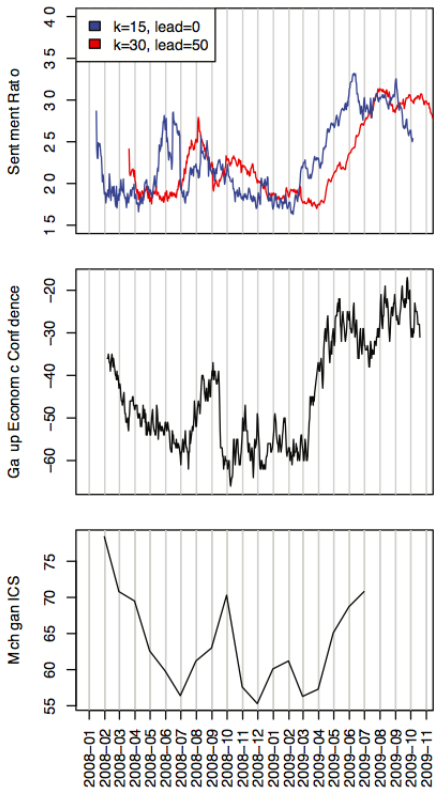


Figure 2.7: Sentiment ratio and consumer confidence surveys. Sentiment information captures broad trends in the survey data [27]

Another approach to predicting polls with lexicons was the one by Lindsay [28] on a proprietary corpus of Facebook posts. First a corpus of 5000 tagged posts labeled positive, negative or neutral was manually created. Then synonyms for sentiment words were gathered depending on their neighbouring words e.g I "hate/love" you. and then these sentiment words were again manually labelled. Having created a lexicon and a corpus, several techniques were used like counting the number of sentiment words, simple tokenization schemes, negation heuristics and feature selection giving them more than 80% precision (and a very low recall). This approach was tested at how it predicts election polls by correlating with the results of the polls conducted during part of the 2008 presidential election. The correlation was pretty good a few days before the poll came out and totally unpredictive a few days later (only 30% - 40%).

Another study by Gilbert and Karahalios [29] showed how estimating emotions from weblogs can provide information about future stock markets, by measuring anxiety, worry and fear from a dataset of over 20 million posts. Their first classifier, a boosted decision tree, was made from a corpus of already labelled posts (anxious, happy, angry, confused, relaxed, etc), and uses a feature set of the first 100 most informative words used as anxious or not anxious indicators. The second classifier was built on a bagged Complement Naive Bayes algorithm and it uses 46,438 word from the mood corpus as feature set. Their classification results were rather low, only 28% and 32% respectively.

An interesting approach was shown by Pimenta et al. [30] for calculating polarity of blog posts. The idea was to take into account an aggregation of a set of documents instead of each individual document. Using POS-tagging for counting the polarity of different POS tags using SentiWordNet [31] together with considering the contextual valence shifters on each word, gave higher classification rates than using the association approach (measuring association with the Pointwise Mutual Information using search engine queries).

A well known study by Pang & Lee [32] for determining sentiment polarity for movie reviews is by using text-categorisation techniques on the subjective parts of the documents which are extracted using the minimum cuts in graphs approach. Their procedure was to first trim out the objective content and then using a support vector machine classifier, trained on a objective-subjective dataset of sentences, to determine their polarity. The trimming of the text was done by first placing each sentence of the text as a node on a graph along with a positive and a negative node. Then, they measured the weight of the edges between sentence nodes and the positive and negative nodes. Next, they assigned scores to edges between sentences by their proximity within the review and finally, using the minimum cut on the graph they removed the objective content from their reviews. This approach had a high classification rate of 90%.

2.5.3 Determining Support Or Opposition

Being able to identify support or opposition is very much related to the purposes of this project.

An approach by Thomas, Pang and Lee [33], focuses on very general types of cross-document classification preferences, and only used as constraints the identity of the speaker and any direct textual references between statements. This indicated that the integration of inter-document relationships increases accuracy. Their corpus was extracted from GovTrack and had many conventions that enabled them to identify speaker, post in reply to, label etc. Their method combines the use of support vector machines for considering each text in isolation, and exploitation of various relationships between speech segments belonging in the same debate using a weighting function that associates speech segments.

A different approach to recognise opposition, has been adopted by new studies, e.g. Yu et al [34], which rather than classifying isolated positions, the classifier tries to find out what is the underlying ideology of the speaker. This was particularly helpful in professional political speeches when the speakers tend not to use expressive language which will indicate an emotion. The belief is that ideology influences the individuals and therefore their views on different issues will tend to be based on the underlying ideology. This is an innovative approach but unfortunately not very helpful for this project since we will be dealing with text from blogs and therefore extracting the ideology of the speaker is not at all applicable.

Martinau and Finin [35] used Support Vector Machines for showing how using the Delta TFIDF (term difference-inverse document frequency) for weighing words gives better classification results. Feature values for the document are created by calculating the difference of a word's TFIDF score for positive and negative training corpora. They tested their approach against the movie review used by Pang & Lee (minimum-cut approach) for sentiment

classification giving a statistical significance of 95% on a two tailed t-test. When tested on determining support or opposition it performed relatively higher than using complex techniques such as party affiliation information, or joining texts from same speaker, or manual co-refference on the named entities.

Somasundaran and Wiebe [36] tried to combine utilising sentiment and arguing opinions in order to classify a text either as supporting or opposing a statement. They created an arguing lexicon from a manually annotated corpus and used features that encode what the opinion is about. Their experiments were made on four different ideological debates. Their classification results had reached up to 63.93% when trained on multiple topics and 70.59%, 63.71%, 60.55%, 63.96% when trained on one topic at a time.

2.5.4 Detemining Agreement Or Disagreement

Being able to automatically detect whether a response to a specific quote agrees or disagrees with it, is a feature that we are looking into incorporating into the classifier.

An approach to do so, followed by Lee et al [37] is treating classification as seeking minimum cuts in the appropriate graphs. This is appealing to the agreement-disagreement classification since we can incorporate preferences among pairs of instances in a provably tractable way which ensures efficiency.

An approach by Galley et al. [38], describes a statistical model for identifying adjacency pairs and deciding if an utterance expresses agreement of disagreement with its pair. Using maximum entropy ranking based on a set of lexical, durational, and structural features that look both forward and backward in the discourse, they classified utterances as agreement or disagreement using these adjacency pairs and features that represent various pragmatic influences of previous agreement or disagreement on the current utterance. Their approach gives an accuracy of 86.9%.

Clint Burfoot [39], shows how using multiple sources of agreement information can be used to classify sentiment. This is done by combining per-document classifications with information about agreement between documents, taking advantage of a minimum-cost cut graph. The idea of same-speaker agreements is that speeches by a given speaker are linked and receive the same final classification. Also, taking into account how a speaker refers to another speaker in the discussion indicates whether he/she agrees with him/her. This is done using an SVM classifier on the tokens immediately surrounding the reference. Additionally, as seen by Yu et al, they rely on the ideology of the person for strengthening or weakening the agreement. Furthermore, they measure the extend to which a pair of speakers use similar words to describe their positions. They tried different combination of the above techniques with a maximum classification rate of 85% on the development set and 80.5% on the test set.

2.5.5 Summary Of The Approaches Taken In Previous Studies

The following table summarises the approaches described above that yielded high classification results.

Detecting	**Technique (up to)**	**Classification rate**	**Corpus used**
Support/ Opposition	Support Vector Machines using bag of words	83%	Congressional floor-debate transcripts
	+ human supplied annotation/ rationals, appraisal groups/ minimum cuts	90%	
	+ relationship between speech segments	88%	
	+ ideology	85%	
	+ Delta TFIDF (on one topic only)	95%	
	+ Utilising sentiment and arguing opinions (overall)	64%	Political and Ideological Debates
	(Guns Rights)	71%	
	(Gay Rights)	64%	
	(Abortion)	61%	
	(Creationism)	64%	
Agreement/ Disagreement	Adjacency pairs using maximum entropy ranking based on lexical, durational and structural features	87%	Congressional floor-debate transcripts

Table 2.2: Summary of classification results

Based on the above results and the nature of the corpora that we will be working with, the techniques that are applicable and can be used, are SVM with Delta tf-idf, Utilising sentiment and arguing opinions, and using local lexical features. There is no information on the speaker's ideology, or any other relationships between speech segments and therefore cannot be used in this context.

However, as the corpus is different we cannot rely only on using the above techniques. The exploration will incorporate different supervised methods (see Section 2.2.2.2), as well as symbolic techniques (see Section 2.2.1) and Machine Learning Techniques (n-grams, lemmas, negation, adjectives) for extracting feature sets.

Furthermore, we decided not to use the approach followed by Somasundaran et al, of Utilising sentiment and arguing opinions, as their corpus is a subset of the corpus we will be using and there is no point in repeating their research. Also, the results were not very good, since the simple use of unigrams provides almost the same classification results with their approach.

2.6 Programming Language And Toolkits Used

The programming language used for experimenting with the various techniques for Support/ Opposition and Agreement/Disagreement classification, is 'Python'. Additionally, Natural Language Toolkit (NLTK) was used for manipulating strings for feature extraction, as well as providing useful implementations to assist Part of Speech Tagging. Extremely helpful was Scikit-learn for machine learning in Python.

Python

After exploring different alternative choices, we decided to use Python[1], since it has a powerful functionality in processing linguistic data, particularly for handling strings. It also has support for graphical programming, numerical processing and web connectivity.

Someone familiar with other programming languages can easily adapt to the syntactic rules of Python as it is considered to be a language with a shallow learning curve.

There are many available machine learning tools for Python, including NLTK and Scikit-learn. As they are very important for implementing this project, it would be much easier to use these machine learning components when writing the program in Python. Installing and accessing their source code is relatively easy and they have good documentation, guiding you through examples and helping you getting started.

NLTK

The Natural Language Toolkit (NLTK)[2] is an open source Python library for Natural Language Processing. There is a very good guide for using the toolkit [3] and familiarising the user to the techniques used; making it ideal for people who are new to machine learning area. It also includes many corpora allowing the user to experiment with the techniques and approaches mentioned in the book.

Scikit-learn: Machine Learning In Python

Scikit-learn [4] is useful for integrating machine learning algorithms in the "tightly-knit scientific Python world" (numpy[5], scipy[6], matplotlib[7]). It provides tools for data mining and analysis in a simple and efficient way making it easily accessible and reusable.

[1] http://www.python.org/
[2] http://nltk.org
[3] http://nltk.org/book
[4] http://scikit-learn.org/stable/index.html
[5] http://www.numpy.org/
[6] http://scipy.org/
[7] http://matplotlib.org/

2.7 Visualisations

As we have mentioned already, one of the objectives of this project is to create visualisations that will show quickly an overall picture of the debate so that users will be able to make observations easily. In order to do so, we plan to make use of Chernoff Faces, Streamgraphs and Line Charts. Line Charts are widely used and therefore no particular explanation is needed; but what exactly is a Chernoff Face and a Streamgraph?

2.7.1 Chernoff Faces

Chernoff Faces were invented by Herman Chernoff for "displaying data in the shape of a human face. The individual parts, such as eyes, ears, mouth and nose represent values of the variables by their shape, size, placement and orientation. The idea behind using faces is that humans easily recognise faces and notice small changes without difficulty". [4]

An example of how they can be used is shown in Figure 2.8, which indicates, by varying the characteristics of the faces, the measurements of skull and teeth, on human races, apes and fossils:

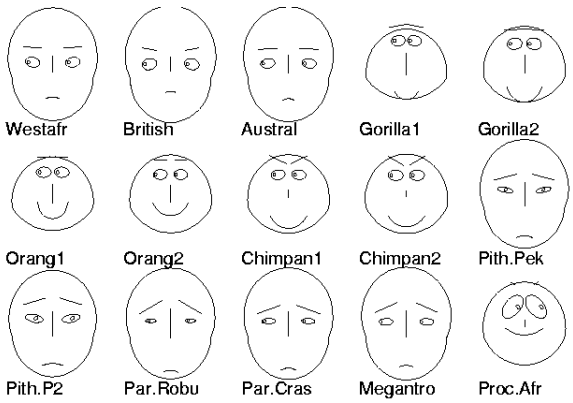


Figure 2.8: Skull and teeth measurements on human races, apes and fossils

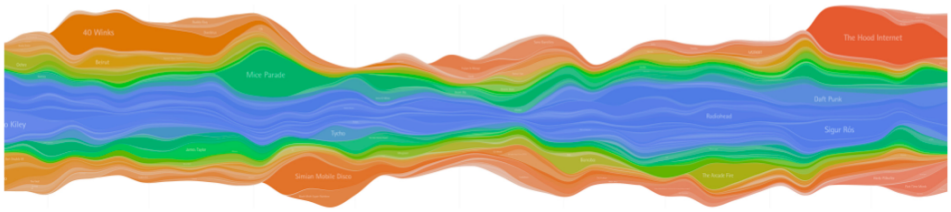
We decided to use Chernoff Faces to visualise the user's attitude and how the user's emotional status changes depending on how the other users respond to his/her comment.

2.7.2 Streamgraphs

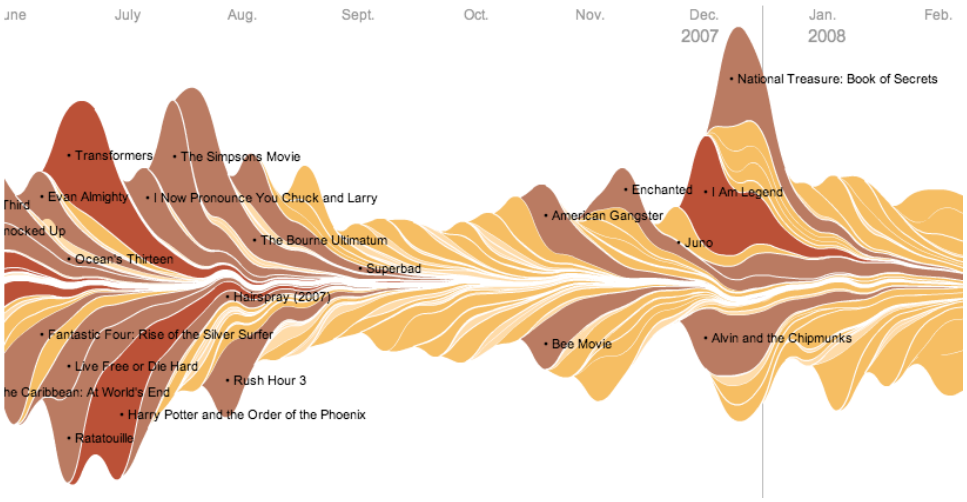
Streamgraphs [40] are used to visualise underlying trends. They are preferred over traditional stacked bar charts as they can emphasise on the legibility of each layer, by arranging the layers in a distinctively organic form.

They were first applied to last.fm data for showing the listening histories of users. The response on the users was very positive and therefore drew the attention of New York Times. NYT, implemented an interactive version of the streamgraph for showing the box office revenue for

7500 movies over the past 21 years. The two visualisations are shown in Figures 2.9 (a) and 2.9 (b) respectively.



(a) Listening History streamgraph



(b) Box Office Revenue Streamgraph

Figure 2.9: (a) Visualisation showing the Listening History streamgraph, (b) Visualisation showing the Box Office Revenue Streamgraph

In order to understand the streamgraphs one must realise that each layer represents a different artist/movie and that x-axis shows time. Additionally using the colour of the layers, different parameters can be shown, based on two dimensions, saturation and hue. For example in Figure 2.9 (a), saturation shows the number of times an artist was listened and hue shows the earliest time that a particular song from that artist was listened. Figure 2.9 (b) shows a simplified version of colouring the graph, where the colour becomes darker when the total domestic gross increases.

As these two graphs, can easily show the underlying trends for listening and movie preferences of users, we decided to use them for showing trends in debating preferences of users.

2.7.3 Technologies Used

When choosing the technologies for implementing the visualisations, we had in mind the following two aspects:

- The debates are to be accessed through websites.
- The visualisations will need to be interactive and in capable of dynamically changing when there is an update on the debate (i.e new argument, new reply, votes etc).

Therefore the main technology used is d3.js with HTML5, Javascript, jQuery and CSS3.

2.7.3.1 D3.js

"**D3.js**[8] is a small, free JavaScript library for manipulating documents based on data. It allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document."

It is used for efficient manipulation of data in a flexible way using technologies such as CSS3, HTML5 and SVG. It is very fat and can handle big datasets including transitions and interactions.

2.7.3.2 HTML5

HTML5 is used for implementing web pages by creating text files that can be read by web browsers. It is used for structuring and presenting the required content on the Internet.

2.7.3.3 Javascript

A scripting language developed by Netscape to for adding interaction to websites by incorporating user interface controls and user friendly features.

2.7.3.4 jQuery

As defined in 'Webopedia' [41], jQuery is a free and open source JavaScript library that is used by Web developers to navigate HTML documents, handle events, perform animations and add Ajax interactions to Web pages.

2.7.3.5 CSS

Cascading Style Sheet or CSS allows web designers to form the document presentation by changing the layout, colours and fonts [42].

[8] `http://d3js.org/`

2.8 Open Questions And Challenges

2.8.1 Challenges Associated With Implementing A Classifier For A Debate

Natural Language Processing, is considered to be a complicated task. Having to deal with a relatively new area like support/opposition and agreement/disagreement, makes it even more complicated and therefore there are certain challenges that are particularly difficult to overcome.

2.8.1.1 Training Corpora

The classifiers to be implemented will try to pick up patterns that distinguish a supporting and an opposing statement (or agreement/disagreement). These patterns will be constructed based on the training corpora. Therefore, the training corpora have to consist of a good indication of what might come up in a debate. Additionally, they has to be annotated correctly since, otherwise, the patterns extracted will be completely rubbish. Finding such corpora, that have already been annotated, is not an easy task as there are not any corpora available that use microblogging data from debates. In order to create a good enough corpus certain conventions were taken and are explained in Section 3.1.

2.8.1.2 Dealing With Different Corpora

Different corpora will have different patterns that show support/opposition or agreement/ disagreement. For example, in an abortion debate, using the word 'fetus' instead of 'baby' is an indication that probably this statement is pro abortion. However, these words make no sense when talking about gay rights, quality of a product etc. Having a corpus that includes all possible topics of discussion is nearly impossible. Consequently, the classifier needs to be able to provide a good estimate based on the corpus provided, or a subset of the corpus provided.

2.8.1.3 Stance Of Speaker Not Clearely Stated

Sometimes, the position a person is taking is not clearly stated. For example, consider the following sentence:

"If a fetus had a right to life, abortionists would be subject to murder charges. While abortionists claim that fetuses should have a right to life, they would never go so far as to charge abortionists with murder. Yet, this is what would be required if we gave fetuses a right to life. Therefore, there is a fundamental inconsistency in this position."

This text was posted in a discussion about being favour of abortion or not. This statement does not clearly specify his/her support of abortion (i.e. it does not specifically say abortion should be allowed), yet it is implied from this text. Training a classifier to do so will be therefore quite a challenge.

2.8.1.4 Neutral Text

Another challenge is dealing with neutral text. Some people who have not yet settled on a particular side of the topic, tend to express their thoughts saying both positive and negative points. Therefore, we cannot really tell whether that particular person supports or opposes the discussion topic.

2.8.1.5 Inconsistencies

Furthermore, there are cases when the majority of the vocabulary used misleads the overall opinion of the speaker:

"The main actress' performance was unbearable and the special effects were horrible but I enjoyed the movie nonetheless."

This sentence consists of mainly negative points for the movie but it should be classified as support.

2.8.1.6 Classifying Multiple Sentences

Related to the above example, is that the techniques developed are usually classifying a single sentence. For detecting the topic-sentiment relation in text, coreference resolution needs to be applied across sentences. Taboada and Grieve [43] found out that the opinion is most likely to be expressed in the middle and the end of a text so they weigh the text differently.

2.8.1.7 Making People Conformant To The Rules Of The Debate

A particularly important challenge for this project is making people to confront to the rules of the debate so that the text is of high quality and there is not too much noise. Problems related to this will be further discussed when trying to build the corpus.

2.8.2 Challenges Associated With Implementing Visualisations For A Debate

When creating a visualisation, there are various pitfalls that have to taken into serious consideration and have to be avoided as much as possible.

2.8.2.1 Misleading Information

Sometimes it is possible for the visualisations to show a more surreal/meretricious version of what the data actual show. For example they may exaggerate and make a big deal out of something, when actually there should not be an issue. For example showing a high negative classification for a debate when in fact only one person has replied.

2.8.2.2 Ambiguities In Interpreting The Information Shown

When the visualisation is not carefully designed, it is possible to provide ambiguities to the user. For example when not clearly explaining what each part of the visualisation is representing, or when colours and sizes are not representative of the actual data.

2.8.2.3 Making Too Complicated Visualisations

It is not a rare situation when somebody overambitious tries to include as much information and details as possible to the visualisation. This makes it very difficult and hard for the user to understand and usually instead of speeding up the process of consuming information, ends up wasting someones time.

2.8.2.4 Prerequicities For Understanding The Visualisation

When someone is very familiar with a certain topic, tends to forget that other people may not be so familiar and what he/she thinks is trivial, actually is not. Therefore, when implementing a visualisation, all the features have to be clearly explained so that users with no previous experience with visualisations are able to understand it.

Chapter 3

Analysis Of Techniques For Sentiment Analysis In Debates

In this section we will go through the different corpora used and explain how they were adjusted to meet the requirements of this project. As the two classifies have different criteria about how their corpora should be, we created two different corpora, one for each classifier. First we will go through the corpora used for the Support/Opposition Classifier and then through the ones used for the Agreement/Disagreement Classifier. Following that, a thorough explanation of the various combinations of techniques used for developing the classifiers will take place. This includes analysis on the choice of features sets for training the classifier, and the choice of machine learning algorithms such as Support Vector Machines, Linear Regression, Naive Bayes, etc when triggered with different parameters. The analysis will first be made on the Support/Opposition Classifier and then on the Agreement/Disagreement Classifier.

3.1 Building Corpora For Training And Testing The Classifiers

As we have already mentioned, two different corpora were created, one for each of the two classifiers. Building the corpora was a rather challenging task since there are not yet any corpora that are based on debates from microblogging websites. Even if text size was not an issue, there are still not many available corpora that can be used to train classifiers for debates. Therefore, for each of the two corpora, certain modifications had to take place before being fed to the classifiers for training. Additionally, we used techniques for making the text conformant to certain syntactic rules like using the lemma of each word and correcting spelling mistakes. In order to make sure that these techniques were helpful, we tested it on an existing classifier for Recognising Textual Entailment.

3.1.1 Building A Corpus For Training And Testing The Support/Opposition Classifier

For the corpus to be used for the Support/Opposition Classifier, we needed texts extracted from social networking sites that incorporate the use of microblogging data. Researching through the corpora used in academic community was disappointing as we could not find corpora that meet these criteria. Therefore, we tried to manually build one using data from Twitter. As this was unsuccessful, we then turned to the corpora used by Somasundaran and Wiebe [36] and an amazon review corpus. In the following sections we will go through the failing attempt to use Twitter and the amendments that had to be made to the corpora to meet the criteria of this project.

3.1.1.1 Using Twitter

Our initial attempt was to use Twitter posts for creating a corpus, since it provides millions of posts everyday with many different topics, thus giving possibilities to create a big corpus with variety in its data set. Additionally, we would be able to extract data for both classifiers as it allows arguments, and replies to arguments to be posted for a particular debate.

To do so, we decided to first extract posts made by Obama as it gives seed to many debates. For collecting the data we used twitter API and given the *id_str* we were able to extract the replies made for a particular post and to create a tree of comments and replies and store the results in an xml file using Python. This would have been particularly helpful since having a particular hierarchy of comments and replies is much similar to the style of the debate system for which this tool is being developed for. Furthermore, we would be able to exploit various relationships between posts as pointed out in section 2.6.

Unfortunately, after creating the xml file, we realised that not many people confronted to the rules of debating and there was too much noise in the data set. To give you an example:

Barack Obama : *"This law is just one step in the broader effort to strengthen our economy and broaden opportunity for everybody." —President Obama*
IamRafaelBraga : *Follow me! (:*

Therefore, we had to abandon this corpus since it would have been too difficult to find a way to delete all the noise and still have a big enough corpus.

