Detecting Sarcasm in English Text

Andrew James Pielage

Artificial Intelligence MSc

2012/2013

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)_____

Table of Contents

1 Introduction

1.1 Project Aim

The aim of the project is to develop and test a computational method for detecting ironic sarcasm in English text. The classifier algorithm will, given a sentence of English text, classify it as being sarcastic or not.

1.2 Project Objectives

- Research related work to identify features and methodologies that can potentially differentiate between sarcastic and non-sarcastic text
- Design a classifier algorithm using identified sarcastic features and knowledge gained from analysis of related work
- Produce classifier algorithm
- Evaluate the classifier algorithm by comparing its classifications of sentences from a corpus against those done by humans
- Produce a report of the project, including future possibilities and improvements

1.3 Minimum Requirements

- A classifier algorithm which given English text input will classify a sentence as being sarcastic or not
- Analysis of textual characteristics that correlate with sarcasm
- Comparison of different approaches
- Evaluation of results from a corpus.

1.3.1 Possible Extensions

- Evaluation of results from a large, diverse corpus, to give more reliable results.
- Investigate if the addition of emoticons, laughter acronyms and onomatopoeic expressions denoting laughter as features improve the accuracy of my classifier
- A classifier algorithm which given English text input will classify a sentence as being sarcastic or as another rhetorical mode, such as litotes

1.4 Deliverables

- A classifier algorithm capable of identifying English sentences that are likely to be sarcastic.
- A corpus of sarcastic and non-sarcastic sentences.

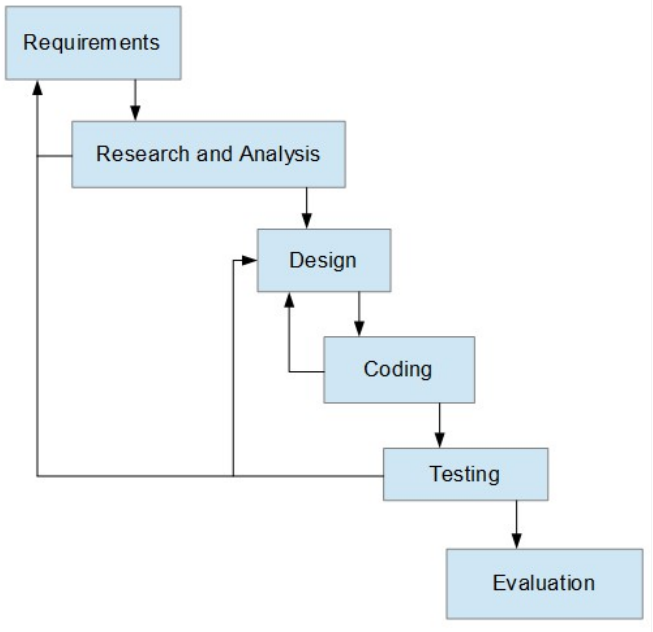
1.5 Research Methods

- Google/Yahoo
- Google Scholar
- Library/Text Books
- Course Materials

1.6 Project Life Cycle

I will use the iterative waterfall model as the basis of my project life cycle. The iterative waterfall model is an evolution of the lamented waterfall model, improving it to make it less prone to causing project failure. The iterative waterfall model does this by planning for the life cycle to continually go back on itself to allow for incremental changes to be made, accounting for the fact that projects frequently change over their life cycles due to changing requirements or unforeseen difficulties that require redesigns to circumvent. Since my project will likely require redesigns given the somewhat unknown aspect I face going into this project, an iterative based model that plans for additional redesign time is somewhat essential.

The model I will follow is as follows:



1.7 Project Schedule

My schedule will follow the rough outline depicted in the project life cycle diagram above. Time will be factored into my schedule to accommodate me spending time redesigning my sarcasm classifier, and also to allow for overrun as a contingency. A Gantt chart representing this can be seen in the appendices for reference, and the project milestones are below:

1.7.1 Milestones

- Complete research and analysis of related work
- Complete Design of Sarcasm Classifier
    - Complete Design of Pattern Extraction
    - Complete Design of Pattern Matching
    - Complete Design of Punctuation Features
- Complete Coding and Testing
- Complete Evaluation
- Reflect on Project and Finish Write up

2 Background Reading

2.1 Introduction

Sarcasm is a nuanced part of language, often difficult for people to pick up on if it isn't particularly obvious. Sarcasm is typically expressed verbally through the use of heavy tonal stress, and certain gestural clues such as rolling of the eyes. These tonal and gestural clues are obviously not available for expressing sarcasm in text, making its detection reliant upon other factors. There is currently no accepted solution to detecting sarcasm in written text, or whether it can actually be dependably detected at all, which is the basis of this project.

2.1.1 Sarcasm

Dictionary.com (Dictionary.com, 2013) provides definitions of sarcasm from two different sources, the Collins English Dictionary and the Random House Dictionary. The Random House Dictionary defines sarcasm as “a harsh or bitter derision or irony” or “a sharply ironical taunt; sneering or cutting remark” . The Collins English Dictionary defines it as “mocking, contemptuous, or ironic language intended to convey insult or scorn”. Another definition of sarcasm, by Merriam-Webster (Merriam-Webster, 2013), is “a mode of satirical wit depending for its effect on bitter, caustic, and often ironic language that is usually directed against an individual”.

As can be seen from the definitions above, and as noted by the sarcasm society (Sarcasm Society, 2013), sarcasm and irony are closely intertwined; it can even be stated that sarcasm is a form of irony. Verbal irony is purposefully stating the opposite to what is meant, such as “great” to mean bad (Dictionary.com, 2013). Sarcasm can be described as an ironic statement used to mock or insult, separating it from irony in that irony is not used to mock or insult. This particular type of ironic sarcasm is what my project will be focused upon classifying: a remark intended to insult by giving the opposite connotation.

2.2 Background to The Problem

2.2.1 Knowledge in NLP

NLP systems, both speech and text based, are separated from typical data processing systems by their use of knowledge of language (Jurafsky and Martin, 2009). The knowledge required for these systems is quite extensive, and is required to properly analyse and design systems around the complexities of natural language.

Knowledge in phonetics and phonology is required for understanding human speech. Phonetics and phonology encompasses the knowledge of linguistic sounds (Jurafsky and Martin, 2009). This covers the way words are pronounced, phonemes, prosody, and knowledge about pronunciation variation. Phonemes are the smallest speech units that differentiate meaning; different languages have different phonemes, due to the

wide variance in languages. Prosody is the term that describes lexical stress, rhythm, intonation and tone. Variation in pronunciation comes in two forms: lexical variation and allophonic variation. Lexical variation is typically sociolinguistic variation, but it also includes ambiguity in words such as “bass”. Allophonic variation is contextual pronunciation.

Morphology is a required knowledge to understand the variations of words, such as contractions and pluralisation (Jurafsky and Martin, 2009). The knowledge of morphology allows a NLP system to determine how words can be broken down into “meaning elements”, components of a word with meaning. Some examples of meaning elements are the 's' at the end of a word denoting a plural, or the 'un' at the beginning of a word denoting a negative.

Structural knowledge, or syntactical knowledge, enables a NLP system to understand how to order and group words for them to make sense (Jurafsky and Martin, 2009). This is required to enable the system to understand that although the words in a sentence may all be correct, they will not be understood unless put in the correct order.

Semantics is the study of meaning, allowing a system with this knowledge to understand the meaning of words. This particular application of semantics is known as lexical semantics, which is used in tandem with compositional semantics to understand that the meaning of a sentence can be composed from the meaning of its parts. The technical name used to describe the meanings of a sentence is utterance semantics. This knowledge is important when understanding how the meaning of words relate to the syntactic structure, such as with the word “by”; the word “by” can be used to denote a temporal relationship or as a description of an agent (Jurafsky and Martin, 2009).