3.1.1.2 Using Political And Ideological Debate Corpora

Our second attempt was to make use of the corpus[1] used by Somasundaran and Wiebe [36], which included debating posts about 'Abortion', 'Creationism', 'Gay Rights', 'Existence of God', 'Gun Rights' and 'Healthcare'. This would again give a variety in topics discussed, making it more flexible in a new topic being presented to it in the future. Each post file included the following:

[1]MPQA corpus available for download at `http://www.cs.pitt.edu/mpqa`

#*stance* = *stance*1
#originalStanceText=Yes
#originalTopic=In favour of Abortion
"Legal abortion protects women with serious illnesses that are vulnerable. Tens of thousands of women have heart disease, kidney disease, severe hypertension, sickle-cell anemia and severe diabetes, and other illnesses that are made worse by childbearing. Legal abortion helps women avert these unavoidable risks to their health and lives."

This at a first glance seems perfect for the purposes of this project. So we created an xml file using these corpora using as hypothesis the original topic and as text the remaining text. The above hypothesis-text pair was converted to have the following format in the xml file:

<pair entailment="YES" id="4" task="IE">
<t>
"Legal abortion protects women with serious illnesses that are vulnerable. Tens of thousands of women have heart disease, kidney disease, severe hypertension, sickle cell anaemia and severe diabetes, and other illnesses that are made worse by childbearing. Legal abortion helps women avert these unavoidable risks to their health and lives."
</t>
<h>
In favour of abortion.
</h>
</pair>

3.1.1.3 Amendments Made On The Corpora Characteristics

Before using the corpus to train the classifier, we had to make sure that it was good enough to be used for the classification purposes of this project.

We tested it against the RTEClassifier provided by the python NLTK for recognising textual entailment. The classifier's success rate was supposed to be 58% but this only gave classification rate of almost 30%.

Therefore, we made some changes to the posts. For example if the hypothesis was *"debate abortion"* we would change it to *"In favour of abortion"*. The corpora had many different topics of discussion, so we used five topics of the dataset for training and testing the classifier giving a classification rate of 58%. Then we tested it against a new corpus, (the remaining sixth topic) which gave an improved result of 50% (507 out of 1000), compared to the 30% that was achieved before.

Conforming To Syntactic Rules

Looking at the corpus data, we realised that there are still some improvements that could be made that could possibly give a higher classification rate. For example, the text and the hypothesis have to confront to the same syntactic rules.

Since this classification method compares the text and the hypothesis treating them as two bags-of-words, we had to make sure that any syntactic mistakes were kept to a minimum.

Therefore we used the PyEnchant spellchecking library for python which is based on the Enchant library. Enchant library provides spell checking (i.e whether a specific word appears in the specified dictionary) for various languages, and additionally can suggest a list of words in case the spell checker returns false. Figure 3.1, shows how PyEnchant was used to spellcheck a word in the text:

```
>>> import enchant
>>> d = enchant.Dict("en_US")
>>> d.check("computer")
True
>>> d.check("computre")
False
>>> d.suggest("computre")
['computer', 'compute', 'computers', 'computed', 'computes', 'compare', 'compere
', 'compete', 'compote', "computer's"]
```

Figure 3.1: PyEnchant Library in use

For correcting a word that is not in the dictionary, we always replace it with the first word from the list of suggested words. If the list is empty, we keep the original word. When looking at the resulting corpus, we realised that any punctuation was completely lost. A word at the end of a sentence which is followed by ‘,’, ‘.’,‘?’,‘!’ etc does not pass the check method and gets replaced without the punctuation. This however, does not cause any problems at this point so it is was not considered to be a problem. An issue of the resulting corpus was when words were separated with a ‘-’. For example, *self-defence* was replaced by *slenderness*. It was therefore necessary to replace all non alphanumeric characters, that separate words, with an empty space.

Using Lemmas

As suggested in the section 2.2.2.1.3, it is better to use the lemma of a word, i.e its basic dictionary form and so we had to take into consideration the morphology of each word.

Morphology of a word is the small meaningful unit that makes up the word. Each word consists of the following:

1. Stems: core meaning of the word
2. Affixes: bits and pieces that adhere to stems

For example the word stems itself has as stem the word *stem* and as affix the letter *s*.

For our corpus, we decided to perform the stemming procedure of cutting off the affixes of each word, in order to have the same morphology for all the words in the text and the hypothesis.

For doing so, we used the stemming 1.0 algorithm provided for python, which implements the Porter, Porter2, Paice-Husk and Lovins stemming algorithms for English.

```
>>> from stemming.porter2 import stem
>>> stem("functionally")
'function'
```

Figure 3.2: Stemming 1.0 in use for the word 'functionality'

Minimising Size Of The Text

Another problem with this corpus was that sometimes some of the texts were quite large (more than 1000 characters). So, we had to check the length of each text and decide whether to keep it in the corpus or not. The reason for doing that, is that when training with larger texts, there is the possibility to give bad classification results when testing on smaller texts since people tend to use different vocabulary when they have a limitation on how much they can write.

The classification results, based on the amended corpora, were the following:

```
Training classifier...
  ==> Training (100 iterations)

      Iteration    Log Likelihood    Accuracy
      ---------------------------------------
             1          -0.69315       0.601
             2          -0.64313       0.601
             3          -0.63570       0.601
             4          -0.62970       0.602
             5          -0.62482       0.604
             6          -0.62081       0.610
             7          -0.61750       0.611
             8          -0.61473       0.612
             9          -0.61241       0.612
            10          -0.61043       0.616
            11          -0.60875       0.616
            12          -0.60730       0.615
            13          -0.60605       0.615
                           ⋮
            88          -0.59530       0.607
            89          -0.59529       0.607
            90          -0.59529       0.607
            91          -0.59528       0.607
            92          -0.59528       0.607
            93          -0.59527       0.607
            94          -0.59527       0.607
            95          -0.59527       0.607
            96          -0.59526       0.607
            97          -0.59526       0.607
            98          -0.59525       0.607
            99          -0.59525       0.607
         Final          -0.59524       0.607
Testing classifier...
Accuracy: 0.5445
639
```

Figure 3.3: Recognising Text Entailement using python NLTK

As expected, the classification results were improved, getting 639 out of 1000 correct.

3.1.1.4 Extending Support/Opposition Corpus With Casual Topics

Having created this corpus, we realised that even though it has a variety in the topic of the debates, they all constitute of serious matters for which people tend to be very passionate about. Therefore, the vocabulary used will be very different compared to a discussion about the quality of a car battery for instance. Consequently, we had to extend the current corpus with topics that are casually discussed. After an exhaustive search to find corpora that meet the criteria of our training corpus, we were able to find a big enough amazon corpus [2] with reviews about the battery life of the kindle, ipod, the accuracy of a gps, the price of a hotel etc. From this corpus, we only kept texts that were up to 200 characters and we had to manually classify it, since unfortunately it was not annotated, creating an additional corpus of about 1000 comments. For creating the final xml version of the corpus, we applied spellchecking and transformed the words to their lemma form.

3.1.2 Building A Corpus For Training And Testing The Agreement/ Disagreement Classifier

The corpus we had at the moment, even though it was good for deciding whether the text supports or opposes a stance, the 'text' and the 'hypothesis' cannot be used as a argument-response pair for training a classifier to check for agreement or disagreement. Therefore, for this kind of classification, a new corpus had to be created.

As with the previous corpus, it was difficult to find an annotated corpus that meets our criteria (relatively short text, conversational structure, as little noise as possible). The closest we could find was a corpus extracted from the Internet Argument Corpus (IAC) created by Abbott et al [44]. The material included in this corpus was extracted from the online debate site 4forums.com. By correlating the information from four different csv files, we extracted a good enough labelled corpus with quote-response pairs that could be used for detecting agreement or disagreement.

For creating the final xml file with the argument-reply pairs, we followed the same procedure with the previous corpus (spell checking, removing non alphanumeric characters etc), except from transforming each word to its lemma form. The reason for not using their lemma form will be explained in the analysis Section 3.3. A typical argument-response pair in the xml file has the following format:

```
<pair disagreement="FALSE" id="4" topic=" abortion">
    <quote>
        The anti-abortionists claim that would be killing a human life to save one.
    </quote>
    <response>
        The anti-abortionists claim a load of **** on many issues. I don't listen to them.
        To put the "life" of an unfertilized egg above that of a person is grotesquely sick
        IMO. I support any such stem cell research wholeheartedly.
    </response>
</pair>
```

[2] `http://archive.ics.uci.edu/ml/`

3.1.3 Summary Of Proposed Corpora For Training And Testing The Classifiers

When trying to implement the corpora for the two classifiers, we were looking for existing corpora that would be able to classify data from social networking sites that use microblogging text. Therefore they had to meet the following characteristics:

(a) Extracted from debates
(b) Relatively short text (no longer than 200 characters)
(c) Not restricted to formal speech
(d) With as little noise as possible

For the Support/ Opposition Classifier, we needed a corpus that would cover a variety of topics so that it would be more likely to classify correctly data from any given corpus. The resulting corpus consisted of political and ideological debates from MPQA corpus that covers data on 'Abortion', 'Creationism', 'Gay Rights', 'Existence of God', 'Guns Control' and 'Healthcare' as well as reviews extracted from amazon on casual topics like Quality of hotels, cars, mobile phones etc.

For the Agreement/Disagreement classifier, we needed a corpus that has an argument-reply format. The corpus used was made from the Internet Argument Corpus, which uses data from debates in 4forums.com. The resulting corpus was constructed by correlating data from different files in order to form argument-reply pairs.

3.2 Analysing Techniques For Support/ Opposition Classification

With a good enough corpora to train a classifier for detecting support or opposition, we were now able to experiment with different feature selection techniques and classification algorithms.
Before getting started, we have to understand how this is different to classifying the polarity of an argument. One might think that when an argument is negative, it must therefore oppose the initial statement or vice versa. Unfortunately, this is not the case, as we can see in Figure 3.2 using a polarity classifier:

Figure 3.4: classifying arguments on abortion topic using sentiment polarity

As you can see, both statements are considered to be negative. This is because, in both cases, the vocabulary used is considered to have negative polarity (abortion on its own is a strong negative word). Therefore, words that are usually used to introduce negative orientation in a text, can be used as supporting arguments; and words which are considered to have neutral orientation, can be supportive or opposing for the topic in discussion. It is more of a matter to find out what vocabulary is used to support a specific topic and what is used to oppose it.

The following section goes through the procedure we followed for targeting the problem of finding representative features for each class. The first step includes experimenting with feature selection methods (tf, idf, tf-idf, Delta tf-idf, n-grams). Then we experiment with different parameters and machine learning algorithms when trained on the entire corpus. Following this, the experiments will be made on a single topic at a time and finally various patterns of POS tagging will be incorporated for investigation.

3.2.1 Feature Selection

In section 2.2.2.1, we have described several methods for choosing features from the training corpus. The aim is to create discriminating and independent features that would give high classification results. In order to find out what the best combination for our corpus is, we experimented with different tokenization methods.

For the experiments we used three different n-grams: unigrams, bigrams and trigrams, and four different feature weighting methods: TF, IDF, TF-IDF, Delta TF-IDF.

For example, given the following two sentences as corpus (ignoring stopwords and stemming of words), the feature set that corresponds to each training sample is calculated below:

corpus: "The movie is the best", "This movie is not the best"

(a) **Using unigrams:**

['The', 'movie', 'is', 'the', 'best'], ['This', 'movie', 'is', 'not', 'the', 'best'].

(i) **Term Frequency (TF)**: refers to the term frequency of each feature in the sentence it belongs to.

"The movie is the best"
feature set: [('The', 0.4), ('movie', 0.2), ('is', 0.2), ('best', 0.2)]
"This movie is not the best"
feature set: [('This', 0.17), ('movie', '0.17'), ('is', 0.17), ('not', 0.17), ('the', 0.17), ('best', 0.17)]

TF tends to scale up frequent terms and scale down infrequent terms. This is an issue since the infrequent terms are usually more informative.

(ii) **Inverse Document Frequency (IDF)**: For each term, it is calculated by the logarithmic value of the number of documents in the corpus divided by the number of documents containing the specific term.

"The movie is the best"
feature set: [('The', -0.4), ('movie', -0.4), ('is', -0.4), ('best', -0.4)]
'This movie is not the best'
feature set:[('This', 0), ('movie', -0.4), ('is', -0.4), ('not', 0), ('the', -0.4), ('best', -0.4)]

This way, terms that appear less frequently in the entire corpus will stand out.

(iii) **Term Frequency - Inverse Document Frequency (TF-IDF)** : For each term, it is calculated as the TF multiplied by the IDF:

'The movie is the best'
feature set: [('The', -0.16), ('movie', -0.08), ('is', -0.08), ('best', -0.08)]
'This movie is not the best'
feature set:[('This', 0), ('movie', -0.07), ('is', -0.07), ('not', 0), ('the', -0.07), ('best', -0.07)]

Using the tf-idf approach, terms that have high frequency in a given document and a low frequency on the entire corpus stand out.

(iv) **Delta Term Frequency - Inverse Document Frequency (Delta TF-IDF)**: For each term, it is calculated by the difference of its TF-IDF in the positive and negative training examples. Given that the first sentence is positive and the second is negative we get the following results.

'The movie is the best'
feature set: [('The', -0.08), ('movie', -0.04), ('is', 0.04), ('best', -0.04)]
'This movie is not the best'
feature set:[('This', -0.03), ('movie', -0.03), ('is', 0.04), ('not', 0.09), ('the', -0.03), ('best', -0.03)]

This technique allows to boost the importance of words that are unevenly distributed between the positive and negative classes and eliminates the importance of evenly distributed words. As a result, features that are significant in either of the two classes stand out.

Sometimes it is better to use bigrams and trigrams for capturing more context and patterns in a sentence. For example: *not interesting, believe God* etc. For the above corpus the tokenization of the documents would be as follows:

(b) **Using bigrams:**

"The movie, movie is, is the, the best", "This movie, movie is, is not, not the, the best".

(c) **Using trigrams:**

"The movie is, movie is the, is the best", "This movie is, movie is not, is not the, not the best".

Then, the above procedure for weighting the features can be applied.

Additionally, a technique called ***normalisation*** is usually used when dealing with tf-idf for avoiding problems when a specific word is frequently used in a specific document. L1 normalisation is the Manhattan distance and L2 normalisation is the Euclidean distance.[3]. Also, sometimes it is better not to use all the features of the document but only those with high information gain. If the features do not add any value or worsen the model, it is better to throw them away. A method to do that is using ***chi square*** feature selection.

Normalisation essentially performs Delta tf-idf by comparing how common a feature is in the support class compared to the opposition class. ***Chi square*** feature selection allows you to select the ***K*** best features (i.e the most informative features).

3.2.1.1 Problem Encountered With Feature Selection

As the feature sets to be used for classifying the corpus are formed using lexical patterns that compose support and opposition phrases, this raises an issue with how the debate title

[3]http://pyevolve.sourceforge.net/wordpress/?p=1747

is phrased, i.e. if it is phrased in a positive or negative way.

For example, consider the following two discussions: *"Should abortion be allowed?"* vs *"Should abortion be banned?"*

Any supporting comments on the first one will obviously be opposing the second one. Therefore, any supporting patterns that are picked up in the second debate will mislead the classifier. This means that the classifier should differentiate between a positive and negative debate and invert the classification in the case of negative. Our initial thought was to use an existing tool that identifies positive and negative statements so that any support phrases of a negative debate will automatically considered to be opposing the topic of discussion.
One of the tools that we tried out was the *uClassify*[4].

Figure 3.5: Use of *uClassify* to determine if phrase is positive or negative

Figure 3.6: Use of *uClassify* to determine if phrase is positive or negative

The results were disappointing. None of the tools that we used were able to correctly distinguish between the two phrases which one was the negative and which one the positive. Creating such a tool, which distinguishes the two phrases, is out of the scope of this project.

[4] `http://www.uclassify.com/`

As a result an assumption of this project is that this information, i.e. whether the debate is phrased in a positive or negative way, should be given manually to the classifier.

3.2.2 NB, Maxent, SVM Using Default Parameters

In order to find out which of the weighting algorithms is more appropriate for our corpus, we created a feature extractor function in python for calculating the weights of the features depending on what technique is chosen (tf, idf, tf-idf, Delta tf-idf). Additionally, the corpus was tokenised into unigrams, bigrams, trigrams or a combination of n-grams, and tested using the three well known classifiers: Naive Bayes, Maximum Entropy and Support Vector Machines.

The results showed, as expected, higher classification rates when the tf-idf and Delta tf-idf weighting were used. The different classification algorithms had relatively close classification results, with Maximum Entropy giving the worst classification results. Vector Machines on the other hand, gave slightly better results than the other two. The most interesting observation was the n-gram choices. SVM gave higher classification rates when using unigrams together with bigrams and trigrams. The Maxent algorithm performed better using unigrams whilst Naive Bayes results were much better when using only trigrams. The following table summarises the best results achieved from the various combinations and experiments.
For full analysis on the results see Appendix Section A.1

Parameters					**Classification rate (up to)**
Maxent	+	1-grams	+	Δ tf-idf	57%
NB	+	3-grams	+	Δ tf-idf	61%
SVM	+	1-2-3-grams	+	Δ tf-idf	65%

Table 3.1: Summary of classification results

Based on the above results we were now more confident that the corpus should be classified using Naive Bayes and Support Vector Machines with Delta tf-idf feature weighting. The next step, is to try the algorithms using different parameters.

3.2.3 Experimenting With Different Parameters Using Sklearn Tool

The NLTK tool that was currently being used was not appropriate for experimenting with different classification algorithms and parameters. A much more flexible and with a wide range of algorithms is the *scikit-learn* tool [3]. Scikit-learn is a collection of Python modules relevant to machine/statistical learning and data mining.

Scikit-learn has the TfidfVectorizer class for converting a collection of documents into a matrix of TF-IDF features. This class allows the user to specify various parameters like:

- ngram_range: (min_n, max_n) for specifying what n-grams to use

- max_features: for specifying the maximum number of features to use
- norm: 'l1', 'l2' or None, gives the option to normalise the vectors
- max_df: [0.0, 1,0], gives the option to ignore terms with term frequency higher than the threshold set
- sublinear_tf: replaces tf with $1 + \log(tf)$

Another positive thing of using this class is that before tf-idf is applied, the result of tokenising and counting the tokens is returned as a sparse matrix which makes the process much faster. A sparse matrix is used when dealing with large amounts of data in a matrix populated primarily with zeros, for saving space and computation time.

Here is how the tokenizer is defined:

```
vectorizer = TfidfVectorizer(ngram_range=(1,3), max_features=200, norm='l2',
                max_df=0.5, sublinear_tf=True)
```

Also, we now had a variety of classification algorithms to choose from. Based on the research we have done on related studies, we extracted the algorithms that are most appropriate when dealing with text classification.

This is the final[5] list of algorithms that we will be dealing with in the following experiments:

- Linear SVC
- Stochastic Gradient Descent (SGD)
- Nearest Centroid
- K Nearest Neighbours
- Multinomial Naive Bayes
- Bernoulli Naive Bayes
- Non Linear SVC
- Logistic Regression

Having the possibility to initialise the algorithms with different parameters, a python program was needed to try out the different combinations and extract the best possible parameters for a given corpus and classifier. Again, scikit-learn gave the solution [6], providing a python program which does exactly that using cross validation on the corpus. This is done by specifying the corpus you want to train, the algorithm you want to examine and the possible parameters with the values you want the algorithm to be triggered. For example, for extracting the best parameters for LinearSVC classification on the entire corpus, we used the following parameters:

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  LinearSVC())
                    ])
```

[5]The original list also had Perceptron and K nearest neighbours which were soon removed due to very low classification results

[6]http://scikit-learn.org/0.13/auto_examples/grid_search_text_feature_extraction.html

```
parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__max_n': (1, 2, 3),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'chi2__k': (50, 100, 200, 500,1000),
    'clf__penalty': ('l1','l2'),
    'clf__C': [1, 10],
    'clf__tol': [1e-6, 1e-1]
}
```

Giving the following results:

```
Best score: 0.629
Best parameters set:
        chi2__k: 500
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure 3.7: Extracting Best Parameters for LinearSVC Classification on all corpus data

Using the best parameters on the corpus we had the following results[7]:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=1e-06, verbose=0)
train time: 0.018s
test time:  0.000s
accuracy-score:   0.637
precision-score:   0.634
recall-score:   0.878
f-measure-score:   0.736
```

Figure 3.8: Extracting Best Parameters for LinearSVC Classification on all corpus data

This procedure was performed on all the classification algorithms. Please see Appendix Section A.2 for the parameters used for each algorithm and their results.

These are the classification results when using the parameters that resulted from the experimentation in Appendix Section A.2, for each algorithm:

[7]small difference in the accuracy is due to the division of the data into train and test data sets

Learning Method	Normalisation Technique	Choice of N-grams	Accuracy
Linear SVC	l2	unigrams	64%
Stochastic Gradient Descent (SGD)	l2	unigrams	62%
Nearest Centroid	l2	unigrams	62%
K Nearest Neighbours	l2	unigrams, bigrams	62%
Multinomial Naive Bayes	l2	unigrams, bigrams, trigrams	63%
Bernoulli Naive Bayes	l2	unigrams, bigrams, trigrams	64%
Logistic Regression	l2	unigrams, bigrams	65%

Table 3.2: Summary of classification results

The results from training the entire corpus using the best parameters for each classifier were still not giving satisfactory results. This meant that something was wrong with the feature selection. Using the classifier with the highest accuracy, we extracted the 20 most informative features for support and opposition:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l2, random_state=None, tol=0.1)
train time: 0.004s
test time:  0.000s
accuracy-score:   0.648
precision-score:   0.644
recall-score:   0.870
f-measure-score:   0.740

top 20 keywords per class:
        -2.2787 behav                   2.4036  appar
        -1.6558 chang                   1.9343  breakfast
        -1.5338 airport                 1.4497  comedi
        -1.3416 awkward                 1.3694  contribut
        -1.2225 damag                   1.2493  christma
        -1.2184 afterlif                1.1870  answer
        -1.2049 cynic                   1.1492  bottl
        -1.1686 confirm                 1.1356  abortions
        -1.1372 conceiv                 1.0871  confus
        -1.1173 discoveri               1.0835  crippl
        -1.1129 daddi                   1.0814  antiabortionist
        -1.0149 armi                    1.0228  amus
        -0.9814 door                    1.0118  brainier
        -0.9779 arrang                  0.9777  confront
        -0.9580 ah                      0.9687  denmark
        -0.9392 combin                  0.9647  bargain
        -0.9391 adapt                   0.9071  action
        -0.9042 chore                   0.8681  distinct
        -0.8769 adulteri                0.8569  civilian
        -0.8635 clarenc                 0.8562  america
```

Figure 3.9: Most Informative Features for support and opposition trained on the entire corpus

This made us realise that the features were not particularly informative since the topics discussed had very different vocabulary for expressing support or opposition. Therefore, the

next step was to examine how successful the classification would be when trained only on one particular topic.

3.2.4 Experimenting With Different Classifiers On One Topic At A Time

In order to find out which classifier to use on each topic we used the classifiers that performed better on the experiments in section 5.2.3. These were Linear SVC, Naive Bayes and Logistic Regression. The topics to be classified are:

- Abortion
- Creationism
- Gay Rights
- Existence of God
- Gun Rights
- Healthcare
- Quality

Additionally, since now we would be dealing with a smaller corpus each time, we needed to perform k fold cross validation. The number of folds created was set depending on the size of the corpus that was being classified. Scikit-learn provides a number of different ways to split the corpus into folds but the one used in my implementation is the *StratifiedKFold*[8]. StratifiedKFold makes folds by preserving the percentage of samples for each class. The reason for choosing this technique is that Delta tf-idf feature selection requires both positive and negative examples to be in each set.

Here are the topic based classification results:

Topic	**Learning Method**	**Normalisation Technique**	**Choice of N-grams**	**Accuracy**
Abortion	Linear SVC	l1	unigrams	75%
Creationism	Logistic Regression	l2	bigrams	77%
Gay Rights	Bernoulli Naive Bayes	l2	unigrams	68%
Existence of God	Linear SVC	l1	unigrams, bigrams, trigrams	62%
Gun Rights	Logistic Regression	l2	unigrams	79%
Healthcare	Logistic Regression	l2	trigrams	60%
Quality	Linear SVC	l2	unigrams, bigrams	79%

Table 3.3: Summary of classification results

[8] `http://scikit-learn.org/dev/modules/generated/sklearn.cross_validation.StratifiedKFold.html`

As seen from Table 3.3, the classification results are much higher in most of the topics. The reason for the increase in the accuracy, is that now the feature set is much more specific and narrowed down to the vocabulary related to a specific topic. Extracting the most informative features on abortion data, we got the following results:

```
top 20 keywords per class:
        -1.9626 arbitrari              1.8392  sexual
        -1.4652 half                   1.3857  live
        -1.4290 atheist                1.3733  can
        -1.3285 cal                    1.3690  obvious
        -1.2264 record                 1.3283  hurt
        -1.1765 seriou                 1.2568  liberti
        -1.1640 histor                 1.0280  issu
        -1.0676 check                  1.0192  hypocrisi
        -1.0445 activist               1.0053  eye
        -0.9702 onc                    1.0050  ago
        -0.9667 beat                   0.9352  consid
        -0.9471 accident               0.9210  implement
        -0.9237 open                   0.9090  banned
        -0.9165 procedur               0.8293  breath
        -0.8902 revers                 0.8162  find
        -0.8697 innoc                  0.8064  everyth
        -0.8595 flawless               0.7664  educ
        -0.8565 iraq                   0.7517  plan
        -0.8332 old                    0.7466  due
        -0.8018 instanc                0.7321  afford
```

Figure 3.10: Most Informative Features for support and opposition when trained on the abortion corpus

For more details on the classification results and the parameters used for each algorithm please see Appendix Section A.3.

3.2.5 Experimenting With Part Of Speech Tagging

As mentioned already in the criteria for selecting appropriate feature sets, it is sometimes more helpful to use words or phrases that tend to more informative. These words or phrases are usually adjectives or adverbs combined with nouns, since they tend to be more emotional. In order to extract these words and phrases, we must first assign a part of speech (POS) tag on each word of the text, and then using patterns to extract the ones we are interested in.

Here's an example of how the nltk ***POS tagger*** attaches the POS tag to the words of a text

```
>>> import nltk
>>> text = nltk.word_tokenize("This is a beautiful scenery")
>>> nltk.pos_tag(text)
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('beautiful', 'JJ'), ('scenery', 'NN')]
>>>
```

Figure 3.11: Using the Part-of-Speech tagger of nltk to attack POS tags to each word of a text

The patterns used in my classification was a combination of unigrams, bigrams and trigrams of the following patterns.

Rule No	First Word	Second Word	Third Word
Rule 1	JJ	NN|NNS	anything
Rule 2	RB|RBR|RBS	JJ	anything but NN|NNS
Rule 3	JJ	JJ	anything but NN|NNS
Rule 4	NN|NNS	JJ	anything but NN|NNS
Rule 5	RB|RBR|RBS	VB|VBD|VBN|VBG	anything
Rule 6	JJ	NN|NNS	-
Rule 7	RB |RBR|RBS	JJ	-
Rule 8	JJ	JJ	-
Rule 9	NN|NNS	JJ	-
Rule 10	RB |RBR|RBS	VB|VBD|VBN|VBG	-
Rule 11	NN|NNS|JJ|RB|RBR|RBS	-	-

Table 3.4: POS Patterns used

where:

Tag	POS
JJ,	Adjective
RB,	Adverb
RBR,	Adverb, comparative
RBS,	Adverb, superlative
NN,	Noun, singular
NNS,	Noun, plural
VB,	Verb
VBD,	Verb, past tense
VBN,	Verb, past principle

Running the following commands allows you to extract the patterns with the appropriate tags:

```
vocabulary1 = set()
vocabulary2 = set()
vocabulary3 = set()

def filter(tree):
    return (tree.node == "RULE")

def myTokenizer(text):
    text = nltk.word_tokenize(text)
```

```
tags = nltk.pos_tag(text)
grammar = []
# Rule 1
grammar.append(r"""RULE: {<JJ>(<NN>|<NNS>)(<...>|<..>)}""")
# Rule 2
grammar.append(r"""RULE: {(<RB>|<RBR>|<RBS>)<JJ>(?!<NN>|<NNS>)(<...>|<..>)}""")
# Rule 3
grammar.append(r"""RULE: {<JJ><JJ>(?!<NN>|<NNS>)(<...>|<..>)}""")
# Rule 4
grammar.append(r"""RULE: {(<NN>|<NNS>)<JJ>(?!<NN>|<NNS>)(<...>|<..>)}""")
# Rule 5
grammar.append(r"""RULE: {(<RB>|<RBR>|<RBS>)(<VB>|<VBD>|<VBN><VBG>)(<...>|<..>)}""")
# Rule 6
grammar.append(r"""RULE: {<JJ>(<NN>|<NNS>)}""")
# Rule 7
grammar.append(r"""RULE: {(<RB>|<RBR>|<RBS>)<JJ>}""")
# Rule 8
grammar.append(r"""RULE: {<JJ><JJ>}""")
# Rule 9
grammar.append(r"""RULE: {(<NN>|<NNS>)<JJ>}""")
# Rule 10
grammar.append(r"""RULE: {(<RB>|<RBR>|<RBS>)(<VB>|<VBD>|<VBN><VBG>)}""")
# Rule 11
grammar.append(r"""RULE: {(<NN>|<NNS>|<JJ>|<RB>|<RBR>|<RBS>)}""")

for item in grammar:
    chunker = nltk.RegexpParser(item)
    chunked = chunker.parse(tags)
    for s in chunked.subtrees(filter):
        if len(s)>2:
            vocabulary3.add(s[0][0]+' '+s[1][0]+' '+s[2][0])
        elif len(s)>1:
            vocabulary2.add(s[0][0]+' '+s[1][0])
        else:
            vocabulary1.add(s[0][0])
```

The above code, allows you to extract patterns of unigrams, bigrams, trigrams with the POS tags you specified in the grammar.

Here are the topic based classification results:

Topic	Learning Method	Normalisation Technique	Choice of N-grams	Accuracy
Abortion	Logistic Regression	l1	unigrams, bigrams, trigrams	74%
Creationism	Logistic Regression	l2	trigrams	76%
Gay Rights	Logistic Regression	l2	unigrams, bigrams, trigrams	67%
Existence of God	Logistic Regression	l1	unigrams	57%
Gun Rights	Multinomial Naive Bayes	l2	unigrams, bigrams, trigrams	78%
Healthcare	Logistic Regression	l2	bigrams	53%
Quality	Logistic Regression	l2	unigrams	78%

Table 3.5: Summary of classification results

As you can see from the results, there was no improvement in classification accuracy when using POS tagging and in some cases we even see a decline. One of the main reasons that this happened is that POS tagging is not very effective when used on short documents. This is because when you have limited amount of words that you can use, there is little noise in the text and usually most of the words are quite informative.

For more details on the classification results and the combination of POS tag patterns use please see Appendix Section A.4.

3.2.6 Summary Of Proposed Techniques For Building A Suppport/ Opposition Classifier

From the series of experiments that we performed, we verified that Delta tf-idf, as it was suggested by Martinau and Finin[35], is indeed a good approach to develop the feature space of the classifier (compared to tf, idf, tf-idf), even when dealing with microblogging data. The feature space performs better when using the lemma of each word.

There is not a clear cut distinction for deciding which type of normalisation method to use ('l1' or 'l2') for achieving the effect of boosting the importance of words between the two classes, or for the choice of maximum features to be used, or what choice of n-grams (unigrams, bigrams, trigrams) to follow, or even the choice of learning method (SVM, Logistic Regression, Naive Bayes, etc.). The results from the experiments indicated that these choices vary depending on the corpus used for training and testing the classifier, since one combination of the above can be ideal for one corpus but perform significantly worse than a different combination on another corpus. It is therefore important to experiment with the above choices before settling

to the ones used for a particular corpus. That is why, for the different topics of our corpora, we used different combinations.

Finally, incorporating POS tagging for developing the feature space, did not provide any improvement in the classification results but instead lowered the accuracy, making it not an ideal approach when dealing with microblogging data.

3.3 Analysing Techniques For Agreement/Disagreement Classification

The second part of this project is about creating a classifier for recognising whether a response agrees or disagrees with the quote that is in reply to. As we have seen in Section 2.5, this is a problem that a few have already tried to target giving relatively good classification results.

The previous studies tend to use information about the speaker like political views, identification of adjacency pairs and hierarchy of previous related comments. This kind of information is not available in this project so the we can only experiment with local lexical features for training the classifier.

In this section, we will first experiment with the lexical features used by Galley et al [38]. Then as we move along, the features will be adjusted to deal with microblogging data.

3.3.1 Using Different Feature Sets For Experimentation

Depending on the corpus provided, each classifier has to create an appropriate feature set for classifying correctly the corpus. In order to find out the best combination of lexical features, we experimented with different approaches. The starting point of our work, is based on the study by Galley et al [38]. As seen in their study, the following features can be used, as they tend to be a good indication whether there is agreement or disagreement. Therefore, we used it as a base for creating our own feature set.

- First word of response
- Last word of response
- Number of adjectives with positive polarity
- Number of adjectives with negative polarity
- Number of instances in the document of each cue phrase listed in Hirschber and Litman, 1994 [45]
- Number of instances in the document of each agreement/disagreement word listed in Cohen, 2002 [46]

Cue phrases are linguistic expressions such as ‘now’ and ‘well’ that function as explicit indicators of the structure of a discourse. For example, ‘now’ may signal the beginning of a subtopic or a return to a previous topic, while ‘well’ may mark subsequent material as a response to prior material, or as an explanatory comment [45]. The cue phrases used for the Agreement/Disagreement Classifier were introduced by Hirschber and Litman[45]:

```
cue_phrases = [‘accordingly’, ‘again’, ‘alright’, ‘also’, ‘altogether’, ‘because’,
‘boy’, ‘but’, ‘consequently’, ‘conversely’, ‘equally’, ‘finally’, fine’, ‘first’,
‘further’, ‘furthermore’, ‘gee’, ‘hence’, ‘hey’, ‘hopefully’,  ‘indeed’, ‘last’,
‘like’, ‘likewise’, ‘listen’, ‘look’, ‘moreover’, ‘namely’, ‘next’, ‘now’, ‘oh’,
‘ok’, ‘or’, ‘overall’, ‘say’, ‘second’, ‘see’, ‘similarly’, ‘so’, ‘then’, ‘therefore’,
‘thus’, ‘too’, ‘well’, ‘where’, ‘alternately’, ‘alternatively’, ‘although’, ‘however’,
‘incidentally’, ‘anyway’, ‘meanwhile’, ‘nevertheless’, ‘nonetheless’, ‘nor’, ‘not’,
‘only’, ‘otherwise’, ‘still’, ‘though’, ‘unless’, ‘whereas’, ‘why’, ‘yet’, ‘wrong’]
```

Agreement/Disagreement words and phrases, are strong indicators of whether the reply is in agreement or disagreement with a statement. The ones used here were extracted from Cohen's paper [46] and are the following:

```
agreement_disagreement = [‘i’,’yeah’, ‘mmhm’, ‘ok’, ‘right’, ‘yes’, ‘fine’, ‘exactly’,
‘agree’, ‘of course’, ‘no’, ‘um’, ‘not’, ‘like’, ‘know’, ‘well’, ‘dont’, ‘really’,
‘just’, ‘was’, ‘to’, ‘im’, ‘that’, ‘you’, ‘actually’, ‘because’, ‘uh’, ‘in’, ‘but’,
‘think’,‘if’, ‘sort’, ‘mean’, ‘she’, ‘kind’, ‘hm’, ‘honestly’, ‘disagree’, ‘definitely’]
```

These features were used when dealing with large documents and therefore had to be adapted for dealing with small texts of about 200 characters. Therefore, in order to find an appropriate feature set we experimented with the following:

- Use adverbs, verbs, nouns instead of just adjectives.
- Instead of having two features, one for the positive and one for the negative number of adjectives, use only one to state whether the number of positive words was greater than the number of negative words.
- Instead of counting the positive/negative words in the document, calculate the overall positive/negative score of the words in the text and use a feature to state if the overall score of positive words is greater than the score of negative words.
- Instead of counting the number each cue phrase or agreement/disagreement word appears in the text, use a ”0” or ”1” value to state whether it appears in the text or not.
- Take into account te punctuation used in the text.

3.3.2 Extracting Words From Text Based On Their POS Tag

In order to find the polarity of the adjectives in the text, we used the method explained in section 3.2.5, of assigning a part of speech tag to each word and then based on the desired pattern, extract the words with the part of speech we are interested in. In this case, the text was tokenised into unigrams as the words were considered separately. In the beginning, only adjectives were taken into consideration. However, since the tagger used is not perfect, there were some key words which were not taken into consideration, and could change the overall polarity of the text. Therefore, the experiments take into consideration adjectives, adverbs, verbs and nouns.

3.3.3 Calculating Sentiment Polarity Of Words

3.3.3.1 Using SentiWordNet

In order to detect the polarity of the adjectives, we used a sentiment classifier [9] which is based on *SentiWordNet*. SentiWordNet is a lexical resource that uses words extracted from WordNet database and provides for each synset (set of synonyms) in WordNet a numerical score for positive and negative orientation. In order to calculate the scores, SentiWordNet computes the number of times each word appears to be negative/positive. Figure 3.12 , shows how the sysnset scores for the adjective 'bad' are extracted and calculated. First the synsets of the word 'bad' are extracted based on the part of speech tag (Adjective, Noun, Verb, Adverb) and then the score for one of the synsets in the list, is calculated. The word 'bad' appears to have 14 senses for adjective.

```
>>> from senti_classifier.senti_classifier import synsets_scores
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets(str("bad"), pos=wn.ADJ)
[Synset('bad.a.01'), Synset('bad.s.02'), Synset('bad.s.03'), Synset('bad.s.04'), Synset('regretful.a.
01'), Synset('bad.s.06'), Synset('bad.s.07'), Synset('bad.s.08'), Synset('bad.s.09'), Synset('bad.s.1
0'), Synset('bad.s.11'), Synset('bad.s.12'), Synset('bad.s.13'), Synset('bad.s.14')]
>>> synsets_scores[sn[0].name]['pos']
0.0
>>> synsets_scores[sn[0].name]['neg']
0.625
>>> synsets_scores[sn[9].name]['pos']
0.0
>>> synsets_scores[sn[9].name]['neg']
1.0
```

Figure 3.12: Using SentiWordNet for extracting the positive and negative sentiment score for the adjective 'bad'.

As you can see from Figure 3.12, different synsets have different positive/negative scores. So, instead of calculating these values from a random synset for each word, a better approach is to calculate the average score for all the provided synsets. The scores for word 'bad' are shown in Table 3.6:

Word	POS Tag	Positive Average Score	Negative Average Score
bad	Adjective	0.02	0.73
bad	Adverb	0.13	0.25
bad	Noun	0.0	0.88
bad	Verb	-	-

Table 3.6: Average Positive and Negative scores for word 'bad'

3.3.3.2 Dealing With Negation In A Sentence

In the case where a sentence has a negation in it, i.e. the word 'not', we had to make sure that the correct polarity scores were calculated. For example, consider the sentence 'This

[9] `https://pypi.python.org/pypi/sentiment_classifier`

product is not as good as we expected it to be', which has the adjective 'good'. Using the sentiment score technique described in the previous section, would provide a higher positive than negative value and therefore consider it to be a sentence with overall positive polarity. However, 'not' negates this and makes the overall polarity negative.

For dealing with this situation, our approach looks whether a negation appears in the sentence and inverts the positive/negative scores of the adjectives, adverbs etc. This requires the text to be split into sentences so that a negation does not affect the overall sentiment of the text. The sentences are considered to be separated with a full stop, question mark or exclamation point.

3.3.3.3 Using uClassify

Even though SentiWordNet has good results for most of the words, there are some words for which its results are not as good as we would want them to be. For example consider the word 'bored'. One would expect it to have a higher negative than positive score. However, this is not the case when using SentiWordNet as seen in Figure 3.13.

```
>>> wn.synsets(str("bored"), pos=wn.ADJ)
[Synset('bored.s.01'), Synset('blase.s.02')]
>>> sn = wn.synsets(str("bored"), pos=wn.ADJ)
>>> synsets_scores[sn[0].name]['pos']
0.0
>>> synsets_scores[sn[1].name]['pos']
0.375
>>> synsets_scores[sn[0].name]['neg']
0.0
>>> synsets_scores[sn[1].name]['neg']
0.25
```

Figure 3.13: Using SentiWordNet for extracting the positive and negative sentiment score for the adjective 'bored'

An alternative method for extracting the positive and negative scores of a word is using uClassify. uClassify provides a link that you can use to classify a word or a set of words and returns the results in json format. This is shown in Figure 3.14.

```
>>> import json
>>> import urllib
>>> data = urllib.urlopen('http://uclassify.com/browse/uClassify/sentiment/ClassifyText?readkey=mtarR
2D4jHPhcmKCFEZzv6691gU&text=bored&output=json&version=1.01').read()
>>> json_data = json.loads(data.encode('utf-8'))
>>> json_data
{u'cls1': {u'positive': 0.0120925, u'negative': 0.987908}, u'success': True, u'errorMessage': u'', u'
version': u'1.01', u'textCoverage': 1, u'statusCode': 2000}
>>> json_data['cls1']['positive']
0.0120925
>>> json_data['cls1']['negative']
0.987908
```

Figure 3.14: Using uClassify for extracting the positive and negative sentiment score for the adjective 'bored'.

As you can see, the results are better when extracted using 'uClassify'.

We repeated the process for a about 100 different responses and compared the results with SentiWordNet. It seemed that uClassify was more trustworthy than SentiWordNet and there was usually a high difference in the positive and negative scores provided, making the distinction more clear. Unfortunately, uClassify only allows 5000 requests per day from a single 'free' account, making this not ideal for experimentation with training large data sets. Additionally, it requires much more time to provide a classification result than SentiWordNet. Training with SentiWordNet usually takes no more than 1 minute, while training with uClassify requires about 30 minutes.

Consequently, the small increase in performance using uClassify was not enough reason to use it over SentiWordNet. Additionally, SentiWordNet is much more flexible, as you can specify the part of speech you are interested in classifying as well as using different meanings of each word.

3.3.4 Resulting Feature Set

After experimentation with different feature sets and implementation choices, that can be found in Appendix A.5, we settled on the following feature set:

- First word of response
- Last word of response
- Overall positive score of adjectives, adverbs, verbs and nouns compared to their negative score
- Existence of each cue phrase listed in Hirschber and Litman, 1994 in the document
- Existence of each agreement/disagreement word listed in Cohen, 2002, in the document
- Existence of question marks

Using these features, with 10-fold cross validation the classifier has an accuracy of up to **88%**

Classifiacation Accuracy:	0.8818
Positive Precision:	0.8448
Positive Recall:	0.9245
Positive F-measure:	0.8829
Negative Precision:	0.9231
Negative Recall:	0.8421
Negative F-measure:	0.8807

Table 3.7: Accuracy Results for Agreement Disagreement Using the Optimal Feature Set

For more details and reasoning behind these choices, as well as how the various feature sets affect the classification results, please see Appendix A.5.

3.3.5 Summary Of The Proposed Techniques For Building An Agreement/ Disagreement Classifier

The technique proposed for building an agreement/disagreement classifier is based on some of the features used by Galley[38]. The feature space includes only lexical features as there

are not any structural or durational features available using our corpus. The lexical features treat the response text as a bag of words from which the following features are extracted:

- First and last word of response
- Indication whether the overall polarity of the text is positive or negative based on adjectives, adverbs, verbs and nouns
- Existence of each cue phrase listed in Hirschber and Litman, 1994 [45]
- Existence of each agreement/disagreement word listed in Cohen, 2002 [46]
- Existence of question marks

For detecting polarity, we used POS Tagging for extracting adverbs, adjectives, nouns and verbs. For determining the polarity of the text we used SentiWordNet. To avoid inconsistencies we took into consideration the existence of negation in each sentence of the text.

The learning method used is Multinomial Naive Bayes.

3.4 Difficulties Encountered

When working on this project, we faced many difficulties since we were relatively new to the machine learning area and had little or no knowledge for most of the aspects concerning automatic sentiment analysis. Therefore, a lot of time was spent trying to understand how to target this concept in terms of algorithms used for classifications, how features are extracted and weighted and how to evaluate the different methodologies. This research was particularly useful for understanding the relative studies on the automatic sentiment analysis since most of the papers take as a given that the reader is familiar with the various definitions and methodologies.

Additionally, a lot of time was spent on creating the corpora that were used since we could not find any corpora in the academic community that met the criteria of the microblogging debating world. Having to adjust existing ones or annotate corpora that were extracted from debating websites was very time consuming.

When implementing the classifiers, a challenging task was the exploration journey of investigating the available techniques in order to reach the one that will be most suitable for the classification task and corpora. As we were dealing with large amounts of data, we tried to use techniques and tools that are not very time consuming, like storing the data in sparse matrices, using machine learning algorithms provided by Scikit instead of NLTK and so on. For doing so, we had to experiment with the different tools as well as with any conventions provided in Python to speed up the process.

When dealing with Support/Opposition Classification, the most difficult part was creating a classifier that when trained on the entire corpora (including all the different topics) to be able to classify a random text based on any possible topic. To do so, we spent endless hours trying to experiment with different combinations for feature sets, different parameters for all the available machine learning algorithms and limits in the number of features sets used. This is shown in the Appendix Sections A.1- A.4. As this did not achieve the required results, we had to repeat the experiments using one Topic at a time and then again by adding POS tagging.

When dealing with Agreement/Disagreement Classification, the most difficult part was to construct the correct feature set that would enable to differentiate the two classes, since this classifier could only rely on local lexical features. A particular challenge was dealing with negation in order to modify the feature set accordingly.

From an engineering perspective, we had to understand how to use each of the tools (e.g. for spell checking, POS Tagging, sentiment polarity, etc.), and find the ones that are more trustworthy and with as little overheads as possible. Additionally, we had to learn how to incorporate them all together in Python and to make sure that they were working correctly.

3.5 Summary Of Analysis

In this chapter, we evaluated different methodologies for building two classifiers, a Support/Opposition Classifier and an Agreement/ Disagreement Classifier. First of all, as the two classifiers have different text requirements, two corpora were constructed with a text size limit of 200 characters.

The Support/ Opposition Classification used data from political and ideological debates combined with data about quality of products, hotels etc. For better use of lexical features, the corpus performs syntax corrections and transformation of words to their lemma form. The classifier was then tested against different feature sets and machine learning algorithms. The features set experimentation included tf, idf, tf-idf, Delta tf-idf, unigrams, bigrams, trigrams combined with normalisation and chi-square feature selection when trained using SVM, Maximum Entropy and Naive Bayes. From these experimentation, the Delta-tf-idf performed better so it was chosen for the experiments that followed. Since the corpora has data from various topics, at first we experimented with different machine learning algorithms when triggered using different parameters on the entire corpora and then only on one topic at a time. The results showed that training on a single topic at a time gives much better classification results, since the classifier tries to pick up phrasal patterns to distinguish between support/opposition. A supporting phrase of one corpus might be neutral (or even worse opposing) for another corpus making it complicated to find a set of features to be used on the entire corpus. We also tried to use feature sets extracted using POS Tagging. This, however, gave worse classification results since when dealing with a small text size it usually means that most of the words are helpful and therefore not considering them will be a mistake. The best classification results achieved were 65% when trained on the entire corpus and up to 79% when trained on one topic at a time.

For the Agreement/Disagreement Classifier the corpora was extracted from the online debate forum, 4forums.com. For dealing with agreement/disagreement classification, instead of using all the words of the text for detecting patters, we looked for the existence of cue phrases and words that indicate agreement/disagreement. Additionally, in the feature set we include the first and last word of the text as they are usually very indicative of agreement/disagreement with the argument they reply to. We also performed POS tagging to extract adverbs, adjectives, nouns and verbs for determining the positive or negative orientation of the text. The machine learning algorithm used was Naive Bayes; achieving classification results of up to 88%.

Chapter 4

Visualisations To Assist A Debate

As we have already mentioned, an important objective of this project is to provide useful visualisations to the user, regarding the debates, that will enable him/her to see an overall picture of the debate in order to make observations and reach conclusions without having to go through the entire history of arguments posted. This will come in handy, especially to people who do not have much time in their hands and want an easy and quick way to extract information. As a result, the visualisations have to be easy to understand by people who do not have any particular knowledge on visualisations, without providing any ambiguities or misleading information.

In the following sections there is a detailed explanation for the purpose of using each visualisation and the information that the user can extract from them. The visualisations use the javascript library D3.js[1]. D3.js binds arbitrary data to DOM and then you can apply data-driven transformations to the document, making it ideal for dynamic visualisations involving complex transitions.