Similar to semantics, pragmatics is the study of how language is used to achieve goals. Knowledge in this field is utilised by NLP systems to comprehend the differences between how a sentence is phrased to give meaning, also known as a speech act. Examples of different speech acts are utterances phrased as a question, statement, or request.

Discourse is the study of linguistic units larger than a single utterance, such as if a question is asked which references a previous utterance. A simple example would be the phrase “like what?”, which requires knowledge of a previous utterance. This is known as conference resolution.

Each of these is required by both humans, and NLP systems, to engage in complex language behaviour (Jurafsky and Martin, 2009). However, a large problem in NLP is resolving ambiguity at one or more of these levels.

2.2.2 Ambiguity in NLP

Most tasks in NLP can be viewed as resolving ambiguity in one or more of these knowledge fields (Jurafsky and Martin). As the knowledge described in the previous section is in essence mandatory to process utterances within NLP systems, ambiguity in one or more of these fields frequently occurs. Methods have

been made to counter this ambiguity however, such as Part-Of-Speech (POS) tags.

POS tags help to resolve issues with words that can have several meanings, such as “duck”, by applying a tag to the word denoting it as a verb or as a noun. This however can lead to ambiguity in itself when actually applying these tags; if given an untagged sentence, it is not always clear which of the available tags for a specific word is the correct tag. This is known as POS ambiguity. POS tagging does not completely solve the ambiguity about words though, certain words have several meanings while only having a single POS tag, such as “crane”. This is known as lexical ambiguity, and is combated by word sense disambiguation.

Syntactic ambiguity is where a sentence can have more than one structure, namely how words connect together. This problem is addressed by probabilistic parsing, that selects the most probable structure for that sentence. Probabilistic parsing, being statistical, does however make mistakes when disambiguating particularly obscure sentences.

Several other types of ambiguity exist that cause problems in NLP. Referential ambiguity is where an expression has several referents without a clear indicator of who is the subject of the expression. Discourse ambiguity is where there is one connective, but several rhetorical relations. Speech act ambiguity is where an utterance does not have a clear speech act (Jurafsky and Martin, 2009).

A key task in NLP is disambiguating these in context, and several different models and approaches are available to help complete this task.

2.2.3 Symbolic vs. Statistical

There is a divide in NLP between the symbolic/rule-based approach, known as the Rationalist approach, and the statistical approach, known as the Empiricist approach. The symbolic approach focuses on creating rules that cover the grammatically correct sentences in a language. This side argues that a corpus cannot be useful to linguists, as they describe the actual use of language, not what is actually correct. This is known as modelling performance rather than competence. Further arguments made by the symbolic approach is that a corpus is finite, and the probability of a sentence cannot describe its grammaticality. The argument that a corpus is finite is correct, but this argument was made before the internet fully came into its own; it is now possible to gather a corpus of over a trillion sentences, which greatly strengthens the statistical side of NLP. Any corpus however is subject to Zipf's law, meaning that there will always be a large number of words that occur infrequently, and a small number that occur very frequently.

The statistical approach is based upon the probability of sentences and words; Empiricists argue that people differ in grammatical judgements, and so acceptability should be prioritised over grammaticality. This approach makes NLP more robust to errors, as it is not based upon stringent rules. The statistics for this approach has to come from somewhere, so corpora are used. The use of corpora, as noted above, does not guarantee grammaticality however. This is not too crucial a problem however as it is not what the empiricists

are after; empiricists prioritise performance over competence. The downside of the statistical nature of this approach is that because language is infinite, and corpora are finite, unseen events become a problem.

The current trend is a mixture of these two approaches, with a lean on the statistical side. The current trends are based upon utilising modern day computer power to work with large, annotated corpora and machine learning methods.

2.2.4 Detecting Sarcasm

Due to the non-literal nature of ironic sarcasm, designing software to recognise it is formidably difficult. This is due to a one-to-one mapping of words to meaning in current corpora and systems; irony is not accounted for. A further problem is that there is an absence of accurately labelled naturally occurring sentences of ironic sarcasm that can be used for research and machine learning systems (González-Ibáñez, Muresan and Wacholder, 2011). Sarcasm is so notorious for being hard to understand that a special character, the sarc mark, was created with the intention of filling a gap in conveyance of writing, in the same vein as an exclamation point or question mark.

Sarcasm, as a form of wit, is not always obvious in its use, causing people to rely on a number of cues. These cues include using a dead pan or overly exaggerative tone, and the use of exaggerative gestures or facial cues such as rolling the eyes. Other cues are typically vocal in nature, including the use of a slower tempo, raised volume and lower pitch. These cues are often combined with particularly positive, or occasionally negative, adjectives, as well as strong verbs or adverbs (Yahoo! Contributor Network, 2008). There is no prescribed way of indicating sarcasm, which leads to there being no consistent gestures, though the vocal cues described above are fairly consistent (Yahoo! Contributor Network, 2008).

As previously described, people speaking face to face can rely upon other cues than language ones; speakers can roll their eyes, place heavy stress upon certain words, slow their speaking rate etc. This is obviously not available when reading from text. This leads to the problem of only having lexical factors as clues to work with, both for the reader to pick up on and for a NLP system to detect. As noted previously, ironic sarcasm is non-literal, which not only gives problems to designing a system to detect it, but also for readers to pick up on it. Assuming the author does not explicitly state that the phrase is meant sarcastically, readers have few clues to pick up on the sarcasm, such as the context of the situation, known as common ground (Clark, 1996), and the words being used to pick up on the sarcasm. An example of some of the clues that readers and systems can pick up on are extreme adjectives and adverbs, such as 'absolutely fantastic', which Utsumi (2000) suggests as being a way of implicitly displaying a negative attitude. As you can tell from the example I used, this is commonly used to express ironic sarcasm.

A sarcastic sentence is sometimes expressed vocally with heavy stress to audibly give clue that the sentence is meant non-literally; this is sometimes mimicked in text by extending the letters that constitute a syllable, an example being “Riiiiight” as demonstrated by Bogart (2008). Another way this is mimicked,

typically by users of the internet writing on Twitter, forums, or product reviews, is through the use of capital letters; capital letters, when used throughout an entire word, is conventionally used to denote tonal stress or shouting. Were the utterance taken from the book quoted above (Bogart, 2008) an actual conversation, the intonation used, as well as the actual world knowledge supposedly known by the recipient, would be the clues required to discern it as being sarcastic.

This leads into another problem with detecting sarcasm, world knowledge. World knowledge refers to knowledge about things and people in the world that is commonly known. Using the same sarcastic sentence used by Bogart (2008), he writes "'and Moses didn't know how to count to ten.'", whereas it is highly likely that Moses could in fact count to 10. Were this taken using the one to one mapping of current NLP systems, this sentence would appear to be a strange non-sequitur, having no direct relation to the previous sentence. World knowledge, among with the other clues of non-literal intent, are required to discern that this sentence is not meant literally.

A further problem with the detection of sarcasm is that many utterances that are supposedly sarcastic do not conform to the definition of verbal irony, namely writing or saying the opposite of what is actually meant (Gibbs and O'Brien, 1991). Whether this is misuse of the term or an evolution of it is debated.

2.3 Related Work

Sarcasm detection in English text is not a thoroughly researched field, due to the prevailing idea that a computer system will not be able to comprehend the non-literal nature of sarcasm, the intricacies of its use, and the fact that even humans have trouble detecting it at times. Nonetheless, some attempts have been made to create a computational classifier for sarcasm, or similar fields (irony, metaphor), using different approaches.

2.3.1 Features

2.3.1.1 Lexical and Pragmatic Features

Due to the complexities described in detecting sarcasm, previous work done by others have all made assertions on how might be a good way to abstract the problem into something that can be solved computationally. This also includes pinpointing the specific ways that allow us to detect sarcasm as human beings so that they can be potentially utilised for detecting sarcasm in a computational manner.

Common ground is one such component of sarcasm that helps us to detect it as human beings. Common ground (Clark, 1996) is the shared knowledge or familiarity between the two participants in a conversation. The absence of common ground between the two participants can easily lead to the misunderstanding of a sarcastic statement; using sarcasm at somebody that was not present for, or has no knowledge of, a certain event, or at a stranger not expecting the mockery inherent in sarcasm, will typically lead them to not correctly comprehend the intent behind the sarcastic statement.

Kreuz and Link (2002) state that this pragmatic factor of common ground can be projected by readers to characters that know each other in written text; the participants from the paper reading the excerpts are capable of reading sarcastic statements more quickly, and with a greater degree of certainty, than when the characters do not know each other. These results can be explained in terms of a principle of inferability: people will only employ sarcasm if they are reasonably certain that the people they are communicating to will interpret it correctly (Kreuz, 1996). This presents a problem; if the use of sarcasm relies solely upon pragmatic factors, creating software that is capable of detecting it would be difficult.

In later work, Kreuz and Caucci (2007) hypothesise another component that can be used to help in the detection of sarcasm: interjections. They state that the use of interjections in excerpts from published works with the phrase “said sarcastically” removed, to prevent instant identification, predict a significant amount of variance in participants' ratings of sarcastic intent. This leads them to suggest that sarcastic statements may be somewhat formulaic, and as such computer software could be written to recognise these lexical factors. The detection of these lexical factors would enable the software to interpret non-literal intent even if the pragmatic components of non-literal language cannot be identified, bypassing one of the largest problems in detecting sarcasm.

This idea was extended upon by González-Ibáñez, Muresan and Wacholder (2011), with them stating that lexical factors alone are not enough to accurately identify sarcasm. Using lexical and pragmatic features, they attempted to use machine learning to separate sarcastic tweets from those issuing a positive or negative sentiment. The pragmatic factors included into their work, given that they utilised twitter as their corpus, consisted of emoticons and the people to whom tweets are being directed. Their work makes note of the fact that human beings themselves are not particularly adept at discerning what is and isn't sarcastic without proper context. This note on how humans require proper context, alongside how the pragmatic feature of directed tweets (it is assumed that the tweeter has some common ground with the tweeted) factored into the results of the paper, provides further proof that common ground is an important and common feature in sarcasm.

Despite the addition of these pragmatic features however, the results from their experiments showed that the lexical and pragmatic features used were not sufficient to accurately differentiate sarcasm from positive or negative tweets. In fact, the results from this paper showed that, for their methodology, humans do not significantly outperform the machine learning techniques used. As previously alluded to, they note that their results provide further proof for the need of context, both common ground between the tweeters and more general world knowledge, when attempting to classify sarcasm through lexical and pragmatic features.

This need for extra context is not a dead-end for using lexical and pragmatic features as a means of classifying sarcasm however; the work done by Carvalho et al. (2009) reveals that a marked increase in the detection of irony detection can be achieved when using what they call “oral or gestural clues” as features. They describe these oral and gestural clues as: emoticons; onomatopoeic expressions for laughter; heavy use

of punctuation marks; quotation marks and positive interjections. Though they did not explicitly look for this, it was noted by González-Ibáñez, Muresan and Wacholder (2011) that smileys and positive emotions feature among the best of the factors they used for distinguishing sarcastic tweets when attempting to classify by the presence of factors. With these features giving positive results, they can be potentially combined with other features to potentially produce a working classifier. Further proof to showing that emoticons in Twitter are a significant predictor on the likelihood of a sentence being sarcastic can be gained from the word of Derks et al. (2008).

2.3.1.2 Pattern-based Features

Davidov, Tsur and Rappoport (2010) propose a different way of trying to classify sarcastic sentences computationally: through pattern based features. Their study experiments with the idea of extracting patterns from sarcastic and non-sarcastic sentences, and then using a k-means clustering strategy to classify a sentence as being sarcastic or not. This clustering method also affords them the ability to not just classify as being sarcastic or not; in the study they use five levels, ranging from not sarcastic to clearly sarcastic. This has the advantage of allowing the user to see not just if a sentence is sarcastic or not, but with what degree of certainty.

Pattern based features are not the only thing that they include in their classifying algorithm however; they also include some punctuation based features. Building upon what previous work has revealed, namely that lexical and pragmatic factors alone are not enough to classify sarcastic sentences but are still useful, they used them in conjunction with their pattern based features to try and make the classifier more accurate.

The results of this study are very promising, with pattern based features providing promising results when run on both structured sentences from Amazon Reviews, and more unstructured ones from Twitter. It is theorised in the paper that the algorithm is as successful as it is due to the ability to have incomplete matches and the fact that their classifier spans a feature space of over 300 dimensions.

The approach by Davidov, Tsur and Rappoport (2010) was not the first study published in the pattern based field however, or the first published to utilise both patterns and lexical/pragmatic features; Carvalho et al. (2009) utilised patterns, lexical and pragmatic factors in their study that was published the year before. In contrast to the more automated and flexible method of gathering the patterns from the corpus being used itself, they used stricter 4-gram patterns, specifically looking for certain lexical and pragmatic features. This was done with the intent of focusing on looking for the opposite meanings present in irony, specifically positive to negative. These specific patterns were based upon previous research done by them to identify specific lexical and pragmatic cues that relate to irony.

As previously stated, their study revealed that oral and gestural clues are a significant predictor of irony, with pragmatic features such as smileys also performing well. The main focus of their work however, namely on structured linguistic knowledge, proved to be ineffective at detecting irony, meaning that additional classifying features will have to be gained from elsewhere. A downside to this study's relevance to

my project is that it is based upon Portuguese, so the morphological proofs will not carry over to English and my project. It is noted however that some of the features they used, such as emoticons, are language independent and so are still usable.

2.3.1.3 Personal Comments

This overview leads me to believe that a classifier based upon SASI, utilising surface patterns extracted from natural language sentences and punctuation as features, would be the most successful out of those researched.

2.3.2 Corpora

Previous work has used generated sarcastic statements, which are often extreme in their nature and would typically not be used in common text or speech (Kreuz and Glucksberg, 1989). This confounds the pragmatic and lexical aspects of sarcasm, meaning that sentences gathered have to gathered from natural sources to prevent this. Naturally occurring sentences for training and testing the presence of sarcasm can be obtained from various sources, Google books, Amazon reviews and Twitter being some sources used by the references above.

2.3.2.1 Twitter

Twitter has a hash tag function where users can tag certain posts with memes or other labels. González-Ibáñez, Muresan and Wacholder (2011) take advantage of Twitter in this regard to obtain utterances of naturally occurring sarcastic and non-sarcastic tweets. They attempt to alleviate the concerns noted by Davidov, Tsur and Rappoport (2010) that data from Twitter is noisy by only selecting the tweets which have the desirable tag at the end of the sentence. They filter the data from Twitter using this criteria and manually pre-process it before use to remove duplicates, spam, foreign language tweets, URLs and messages that have the tags as part of the message.