4.1 Streamgraph

A Streamgraph[2] is an alternative way to the traditional stacked bar chart, whose purpose is to easily show underlying trends in the data, without giving too much emphasis on the details. It is preferred than the traditional stacked bar graph because the aesthetics play a key role in a visualisation for creating an engaging and informative graphic.

The reason why a streamgraph would be useful in a debate system, lies in the idea of showing how popular the topics of the system are, throughout time. This is helpful for the administrators, who post the questions, as they can find out which are the topics that are of most interest to the users and therefore add new debates. Additionally, by looking at the history of a streamgraph, a sudden increase in a particular topic reminds the user that during that period an important event has occurred. For example, if there is an increase in 'politics' then perhaps it was the election period, or an increase in 'gadgets', that a new product was released.

[1] http://d3js.org/

[2] http://www.leebyron.com/else/streamgraph/

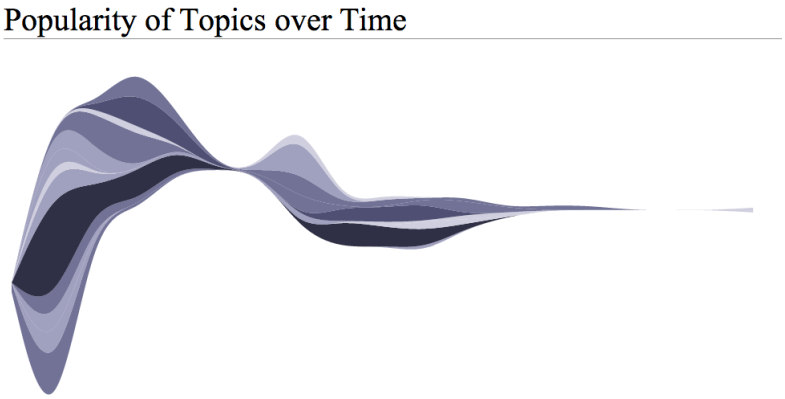


Figure 4.1: Streamgraph for the topics discussed on quaestio-it.com

In order to understand the visualisation, one must have in mind the following:

- There is no negative value.
- Each slope represents a different topic.
- The colour of each slope shows the overall popularity of the topic throughout time. The darker the colour, the more responses on that topic.
- The height of the slope shows the popularity of the topic at each period in time
- Hovering over each slope indicates the topic and the date.

Figure 4.2: Hovering feature of streamgraph

4.1.1 Implementation Details

The streamgraph was implemented based on the visualisation created by Mike Bostock [47] which is based on random values.

For the purposes of the streamgraph implemented here, the data are read using a 'comma separated value (CSV)[3]' form. We first sort them according to the date they were created and then create a HashMap containing dates as keys. Each key on the hashmap has as a value a hashmap, with keys the topics that were discussed on that particular date. The value corresponding to each topic indicates the number of posts made on a specific date for that topic. The topics are calculated on runtime allowing new topics to be added on the graph automatically. The structure is shown in Figure 4.1.

[3] `http://en.wikipedia.org/wiki/Comma-separated_values`

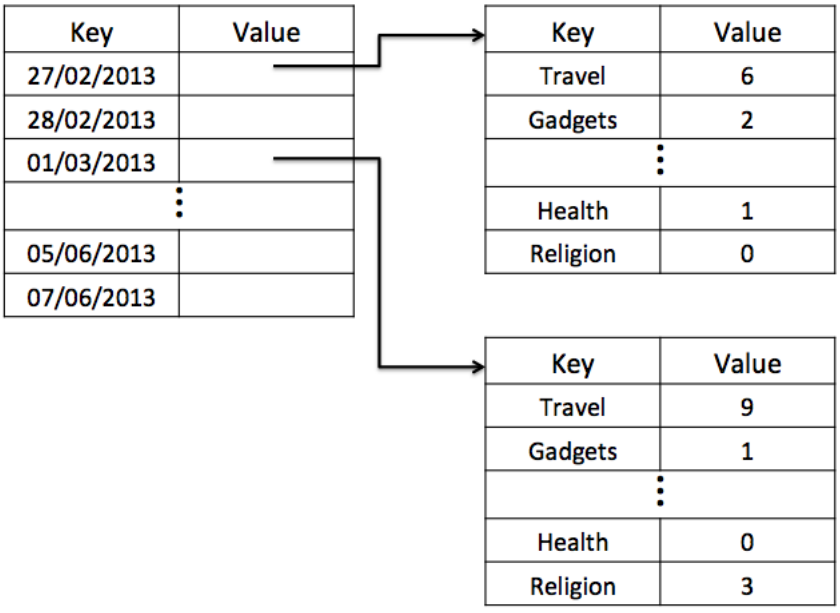


Figure 4.3: Structure of HashMaps used for storing data in Streamgraph.

Having created the HashMaps for easily accessing the data, we created the data required to represent each one of the layers, by accumulating the data to be used for the stacked layers shown in the graph. These accumulated data were used for creating the areas and the paths using d3.js.

The colours of the layers are proportional to the number of comments posted on the most popular topic over the entire period shown, and they might change as new topics or comments are added to the website. For example, let's say that until now the maximum number of comments were made by Travel, adding up to 100 comments, and 'Health' has only 10 comments in total. If after some time the amount of comments of the most popular topic changes to 1000 but no change in the amount of comments related to 'Health' (remain 10), then the colour for Health will become lighter. The scale is shown on Figure 4.4 based on the current maximum of 69 comments.

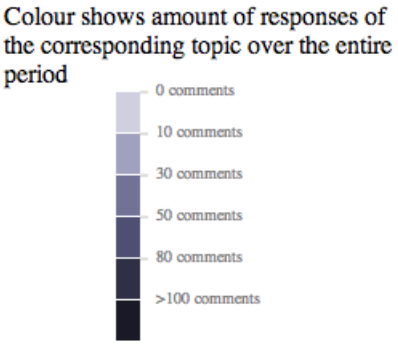


Figure 4.4: Scale of colouring used for streamgraph visualisation based on the maximum of 69 comments

For implementing the 'hovering' effect we used the 'mouseover' and 'mouseout' events for capturing movement over the layers. Then, using the coordinates of the mouse, we calculated the date and topic corresponding to that position.

4.2 Line Chart

Another interesting visualisation that is applicable in a debating system, is one that shows the history of the most popular responses for a specific discussion. For example, as new arguments come along and more people add votes and replies to the discussion, the winning arguments tend to fluctuate. In some cases, an argument is made that changes the state of the discussion and after that, users tend to agree. This will be particularly helpful when somebody wants to find out what is the underlying opinion on the debate. For example, somebody looking for reviews in a specific product, instead of having to read all the arguments, comments etc. that the users have made, it is enough to just look at the chart and see the most popular responses throughout the history of the debate, and which is the current winning argument. Also, the scoring value of that argument will be shown, to avoid misleading information being introduced. An argument may be the most popular in the debate but can still have a low score which means user's opinion is not shared amongst the majority of users; or that not enough people have casted their opinion. As the total number of arguments is shown, the user can distinguish between the two cases.

This visualisation shows on a traditional x,y axis which is the argument that wins the discussion based on the users' responses. Hovering over each response, allows you to see the argument that at that time was winning the debate.

As the debate can be both dual-sided or multi-sided, two different visualisations had to be implemented. In the case of dual-sided the arguments can either support or oppose the debate, and therefore two lines are shown on the graph. The red one represents the most popular opposing argument and the green one the most popular supporting argument. This is shown in Figure 4.5.

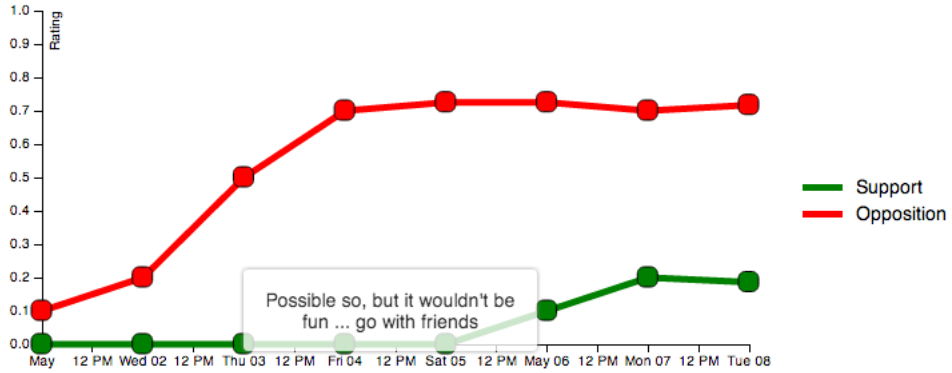


Figure 4.5: Visualisation showing the winning supporting and opposing arguments of a Debate throught time

When dealing with debates where the arguments are neither support or opposition, for example asking 'Which is the best US Series', there is only one line on the chart, for showing the most popular arguments throughout time. This is shown in Figure 4.6.

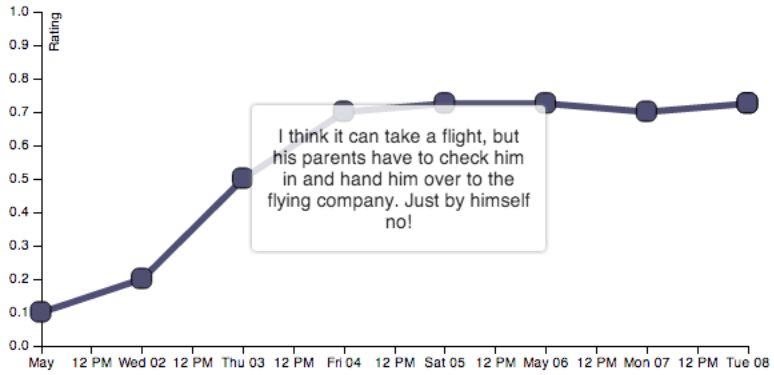


Figure 4.6: Visualisation showing the winning arguments of a Debate through time

4.2.1 Implementation Details

The implementation of the Line Chart is based on a visualisation by Mike Bostock[48], which was amended to deal with two lines on the graph.

The data are again read from a CSV file. Based on the dates that are available, the x-axis is scaled accordingly to fit the dimensions required. D3.js has a good API for creating this kind of visualisations, so, once you understand how d3.js works and how the SVG are placed on DOM, it is pretty straightforward to implement.

The graphs were implemented in such a way that would allow the scale on the axis to be updated according to the data provided. For exampe, the x-axis only shows the dates for which the particular debate had arguments or updates on arguments. This is because, the users are only interested for the days for which there is activity.

The 'hovering' effect was again based on the coordinates of the mouse and the corresponding element on the graph. d3.js allows you to bind data on the objects when created for allowing this kind of interaction with them. Therefore, we store the text corresponding to the specific coordinate for convenient retrieval when hovering over the object.

4.3 Chernoff Faces

Chernoff Faces are used to show the attitude of a user towards other users and how the attitude of other users towards him/her make him/her feel. In order to show this, two Chernoff Faces were implemented for representing the attidar of a user.

For creating the first Chernoff Face that shows the user's attitude, we used the following features to indicate various parameters.

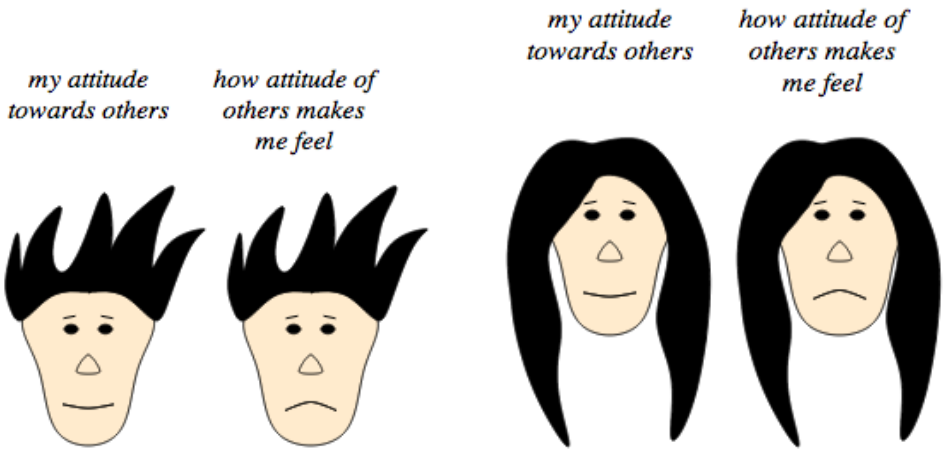
- Face: Total activity of user in terms of questions posted, arguments posted, replies to other users' arguments, thumbs up/thumbs down. Face size grows as the total activity increases.
- Mouth and Eyebrows: fluctuates from happy to angry, shows attitude of user's posts (agreement vs disagreement) and votes (positive/negative).
- Nose: Overall time spent on site. Nose size grows as time spent on site increases.

In order to create the second Chernoff Face, which shows how the attitude of other users towards user, make him/her feel, we used the following feature parameters.

- Face: Total activity of other users in terms of arguments posted for discussions made by user, replies to user's arguments, thumbs up/thumbs down or user's comments. Face size grows as the total activity increases.
- Mouth and Eyebrows: fluctuates from happy to sad, shows how the attitude of other users (agreement vs disagreement) and votes (positive/negative) towards user's comments make him/her feel.
- Nose: Overall time spent on site. Nose size grows as time spent on site increases.

Since it is easier for a user to cast votes, voting up or down has not the same value as posting an argument or a reply, weighting only 0.2 instead of 1 point.

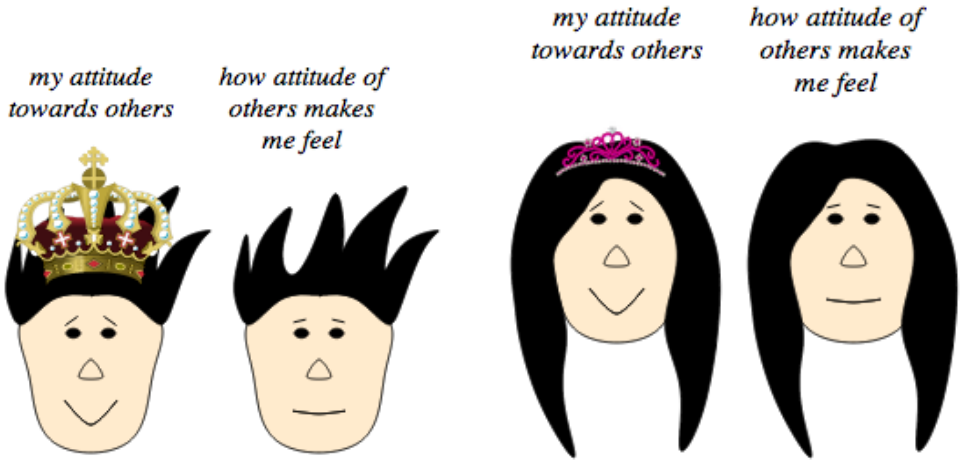
In order to make the Chernoff Faces more personal, if the user provides a gender when registering, the hair style represents if he/she is male or female. Otherwise, the default option is set to male.



(a) Male Attidar (b) Female Attidar

Figure 4.7: (a) Visualisation showing the attidar of male user, (b) Visualisation showing the attidar of female user

An additional feature of this visualisation is to give an incentive to the users to be more active in the debates. In the case where the values corresponding to the size of the faces exceed 40, the user is rewarded with a crown or a tiara depending on his/her gender. This is shown in Figure 4.8.



(a) Male Attidar with Crown (b) Female Attidar with Tiara

Figure 4.8: (a) Visualisation showing the attidar of male user with a crown, (b) Visualisation showing the attidar of female user with a tiara

4.3.1 Implementation Details

This visualisation was based on a visualisation created by Lars Kotthoff[49].

The features used for the Chernoff Faces were implemented by Lars Koothoff so we had to familiarise with the conventions he used and learn how to scale the data to meet his requirements. We also tried to improve the features of the face so that they are more closely related to the human features, since the outcome of the face is to indicate emotion. Figure 4.9 shows an example of a Chernoff Face before the alterations were made. We thought that this version was not very appealing and that the user would not be comfortable having one as his/her attidar. One of the most time consuming features to add on the original implementation was the hair. As it was not used as a varying feature, we decided to use it to represent gender. For implementing it we had to figure out the coordinates and add them together on a path for creating the spikes on the boy and the curves on the girl.

Figure 4.9: Original Version of Chernoff Face provided by Lars Kottoff[49]

Finally, we added the crown/tiara features for users which are very active.

The data used were extracted from a JSON file and then, by performing various calculations to meet the criteria set in Section 4.3, the values were scaled to appropriate values for creating the different features. Setting the correct scale for each feature was very important and required careful handling and experimentation. This is because we wanted the change of each feature to be proportional to the change in the activity of the user.

4.4 Difficulties Encountered

When working on the visualisations, the most challenging task was trying to avoid introducing ambiguities, or misleading information. Therefore, when using sizes we tried to adjust the proportions accordingly to reflect the correct information, or when using colours to clearly explain what each colour represents.

When working on the streamgraph and the Chernoff Faces visualisations, we added explanatory sections to assist the user, since, as they are not conventional visualisations, it is not very easy for someone who has not seen one before to understand them automatically. Trying to make the explanatory pictures as concise as possible so that not too much time is wasted on trying to understand the graphs, was therefore quite a challenge.

From an implementation point of view the difficulty lies in the way the data is fed to the d3.js API and then manipulated for creating the resulting shapes, lines, etc. of the visualisations and making them interactive (i.e. to show the correct information when hovering over the slope).

The most difficult task, though, was choosing the right visualisations for showing the classification results, since we wanted them to be engaging and helpful in a way that the user will be happy to use them. A huge impact on this is making them easy to use, interesting and appealing. One example of how this is achieved is finding the correct colours that stand out, show the information required but at the same time are not tiring to they eye and are likeable by most.

Chapter 5

Evaluation On An Existing System

Evaluating the classifiers is one of the most important stages of the project. The accuracy results that were calculated in implementation stage, are only based on the trained corpora. A good classifier has to be tested on a new corpus to make sure that overfitting was avoided and that the classifier is not prone to the idiosyncrasies of the training data. This will ensure that the classifiers are trustworthy enough and can be embedded in a debate system.

Additionally, it would be useful to view the visualisations on actual data that constitute complete debates and not just random texts from numerous debates.

The debating website used for carrying out the evaluation is "quaestio-it.com" [1] debate system.

5.1 Overview Of quaestio-it.com

quaestio-it.com[50] is a general purpose debating platform that offers users to express their opinion. One of the reasons for choosing it instead of other available ones, was that it was implemented within Imperial College and therefore we had easy access to the data. Additionally, the people working on the debating system were located at Imperial College making it convenient for arranging meeting and discussing the classifiers and the visualisations.

queastio-it.com allows a user to post new debates, answer with new arguments on an existing debate, reply to arguments made by other users and vote up/down an argument or reply of another user. Figure 5.1 shows a typical visualisation tree of a discussion on the website. The first layer of responses represents the arguments posted by users to support or oppose an argument. Any following layers are replies that show agreement or disagreement to the user's argument.

[1] http://www.quaestio-it.com

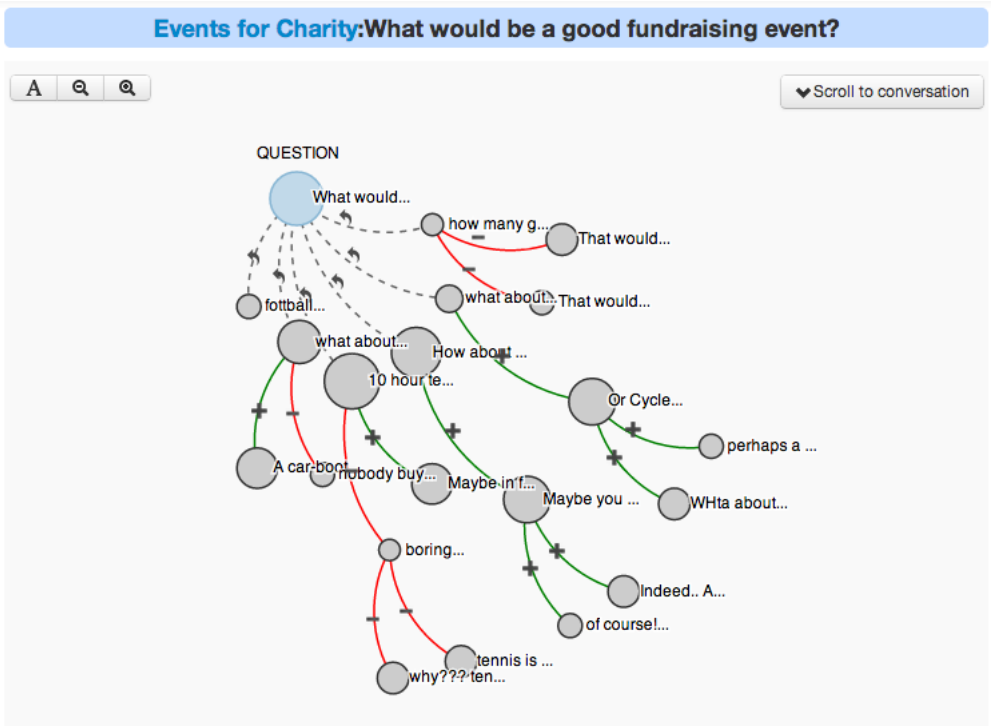


Figure 5.1: Visualisation of a debate in quaestio-it.com

When replying to arguments, this debating system asks the user to mark his/her reply as for or against the argument.

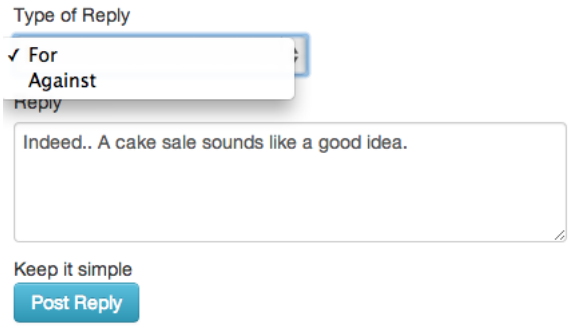


Figure 5.2: Manual classification of comment as for or against the user's argument

The user's choice is shown on the tree using a '+' or a '-' indicating a comment as being 'for' or 'against' the argument respectively.

For the first layer of responses the classifier that can be used to detect whether an argument supports or opposes the question, is the Support/Opposition Classifier that was implemented in Section 3.2. Unfortunately there is not enough corpus on the website for testing the performance of the Support/Opposition Classifier. The latest csv export of arguments posted

on the website included 29 comments (only first layer comments considered which belong to dual-sided debates). Additionally, since the website does not provide annotation for those, we have to annotate them manually before evaluating the classifier.

For the following layers the Agreement/Disagreement Classifier can be used and avoid the burden imposed to the user of having to manually annotate his/her comment as being in agreement or disagreement. These comments have already been annotated so they form a good enough new corpus for evaluating the Agreement/Disagreement Classifier. An accuracy higher than 75% is enough to make it trustworthy in order to replace the manual annotation.

5.2 Testing Support/Opposition Classifier On A New Corpus

The Support/Opposition Classifier as it was implemented in Section 3.2, it is trained on a specific corpus each time. Therefore, in order to classify a text, the topic of the discussion has to be specified. The classifier has three options; corpus, question and argument. This is shown in Figure 5.3:

```
Options:
  -h, --help            show this help message and exit
  --corpus=CORPUS       Select classifier according to corpus.
  --question=QUESTION   Please provide debate question.
  --argument=ARGUMENT   Please provide the argument to be classified.
```

Figure 5.3: Options provided by the Support/Opposition Classifier

Question and Argument options are mandatory. The user can call the classifier using the following command:

```
python SupportOppositionClassifier.py --corpus=abortion --question=‘‘Should abortion
be allowed?’’ --argument=‘‘Life starts at conception. Having an abortion is therefore
removing a human’s life’’
```

If ‘question’ and ‘argument’ options are not set then an error message appears and the program exits.

```
Options:
  -h, --help            show this help message and exit
  --corpus=CORPUS       Select classifier according to corpus.
  --question=QUESTION   Please provide debate question.
  --argument=ARGUMENT   Please provide the argument to be classified.

Usage: SupportOppositionClassifier.py [options]

SupportOppositionClassifier.py: error: Debate Question Not Given
```

Figure 5.4: Question Error provided by Support/Opposition Classifier

```
Options:
  -h, --help            show this help message and exit
  --corpus=CORPUS       Select classifier according to corpus.
  --question=QUESTION   Please provide debate question.
  --argument=ARGUMENT   Please provide the argument to be classified.

Usage: SupportOppositionClassifier.py [options]

SupportOppositionClassifier.py: error: Argument Not Given
```

Figure 5.5: Argument Error provided by Support Opposition Classifier

5.2.1 Dealing With Non Dual-Sided Debates

Additionally, the debate type has a significant role for the classifier. The classifier can classify arguments for dual-sided debates. These include debates that do not have multiple answers but instead have two sides that either support or oppose the debate.