Despite any filtering and pre-processing before using tweets tagged by their tweeters, the problem with this method of gathering a corpus remains that relying on the tweets tagged by their authors may lead to a noisy corpus due to misuse of the term sarcasm by the author. This misuse of the sarcasm hash tag is possibly quite a frequent occurrence, given the difficulty human beings have simply discerning it, and also the propensity of internet users to submit what they type before properly proofing or thinking over what they have typed. It is also noted by Davidov, Tsur and Rappaport (2010) that the “sarcasm” hash tag is infrequently used by the authors of tweets, due to it having little publicity.

2.3.2.2 Amazon Reviews

Amazon is another means by which naturally occurring text can be gathered, as utilised by Davidov, Tsur and Rappoport (2010). Amazon sells a large variety of products, and allows users to post their own

reviews of the product on the product page. This provides a good variety of contexts for the extraction of sentences for use in their study of classifying sarcasm. They note that in contrast to tweets, Amazon reviews are typically much longer, have better structure, have better grammar, and also have the bonus of a known context (the product being reviewed).

Each review on Amazon has a star rating of 1-5, with five stars being a very good product. This meta-data can be exploited, as done by Davidov, Tsur and Rappoport (2010), to search for positive sentiments in negative reviews. This ties back to the work done by both Carvalho et al. (2009) and González-Ibáñez, Muresan and Wacholder (2011), with both of their studies noting that the lexical feature of positive interjections is a potential indicator of sarcasm.

2.3.2.3 Google Books

Google Books contains over 100,00 published works of various genres, providing a good variety of contexts to pull sarcastic excerpts from. Google books allows users to search for books with particular phrases or words contained within them, with this facility being used by Kreuz and Caucci (2007) to extract excerpts from the books available that contain the phrase "said sarcastically". To provide some context to each sarcastic utterance, Kreuz and Caucci (2007) take the paragraph the key phrase appears in, as well as the preceding and following two paragraphs.

A brief search using the same term "said sarcastically" as used by Kreuz and Caucci (2007), will reveal that this is not a particularly accurate means of gathering a corpus without supervision however; a large number of authors do not use the term sarcastically in the way I am looking for, namely in the form of verbal irony. Authors instead commonly use it to mean something amusingly witty, "'I like where this road ended, but man does somebody have to fix all those damn bumps along the way.' I said sarcastically" (Clifford, 2002), or said with a certain intonation, "She walked over to the bathing pool and said sarcastically, 'Well, who do we have here?'" (Ellis and Ellis, 2010).

This is noted by Kreuz and Caucci (2007) in their work, arguing that authors may not explicitly use the term "said sarcastically" in their work, as it can reflect poorly on their esteem of their readers and somewhat dampen the comedic value of the statement. There are arguments on both sides for whether or not authors should include the phrase "said sarcastically", or any similar deliberate pointers to the same affect, in their work. It is argued that a writer should have provided enough context to have given the non-literal intent without explicitly stating it as such, leaving the detection up to the reader. The counter to this stance is that sarcasm and irony are hard to portray in text, and so authors will still need to use confirming expressions to point out these sarcastic utterances. Nonetheless, there are plenty of books available via Google Books with the phrase present, with the previously stated caveat that the phrase is not always used correctly or in the way I am looking for.

2.3.2.4 Website Comments

Another potential corpus is to use the comments sections present in both forums and websites, such as Youtube and newspaper websites as was used by Carvalho et al. (2009). From a single popular newspaper website, they managed to gather roughly one million sentences from a five month period. Their study was focused on recognising irony in relation to named-entities, with them using this corpus in conjunction with a lexicon of frequently mentioned politician names to extract their desired sentences automatically.

2.3.2.5 Data Enrichment

Davidov, Tsur and Rappoport (2010) propose a method of expanding a small corpus without resorting to noisy and expensive annotation. They posit that sarcastic sentences frequently co-appear in texts with other sarcastic sentences. Under this hypothesis, they perform an automated web search using each sarcastic sentence from their training set and collected up to 50 excerpts that the search engine returned. If the sarcastic sentence being searched for was more than 6 words, only the first 6 words were used for the search to prevent the search becoming too specific. Each of these sentences was then added to the training set with the same sarcasm label as the search query that acquired it.

This method does not fix a problem that was present in the corpus used by Davidov, Tsur and Rappoport (2010) however, that of a ratio imbalance in their corpus. Due to the rarity of sarcastic sentences in comparison to other sentences, their corpus has far more non-sarcastic sentences than sarcastic ones, with data enrichment not solving this problem.

2.3.2.6 Personal Comments

Each of these corpora described would appear suitable for my given task of collecting natural sarcastic sentences. The one exception to this being data enrichment, as unless it is used to fix a ratio imbalance by only collecting sentences that fall into the smaller class, all it succeeds in doing is gathering a larger corpus. Given the success had with the other corpora, this would not seem to be a necessary measure to be taken.

2.3.3 Methodology

The studies referenced use both different features and corpora in their attempts to design a computational classifier of sarcasm, with varying results. In accordance with these differences, different methodologies are used by each; the primary difference being when searching for lexical and pragmatic features versus searching for patterns.

2.3.3.1 Lexical and Pragmatic Features

Kreuz and Caucci's (2007) study on the lexical influences on the perception of sarcasm used two independent judges to code in a binary fashion each of the excerpts obtained from Google Books in their

corpus in three dimensions. The binary system that they used indicated whether or not there were adjectives and adverbs, interjections, exclamation points and question marks within each excerpt. A number of students were then used to mark a selection of their corpus, always including some control excerpts, upon a 7 point scale of how sarcastic the excerpt was, given that the term "said sarcastically" was removed from the excerpts that it was present in. To help the students, where the phrase "said sarcastically" had been removed, the phrase to which it was referring to was emphasised in bold. The students were tested in groups, and were given ample time to work through their provided excerpts in their own time. Each student was not given a definitive definition of the term sarcasm, instead being told to rely upon their own understanding of it.

To determine the importance of the lexical factors tested in the study, Kreuz and Caucci (2007) performed a regression analysis using the mean sarcasm rating of each excerpt as the criterion variable. Five predictor variables were employed for the regression analysis: the number of words, the number of bold-faced words, the presence of adjectives and adverbs, the presence of interjections and, the use of exclamation points and question marks. As done before by the judges, variables 3-5 were coded in a binary.

In the study by González-Ibáñez, Muresan and Wacholder (2011) attempt to empirically identify lexical and pragmatic features to classify tweets as sarcastic or not. They gather their lexical factors using unigrams and dictionary based features, and use positive and negative smileys, as well as the *ToUser* Twitter command, which marks if a tweet is a reply to another tweet, as their pragmatic features.

The Linguistic Inquiry and Word Count (Pennebaker, Francis and Booth, 2007) is used for the dictionary based features, which groups a set of 64 word categories into four general classes: Linguistic Processes, which consist of adverbs and pronouns; Psychological Processes, which consists of positive and negative emotions; Personal Concerns, which consist of achievements and work; and Spoken Categories, which consist of assent and non-fluencies. In conjunction with the Linguistic Inquiry and Word Count, González-Ibáñez, Muresan and Wacholder use WordNet Affect (Strapparava and Valitutti, 2004) and a list of interjections and punctuations as used by Kreuz and Caucci (2007) for their dictionary based features. This combination when merged into a single dictionary covered 85% of the words in the 2700 tweets used in the study, providing a good coverage despite the disposition of internet users to create new words and use abbreviations.

To quantify which of the features listed above has the greatest impact discriminating between sarcastic, positive, and negative tweets, feature ranking was performed. The two measures used in the study for the feature ranking were the presence and frequency of the factors in each tweet, with both a three way comparison and a two way comparison being used for the three categories.