For example, asking users to provide arguments on the debate: "Does owing a gun makes you safer?" is an example of dual-sided debate. A debate like "Which is the best city in Europe?" is not a dual-sided debate, since the answers can be for more than two choices.

As a result, before calling the classifier, we have to make sure that we are dealing with a dual-sided debate. A good enough way to detect whether a debate is dual-sided or not, is by checking the verb used to begin the question of the debate [51]. This verb can be one of the three following types:

(a) ***Be verbs***: ['am', 'is', 'are', 'been', 'being', 'was', 'were']

(b) ***Modal verbs***: ['can', 'could', 'shall', 'should', 'will', 'would', 'may', 'might']

(c) ***Auxiliary verbs***: ['do', 'did', 'does', 'have', 'had', 'has']

There are exceptions where this approach does not work, but in general it provides an accuracy of up to 90%.

In the case where the question is not dual-sided, the classifier outputs the following message:

```
Options:
  -h, --help            show this help message and exit
  --corpus=CORPUS       Select classifier according to corpus.
  --question=QUESTION   Please provide debate question.
  --argument=ARGUMENT   Please provide the argument to be classified.

This is not a binary question. Arguments provided for this type of questions cannot be classified usi
ng this classifier
```

Figure 5.6: Output message when the question is not dual-sided

5.2.2 Dealing With Missing Topic Of Corpus

Instead of showing an error message when the user does not provide the corpus related to the argument to be classified, we decided that it would be better to provide a result, even if it

is not as likely to be correct as when if the topic was given. For finding out the best way to approach it, we experimented with four different approaches.

5.2.2.1 Use Default Classifier

This option uses the entire corpus, including all the different topics (abortion, gay rights, healthcare etc) and use the classifier and parameters that were obtained from the experimentation in Appendix Section A.2.

The default option uses logistic regression algorithm with the following parameters:

```
parameters = {
    'vect__max_df': 0.5,
    'vect__max_features': None,
    'vect__max_n': 1,
    'tfidf__use_idf': False,
    'tfidf__norm': 'l2',
    'chi2__k': 1000,
    'clf__penalty': 'l2',
    'clf__C': 1,
    'clf__tol': 0.1,
}
```

These are the classification results when using a default classifier, tested on data from quaestio-it.com.

Classification Accuracy:	0.3929
Positive Precision:	0.5385
Positive Recall:	0.3889
Positive F-measure:	0.4516
Negative Precision:	0.4444
Negative Recall:	0.4444
Negative F-measure:	0.4444

Table 5.1: Accuracy Results for Support Opposition Classifier Using default classifier when no topic is provided

The results were disappointing. This emphasises that different corpora have different feature patterns that show support or opposition. Therefore, using a default classifier is not a good enough implementation decision.

5.2.2.2 Use Classifer Based On Top Features

The alternative choice, is to use the top features as they were extracted for each topic in Appendix Section A.3. These features are the ones that are more informative for each one of the topics. Therefore, we thought that a good approach is to compare the set of words in the argument provided and each one of the set of words. The set of features that has the most in

common with the argument, is set as the topic of the corpus. Depending on that choice, the respective corpus is chosen along with the appropriate classifier.

These are the classification results when using a classifier based on the most informative features, tested on data from quaestio-it.com.

Classification Accuracy:	0.50
Positive Precision:	0.25
Positive Recall:	0.375
Positive F-measure:	0.3
Negative Precision:	0.6875
Negative Recall:	0.6111
Negative F-measure:	0.6471

Table 5.2: Accuracy Results for Support Opposition Classifier when no topic is provided, using classifier with most in common top features

The results were better than using a default classifier. However, this implementation choice is still not good enough as it is no better than a random choice.

5.2.2.3 Finding Out The Most General Corpus

An alternative approach is to find which classifier, when trained on a different topic, is the most general one. This means that it performs better than others when presented to a new corpus, with a different topic of discussion. In order to find out which one it is, we used one corpus at a time and used the rest to find out it's accuracy.

Summary of accuracy results:

Corpus Topic	**Accuracy**
Abortion	57%
Creationism:	39%
Gay Rights:	52%
Existence of God:	51%
Gun Rights:	55%
Healthcare:	41%
Quality:	52%

Table 5.3: Training using abortion corpus and testing with the remaing corpora

Please see Appendix Section A.6, to see the detailed results for each training and testing corpora.

Results from testing the most general corpus, using data from quaestio-it.com:

Classification Accuracy:	0.4642
Positive Precision:	1
Positive Recall:	0.4444
Positive F-measure:	0.6153
Negative Precision:	0.0065
Negative Recall:	1
Negative F-measure:	0.1176

Table 5.4: Accuracy Results for Support Opposition Classifier when no topic is provided, using most general classifier

The results, were not satisfactory, indicating once more, that training on a single corpus is not good enough to be used to classify various topics.

5.2.2.4 Use Classifer Based On Majority Of Votes From Classifiers Trained On Different Corpora

This time, instead of choosing only one classifier trained on a choice of corpora, we decided to run all the different classifiers as they were resulted in Section 3.2.4. Therefore, eight classifiers run in total, one for each different corpus. Then, the majority of replies is used to classify the text. If more classifiers respond with support, then the final response will be support and oppose otherwise.

These are the classification results when using the majority of votes from the different classifiers, tested on data from quaestio-it.com.

Classification Accuracy:	0.6429
Positive Precision:	0.75
Positive Recall:	0.5625
Positive F-measure:	0.6428
Negative Precision:	0.6154
Negative Recall:	0.7273
Negative F-measure:	0.6667

Table 5.5: Accuracy Results for Support Opposition Classifier when no topic is provided, using classifier with most in common top features

The results were improved to 65% making this a safer choice, in the case when no topic is provided.

5.3 Testing Agreement/Disagreement Classifier On Corpus From quaestio-it.com

The data for testing the classifier, were extracted from the website in a csv file which included the classifications as they were entered by the users. Each data was modified to meet the standards of the trained data (i.e. removing urls, replacing don't with do not etc) and then the result of classifying each text with the Agreement/Disagreement classifier was compared to the value given by the user.

The results obtained were the following:

Classifiacation Accuracy:	0.8462
Positive Precision:	0.8333
Positive Recall:	0.8333
Positive F-measure:	0.8333
Negative Precision:	0.8571
Negative Recall:	0.8571
Negative F-measure:	0.8571

Table 5.6: Classification results of Agreement/Disagreement Classifier on quaestio-it.com corpus

The most informative features of the classifier are the following:

```
Most Informative Features
                    dont = 1              0 : 1      =     24.7 : 1.0
                     not = 1              0 : 1      =     12.8 : 1.0
                   wrong = 1              0 : 1      =     11.9 : 1.0
                lastword = 'wrong'        0 : 1      =     11.1 : 1.0
                    only = 1              0 : 1      =     10.5 : 1.0
                 exactly = 1              0 : 1      =      9.8 : 1.0
                    well = 1              1 : 0      =      8.9 : 1.0
                lastword = 'what'         0 : 1      =      7.9 : 1.0
                     she = 1              0 : 1      =      7.8 : 1.0
                   right = 1              1 : 0      =      7.5 : 1.0
               firstword = 'well'         1 : 0      =      7.5 : 1.0
               firstword = 'my'           1 : 0      =      6.1 : 1.0
               firstword = 'the'          0 : 1      =      5.7 : 1.0
               firstword = 'no'           0 : 1      =      5.7 : 1.0
               firstword = 'she'          0 : 1      =      5.7 : 1.0
               firstword = 'because'      1 : 0      =      5.4 : 1.0
               firstword = 'yes'          1 : 0      =      5.3 : 1.0
               firstword = 'actually'     0 : 1      =      5.1 : 1.0
                lastword = 'right'        1 : 0      =      4.9 : 1.0
                      no = 1              0 : 1      =      4.7 : 1.0
```

Figure 5.7: Most Informative Features of Agreement Disagreement Classifier

The results were very satisfactory since any results above 75% are good enough to make the classifier trustworthy enough to be embedded in a debating system. However, we were interested to see the arguments which the classifier failed to classify.

5.3.1 Misclassifications Made By The Classifier

The following cases show the situations where the Agreement/Disagreement Classifier makes mistakes.

(a) **Human Error**

It turned out that in some cases the comments were in fact correctly classified by the Agreement/Disagreement Classifier but were not annotated correctly by the users. For example, the next two cases were mistakenly annotated as negative and positive respectively.

Figure 5.8: Comment annotated as negative instead of positive by a user of quaestio-it

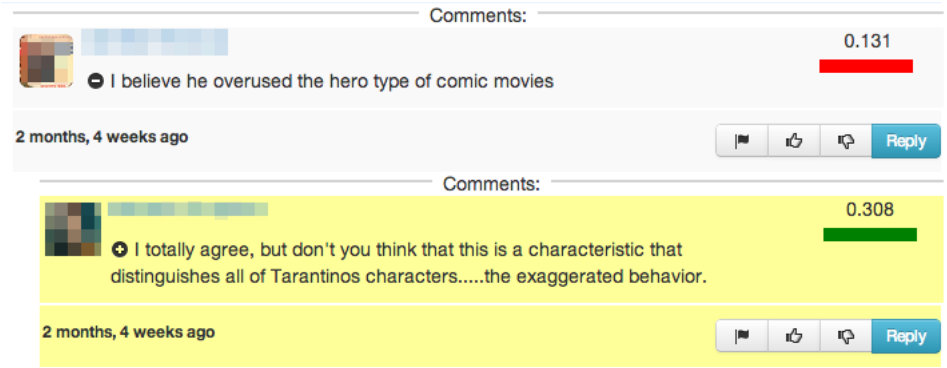


Figure 5.9: Comment annotated as positive instead of negative by a user of quaestio-it

In those cases, the classifier actually performed better than the manual annotation given by the users, since the two comments were correctly classified as positive and negative respectively by the Agreement/Disagreement Classifier. This points out that occasionally, the decision whether a comment is in agreement with another or not, is subjective. For the example in Figure 5.8, we asked the user why he annotated that comment as negative and he told us that "since the bug had already been fixed, the argument made by the other user was not really valid or helpful". However just looking at his response, there is

no indication of disagreement in the vocabulary he uses and that is why the classifier, and any other person we asked their opinion on that, considered that comment to be positive.

(b) **Missing explanation**

Looking at the results, we noticed one other case where the classifier fails to give a correct classification. This happens when the user does not provide any reasoning for his/her choice. This is shown in the following example:

Figure 5.10: Comment annotated as negative by a user of quaestio-it, without giving enough reasoning

'Liverpool' reply was classified as positive using the Agreement/Disagreement classifier when in fact it disagrees with the argument made. These situations cannot be easily classified as they do not include any cue phrases or agreement/disagreement words or even any connectives that would indicate the position it is taking. It does not even include any adjectives or adverbs to indicate any positive or negative orientation. Even manually, a human would be in a difficult situation and would be impossible to classify it without looking at the argument that is in reply to. The agreement/disagreement classifier does not take into account the comment that the text to be classified is in reply to. This information is not available in the csv file provided from the site and even if it were, it would be extremely difficult to find the topic that the comment supports and then compare it to the topic the response is supporting. Based on the training data, when a text does not include any indicative features, it is considered to be neutral, and the classifier classifies it as positive.

(c) **Missing agreement/disagreement indication**

Sometimes, the users provide enough information to support their reply but fail to say whether they agree or disagree with the comment they reply to. Consider the following example:

Figure 5.11: Comment annotated as negative by a user of quaestio-it, without giving indication of agreement/disagreement

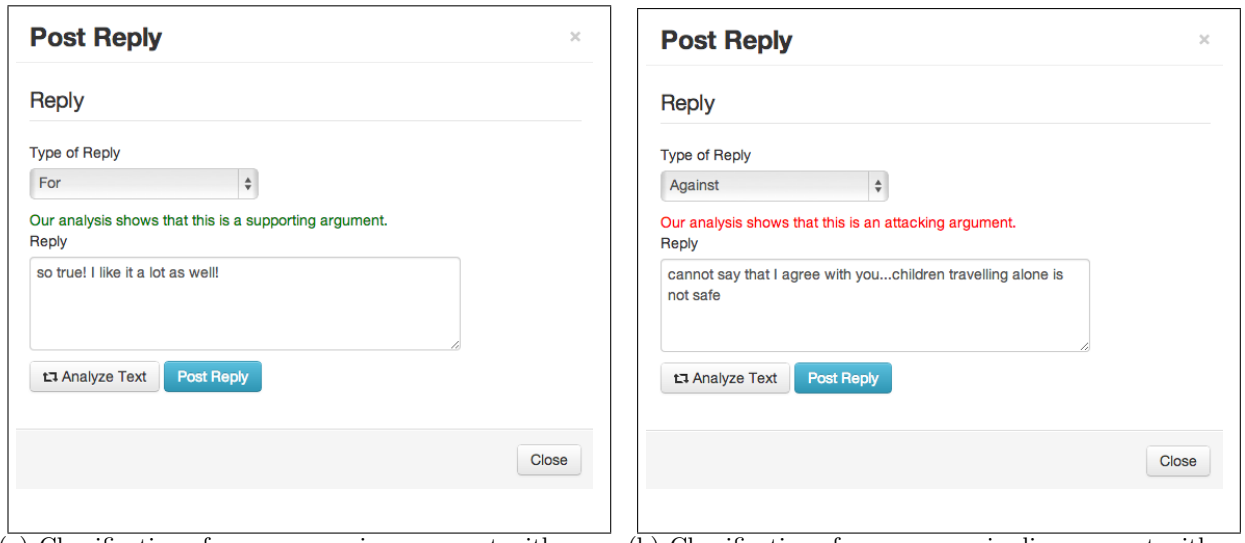
Just by looking at the reply even a human would classify that as a comment that shows agreement and, like the previous example, can only tell that it is in disagreement when looking at the the comment that it is in reply to.

5.4 Integrating Agreement/Disagreement Classifier

The results for Agreement/Disagreement Classification, as they were evaluated on quaestio-it.com, were good enough to make it reliable in order to be integrated into the system.

The integration of the classifier was easily made, since they use Python as well and therefore we only had to explain to them how to train and call the classifier and which tools or programs to import into their system.

When a user decides to reply to an argument in a debate, he/she sees the same pop-up window as shown in Section 5.2 but now have this option of analysing their response automatically instead of choosing it on their own. This is shown in Figure 5.12.



(a) Classification of a rsponse as in agreement with argument

(b) Classification of a rsponse as in disagreement with argument

Figure 5.12: Use of Agreement/Disagreement Classifier in quaestio-it.com The user can automatically classify whether his/her comment is for or against the argument he/she replies to.

As seen in Figure 5.12, the classifier tells the user how the classifier classifies the response and can decide to keep the classification or manually change it to the one he/she believes is the correct one.

5.5 Integrating Visualisations

The owners of the debating website were satisfied with the visualisations and agreed to use them on their website. However, their database could not provide enough information for showing the history of popular arguments in a debate since is did not store the previous values. Therefore, the line chart will be included in their next update, when the database has changed.

5.5.1 Integration of Streamgraph

The user is also now able to view the popularity of the topics by clicking the available button on the website. The current visualisation based on the overall activity of the users on the website is shown in Figure 5.13.

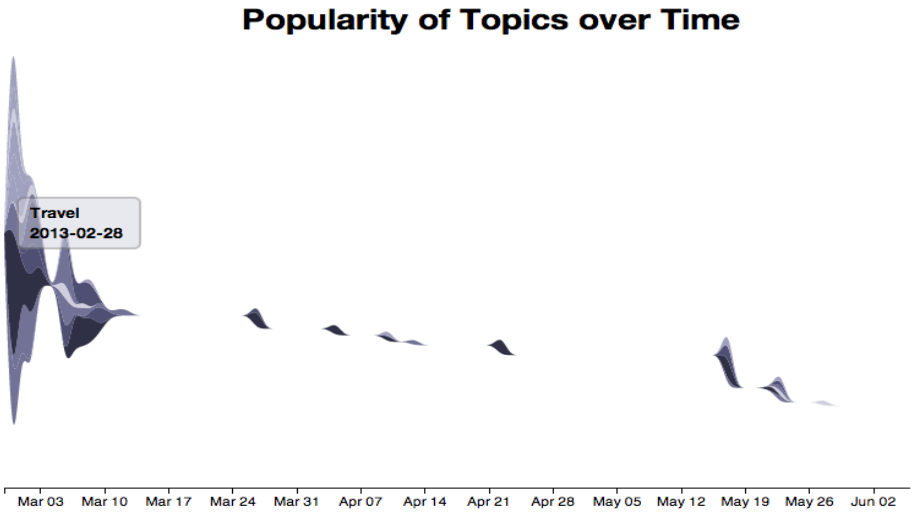


Figure 5.13: Current Streamgprah Visualisation on quaestio-it.com

Figure 5.13 shows that the activity of the users when the website was first introduced, was at its highest levels. It seems, however, that, from the middle of March until today, not so many users have been using the website. Additionally, the most popular topic of discussion is travel, as is it the one with the darker colour and the widest slope.

5.5.2 Integration Of Chernoff Faces

The user can oversee his overall activity and the activity towards him/her by other users by visiting his/her profile. Then, by clicking on the 'show attidar' button, the corresponding Chernoff Faces appear. The Chernoff Faces show the corresponding activities over the last week. Figure 5.14 shows the attidar of a user.

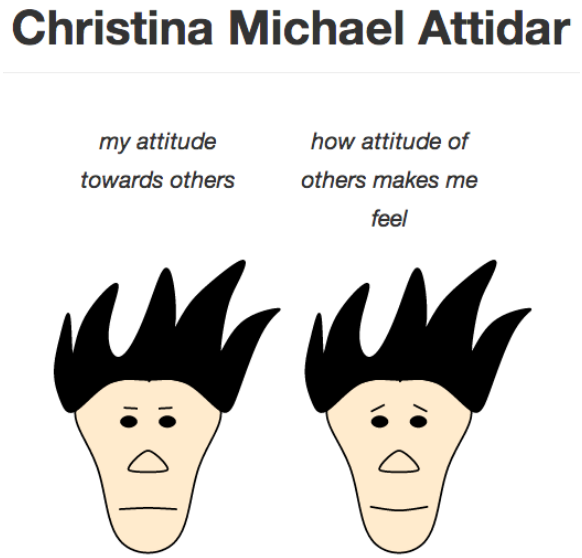


Figure 5.14: Attidar of a user based on his/her avtivity over the last week.

The attidar in Figure 5.14, shows that the attitude of the user is relatively aggressive towards other users. The attitude of other users towards him/her however, is relatively positive, since the face has a hint of a smile. Looking at the size of the face of the Chernoff Faces, neither the user nor the other users have been very active despite that the user has been spending a lot of time on the website (deducing from the size of the nose).

5.6 Difficulties Encountered

One of the difficulties that we had to overcome in order to evaluate the Support/Opposition Classifier was having to deal with situations were the topic is not provided. When developing the classifier we made an assumption that the topic will be provided but this was not the case with quaestio-it.com data. Therefore, we had to incorporate this situation. As a result, we were not able to evaluate the classifier based on the initial considerations that the classifier was built upon.

Additionally, as the website allows non dual-sided debates to be posted, there were not a lot of arguments for evaluating the Support/Opposition Classifier. Also, they were not annotated by the user so we had to manually annotate them, resulting to possible misclassifications from our part. This is because the stance a person is taking is not always clear.

For the Agreement/ Disagreement Classification, as we have already mentioned in Section 5.3.1, the difficulty lied in finding out the situations where the user had made a false annotation. This lowered the accuracy results even though the classifier was in fact correct.

A common problem regarding the two classifiers, was dealing with situations where the user did not provide any reasoning for his/her response. As the classifiers use features that depend on patterns that they pick up from the trained corpora, it is not easy to classify the ones that do not have any indication of support/opposition or agreement/disagreement.

5.7 Summary Of Evaluation On An Existing System

The Support/Opposition Classifier was evaluated on the arguments posted on dual-sided debates of quaestio-it.com. For doing so, we had to find out whether the debate was dual-sided, since there is no point in classifying arguments made for non dual-sided debates. Also, we had to change the decision making process of calling the correct classifier based on the topic of the debate, since it was not provided. For doing so, we used the majority of votes of the classification results given by each of the classifiers when trained on different corpora. This approach, combined with the classifiers described in Section 3.2, gave classification results of 65%.

For evaluating the Agreement/Disagreement Classifier, we did not face any difficulties with conventions used by quaestio-it.com (like missing topic of debate). The classifier was tested on the replies posted to arguments on all the debates, regardless whether they were dual-sided or not. The classifier described in Section 3.3 gave classification results of 85%.

Chapter 6

Conclusions And Future Work

This project aimed at targeting sentiment analysis on debates extracted from social networking sites that make use of microblogging text. This included finding ways to classify an argument as being in Support/Opposition with a debate question, and a reply as being in Agreement/Disagreement with an argument; as well as finding different ways to visualise important information extracted from the classification results.

6.1 Achievements Summary

The overall project was successful since we were able to complete in time all the objectives that were set. For doing so, we managed to incorporate and adapt previous studies to make them suitable for dealing with microblogging data, by experimenting with various combinations of machine learning algorithms, feature sets and symbolic techniques. Additionally, we incorporated an alternative way of visualising debates in an engaging and interesting way. Finally we were able to evaluate the appropriateness of the classifiers and visualisations using an existing debating website.

Particularly, this project has achieved the following:

- Achieved a classification accuracy of 65% when trained on multiple corpora and up to 79% when trained on one topic at a time for Support/Opposition Classification (compared to 64% and 71% respectively of other studies using the same corpora).
- Achieved a relatively high accuracy score of 88% in Agreement/Disagreement Classification, without having any durational or structural features that were available to other studies.
- Managed to create three interactive visualisations for providing information to the user in an appealing and interesting way; resulting in relieving him/her of the burden to waste time and effort to extract this information.
- Evaluated the classifiers and visualisations on an existing debating website achieving 64%, and 84% accuracy scores for the two classifiers. The classifier scores and the visualisations were deemed good enough to be integrated into the debating website. Some of them have already been integrated and the rest will be integrated in the next update.

6.2 Future Work

No matter how successful a project might be, there is always room for improvement. The following features, are some of the possible extensions that in my opinion can be useful, which given that we had more time, we would have included in this project.

- One of the most important additions to this project is to ***enhance the corpora***. This way the classifiers will be in position to pick up any additional patterns that can be used to indicate Support/Opposition and Agreement/Disagreement. Having a robust corpora allows the classifiers to be more adaptable to new topics of discussion. There are many available microblogging websites from which the corpora can be extracted. The process of annotating the corpora can be very tiring and time consuming, and if you do not have the time, there are companies you can pay for this kind of service.
- Adjust the Support/Opposition Classifier to ***deal with non dual-sided debates***. The classifier as it is at the moment, can only deal with debates where the answer can be one of two sides like 'yes/no', 'for/against'.
- Incorporate ***structural features*** of the debates by taking into consideration previous responses of users. For example, a user that always agrees/disagrees with another user, is more likely to agree/disagree in the future as well, or if a user's responses on debates regarding a particular topic are usually supporting/opposing it is probably to be the same in related debates.
- Extract possible ***patterns where the classifiers always make a mistake*** based on the evaluated corpus and try to deal with them. However, this requires careful handling to avoid overfitting and making the classifiers prone to idiosyncrasies of the evaluating corpora.
- The Streamgraph visualisation, as it is at the moment, shows the overall topic popularity for the entire period of the data that are provided. An extension would be to allow the user to ***select the period*** for which he/she is interested in seeing the popularity for, like a periodic graph.
- For the Chernoff Faces, instead of having a single crown/tiara as a reward for users, ***multiple rewards*** can be provided in a gold, silver, bronze manner so that there is a clearer distinction.