González-Ibáñez, Muresan and Wacholder (2011) also performed classification experiments to investigate the usefulness of lexical and pragmatic features when utilising machine learning . For their classification experiments they used two standard classifiers: a support vector machine with sequential minimal optimisation, and logistic regression. As stated previously, unigrams were used as a feature for the

classification experiment; bigrams and trigrams were both noted to not provide any better results than unigrams. Both classifiers were trained on a balanced dataset and tested through five-fold cross-validation.

2.3.3.1.1 Comparison against Human Classification

They asked 3 judges to attempt to classify 10% of the corpus (90 sarcastic tweets, 90 positive tweets, 90 negative tweets) as either sarcastic, positive or negative. The judges were also allowed to state whether they were unsure if a tweet that they classified belonged in its class, and were permitted to comment on the difficulty of the task.

In response to the studies by Derks et al. (2008) and Carvalho (2009), González-Ibáñez, Muresan and Wacholder (2011) also performed classification experiments utilising the above machine learning classifiers with tweets that contained emoticons, creating a new, smaller dataset to accommodate this experiment. This experiment was done with the intention of comparing it against human classification, to study the affect of emoticons on human detection and computational classification. The same procedure as above was used for this experiment, with the exception that only two judges were used this time, and the computational model was not retrained for this new data; the same trained model as utilised in the standard human vs computational classifier experiment.

2.3.3.2 Pattern-based Features

Carvalho et al. (2009) utilise 8 patterns based upon research they conducted to find lexical and pragmatic features that are potential clues of irony. The 8 patterns are a mix of structured 4-gram patterns that search for a particular lexical feature and takes the words before or after it, or are simply just particular lexical features. The features they used are: diminutive forms, demonstrative determiners, interjections, verb morphology, cross-constructions, heavy punctuation, quotation marks, and laughter expressions. All of the patterns in the study restrict the polarity of their matching sentences to positive; they focus on finding positive patterns that are being used in a negative context. This typically took the form of a matching sentence requiring the presence of at least one prior positive adjective or noun in the 4-gram window of four word, while excluding the occurrence of any negative element within the same window. To find the positive polarity adjectives and nouns, they created a sentiment lexicon with manually annotated polarities.

The diminutive forms, demonstrative determiners, verb morphology and cross-constructions are all features based in Portuguese, and so will not likely transfer across into English. The remaining features however, are language independent, and so can still be used for research purposes. To search for interjections, they simply used a lexicon of positive interjections such as “bravo”. Their heavy punctuation feature consisted of searching for sentences containing more than one exclamation point or question mark. To find positive instances of quotation marks, they searched for sentences containing one or two quoted words, with at least one of the words being a positive adjective or noun. The laughter expressions features were found by searching for the acronym “lol” and its variations, onomatopoeic expressions, and positive

emoticons.

Once they had gathered their corpus, as explained above with the requirement that each sentence must have at least one of the named politicians from their lexicon, they generated several diminutive forms for each named-entity. These were then used for checking matches of their diminutive form feature. The rest of the features were searched for by scanning the corpus for sentences matching any of their 8 patterns. For any pattern that matched over 100 sentences, the particular pattern was then manually evaluated through a four way classification system: the pattern is ironic, the pattern is not ironic, the pattern is ambiguously ironic, and undecided.

The semi-supervised sarcasm identification algorithm, SASI, developed by Tsur, Davidov and Rappoport (2010) employs two modules: semi-supervised pattern acquisition for identifying sarcastic patterns that can serve as features for a classifier, and a classification stage that classifies each sentence to a sarcastic class.

Similar to how Kreuz and Caucci (2007) had their testers annotate their excerpts, each sentence to be used as a seed is annotated with a number within the discrete range of 1 to 5, with 5 indicating a clearly sarcastic sentence and 1 an absence of sarcasm. From these labelled sentences, the syntactic and pattern-based features are extracted for utilisation as a feature vector. To allow more general patterns to be used, as with the method described below for Google Books by Kreuz and Caucci (2007), any explicit target, such as the product, author, company and book name is replaced with a generic meta tag. In the case of Twitter, they remove the hash tags, users and URLs (Davidov, Tsur and Rappoport, 2010). These patterns were extracted by searching through each sentence in their corpus for words that fell into one of two categories: content words or high frequency words. A word falls into one of these two classes based upon its frequency within the entire training corpus. The patterns were constructed of a number of these high frequency and content words, with only a certain number of each word class being allowed in each pattern.

Filtering was then done to remove the patterns that are too general and also too specific. Patterns that are too general are filtered out by removing the patterns that get extracted from both a clearly sarcastic sentence, and a non-sarcastic sentence, whereas the patterns that are too specific are removed by deleting the patterns that are extracted from only one Amazon product review. Once the patterns had been filtered, the remaining patterns are matched against both the training and test sets to create feature vectors, with a value of 1 being given to a perfect match of the pattern to the sentence, a value of 0 for no match, and a value between 1 and 0 for a partial match.

Punctuation based features are also included into the feature vectors. The features used in the study are the number of words in each sentence, the number of exclamation points, the number of question marks, the number of quotes in the sentence, and the number of capitalised words in the sentence. These are normalised to be in the range of 0-1 in line with the pattern vectors by dividing by the maximum observed value of each category, thus allowing each to have the same weight as a single pattern feature.

Once each sentence in both the training and test sets has a vector, each test set vector is compared against its nearest neighbours from the training set in a k-nearest neighbours like strategy. The k nearest vectors are found by Euclidean distance, with the test sentence being classified as one of the five sarcasm levels by a weighted average score calculated from these closest vectors.

2.3.3.3 Comments

None of the methodologies described seem to be too time consuming for my personal skill level and project time constraints; each methodology essentially comes down to searching for particular features in text as all of the papers wished to avoid expensive annotation. Given the success of the SASI classifier however, and as it continues the trend of avoiding time consuming annotation, it still seems to be the strongest contender as the basis of my classifier.

2.3.4 Results

The results from each of the studies, while not all coming up with a robust, working classifier, all reveal information of potential use that, when combined together, may create a better sarcastic sentence classifier.

2.3.4.1 Lexical and Pragmatic Features

The results from the studies on lexical and pragmatic features (Kreuz and Caucci, 2007)(González-Ibáñez, Muresan and Wacholder, 2011) conclude, that while lexical and pragmatic features are present in sarcastic sentences, alone they are not enough to accurately classify a sentence as sarcastic or not without established common ground and context.

Kreuz and Caucci (2007), with their study investigating if there are lexical features that can identify sarcasm, found that their human testers were capable of differentiating between the sarcastic and non-sarcastic sentences. They found from their regression analysis that the length of the excerpts were not a large influence on the judgement of the testers, stating that longer excerpts gave more contextual clues and nothing else. They found however that only positive interjections were a significant predictor of sarcastic intent, with their other features of adjectives, adverbs, and punctuation being of little consequence.

The feature ranking done by González-Ibáñez, Muresan and Wacholder (2011) revealed that for differentiating between sarcastic, negative, and positive tweets, negative emotion, positive emotion, negation, emoticons, auxiliary verbs and punctuation marks are in the top ten features. For differentiating between sarcastic and non-sarcastic specifically, the five best features are: positive emotions, the present linguistic process, questions, the ToUser command, and the affect psychological process. Their results show that in sarcastic tweets a positive emotion while using negation words, and the pragmatic factor of knowing who the tweeter is talking to, indicating common ground, are important distinguishing features.

In their classification experiments, González-Ibáñez, Muresan and Wacholder (2011) utilised the

presence of their features, the frequency of their features, and unigrams, to investigate the usefulness of lexical and pragmatic features in machine learning for the given task. Using support vector machine with sequential minimal optimisation, they achieved an accuracy of 65.44 with unigrams, with frequency and presence being close behind. For logistic regression, presence gave the best accuracy of 63.17, with again the others being close behind. With results only just above a 50/50 chance, they conclude that lexical and pragmatic features alone are not sufficient to classify sarcastic from non-sarcastic sentences. They speculate that this may be due to the difficulty of identifying sarcastic utterances in isolation, without the use of contextual evidence.

2.3.4.1.1 Comparison against Human Classification

The three judges managed to achieve an overall agreement of 71.67%, with a mean accuracy of 66.85% with 0.37% uncertainty. The accuracy drops to 59.44% however when only looking at the tweets that all three judges agreed on.

After training the machine learning algorithms used in their main classification experiment, namely support vector machine with sequential minimal optimisation and logistic regression, on the remaining 90% of their corpus, they ran these classifiers on the same 10% that the human judges classified, using the judges' results as the human baseline interval. Only support vector machines utilising unigrams (68.33%) and presence (67.78%), and logistic regression utilising presence (67.22%) as features achieved better results than the human baseline interval. These results, as can be seen, are very similar, with not much separating the results of human versus computational classification.

The results of their experiments with emoticons are a little different to the above. The two human judges achieved an overall agreement of 89%, with the results showing that emoticons do help people to distinguish sarcastic from non-sarcastic tweets; the judges achieved an accuracy of 73%, with an uncertainty of 10%, and achieved an accuracy of 70% where they both agreed. The computational classifier in achieved a similar accuracy of 71% with the support vector machine utilising unigrams as features, thus showing that both humans and the computational classifier were more capable of classifying sarcasm when emoticons are included in the message.

The judges in the paper note that the lack of context and the brevity of tweets are big factors in their difficulties distinguishing the sarcastic from non-sarcastic tweets. One of the judges is stated to have explained that world knowledge about recent events had to be called on at times to make judgements about whether a tweet was sarcastic or not. This leads the authors of the paper, as previously stated, to suggest that information about common ground between tweeters and world knowledge are necessary to create an accurate computational classifier of sarcasm.

2.3.4.2 Pattern-based Features

Carvalho et al. (2009) reveal in their experiment that classifying sentences as ironic or not by

searching for structured patterns, in Portuguese at least, does not work; the patterns involving more structured linguistic knowledge, namely diminutive forms, demonstrative determiners, verb morphology and cross constructions, simply do not find many, or any, matches. They state that this is likely due to the strictness of the structure, with them showing that removing one of the constraints on the pattern, diminutive forms, allows it to match a large number of sentences. This defeats the purpose of the structure though, with them stating that the large majority of these new matches simply being normal sentences.

Despite this failure, the unstructured patterns searching for positive interjections, punctuation, quotation marks and laugh expressions, all match a large number of sentences. The accuracy of these patterns varies rather wildly however; the worst results coming from interjections and punctuation: correctly identifying 44.88% and 45.71%; incorrectly identifying 13.39% and 27.53%; was undecided on 40.94% and 26.75%, and misclassified due to ambiguity 0.79% and 0% respectively. In contrast, the results from the quote and laugh patterns are far more promising: the quotes pattern correctly identified 68.29%, incorrectly identified 21.95%, was undecided on a mere 2.73%, and was ambiguous on 7.03%; whereas the laugh expressions pattern correctly identified 85.40%, incorrectly identified 0.55%, was undecided on 11.13%, and was misclassified 2.92%. It is worth noting however that these are classified by humans, the automated part merely matches these sentences.

The larger than average misclassification rate for the quotes pattern was largely attributed to two typical situations: quotation marks being used to delimit a multi-word expression; or being used to differentiate a technical term or brand. The misclassification rate for punctuation was attributed to people reinforcing rhetorical questions, which are not always ironical. As can be clearly seen however, laughter expressions such as “lol”, onomatopoeic expressions and emoticons are a good indicator of irony, which ties into sarcasm, with the added bonus of having a low rate of misclassification.

It is noted in the study that they had trouble distinguishing if the sentences returned by the interjection and punctuation patterns were ironic or not. This is due to them simply requiring context for them to classify it as ironic or not, providing further credence to the theory that either world knowledge and common ground are needed to classify utilising lexical or pragmatic features, or to try and circumvent the need for it by utilising a pattern-based method such as SASI (Davidov, Tsur and Rappoport, 2010).

The results achieved by the SASI classifier are extremely promising, both achieving accuracies exceeding those of the other studies researched and also being a more robust and flexible classifier, seemingly usable across multiple corpora. The SASI classifier managed to achieve an accuracy of 0.947 and an F-score of 0.827 on the Amazon Reviews corpus, and an accuracy of 0.896 and an F-score of 0.545 on the Twitter corpus.

It is noted as being interesting that while the data enrichment strategy used in the study does improve the accuracy and F-score, it is only by a very small amount (0.945 and 0.812 to 0.947 to 0.827). It is also noted that the high accuracy is achieved due to the biased corpus; there are far more non-sarcastic than

sarcastic sentences due to their rarity.

2.4 Conclusion

Classifiers based on lexical and pragmatic features alone have proved to be ineffective at distinguishing between sarcastic and non-sarcastic sentences, but have also revealed that they do give some indication of its presence, particularly emoticons and features that represent pragmatic features. Pattern based features have also proved to be ineffective, despite being theoretically sound, when the patterns are structured. The only strategy I found in my research that was provably successful was the pattern based SASI classifier.

With these revelations revealed, and due to the success of the SASI algorithm, I will utilise its strategy of using surface patterns extracted from the text itself before clustering and classifying using k-nearest neighbours. Despite making only a small contribution to the accuracy and F-score, I will continue to use the punctuation based features used in the paper (Davidov, Tsur and Rappoport, 2010). I will however not use perform data enrichment as utilised in the paper; the results showed that having a larger corpus of similar language, with no balancing of sarcastic and non-sarcastic sentences, made very little difference to the accuracy and F-score.

The favourable results revealed regarding emoticons and laughter acronyms such as “lol” may lead to, if incorporated into SASI, more accurate classifications. As such I will have this as an extension to my project, investigating if, when implemented in a similar manner to the punctuation based features, the accuracy and F-scores improve or degrade.

3 Classifier Design

As explained in my analysis of related work, I will base my classifying algorithm upon SASI (Davidov, Tsur and Rappoport, 2010). The SASI classifier requires a corpus of natural sarcastic and non-sarcastic sentences, split into five levels based upon how sarcastic they are (1 being not sarcastic, and 5 being clearly sarcastic). From here, SASI extracts patterns from the corpus, before matching the these patterns to sentences to create vectors, and classify based upon the k-nearest neighbouring vectors.

3.1 Appropriate Technologies

To recreate the SASI classifier, I will need to utilise a programming language that supports some form of text comparison and manipulation to create my classifier. Java is an object orientated programming language that supports string comparison and manipulation, allowing it to function as the programming language I will create my classifier with. Other languages also support these features, Python being a popular choice due to it possessing the Natural Language Toolkit, however I am most comfortable programming in Java. Given the time constraints of this project, Java is likely the best choice to ensure the project is completed to schedule.

3.2 Corpora

As used by Kreuz and Caucci (2007), I will utilise Google Books to gather my corpus. This is to test the SASI algorithm in a way not covered in their original study, that of on utterances in a more conversational context. The corpora gained from Amazon Review is in the context of a single person expressing their ideas of a product, they are dictating to nobody in particular without expecting a response. While Twitter can be somewhat conversational, it is commonly used to comment on a particular world or personal event without expressly expecting to have to provide responses to any responses returned. As such, among sarcastic tweets, there is likely a large ratio of tweeters commenting on a world or life event, and those responding to a tweet another Twitter user has tweeted about a world or life event, as compared to those done in a conversational manner.