Bibliography

[1] Erik Boiy; Pieter Hens; Koen Deschacht; Marie-Francine Moens. *Automatic Sentiment Analysis in On-line Text.* In Proceedings of ELPUB2007 Conference on Electronic Publishing, Vienna-Austria, June 2007.

[2] Bird, Steven, Edward Loper and Ewan Klein (2009). *Natural Language with Python.* O'Reily Media Inc.

[3] Pedregosa et al. *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.

[4] "Chernoff Face" *Wikipedia, The Free Encyclopedia.* Wikipedia Foundation, Inc. 22 July 2004. Web. 10 Aug. 2004

[5] "Sentiment Analysis." *Wikipedia, The Free Encyclopedia.*

[6] "Natural Language" *Wikipedia, The Free Encyclopedia.*

[7] *Stanford Linguistics*. URL: https://linguistics.stanford.edu [2012]

[8] "Text mining" *Wikipedia, The Free Encyclopedia.*

[9] Turney, P., *Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews,* Proceedings of ACL-02, 40th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2002.

[10] Fellbaum, C. (ed.), *Wordnet: An elecotrinic lexical database,* Language, Speech, and Communication Series, MIT Press, Cambridge, 1998.

[11] Kamps, J.; Marx, M.; Mokken, R. J.; De Rijke, M., *Using WordNet to measure semantic orientation of adjectives,* LREC 2004, volume IV.

[12] Mulder, M.; Nijholt, A.; Den Uyl, M.; Terpstra, P., *A lexical grammaticaimplemntation of affec.* Proceedings of TSD-04, the 7th International Conference Text, Speech and Dialogue, Lecture Notes in Computer Science, vol. 3206, Springer-Verlag, Brno.

[13] Mitchell, Thomas M. *Machine Learning* McGraw-Hill, Inc. New York, NY, USA, 1997

[14] "Feature (machine learning)" *Wikipedia, The Free Encyclopedia.* Wikipedia Foundation, Inc. 22 July 2004. Web. 10 Aug. 2004

[15] "n-gram" *Wikipedia, The Free Encyclopedia.* Wikipedia Foundation, Inc. 22 July 2004. Web. 10 Aug. 2004

[16] Pang, B.; Lee, L.; Vaithyanathan, S. *Thumbs up? Sentiment classification using machine learning techniques.* In Proceedings of EMNLP-02, the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Philadelphia, US, 2002

[17] Riloff, E.; Wiebe, J.; Wilson, T. *Learning subjective nouns using extraction pattern bootstrapping.* In Walter Daelemans and Miles Osborne, editors, Proceedings of CONLL-03, 7th Conference on Natural Language Learning, Edmonton, CA, 2003.

[18] Hu, M.; Liu, B, *Mining opinion features in customer reviews.* In Proceedings of AAAI-04, the 19th National Conference on Artificial Intelligience, San Jose, US, 2004.

[19] *Learning subjective adjectives from corpora* In Proceedings of AAAI-00, 17th Conference of the American Association for Artificial Intelligence, AAAI Press / The MIT Press, Austin, US, 2000

[20] Salvetti, F.; Lewis, S Reichenbach, C. *Impact of lexical filtering on overall opinion polarity identification.* In Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications, Stanford, US, 2004

[21] Berwick, R. *An idiot's guide to Support vector machines (SVMs).* URL: http://www.cs.ucf.edu/courses/cap6412/fall2009/papers/Berwick2003.pdf

[22] Hatzivassiloglou, V.; McKeown, K. R. *Predicting the semantic orientation of adjectives.* In Proceedings of ACL-97, 35th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Madrid, ES, 1997.

[23] Popescu, A.; Etzioni, O. *Extracting product features and opinions from reviews.* In Proceedings of HLT-EMNLP-05, the Human Language Technology Conference/Conference on Empirical Methods in Natural Language Processing, Vancouver, CA, 2005.

[24] Maja Pantic *Computer Based Courseeork Manual* Machine Learning (Course 395), Imperial College, 2012

[25] Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; Witten, I. H. *The WEKA Data Mining Software: An Update,* SIGKDD Explorations, Volume 11, Issue 1, 2009

[26] The Apache Software Foundation URL: http://opennlp.apache.org/

[27] O'Connor, B.; Balasubranamyan, R.; Routledge, B. R.; Smithi, N. A., *From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series* In Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media

[28] Lindsay, R. *Predicting polls with Lexicon.* URL: http://languagewrong.tumblr.com/post/55722687/predicting-polls-with-lexicon

[29] Gilbert, E.; Karahalios, K., *Widespread worry and the stock market.* In Proceedings of the International Conference on Weblogs and Social Media, 2010.

[30] Pimenta F. S.; ObradovIc, D.; Schirru, R.; Baumann, S.; Dengel, A., *Automatic Sentiment Monitoring of Specific Topics in the Blogosphere* Proceedings of the 1st Workshop on

Dynamic Networks and Knowledge Discovery co-located with ECML PKDD, Barcelona, Spain, 2010

[31] Esuli, A.; Sebastiani, F., *SentiWordNet: A publicly available lexical resource for opinion mining* Proceedings of LREC-06, 5th Conference on Language Resources and Evaluation, Genova, IT, 2006

[32] Pang, B.; Lee, L., *A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts* Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, 2004

[33] Thomas, M.; Pang, B.; Lee, L. *Get out the vote: Determining support or opposition from Congressional floor-debate transcripts* Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, (EMNLP 2006), Sydney, July 2006 URL: http://www.cs.cornell.edu/home/llee/data/convote.html

[34] Yu, B.; Kaufmann, S.; Deirmeier, D., *Classifying Party Affiliation from Political Speech* Journal of Information Technology & Politics, Vol. 5, No. 1, 2008

[35] Martinau, J.; Finin, T., *Delta TFIDF: An Improved Feature Space for Sentiment Analysis* Proceedings of the Third AAAI Internatonal Conference on Weblogs and Social Media

[36] Somasundaran, S.; Wiebe, J. *Recognising Stances in Ideological On-Line Debates* Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, Los Angeles, CA, June, 2010 URL: http://www.aclweb.org/anthology/W10-0214

[37] Bansa, M,; Cardie, C.; Lee, L., *The power of negative thinking: Exploiting label disagreement in the min-cut classification framework* Proceedings of COLING: Companion volume: Posters, pp. 15–18, 2008

[38] Galley, M.; McKeown, K.; Hirschberg, J.; Shriberg, E., *Identifying Agreement and Disagreement in Conversational Speech: Use of Bayesian Networks to Model Pragmatic Dependencies* Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, 2004

[39] Burfoot, C. *Using multiple sources of agreement information for sentiment classification of political transcripts* Proceedings of the Australasian Language Technology Workshop, Vol 6, 2008

[40] Byron, L.; Wattenberg, M., *Stacked Graphs - Geometry & Aesthetics* URL: http://leebyron.com/else/streamgraph/

[41] *jQuery*, Webopedia, Everything you need to know it right here, URL: `http://www.webopedia.com/TERM/J/jQuery.html`

[42] *What is CSS?*, World Wide Web Consortium, URL: `http://www.w3.org/standards/webdesign/htmlcss#whatcss`

[43] Taboada, M; Grieve, J. *Analysing appraisal automatically* In Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications, Stanford, US, 2004

[44] Walker, M. A.; Anand, P.; Fox Tree, J. E.; Abbott, R.; King. J. *A Corpus for Research on Deliberation and Debate* Proceeding of Extra-propositional aspects of meaning in computational linguistics (ExPromM 2012), ACL-HLT 2012

[45] Hirschber, J.; Litman, D. *Empirical studies on the disambiguation of cue phrases* Computational Linguistics, 19(3):501–530, 1994

[46] Cohen, S. *A computerized scale for monitoring levels of agreement during a conversation.* In Proc. of the 26th Penn Linguistics Colloquium, 2002

[47] Bostock, M. *Streamgraph*, URL: http://bl.ocks.org/mbostock/4060954

[48] Bostock, M. *Line Chart*, URL: http://bl.ocks.org/mbostock/3883245

[49] Lars Kotthoff *Chernoff Faces for D3*, URL: http://bl.ocks.org/larskotthoff/2011590

[50] Computational Logic and Argumentation, Department of Computing, Imperial College, *Quaestio-it, Find the right answers*, URL: http://www.quaestio-it.com

[51] Jing, H.; Decheng, D. *Summarization of Yes/No Questions Using a Feature Function Model.* JMLR: Workshop and Conference Proceedings 20 (2011)

[52] *convince me start a debate.* URL: http://www.convinceme.net [2007]

[53] *InformationWeek, The Business Value of Technology.* URL: http://www.informationweek.com/ [2012]

[54] Hatzivassiloglou, V.; Wiebe, J., *Effects of adjective orientation and gradability on sentence subjectivity,* Proceedings of the 18th International Conference on Computational Linguistics, ACL, New Brunswick, NJ, 2000.

Appendix A

Experiment Results

A.1 Feature Selection Experiments For Support/Opposition Classifier

These experiments were conducted for Section 3.2.2.

1.1.1 Support Vector Machines

	TF	IDF	TF-IDF	Delta TF-IDF
1-grams	61%	60%	62%	**64%**
2-rams	53%	54%	55%	57%
3-grams	57%	58%	60%	61%
1,2,3-grams	59%	61%	63%	**65%**

Table A.1: Support Vector Machines tested on all corpus

1.1.2 Naive Bayes

	TF	IDF	TF-IDF	Delta TF-IDF
1-grams	50%	49%	58%	58%
2-rams	55%	56%	56%	57%
3-grams	57%	57%	**60%**	**61%**
1,2,3-grams	57%	55%	57%	58%

Table A.2: Naive Bayes tested on all corpus

1.1.3 Maximum Entropy

	TF	IDF	TF-IDF	Delta TF-IDF
1-grams	50%	56%	55%	55%
2-rams	48%	47%	49%	49%
3-grams	53%	55%	**56%**	**57%**
1,2,3-grams	51%	53%	54%	55%

Table A.3: Maximum Entropy tested on all corpus

A.2 Extracting Best Parameters When Trained On All the Corpus Data

These experiments were conducted for Section 3.2.3.

1.2.1 Linear SVC

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  LinearSVC())
                    ])

parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__max_n': (1, 2, 3),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'chi2__k': (50, 100, 200, 500,1000),
    'clf__penalty': ('l1','l2'),
    'clf__C': [1, 10],
    'clf__tol': [1e-6, 1e-1]
}
```

Giving the following results (elapsed time 1h42m):

```
Best score: 0.629
Best parameters set:
        chi2__k: 500
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.1: Extracting Best Parameters for LinearSVC Classification on all corpus data

Validating these parameters on my corpus we had the following results[1]:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=1e-06, verbose=0)
train time: 0.018s
test time:  0.000s
accuracy-score:   0.637
precision-score:   0.634
recall-score:   0.878
f-measure-score:   0.736
```

Figure A.2: Extracting Best Parameters for LinearSVC Classification on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	61%	61%	61%	**64%**	64%
2-rams	58%	58%	58%	59%	58%
3-grams	58%	58%	58%	58%	58%
1,2-grams	60%	60%	62%	62%	62%
2,3-grams	58%	58%	58%	58%	58%
1,2,3-grams	60%	60%	61%	63%	63%

Table A.4: Additional Experiments with LinearSVC

1.2.2 Stochastic Gradient Descent (SGD)

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
```

[1] small difference in the accuracy is due to the division of the data into train and test data sets

```
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  SGDClassifier())
                    ])

parameters = {
    'chi2__k': (50, 100, 200, 500, 1000),
    'clf__alpha': [1e-06, 0.1],
    'clf__loss': ('hinge', 'log'),
    'clf__n_iter': (5, 10, 50, 80),
    'clf__penalty': ('l1', 'l2', 'elasticnet'),
    'tfidf__norm': ('l1', 'l2'),
    'tfidf__use_idf': (True, False),
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__max_n': (1, 2, 3)
}
```

Giving the following results (elapsed time 14h31m):

```
Best score: 0.614
Best parameters set:
        chi2__k: 50
        clf__alpha: 1e-06
        clf__loss: 'hinge'
        clf__n_iter: 80
        clf__penalty: 'elasticnet'
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.3: Extracting Best Parameters for Stochastic Gradient Descent on all corpus data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Stochastic Gradient Descent
________________________________________________________________________________
Training:
SGDClassifier(alpha=0.0001, class_weight=None, epsilon=0.1, eta0=0.0,
       fit_intercept=True, l1_ratio=0.15, learning_rate=optimal,
       loss=hinge, n_iter=80, n_jobs=1, penalty=elasticnet, power_t=0.5,
       random_state=None, rho=None, shuffle=False, verbose=0,
       warm_start=False)
train time: 0.039s
test time:  0.000s
accuracy-score:   0.618
precision-score:   0.612
recall-score:   0.916
f-measure-score:   0.734
```

Figure A.4: Extracting Best Parameters for Stochastic Gradient Descent on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	**62%**	61%	59%	60%	61%
2-rams	58%	58%	59%	59%	58%
3-grams	58%	58%	58%	58%	58%
1,2-grams	61%	61%	60%	60%	60%
2,3-grams	58%	58%	59%	60%	59%
1,2,3-grams	61%	61%	61%	60%	61%

Table A.5: Additional Experiments with Stochastic Gradient Descent

1.2.3 Nearest Centroid

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  NearestCentroid())
                    ])

parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__max_n': (1, 2, 3),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'chi2__k': (50, 100, 200, 500,1000),
    'clf__n_iter': (5, 10, 50, 80),
}
```

Giving the following results (elapsed time 0h47m):

```
Best score: 0.613
Best parameters set:
        chi2__k: 1000
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.5: Extracting Best Parameters for Nearest Centroid Classification on all corpus data

Validating these parameters on my corpus we had the following results:

```
================================================================================
NearestCentroid (aka Rocchio classifier)
--------------------------------------------------------------------------------
Training:
NearestCentroid(metric=euclidean, shrink_threshold=None)
train time: 0.004s
test time:  0.001s
accuracy-score:   0.624
precision-score:   0.678
recall-score:   0.660
f-measure-score:   0.669
```

Figure A.6: Extracting Best Parameters for Nearest Centroid Classification on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	57%	59%	60%	**62%**	**62%**
2-rams	46%	45%	58%	60%	58%
3-grams	58%	58%	58%	58%	58%
1,2-grams	56%	58%	58%	61%	61%
2,3-grams	45%	45%	59%	60%	58%
1,2,3-grams	56%	58%	60%	60%	60%

Table A.6: Additional Experiments with Nearest Centroid

1.2.4 K Nearest Neighbour

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  KNeighborsClassifier())
                    ])

parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__max_n': (1, 2, 3),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'chi2__k': (50, 100, 200, 500,1000),
    'clf__n_neighbors': (5, 10, 15, 20)
}
```

Giving the following results (elapsed time 2h58m):

```
Best score: 0.614
Best parameters set:
        chi2__k: 200
        clf__n_neighbors: 20
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.7: Extracting Best Parameters for K Nearest Neighbour Classification on all corpus data

Validating these parameters on my corpus we had the following results:

```
================================================================================
K Nearest Neighbour
________________________________________________________________________________
Training:
KNeighborsClassifier(algorithm=auto, leaf_size=30, n_neighbors=20, p=2,
           warn_on_equidistant=True, weights=uniform)
train time: 0.001s
test time:  0.550s
accuracy-score:   0.604
precision-score:   0.614
recall-score:   0.844
f-measure-score:   0.711
```

Figure A.8: Extracting Best Parameters for K Nearest Neighbour Classification on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	61%	63%	60%	58%	59%
2-rams	57%	58%	58%	58%	59%
3-grams	43%	58%	58%	43%	43%
1,2-grams	54%	59%	**62%**	61%	57%
2,3-grams	57%	58%	57%	58%	59%
1,2,3-grams	55%	56%	61%	60%	57%

Table A.7: Additional Experiments with K Nearest Neighbour

1.2.5 Multinomial Naive Bayes

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  MultinomialNB())
```

```
                    ])

parameters = {
     'chi2__k': (50, 100, 200, 500, 1000),
     'clf__alpha': (1, 0.1, 0, 1, 0.001),
     'clf__fit_prior': (True, False),
     'tfidf__norm': ('l1', 'l2'),
     'tfidf__use_idf': (True, False),
     'vect__max_df': (0.5, 0.75, 1.0),
     'vect__max_features': (None, 5000, 10000, 50000),
     'vect__max_n': (1, 2, 3)
}
```

Giving the following results (elapsed time 2h21m):

```
Best score: 0.627
Best parameters set:
        chi2__k: 1000
        clf__alpha: 0.1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.9: Extracting Best Parameters for Multinomial Naive Bayes Classification on all corpus data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Naive Bayes
--------------------------------------------------------------------------------
Training:
MultinomialNB(alpha=0.1, fit_prior=True)
train time: 0.015s
test time:  0.004s
accuracy-score:   0.611
precision-score:   0.625
recall-score:   0.809
f-measure-score:   0.705
```

Figure A.10: Extracting Best Parameters for Multinomial Naive Bayes Classification on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	100	200	500	1000
1-grams	60%	60%	60%	61%	61%
2-rams	58%	59%	59%	59%	58%
3-grams	58%	58%	58%	58%	58%
1,2-grams	60%	60%	61%	60%	**63%**
2,3-grams	58%	60%	59%	60%	59%
1,2,3-grams	60%	60%	62%	61%	**62%**

Table A.8: Additional Experiments with Multinomial Naive Bayes

1.2.6 Bernoulli Naive Bayes

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  BernoulliNB())
                    ])

parameters = {
     'chi2__k': (50, 100, 200, 500, 1000),
     'clf__alpha': (1, 0.1, 0, 1, 0.001),
     'clf__fit_prior': (True, False),
     'tfidf__norm': ('l1', 'l2'),
     'tfidf__use_idf': (True, False),
     'vect__max_df': (0.5, 0.75, 1.0),
     'vect__max_features': (None, 5000, 10000, 50000),
     'vect__max_n': (1, 2, 3)
}
```

Giving the following results (elapsed time 1h42m):

```
Best score: 0.628
Best parameters set:
        chi2__k: 1000
        clf__alpha: 1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.11: Extracting Best Parameters for Bernoulli Naive Bayes Classification on all corpus data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=1, binarize=0.0, fit_prior=True)
train time: 0.002s
test time:  0.001s
accuracy-score:   0.622
precision-score:   0.638
recall-score:   0.794
f-measure-score:   0.707
```

Figure A.12: Extracting Best Parameters for Bernoulli Naive Bayes Classification on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	61%	60%	62%	61%	62%
2-rams	58%	59%	59%	59%	58%
3-grams	58%	58%	58%	58%	58%
1,2-grams	61%	60%	62%	60%	**64%**
2,3-grams	58%	59%	59%	60%	57%
1,2,3-grams	62%	59%	62%	60%	**64%**

Table A.9: Additional Experiments with Bernoulli Naive Bayes

1.2.7 Logistic Regression

```
pipeline = Pipeline([
                    ('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('chi2', SelectKBest(chi2)),
                    ('clf',  LogisticRegression())
                    ])

parameters = {
    'vect__max_df': (0.5, 0.75, 1.0),
    'vect__max_features': (None, 5000, 10000, 50000),
    'vect__max_n': (1, 2, 3),
    'tfidf__use_idf': (True, False),
    'tfidf__norm': ('l1', 'l2'),
    'chi2__k': (50, 100, 200, 500,1000),
    'clf__penalty': ('l1','l2'),
    'clf__C': [1, 10],
    'clf__tol': (1e-6, 1e-1),
}
```

Giving the following results (elapsed time 1h53m):

```
Best score: 0.635
Best parameters set:
        chi2__k: 1000
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.13: Extracting Best Parameters for Logistic Regression on all corpus data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l2, random_state=None, tol=0.1)
train time: 0.004s
test time:  0.000s
accuracy-score:   0.648
precision-score:   0.644
recall-score:   0.870
f-measure-score:   0.740
```

Figure A.14: Extracting Best Parameters for Logistic Regression Classification on all corpus data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	61%	60%	62%	64%	**65%**
2-rams	58%	59%	59%	58%	58%
3-grams	58%	58%	58%	58%	58%
1,2-grams	61%	60%	60%	62%	**64%**
2,3-grams	58%	59%	59%	58%	59%
1,2,3-grams	61%	60%	60%	62%	64%

Table A.10: Additional Experiments with Logistic Regression

A.3 Extracting Best Parameters When Trained On One Topic At A Time

These experiments were conducted for Section 3.2.4.

1.3.1 Abortion

A.3.1.1 Linear SVC

Giving the following results (elapsed time 1h20m):

```
Best score: 0.630
Best parameters set:
        chi2__k: 1000
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.15: Extracting Best Parameters for Linear SVC Classification on abortion data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1.0, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=1e-06, verbose=0)
accuracy-score:   0.752
precision-score:   0.732
recall-score:   0.909
f-measure-score:   0.807
```

Figure A.16: Results for Linear SVC Classification on abortion data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	66%	67%	72%	74%	**75%**
2-rams	59%	63%	61%	58%	58%
3-grams	58%	57%	58%	60%	59%
1,2-grams	64%	65%	67%	70%	74%
2,3-grams	59%	60%	60%	57%	61%
1,2,3-grams	68%	64%	69%	65%	70%

Table A.11: Additional Experiments with Linear SVC on abortion corpus

A.3.1.2 Multinomial Naive Bayes

Giving the following results (elapsed time 1h25m):

```
Best score: 0.643
Best parameters set:
        chi2__k: 1000
        clf__alpha: 0.1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.17: Extracting Best Parameters for Multinomial Naive Bayes Classification on abortion data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=0.1, fit_prior=True)
accuracy-score:   0.692
precision-score:   0.703
recall-score:   0.801
f-measure-score:   0.744
```

Figure A.18: Results for Multinomial Naive Bayes Classification on abortion data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	66%	66%	71%	**74%**	60%
2-rams	61%	60%	64%	61%	60%
3-grams	58%	58%	60%	59%	59%
1,2-grams	65%	66%	69%	70%	70%
2,3-grams	59%	60%	62%	62%	62%
1,2,3-grams	64%	64%	66%	68%	69%

Table A.12: Additional Experiments with Multinomial Naive Bayes on abortion corpus

A.3.1.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 1h35m):

```
Best score: 0.623
Best parameters set:
        chi2__k: 500
        clf__alpha: 1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.19: Extracting Best Parameters for Bernoulli Naive Bayes Classification on abortion data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=1, binarize=0.0, fit_prior=True)
accuracy-score:   0.668
precision-score:   0.674
recall-score:   0.780
f-measure-score:   0.721
```

Figure A.20: Results for Bernoulli Naive Bayes Classification on abortion data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	65%	67%	68%	67%	66%
2-rams	62%	61%	63%	61%	61%
3-grams	58%	58%	59%	58%	58%
1,2-grams	61%	63%	64%	67%	**69%**
2,3-grams	60%	63%	63%	62%	58%
1,2,3-grams	61%	62%	64%	64%	63%

Table A.13: Additional Experiments with Bernoulli Naive Bayes on abortion corpus

A.3.1.4 Logistic Regression

Giving the following results (elapsed time 1h15m):

```
Best score: 0.632
Best parameters set:
        chi2__k: 1000
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.21: Extracting Best Parameters for Logistic Regression Classification on abortion data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l2, random_state=None, tol=1e-06)
accuracy-score:   0.724
precision-score:   0.725
recall-score:   0.855
f-measure-score:   0.780
```

Figure A.22: Results for Logistic Regression Classification on abortion data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	66%	70%	71%	73%	72%
2-rams	61%	61%	64%	62%	62%
3-grams	58%	58%	60%	60%	59%
1,2-grams	65%	69%	70%	72%	72%
2,3-grams	60%	60%	61%	60%	63%
1,2,3-grams	63%	69%	69%	**74%**	72%

Table A.14: Additional Experiments with Logistic Regression on abortion corpus

Most informative features:

```
top 20 keywords per class:
        -1.9626 arbitrari               1.8392  sexual
        -1.4652 half                    1.3857  live
        -1.4290 atheist                 1.3733  can
        -1.3285 cal                     1.3690  obvious
        -1.2264 record                  1.3283  hurt
        -1.1765 seriou                  1.2568  liberti
        -1.1640 histor                  1.0280  issu
        -1.0676 check                   1.0192  hypocrisi
        -1.0445 activist                1.0053  eye
        -0.9702 onc                     1.0050  ago
        -0.9667 beat                    0.9352  consid
        -0.9471 accident                0.9210  implement
        -0.9237 open                    0.9090  banned
        -0.9165 procedur                0.8293  breath
        -0.8902 revers                  0.8162  find
        -0.8697 innoc                   0.8064  everyth
        -0.8595 flawless                0.7664  educ
        -0.8565 iraq                    0.7517  plan
        -0.8332 old                     0.7466  due
        -0.8018 instanc                 0.7321  afford
```