In the same way that Kreuz and Caucci (2007) gathered their corpus from Google Books, I will utilise the in-built function available using Google Books to search for books containing sentences with the phrase “said sarcastically”. Google Books unfortunately does not support a copy-paste function, thus requiring that I manually type out each sentence. Given that I will be required to pre-process the data into five separate classes, and in other ways (described below), this is not a terrible inconvenience and has not swayed me against using Google Books as my corpus.

I gathered 100 non-sarcastic utterances as my negative training set, and 100 utterances of levels 3-5 as my positive training set. I will use 12 sarcastic utterances of levels 3-5 and 12 non-sarcastic utterances of levels 1-2 for my training set, roughly 10% of the corpus. Each sentence was classified as being sarcastic or

not by a human judge.

3.3 Methodology

3.3.1 Data Pre-Processing

SASI is designed to extract and utilise surface patterns to classify sentences as sarcastic or not. As such, specific patterns are avoided, with general patterns being targeted. As characters in books are likely to address who they are talking to by name, names are replaced with a meta-tag of *[name]*, to help achieve a more general pattern. Characters in books also quite frequently have names created by their authors, thus requiring that changing names to the meta-tags be made by hand, as a corpus of fantasy names does not exist for me to exploit. It is worth noting that I do not replace common nicknames such as Babydoll with the *[name]* meta-tag; I posit that these can potentially be a sign of sarcasm.

Tokenisation is a general NLP problem that I quickly ran into while testing my code; Java lacks a means of distinguishing between an apostrophe and a single quote mark. As there was no automatic solution to this tokenisation problem that I could think of without also affecting apostrophes, and the fact that I already had to manually pre-process the text to remove names, I resorted to preprocessing my data by manually changing single quotes to double quotes.

As Java, to my knowledge, lacks a means of recognising italicised words, which are a potential indicator of sarcasm, I shall pre-process these words by changing them to all capitals. This allows me to highlight the sarcastic intent in a way that Java can read in.

3.3.2 Patterns

The majority of features utilised by SASI are patterns. I automatically extracted the patterns from my corpus by following the algorithm described by Davidov, Tsur and Rapooport (2010) in their paper. This consists of classifying words into one of two categories before creating patterns based on combinations of these two classes that are present in the corpus.

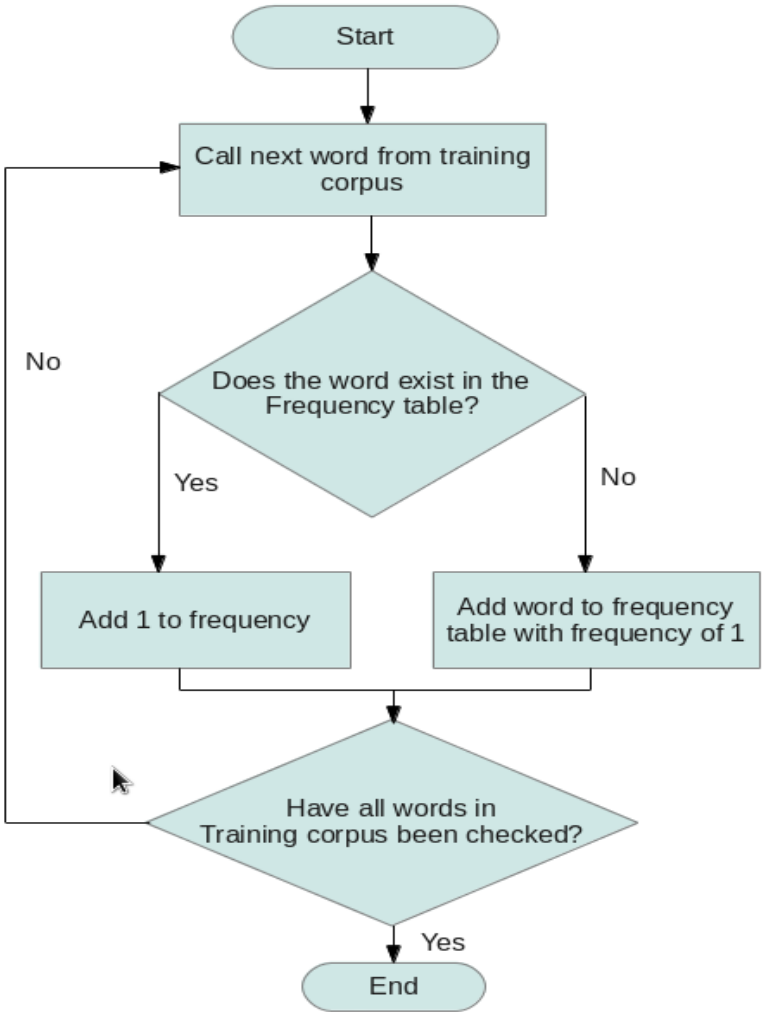
3.3.2.1 Word Classification

Words are classified into one of two classes: High Frequency Words (HFW) and Content Words (CW). Words are classified as being in one of these categories based upon their frequency within the corpus, with words above a threshold value being classified as HFWs, and words below the same threshold being classified as CWs. The *[name]* meta-tag, as well as punctuation, are both also classified as being HFW. Once classified those words that were classified as a CW are replaced in the text by “CW”, with the HFW words remaining as they are.

I accomplished this in Java by first splitting the sentences by words and placed them all into an ArrayList, such that I had an ArrayList of every word in my corpus. An ArrayList was chosen over a simple

array due to the unknown quantity of total words that I would have. From this point I utilised a HashMap to count the frequency, utilising the structure of a HashMap by adding 1 to the value each time the key is found when searching through the ArrayList of words. This took the form of:

3.3.2.1.1 Get Frequency Flowchart



3.3.2.2 Word Classification

With the frequencies gathered, the next step was to classify the words as a HFW or CW, based upon their frequency. A new two dimensional array was created, of the same size as the frequencies HashMap, due to it containing no duplicated words. This was done with the intention of having an array containing each word, with the label of HFW or CW being in the second dimension. As noted previously, the words are classified by their frequency being above or below a certain threshold, which given my small corpus, I chose to be 3.

3.3.2.3 Word Replacement

With the words now each classified as being a HFW or CW, the words classified as a CW needed to be replaced with the *CW* tag to facilitate the extraction of patterns. This was accomplished by first splitting the training set by both sentence and word into a two dimensional ArrayList, such that the ArrayList

contained every training sentence with each of it's words being an individual element, allowing me to perform String comparisons on a word by word basis.

I ran into a tokenisation problem here, namely in regards to how question marks and brackets were read in; both have functions within regular expressions, which String comparison calls upon to perform its comparisons. To solve this, while iterating through the words within each sentence, if a question mark or bracket was found, a preceding "\\" was inserted, to clarify it as a character and not a command.