Figure A.23: Most Informative Features for support and opposition when trained on the abortion corpus

1.3.2 Creationism

A.3.2.1 Linear SVC

Giving the following results (elapsed time 1h20m):

```
Best score: 0.671
Best parameters set:
        chi2__k: 200
        clf__C: 10
        clf__penalty: 'l2'
        clf__tol: 0.1
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.24: Extracting Best Parameters for Linear SVC Classification on creationism data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=10, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=0.1, verbose=0)
accuracy-score:   0.704
precision-score:   0.549
recall-score:   0.543
f-measure-score:   0.272
```

Figure A.25: Results for Linear SVC Classification on creationism data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	74%	75%	70%	71%	70%
2-rams	73%	70%	68%	67%	73%
3-grams	**76%**	**76%**	**76%**	75%	75%
1,2-grams	76%	76%	74%	**76%**	74%
2,3-grams	72%	71%	69%	69%	72%
1,2,3-grams	**76%**	**76%**	**76%**	**76%**	75%

Table A.15: Additional Experiments with Linear SVC on creationism corpus

A.3.2.2 Multinomial Naive Bayes

Giving the following results (elapsed time 0h25m):

```
Best score: 0.671
Best parameters set:
        chi2__k: 500
        clf__alpha: 1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.26: Extracting Best Parameters for Multinomial Naive Bayes Classification on creationism data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=1, fit_prior=True)
accuracy-score:   0.764
precision-score:   0.605
recall-score:   0.551
f-measure-score:   0.285
```

Figure A.27: Results for Multinomial Naive Bayes Classification on creationism data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	73%	72%	75%	70%	71%
2-rams	75%	75%	74%	71%	70%
3-grams	76%	76%	76%	75%	75%
1,2-grams	75%	75%	74%	76%	76%
2,3-grams	75%	76%	76%	74%	72%
1,2,3-grams	76%	75%	76%	76%	75%

Table A.16: Additional Experiments with Multinomial Naive Bayes on creationism corpus

A.3.2.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 0h35m):

```
Best score: 0.665
Best parameters set:
        chi2__k: 500
        clf__alpha: 0
        clf__fit_prior: True
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.28: Extracting Best Parameters for Bernoulli Naive Bayes Classification on creation data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=0, binarize=0.0, fit_prior=True)
accuracy-score:   0.696
precision-score:   0.585
recall-score:   0.584
f-measure-score:   0.292
```

Figure A.29: Results for Bernoulli Naive Bayes Classification on creationism data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	62%	60%	64%	70%	75%
2-rams	74%	74%	**76%**	74%	72%
3-grams	**76%**	**76%**	**76%**	75%	75%
1,2-grams	70%	67%	66%	67%	68%
2,3-grams	74%	72%	74%	75%	74%
1,2,3-grams	69%	66%	67%	66%	66%

Table A.17: Additional Experiments with Bernoulli Naive Bayes on creationism corpus

A.3.2.4 Logistic Regression

Giving the following results (elapsed time 0h55m):

```
Best score: 0.679
Best parameters set:
        chi2__k: 800
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.30: Extracting Best Parameters for Logistic Regression Classification on creationism data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l2, random_state=None, tol=1e-06)
accuracy-score:   0.752
precision-score:   0.615
recall-score:   0.608
f-measure-score:   0.304
```

Figure A.31: Results for Logistic Regression Classification on creationism data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	74%	74%	74%	72%	75%
2-rams	76%	**77%**	**77%**	**77%**	76%
3-grams	76%	76%	76%	76%	76%
1,2-grams	75%	76%	75%	74%	74%
2,3-grams	76%	76%	76%	76%	76%
1,2,3-grams	76%	76%	75%	76%	76%

Table A.18: Additional Experiments with Logistic Regression on creationism corpus

Most informative features:

```
top 20 keywords per class:
        -0.9076 evolut process          0.9044  evolut real
        -0.6967 child abus              0.8405  god not
        -0.6507 easi understand         0.8205  day creation
        -0.6096 far tell                0.7812  even second
        -0.5968 law land                0.7762  design evolut
        -0.5681 creation evolut         0.7292  discoveri nonsens
        -0.5516 happen not              0.7234  evolut true
        -0.5469 christian belief                0.7203  featur earth
        -0.5238 biolog make             0.6682  comprehend it
        -0.5208 evil could              0.6650  no scientif
        -0.5195 cannot explain          0.6494  not give
        -0.5180 not idea                0.6249  creationist not
        -0.5105 filter atmospher                0.6242  could creat
        -0.5035 evolut realli           0.6193  first caus
        -0.5031 believ liter            0.6014  not demonstr
        -0.5001 day evolut              0.6009  evolut occur
        -0.4933 know god                0.5883  allow peopl
        -0.4933 much sure               0.5865  base faith
        -0.4705 not involv              0.5864  fact understand
        -0.4652 evolutionari theori             0.5859  appear sky
```

Figure A.32: Most Informative Features for support and opposition when trained on the creationism corpus

1.3.3 Gay Rights

A.3.3.1 Linear SVC

Giving the following results (elapsed time 0h54m):

```
Best score: 0.669
Best parameters set:
        chi2__k: 500
        clf__C: 10
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.33: Extracting Best Parameters for Linear SVC Classification on gay rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L2 penalty Linear SVC
--------------------------------------------------------------------------------
Training:
LinearSVC(C=10, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=1e-06, verbose=0)
accuracy-score:   0.616
precision-score:   0.581
recall-score:   0.545
f-measure-score:   0.281
```

Figure A.34: Results for Linear SVC Classification on gay rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	64%	64%	62%	62%	66%
2-rams	61%	60%	62%	63%	63%
3-grams	62%	62%	63%	63%	63%
1,2-grams	62%	**66%**	65%	65%	65%
2,3-grams	60%	59%	62%	62%	64%
1,2,3-grams	61%	62%	64%	65%	65%

Table A.19: Additional Experiments with Linear SVC on gay rights corpus

A.3.3.2 Multinomial Naive Bayes

Giving the following results (elapsed time 0h25m):

```
Best score: 0.660
Best parameters set:
        chi2__k: 500
        clf__alpha: 0.1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.35: Extracting Best Parameters for Multinomial Naive Bayes Classification on gay rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=0.1, fit_prior=True)
accuracy-score:   0.608
precision-score:   0.571
recall-score:   0.535
f-measure-score:   0.276
```

Figure A.36: Results for Multinomial Naive Bayes Classification on gay rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	60%	63%	**64%**	61%	62%
2-rams	61%	61%	62%	62%	62%
3-grams	61%	61%	61%	61%	62%
1,2-grams	62%	62%	62%	62%	63%
2,3-grams	62%	61%	62%	64%	63%
1,2,3-grams	61%	62%	62%	63%	**64%**

Table A.20: Additional Experiments with Multinomial Naive Bayes on gay rights corpus

A.3.3.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 0h35m):

```
Best score: 0.652
Best parameters set:
        chi2__k: 500
        clf__alpha: 1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.37: Extracting Best Parameters for Bernoulli Naive Bayes Classification on gay rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=1, binarize=0.0, fit_prior=True)
accuracy-score:   0.676
precision-score:   0.682
recall-score:   0.606
f-measure-score:   0.321
```

Figure A.38: Results for Bernoulli Naive Bayes Classification on gay rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	65%	64%	64%	64%	**68%**
2-rams	63%	64%	63%	61%	61%
3-grams	62%	62%	62%	61%	61%
1,2-grams	64%	64%	64%	63%	63%
2,3-grams	62%	63%	62%	63%	61%
1,2,3-grams	66%	61%	64%	63%	63%

Table A.21: Additional Experiments with Bernoulli Naive Bayes on gay rights corpus

A.3.3.4 Logistic Regression

Giving the following results (elapsed time 0h55m):

```
Best score: 0.661
Best parameters set:
        chi2__k: 500
        clf__C: 10
        clf__penalty: 'l1'
        clf__tol: 0.1
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: 10000
        vect__max_n: 1
```

Figure A.39: Extracting Best Parameters for Logistic Regression Classification on gay rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l1, random_state=None, tol=0.1)
accuracy-score:   0.656
precision-score:   0.654
recall-score:   0.593
f-measure-score:   0.311
```

Figure A.40: Results for Logistic Regression Classification on gayrights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	65%	64%	66%	66%	64%
2-rams	61%	60%	63%	64%	61%
3-grams	62%	62%	62%	62%	63%
1,2-grams	65%	66%	**67%**	65%	66%
2,3-grams	61%	60%	61%	61%	64%
1,2,3-grams	65%	65%	65%	64%	64%

Table A.22: Additional Experiments with Logistic Regression on gay rights corpus

Most informative features:

```
top 10 keywords per class:
        -46.0931        anim speci              26.7507 allow marri
        -37.5824        ani studi               23.6606 aggress behavior
        -29.1937        apolog                  19.7204 abstract argument
        -26.1366        argument gay            17.9242 ask ani
        -25.8212        ani choic               17.7177 advoc
        -24.9507        ask not                 15.5675 affect
        -24.0294        adequ                   14.1970 asexu
        -23.7393        babi                    13.2856 away choic
        -20.9499        anoth respons           12.7451 anyth neither
        -18.6528        attach stereotyp                12.4135 accept ani
        -18.2702        anti shrimp             11.3235 almost
        -18.1316        adulter                 11.2964 agenda globe
        -17.2255        act                     10.6306 affect right
        -16.3461        again should            10.5772 actual look
        -16.0366        alreadi exist           10.1778 actual right
        -15.8701        argument wors           9.8801  alway benefit
        -15.2593        assert                  9.0674  actual want
        -14.4567        among influenc          7.7854  actual plan
        -12.7721        actual understand               7.5980  ab sinc
        -12.4857        adopt mere              7.3001  approv coupl
```

Figure A.41: Most Informative Features for support and opposition when trained on the gayRights corpus

1.3.4 Existence of God

A.3.4.1 Linear SVC

Giving the following results (elapsed time 0h54m):

```
Best score: 0.596
Best parameters set:
        chi2__k: 100
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.42: Extracting Best Parameters for Linear SVC Classification on Existence of God data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=1e-06, verbose=0)
accuracy-score:   0.548
precision-score:   0.583
recall-score:   0.568
f-measure-score:   0.288
```

Figure A.43: Results for Linear SVC Classification on Existence of God data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	55%	52%	52%	54%	50%
2-rams	59%	57%	54%	51%	51%
3-grams	54%	52%	52%	52%	51%
1,2-grams	57%	59%	58%	56%	56%
2,3-grams	60%	59%	59%	55%	54%
1,2,3-grams	56%	62%	**62%**	59%	56%

Table A.23: Additional Experiments with Linear SVC on Existence of God corpus

A.3.4.2 Multinomial Naive Bayes

Giving the following results (elapsed time 0h25m):

```
Best score: 0.599
Best parameters set:
        chi2__k: 500
        clf__alpha: 0.1
        clf__fit_prior: True
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.44: Extracting Best Parameters for Multinomial Naive Bayes Classification on Existence of God data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=0.1, fit_prior=True)
accuracy-score:   0.548
precision-score:   0.618
recall-score:   0.542
f-measure-score:   0.287
```

Figure A.45: Results for Multinomial Naive Bayes Classification on Existence of God data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	52%	54%	49%	52%	50%
2-rams	57%	58%	52%	49%	50%
3-grams	53%	52%	52%	51%	51%
1,2-grams	52%	51%	52%	49%	50%
2,3-grams	**59%**	56%	56%	51%	50%
1,2,3-grams	53%	55%	53%	53%	51%

Table A.24: Additional Experiments with Multinomial Naive Bayes on Existence of God corpus

A.3.4.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 0h35m):

```
Best score: 0.579
Best parameters set:
        chi2__k: 200
        clf__alpha: 1
        clf__fit_prior: False
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.46: Extracting Best Parameters for Bernoulli Naive Bayes Classification on Existence of God data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=1, binarize=0.0, fit_prior=True)
accuracy-score:   0.472
precision-score:   0.475
recall-score:   0.483
f-measure-score:   0.239
```

Figure A.47: Results for Bernoulli Naive Bayes Classification on Existence of God data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	49%	50%	47%	50%	52%
2-rams	57%	57%	48%	47%	57%
3-grams	53%	52%	39%	52%	52%
1,2-grams	52%	54%	54%	50%	50%
2,3-grams	58%	56%	**60%**	51%	49%
1,2,3-grams	58%	58%	56%	48%	47%

Table A.25: Additional Experiments with Bernoulli Naive Bayes on Existence of God corpus

A.3.4.4 Logistic Regression

Giving the following results (elapsed time 0h55m):

```
Best score: 0.603
Best parameters set:
        chi2__k: 500
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.48: Extracting Best Parameters for Logistic Regression Classification on Existence of God data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
--------------------------------------------------------------------------------
Training:
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l2, random_state=None, tol=0.1)
accuracy-score:   0.528
precision-score:   0.529
recall-score:   0.533
f-measure-score:   0.265
```

Figure A.49: Results for Logistic Regression Classification on Existence of God data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	48%	50%	53%	53%	54%
2-rams	58%	57%	58%	57%	58%
3-grams	54%	53%	53%	52%	52%
1,2-grams	58%	57%	58%	54%	55%
2,3-grams	**60%**	59%	58%	58%	57%
1,2,3-grams	59%	57%	56%	56%	54%

Table A.26: Additional Experiments with Logistic Regression on Existence of God corpus

Most informative features:

```
top 20 keywords per class:
        -2.0067 among                   2.3989  argument
        -1.6571 associ god              1.6104  action
        -1.3722 atheist                 1.4101  across
        -1.1404 accept thi              1.3102  around
        -1.0442 absenc                  1.1482  around truth
        -1.0322 actual                  1.1016  argument becaus
        -1.0287 also                    1.0555  ani god
        -0.9209 amaz                    0.9998  belief no
        -0.8786 amount dumb joke                0.9981  argument provid
        -0.8686 answer question         0.9029  becaus no proof
        -0.8464 assum god               0.8749  becaus
        -0.7765 accept initi            0.8516  be
        -0.7700 bang theori             0.8325  ask god help
        -0.7324 assum                   0.7624  alway
        -0.7283 argument seem           0.7620  africa
        -0.7086 belief god              0.7331  attack
        -0.6994 bad thing happen                0.7245  accid
        -0.6976 argu                    0.6954  argument onli
        -0.6841 angel                   0.6928  alreadi
        -0.6779 becaus cannot           0.6925  anyth not
```

Figure A.50: Most Informative Features for support and opposition when trained on the Existence of God corpus

1.3.5 Gun Rights

A.3.5.1 Linear SVC

Giving the following results (elapsed time 0h54m):

```
Best score: 0.746
Best parameters set:
        chi2__k: 200
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.51: Extracting Best Parameters for Linear SVC Classification on gun rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=0.1, verbose=0)
accuracy-score:   0.756
precision-score:   0.564
recall-score:   0.534
f-measure-score:   0.274
```

Figure A.52: Results for Linear SVC Classification on gun rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	74%	75%	**76%**	74%	75%
2-rams	74%	74%	73%	75%	74%
3-grams	**76%**	74%	74%	74%	73%
1,2-grams	72%	71%	74%	**76%**	74%
2,3-grams	74%	75%	**76%**	75%	**76%**
1,2,3-grams	72%	72%	74%	73%	74%

Table A.27: Additional Experiments with Linear SVC on gun rights corpus

A.3.5.2 Multinomial Naive Bayes

Giving the following results (elapsed time 0h25m):

```
Best score: 0.748
Best parameters set:
        chi2__k: 500
        clf__alpha: 0.1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.53: Extracting Best Parameters for Multinomial Naive Bayes Classification on gun rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=0.1, fit_prior=True)
accuracy-score:   0.756
precision-score:   0.378
recall-score:   0.500
f-measure-score:   0.214
```

Figure A.54: Results for Multinomial Naive Bayes Classification on gun rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	76%	76%	77%	**78%**	77%
2-rams	76%	76%	76%	76%	74%
3-grams	76%	76%	76%	76%	75%
1,2-grams	76%	76%	76%	76%	75%
2,3-grams	76%	76%	76%	76%	75%
1,2,3-grams	76%	76%	76%	76%	76%

Table A.28: Additional Experiments with Multinomial Naive Bayes on gun rights corpus

A.3.5.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 0h35m):

```
Best score: 0.748
Best parameters set:
        chi2__k: 500
        clf__alpha: 1
        clf__fit_prior: True
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.55: Extracting Best Parameters for Bernoulli Naive Bayes Classification on gun rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=1, binarize=0.0, fit_prior=True)
accuracy-score:   0.768
precision-score:   0.713
recall-score:   0.561
f-measure-score:   0.311

[(0.7679999999999846, 'BernoulliNB')]
```

Figure A.56: Results for Bernoulli Naive Bayes Classification on gun rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	73%	76%	77%	77%	77%
2-rams	74%	76%	76%	75%	57%
3-grams	76%	76%	75%	75%	76%
1,2-grams	73%	**78%**	**78%**	77%	76%
2,3-grams	74%	76%	76%	75%	76%
1,2,3-grams	72%	76%	**78%**	76%	76%

Table A.29: Additional Experiments with Bernoulli Naive Bayes on gun rights corpus

A.3.5.4 Logistic Regression

Giving the following results (elapsed time 0h55m):

```
Best score: 0.748
Best parameters set:
        chi2__k: 500
        clf__C: 10
        clf__penalty: 'l2'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.57: Extracting Best Parameters for Logistic Regression Classification on gun rights data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l2, random_state=None, tol=0.1)
accuracy-score:   0.740
precision-score:   0.627
recall-score:   0.532
f-measure-score:   0.285
```

Figure A.58: Results for Logistic Regression Classification on gun rights data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	76%	**79%**	78%	74%	75%
2-rams	74%	74%	73%	74%	74%
3-grams	75%	74%	75%	75%	73%
1,2-grams	75%	75%	73%	76%	76%
2,3-grams	74%	76%	74%	72%	75%
1,2,3-grams	74%	75%	76%	73%	76%

Table A.30: Additional Experiments with Logistic Regression on gun rights corpus

Most informative features:

```
top 20 keywords per class:
        -5.3375 arm                     3.4828  afford
        -4.4248 act                     3.4346  accord
        -4.1654 assum                   2.9143  background
        -4.0571 advantag                2.7146  also
        -4.0382 amend                   2.6083  alcohol
        -3.9782 advoc                   2.5830  action
        -3.9736 ani                     2.5414  attract
        -3.7232 actual                  2.2606  abil
        -3.7138 ahead                   -0.9132 attack
        -3.6962 ammunit                 -0.9132 aliv
        -3.5267 appar                   -0.9132 activ
        -3.4957 ask                     -1.0120 american
        -3.4421 accident                -1.0556 anim
        -3.3643 back                    -1.1982 ban
        -3.3297 appear                  -1.2575 anyth
        -3.2599 appli                   -1.2575 alreadi
        -3.2580 around                  -1.2769 argument
        -3.2281 access                  -1.2769 anybodi
        -3.1273 although                -1.2769 airplan
        -3.0224 awar                    -1.2769 addit
```

Figure A.59: Most Informative Features for support and opposition when trained on the gun rights corpus

1.3.6 Healthcare

A.3.6.1 Linear SVC

Giving the following results (elapsed time 0h54m):

```
Best score: 0.594
Best parameters set:
        chi2__k: 50
        clf__C: 1
        clf__penalty: 'l1'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 1.0
        vect__max_features: None
        vect__max_n: 1
```

Figure A.60: Extracting Best Parameters for Linear SVC Classification on healthcare data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L1 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l1,
     random_state=None, tol=0.1, verbose=0)
accuracy-score:   0.568
precision-score:   0.571
recall-score:   0.544
f-measure-score:   0.279
```

Figure A.61: Results for Linear SVC Classification on healthcare data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	57%	53%	52%	44%	45%
2-rams	51%	49%	46%	50%	48%
3-grams	59%	52%	52%	45%	45%
1,2-grams	54%	53%	52%	49%	46%
2,3-grams	**56%**	50%	47%	50%	49%
1,2,3-grams	50%	52%	53%	54%	54%

Table A.31: Additional Experiments with Linear SVC on healthcare corpus

A.3.6.2 Multinomial Naive Bayes

Giving the following results (elapsed time 0h25m):

```
Best score: 0.597
Best parameters set:
        chi2__k: 50
        clf__alpha: 0
        clf__fit_prior: False
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.75
        vect__max_features: None
        vect__max_n: 1
```

Figure A.62: Extracting Best Parameters for Multinomial Naive Bayes Classification on healthcare data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=0, fit_prior=True)
accuracy-score:   0.524
precision-score:   0.521
recall-score:   0.512
f-measure-score:   0.258
```

Figure A.63: Results for Multinomial Naive Bayes Classification on healthcare data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	53%	52%	47%	46%	48%
2-rams	52%	54%	52%	52%	52%
3-grams	52%	50%	50%	51%	49%
1,2-grams	**57%**	55%	53%	51%	50%
2,3-grams	51%	52%	54%	53%	54%
1,2,3-grams	52%	53%	55%	51%	48%

Table A.32: Additional Experiments with Multinomial Naive Bayes on healthcare corpus

A.3.6.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 0h35m):

```
Best score: 0.597
Best parameters set:
        chi2__k: 500
        clf__alpha: 0.001
        clf__fit_prior: True
        tfidf__norm: 'l1'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.64: Extracting Best Parameters for Bernoulli Naive Bayes Classification on healthcare data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
--------------------------------------------------------------------------------
Training:
BernoulliNB(alpha=0.001, binarize=0.0, fit_prior=True)
accuracy-score:   0.532
precision-score:   0.529
recall-score:   0.525
f-measure-score:   0.263
```

Figure A.65: Results for Bernoulli Naive Bayes Classification on healthcare data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	50%	50%	56%	54%	53%
2-rams	55%	50%	52%	45%	52%
3-grams	51%	45%	54%	50%	54%
1,2-grams	53%	44%	50%	55%	53%
2,3-grams	56%	51%	52%	48%	44%
1,2,3-grams	**57%**	51%	55%	55%	55%

Table A.33: Additional Experiments with Bernoulli Naive Bayes on healthcare corpus

A.3.6.4 Logistic Regression

Giving the following results (elapsed time 0h55m):

```
Best score: 0.600
Best parameters set:
        chi2__k: 500
        clf__C: 1
        clf__penalty: 'l1'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 1.0
        vect__max_features: 5000
        vect__max_n: 1
```

Figure A.66: Extracting Best Parameters for Logistic Regression Classification on healthcare data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=0.001, binarize=0.0, fit_prior=True)
accuracy-score:   0.536
precision-score:   0.534
recall-score:   0.530
f-measure-score:   0.266
```

Figure A.67: Results for Logistic Regression Classification on healthcare data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	48%	50%	51%	48%	-%
2-rams	49%	47%	49%	48%	52%
3-grams	**60%**	58%	58%	58%	58%
1,2-grams	50%	51%	48%	48%	44%
2,3-grams	55%	53%	48%	46%	46%
1,2,3-grams	51%	51%	51%	48%	47%

Table A.34: Additional Experiments with Logistic Regression on healthcare corpus

Most informative features:

```
top 20 keywords per class:
        -4.8785 doctor health care              4.7100  check sourc first
        -4.2181 care doe not            4.3824  cost becaus american
        -3.9431 coverag nation review           4.3727  american medic associ
        -3.7982 competit privat insur           3.4352  cost ensur individu
        -3.7555 care cost ensur         3.4241  better health care
        -3.4421 cost health insur               3.4169  afford health care
        -3.3897 cost gain enforc                3.2366  cost not attribut
        -3.2593 becaus fact not         3.2189  answer whether univers
        -3.2376 care cost not           3.1488  doctor univers health
        -2.8813 alreadi univers health          3.0790  care american balanc
        -2.8632 burden tax payer                2.9538  care seattl time
        -2.6464 american prospect decemb                2.9381  due indebt life
        -2.6354 basic human right               2.8851  care cost becaus
        -2.6006 canada wrong model              2.8669  care basic human
        -2.5873 america futur public            2.8370  doe not mean
        -2.5394 anyth less highway              2.8288  care reform beginn
        -2.4860 coverag ensur peopl             2.7357  care not right
        -2.3577 becaus parkway dure             1.9387  care system not
        -1.8221 cost public insur               1.8383  case public plan
        -1.6054 actual read primari             1.8268  cost univers health
```

Figure A.68: Most Informative Features for support and opposition when trained on the healthcare corpus

1.3.7 Quality

A.3.7.1 Linear SVC

Giving the following results (elapsed time 0h54m):

```
Best score: 0.746
Best parameters set:
        chi2__k: 500
        clf__C: 1
        clf__penalty: 'l2'
        clf__tol: 1e-06
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.69: Extracting Best Parameters for Linear SVC Classification on quality data

Validating these parameters on my corpus we had the following results:

```
================================================================================
L2 penalty Linear SVC
________________________________________________________________________________
Training:
LinearSVC(C=1, class_weight=None, dual=False, fit_intercept=True,
     intercept_scaling=1, loss=l2, multi_class=ovr, penalty=l2,
     random_state=None, tol=1e-06, verbose=0)
accuracy-score:   0.788
precision-score:   0.733
recall-score:   0.696
f-measure-score:   0.357
```

Figure A.70: Results for Linear SVC Classification on quality data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	74%	75%	70%	**79%**	-
2-rams	72%	71%	72%	69%	69%
3-grams	74%	75%	75%	75%	45%
1,2-grams	76%	74%	74%	76%	76%
2,3-grams	72%	73%	73%	72%	72%
1,2,3-grams	75%	75%	74%	75%	**79%**

Table A.35: Additional Experiments with Linear SVC on quality corpus

A.3.7.2 Multinomial Naive Bayes

Giving the following results (elapsed time 0h25m):

```
Best score: 0.741
Best parameters set:
        chi2__k: 500
        clf__alpha: 1
        clf__fit_prior: False
        tfidf__norm: 'l1'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.71: Extracting Best Parameters for Multinomial Naive Bayes Classification on quality data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Multinomial Naive Bayes
________________________________________________________________________________
Training:
MultinomialNB(alpha=1, fit_prior=True)
accuracy-score:   0.720
precision-score:   0.360
recall-score:   0.500
f-measure-score:   0.209
```

Figure A.72: Results for Multinomial Naive Bayes Classification on quality data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	72%	72%	72%	72%	-
2-rams	70%	70%	70%	71%	70%
3-grams	**73%**	**73%**	**73%**	**73%**	**73%**
1,2-grams	72%	72%	72%	72%	72%
2,3-grams	70%	71%	70%	70%	70%
1,2,3-grams	72%	72%	72%	72%	72%

Table A.36: Additional Experiments with Multinomial Naive Bayes on quality corpus

A.3.7.3 Bernoulli Naive Bayes

Giving the following results (elapsed time 0h35m):

```
Best score: 0.749
Best parameters set:
        chi2__k: 200
        clf__alpha: 1
        clf__fit_prior: True
        tfidf__norm: 'l2'
        tfidf__use_idf: True
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.73: Extracting Best Parameters for Bernoulli Naive Bayes Classification on quality data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Bernoulli Naive Bayes
________________________________________________________________________________
Training:
BernoulliNB(alpha=1, binarize=0.0, fit_prior=True)
accuracy-score:   0.720
precision-score:   0.631
recall-score:   0.607
f-measure-score:   0.309
```

Figure A.74: Results for Bernoulli Naive Bayes Classification on quality data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	71%	69%	72%	**79%**	-
2-rams	74%	74%	74%	74%	72%
3-grams	75%	74%	72%	72%	72%
1,2-grams	73%	76%	72%	71%	-
2,3-grams	75%	74%	74%	73%	72%
1,2,3-grams	74%	76%	74%	73%	73%

Table A.37: Additional Experiments with Bernoulli Naive Bayes on quality corpus

A.3.7.4 Logistic Regression

Giving the following results (elapsed time 0h55m):

```
Best score: 0.752
Best parameters set:
        chi2__k: 200
        clf__C: 10
        clf__penalty: 'l1'
        clf__tol: 0.1
        tfidf__norm: 'l2'
        tfidf__use_idf: False
        vect__max_df: 0.5
        vect__max_features: None
        vect__max_n: 1
```

Figure A.75: Extracting Best Parameters for Logistic Regression Classification on quality data

Validating these parameters on my corpus we had the following results:

```
================================================================================
Logistic Regression
________________________________________________________________________________
Training:
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, penalty=l1, random_state=None, tol=0.1)
accuracy-score:   0.752
precision-score:   0.722
recall-score:   0.658
f-measure-score:   0.343
```

Figure A.76: Results for Logistic Regression Classification on quality data

Additional Experiments using different n-grams and maximum feature selection:

	50	**100**	**200**	**500**	**1000**
1-grams	72%	70%	75%	74%	-
2-rams	73%	75%	73%	69%	75%
3-grams	75%	75%	75%	75%	75%
1,2-grams	72%	74%	71%	73%	**76%**
2,3-grams	74%	71%	73%	73%	69%
1,2,3-grams	72%	74%	72%	73%	74%

Table A.38: Additional Experiments with Logistic Regression on quality corpus

Most informative features:

```
top 20 keywords per class:
        -3.2747 firm                    2.2659  daili
        -2.1655 front                   2.0897  believ
        -2.1210 camera                  2.0344  fantast
        -2.0952 favorit                 1.6185  cylind
        -1.8838 interview               1.5383  batteri
        -1.6279 luxuri                  1.5018  gen
        -1.5906 famili                  1.4136  acceler
        -1.4671 determin                1.3989  enabl
        -1.3275 flaw                    1.3774  chang
        -1.3229 impress                 1.3016  area
        -1.1836 live                    1.2990  averag
        -1.1778 flip                    1.2634  dryer
        -1.1732 dealer                  1.2378  advertis
        -1.1708 expect                  1.2274  click
        -1.1288 caught                  1.1908  english
        -1.1170 fashion                 1.1359  clear
        -1.0847 level                   1.1354  menu
        -1.0769 ampl                    1.0756  cheap
        -1.0586 hot                     1.0435  dine
        -1.0400 besid                   1.0207  fresh
```

Figure A.77: Most Informative Features for support and opposition when trained on the quality corpus

A.4 Experiment Results When Using Part Of Speech Tagging

These experiments were conducted for Section 3.2.5.

1.4.1 Abortion

A.4.1.1 Linear SVC

	50	**100**	**200**	**500**	**1000**
1-grams	65%	71%	72%	73%	73%
2-rams	61%	59%	61%	57%	56%
3-grams	58%	59%	59%	59%	58%
1,2-grams	65%	70%	68%	72%	73%
2,3-grams	61%	59%	61%	58%	58%
1,2,3-grams	65%	71%	72%	72%	73%

Table A.39: Experiments with LinearSVC using POS tagging on abortion corpus

A.4.1.2 Multinomial Naive Bayes

	50	**100**	**200**	**500**	**1000**
1-grams	65%	70%	68%	72%	73%
2-rams	61%	59%	61%	58%	58%
3-grams	58%	59%	59%	59%	58%
1,2-grams	64%	64%	70%	72%	74%
2,3-grams	61%	59%	62%	60%	60%
1,2,3-grams	64%	67%	67%	71%	72%

Table A.40: Experiments with Multinomial Naive Bayes using POS tagging on abortion corpus

A.4.1.3 Bernoulli Naive Bayes

	50	**100**	**200**	**500**	**1000**
1-grams	58%	58%	59%	59%	63%
2-rams	60%	60%	62%	62%	62%
3-grams	58%	59%	59%	59%	59%
1,2-grams	66%	67%	72%	68%	72%
2,3-grams	60%	63%	62%	62%	62%
1,2,3-grams	66%	68%	72%	68%	72%

Table A.41: Experiments with Bernoulli Naive Bayes using POS tagging on abortion corpus

A.4.1.4 Logistic Regression

	50	**100**	**200**	**500**	**1000**
1-grams	66%	68%	69%	69%	74%
2-rams	61%	62%	62%	60%	59%
3-grams	58%	59%	59%	59%	59%
1,2-grams	66%	68%	69%	69%	74%
2,3-grams	61%	62%	62%	61%	60%
1,2,3-grams	66%	68%	69%	69%	74%

Table A.42: Experiments with LogisticRegression using POS tagging on abortion corpus

1.4.2 Creationism

A.4.2.1 Linear SVC

	50	100	200	500	1000
1-grams	75%	74%	74%	74%	73%
2-rams	69%	67%	66%	64%	64%
3-grams	76%	74%	74%	73%	-
1,2-grams	77%	76%	76%	75%	73%
2,3-grams	73%	69%	67%	66%	64%
1,2,3-grams	77%	76%	77%	76%	74%

Table A.43: Experiments with LinearSVC using POS tagging on creationism corpus

A.4.2.2 Multinomial Naive Bayes

	50	100	200	500	1000
1-grams	76%	74%	72%	70%	70%
2-rams	70%	68%	68%	63%	44%
3-grams	76%	75%	74%	69%	-
1,2-grams	76%	75%	72%	70%	70%
2,3-grams	76%	74%	73%	70%	70%
1,2,3-grams	74%	74%	77%	72%	70%

Table A.44: Experiments with Multinomial Naive Bayes using POS tagging on creationism corpus

A.4.2.3 Bernoulli Naive Bayes

	50	100	200	500	1000
1-grams	70%	66%	68%	71%	75%
2-rams	69%	64%	66%	67%	70%
3-grams	76%	74%	74%	73%	-
1,2-grams	70%	66%	69%	71%	75%
2,3-grams	69%	64%	66%	67%	70%
1,2,3-grams	70%	66%	68%	71%	72%

Table A.45: Experiments with Bernoulli Naive Bayes using POS tagging on creationism corpus

A.4.2.4 Logistic Regression

	50	100	200	500	1000
1-grams	76%	76%	74%	75%	74%
2-rams	75%	75%	74%	70%	70%
3-grams	76%	76%	76%	76%	-
1,2-grams	76%	76%	76%	76%	76%
2,3-grams	76%	76%	76%	75%	74%
1,2,3-grams	76%	76%	77%	77%	77%

Table A.46: Experiments with Logistic Regression using POS tagging on creationism corpus

1.4.3 Gay Rights

A.4.3.1 Linear SVC

	50	100	200	500	1000
1-grams	62%	62%	64%	63%	63%
2-rams	60%	60%	60%	60%	60%
3-grams	61%	61%	61%	60%	60%
1,2-grams	62%	62%	64%	63%	63%
2,3-grams	60%	60%	60%	60%	59%
1,2,3-grams	62%	62%	64%	63%	63%

Table A.47: Experiments with Linear SVC using POS tagging on gay rights corpus

A.4.3.2 Multinomial Naive Bayes

	50	100	200	500	1000
1-grams	61%	64%	65%	61%	62%
2-rams	60%	60%	61%	62%	57%
3-grams	61%	61%	61%	60%	62%
1,2-grams	62%	64%	64%	66%	62%
2,3-grams	60%	60%	61%	62%	57%
1,2,3-grams	62%	64%	65%	61%	62%

Table A.48: Experiments with Multinomial Naive Bayes using POS tagging on gay rights corpus

A.4.3.3 Bernoulli Naive Bayes

	50	100	200	500	1000
1-grams	60%	61%	60%	61%	61%
2-rams	61%	61%	60%	60%	60%
3-grams	61%	61%	61%	61%	60%
1,2-grams	60%	61%	60%	61%	61%
2,3-grams	60%	61%	60%	61%	61%
1,2,3-grams	61%	61%	60%	60%	601

Table A.49: Experiments with Bernoulli Naive Bayes using POS tagging on gay rights corpus

A.4.3.4 Logistic Regression

	50	100	200	500	1000
1-grams	64%	62%	62%	66%	64%
2-rams	60%	60%	60%	60%	58%
3-grams	61%	61%	61%	60%	60%
1,2-grams	66%	67%	63%	64%	65%
2,3-grams	60%	60%	60%	60%	60%
1,2,3-grams	65%	63%	64%	63%	64%

Table A.50: Experiments with Logistic Regression using POS tagging on gay rights corpus

1.4.4 Existence of God

A.4.4.1 Linear SVC

	50	100	200	500	1000
1-grams	52%	53%	52%	52%	53%
2-rams	40%	42%	50%	52%	-
3-grams	54%	53%	54%	54%	-
1,2-grams	52%	50%	51%	50%	52%
2,3-grams	40%	48%	46%	50%	-
1,2,3-grams	52%	50%	50%	52%	52%

Table A.51: Experiments with Linear SVC using POS tagging on Existence of God corpus

A.4.4.2 Multinomial Naive Bayes

	50	100	200	500	1000
1-grams	52%	52%	48%	47%	53%
2-rams	45%	46%	50%	54%	-%
3-grams	54%	53%	54%	55%	-
1,2-grams	52%	52%	48%	47%	54%
2,3-grams	45%	46%	50%	55%	-
1,2,3-grams	52%	52%	48%	47%	54%

Table A.52: Experiments with Multinomial Naive Bayes using POS tagging on Existence of God corpus

A.4.4.3 Bernoulli Naive Bayes

	50	100	200	500	1000
1-grams	52%	53%	53%	54%	52%
2-rams	50%	49%	52%	52%	-
3-grams	54%	53%	54%	54%	-
1,2-grams	53%	53%	53%	53%	52%
2,3-grams	50%	49%	52%	52%	-
1,2,3-grams	53%	53%	53%	53%	52%

Table A.53: Experiments with Bernoulli Naive Bayes using POS tagging on Existence of God corpus

A.4.4.4 LogisticRegression

	50	100	200	500	1000
1-grams	48%	51%	57%	54%	55%
2-rams	51%	50%	49%	51%	-
3-grams	54%	53%	52%	54%	-
1,2-grams	56%	51%	48%	50%	52%
2,3-grams	51%	50%	49%	49%	51%
1,2,3-grams	56%	51%	48%	52%	54%

Table A.54: Experiments with Logistic Regression using POS tagging on Existence of God corpus

1.4.5 Gun Rights

A.4.5.1 Linear SVC

	50	100	200	500	1000
1-grams	74%	76%	76%	76%	75%
2-rams	75%	76%	73%	72%	72%
3-grams	75%	75%	75%	74%	-
1,2-grams	74%	75%	76%	76%	75%
2,3-grams	74%	74%	74%	74%	73%
1,2,3-grams	73%	74%	76%	73%	73%

Table A.55: Experiments with Linear SVC using POS tagging on gun rights corpus

A.4.5.2 Multinomial Naive Bayes

	50	100	200	500	1000
1-grams	76%	76%	76%	76%	76%
2-rams	75%	74%	74%	74%	64%
3-grams	76%	76%	73%	73%	-
1,2-grams	76%	76%	76%	76%	76%
2,3-grams	75%	75%	74%	73%	72%
1,2,3-grams	76%	76%	76%	76%	78%

Table A.56: Experiments with Multinomial Naive Bayes using POS tagging on gun rights corpus

A.4.5.3 Bernoulli Naive Bayes

	50	100	200	500	1000
1-grams	74%	74%	76%	74%	74%
2-rams	75%	73%	74%	73%	74%
3-grams	75%	75%	75%	74%	-
1,2-grams	74%	74%	76%	74%	74%
2,3-grams	74%	74%	74%	74%	73%
1,2,3-grams	75%	75%	75%	74%	75%

Table A.57: Experiments with Bernoulli Naive Bayes using POS tagging on gun rights corpus

A.4.5.4 LogisticRegression

	50	100	200	500	1000
1-grams	74%	74%	76%	74%	74%
2-rams	75%	73%	74%	73%	74%
3-grams	75%	75%	75%	74%	-
1,2-grams	74%	76%	75%	74%	74%
2,3-grams	74%	74%	74%	74%	73%
1,2,3-grams	75%	75%	75%	74%	75%

Table A.58: Experiments with Logistic Regression using POS tagging on gun rights corpus

1.4.6 Healthcare

A.4.6.1 Linear SVC

	50	100	200	500	1000
1-grams	56%	52%	50%	47%	49%
2-rams	48%	50%	53%	52%	-
3-grams	59%	52%	46%	54%	-
1,2-grams	56%	52%	50%	47%	49%
2,3-grams	48%	50%	53%	53%	55%
1,2,3-grams	56%	52%	50%	47%	49%

Table A.59: Experiments with Linear SVC using POS tagging on healthcare corpus

A.4.6.2 Multinomial Naive Bayes

	50	100	200	500	1000
1-grams	55%	52%	48%	45%	42%
2-rams	52%	53%	53%	53%	-
3-grams	52%	52%	52%	48%	-
1,2-grams	55%	52%	48%	45%	42%
2,3-grams	52%	53%	53%	53%	49%
1,2,3-grams	55%	52%	48%	45%	42%

Table A.60: Experiments with Multinomial Naive Bayes using POS tagging on healthcare corpus

A.4.6.3 Bernoulli Naive Bayes

	50	100	200	500	1000
1-grams	47%	50%	51%	50%	52%
2-rams	53%	52%	52%	52%	-
3-grams	52%	52%	52%	52%	-
1,2-grams	47%	50%	51%	50%	52%
2,3-grams	53%	52%	52%	52%	54%
1,2,3-grams	47%	50%	51%	50%	52%

Table A.61: Experiments with Bernoulli Naive Bayes using POS tagging on healthcare corpus

A.4.6.4 LogisticRegression

	50	100	200	500	1000
1-grams	51%	51%	50%	45%	47%
2-rams	53%	53%	53%	53%	-
3-grams	51%	51%	51%	51%	-
1,2-grams	50%	50%	50%	48%	44%
2,3-grams	53%	53%	53%	53%	52%
1,2,3-grams	50%	50%	49%	46%	46%

Table A.62: Experiments with Logistic Regression using POS tagging on healthcare corpus

1.4.7 Quality

A.4.7.1 Linear SVC

	50	100	200	500	1000
1-grams	73%	74%	75%	75%	77%
2-rams	72%	72%	73%	74%	-
3-grams	72%	72%	72%	72%	-
1,2-grams	74%	74%	74%	74%	76%
2,3-grams	73%	72%	72%	72%	74%
1,2,3-grams	73%	72%	74%	76%	75%

Table A.63: Experiments with Linear SVC using POS tagging on quality corpus

A.4.7.2 Multinomial Naive Bayes

	50	100	200	500	1000
1-grams	72%	72%	72%	72%	72%
2-rams	73%	74%	72%	72%	-
3-grams	72%	72%	72%	72%	-
1,2-grams	72%	72%	72%	72%	72%
2,3-grams	73%	73%	72%	75%	76%
1,2,3-grams	72%	72%	72%	72%	76%

Table A.64: Experiments with Multinomial Naive Bayes using POS tagging on quality corpus

A.4.7.3 Bernoulli Naive Bayes

	50	100	200	500	1000
1-grams	73%	72%	74%	77%	75%
2-rams	72%	72%	72%	74%	-
3-grams	72%	72%	72%	72%	-
1,2-grams	72%	73%	72%	75%	76%
2,3-grams	72%	72%	72%	73%	76%
1,2,3-grams	73%	73%	72%	73%	74%

Table A.65: Experiments with Bernoulli Naive Bayes using POS tagging on quality corpus

A.4.7.4 Logistic Regression

	50	100	200	500	1000
1-grams	72%	72%	78%	72%	78%
2-rams	74%	73%	73%	75%	-
3-grams	73%	73%	73%	73%	-
1,2-grams	70%	74%	78%	74%	77%
2,3-grams	76%	73%	74%	75%	75%
1,2,3-grams	70%	74%	76%	75%	74%

Table A.66: Experiments with Logistic Regression using POS tagging on quality corpus

A.5 Extracting Feature Set For Agreement-Disagreement Classification

These experiments were conducted for Section 3.3.1.

1.5.1 Using feature sets suggested in papers

Using the following features:

- First word of response
- Last word of response
- Number of adjectives with positive polarity
- Number of adjectives with negative polarity
- Number of instances in the document of each cue phrase listed in Hirschber and Litman, 1994
- Number of instances in the document of each agreement/disagreement word listed in Cohen, 2002

Giving the following results:

Classifiacation Accuracy:	0.8182
Positive Precision:	0.8448
Positive Recall:	0.8167
Positive F-measure:	0.8305
Negative Precision:	0.7885
Negative Recall:	0.8200
Negative F-measure:	0.8039

Table A.67: Accuracy Results for Agreement Disagreement Initial Feature Set

1.5.2 Adding Adverbs and Verbs

Instead of using only adjectives, we decided to also use adverbs and verbs because words like *horribly, interestingly, agree, disagreem love, hate, etc*, which are usually very informative, were omitted and were not taken into consideration for the positive or negative orientation of the text.
Adding adverbs gave a classification rate of **83%**.
Adding verbs gave a classification rate of **82%**

1.5.3 Using postive and negative score of each sentiment word

In this experiment, instead of counting the positive and the negative words in the document, we decided to calculate the overall positive and negative score that the sentiment words (adjectives, adverbs, verbs) have in the text.

Using only adjectives gave a classification rate of **83%**
Using adjectives and adverbs gave a classification rate of **82%**

1.5.4 More positive/negative informative words

This time, instead of having two features, one for the positive and one for the negative number of adjectives, we used only to state whether the number of positive adjectives was greater than the number of negative adjectives.

Using only adjectives gave a classification rate of **84%**
Using adjectives and adverbs gave a classification rate of **83%**
Using only adjectives gave a classification rate of me score **83%**
Using adjectives and adverbs gave a classification rate of me score **84%**

1.5.5 Existence of cue phrases and agreement/disagreement word

This time, instead of counting the number each cue phrase or agreement/disagreement word appears in the text, we used a '0' or '1' value to state whether it appears in the text or not. The reason behind this idea is that the text is not large and therefore the importance of a word does not lie in the number of times it occurs, but whether it occurs in the text. Using the following features:
Using only adjectives gave a classification rate of **84%**
Using adjectives and adverbs gave a classification rate of **85%**
Using adjectives, adverbs and verbs gave a classification rate of **84%**
Using adjectives, adverbs and nouns gave a classification rate of **85%**
Using adjectives, adverbs verbs and nouns gave a classification rate of **87%**

1.5.6 Adding punctuation in feature set

This time, in the feature set, we include existence of question mark and/or exclamation mark in the feature set:

Include features for exclamation mark (!) and question mark (?) gave a classification rate of **82%**
Include feature for exclamation mark (!) gave a classification rate of **80%**
Include feature for question mark (?) gave a classification rate of **88%**

A.6 Experiments For Finding Out The Most General Classifier

These experiments were conducted for Section 5.2.2.3.

1.6.1 Train using abortion corpus

Using creationism data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Creationism:	38%
Gay Rights:	61%
Existence of God:	49%
Gun Rights:	69%
Healthcare:	53%
Quality:	60%

Table A.68: Training using abortion corpus and testing with the remaing corpora

1.6.2 Train using creationism corpus

Using creationism data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Abortion:	47%
Gay Rights:	35%
Existence of God:	50%
Gun Rights:	27%
Healthcare:	48%
Quality:	42%

Table A.69: Training using creationism corpus and testing with the remaing corpora

1.6.3 Train using gay rights corpus

Using gay rights data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Abortion:	55%
Creationism:	46%
Existence of God:	46%
Gun Rights:	56%
Healthcare:	49%
Quality:	59%

Table A.70: Training using gay rights corpus and testing with the remaing corpora

1.6.4 Train using god corpus

Using god data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Abortion:	46%
Creationism:	57%
Gay Rights:	45%
Gun Rights:	45%
Healthcare:	52%
Quality:	54%

Table A.71: Training using god corpus and testing with the remaing corpora

1.6.5 Train using gun rights corpus

Using gun rights data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Abortion:	54%
Creationism:	35%
Gay Rights:	64%
Existence of God:	48%
Healthcare:	53%
Quality:	59%

Table A.72: Training using gun rights corpus and testing with the remaing corpora

1.6.6 Train using healthcare corpus

Using healthcare data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Abortion:	46%
Creationism:	65%
Gay Rights:	35%
Existence of God:	50%
Gun Rights:	26%
Quality:	43%

Table A.73: Training using healthcare corpus and testing with the remaing corpora

1.6.7 Train using quality corpus

Using quality data as the training corpus and then test with the rest of the corpora:

Corpus Topic	Accuracy
Abortion:	50%
Creationism:	45%
Gay Rights:	48%
Existence of God:	55%
Gun Rights:	56%
Healthcare:	54%

Table A.74: Training using quality corpus and testing with the remaing corpora