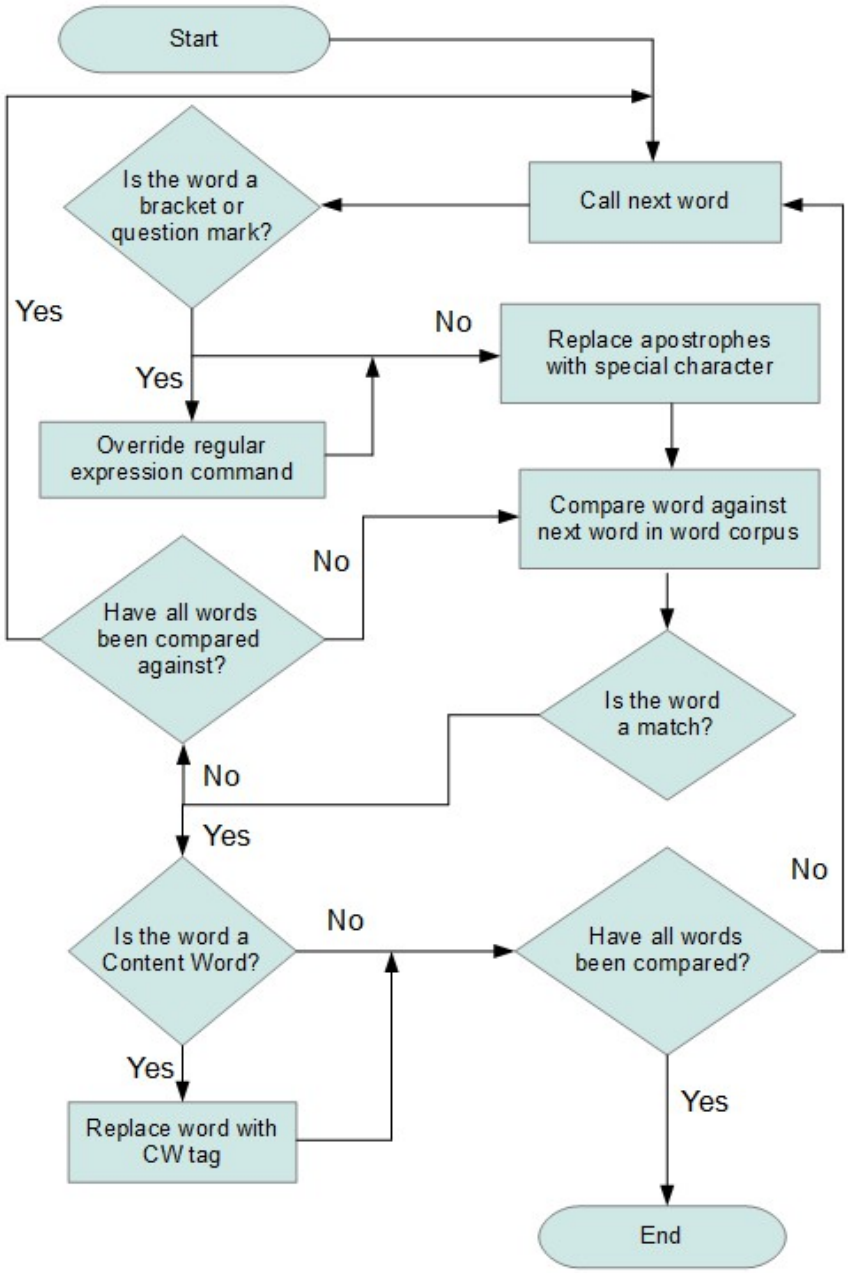
Another tokenisation error was also found at this stage, which was alluded to earlier; apostrophes break word boundaries. As I wanted words such as "It's" to not be counted as two separate words, I decided to, rather than define my own word boundary through regular expressions, use a rather simple fix: I replaced apostrophes with the "*á*" character due to its almost non-existent use in English text, and its status as a alphanumeric character not breaking the word boundary.

With these problems solved, the actual word replacement could begin. This was achieved through the aforementioned String comparison. Each word in each sentence was compared against the words in the word corpus for matches. Where matches were found, the class was checked, with a word class of "CW" requiring that the word be replaced with the "CW" tag through the *replaceAll* method.

An oversight on my part that was revealed through testing, was that this method of word classification and replacement does not take into account new words appearing in the sentences being tested. This oversight was corrected through the implementation of a new method that is run on the test sentences after the initial word replacement method.

The method goes through each word of each test sentence, using String comparison to check if the word exists within the word corpus ("CW" tags are skipped). As spaces are also not included in the corpus, the first letter of the new word is taken to check that it is alphanumeric, before the word is replaced with the "CW" tag.

3.3.2.3.1 Replace Words Flowchart



3.3.2.4 Pattern Extraction

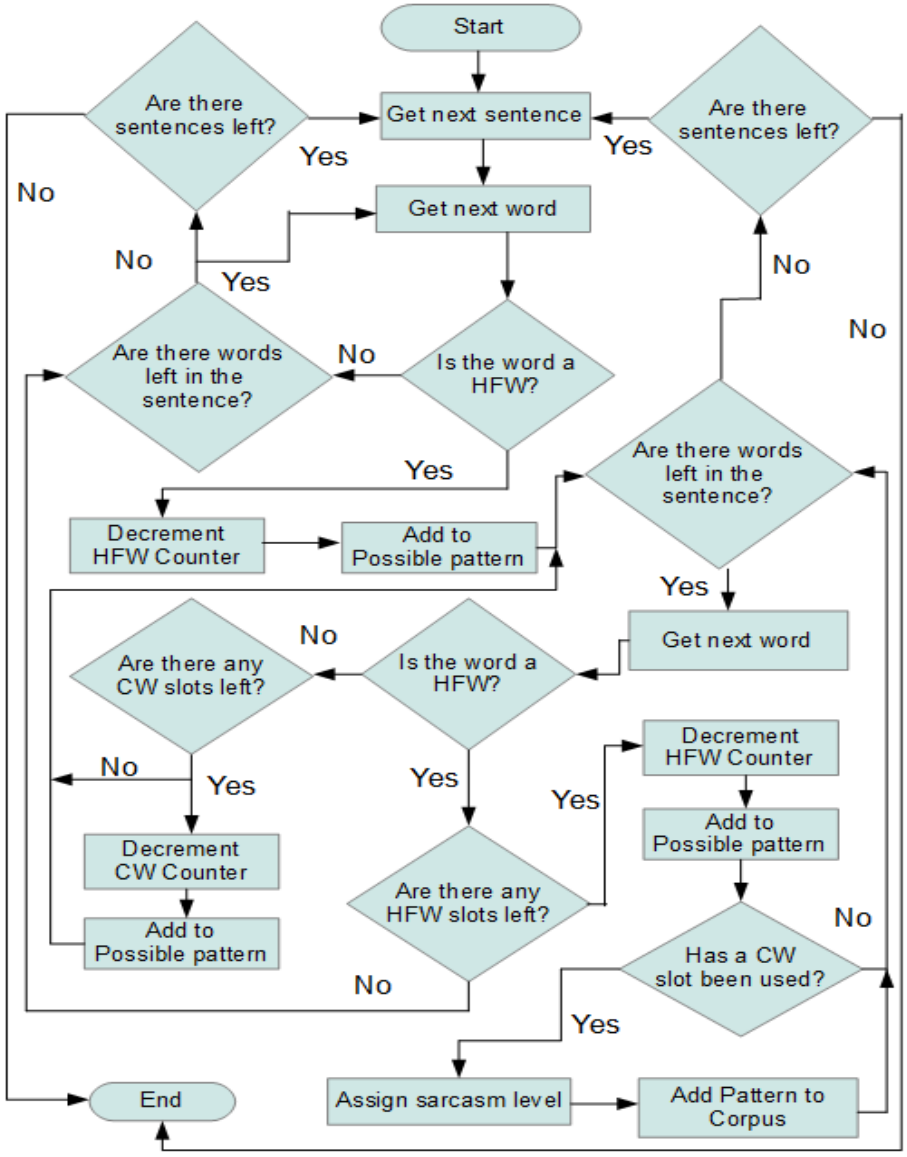
Once all of the words in the corpus have been classified and the content words have been replaced, the extraction of the patterns can begin. The patterns are made up of 2-6 HFWs and 1-6 CWs. Each pattern must begin and end with a HFW, to prevent capture of any multi-word expressions. As such the smallest pattern that can be extracted is HFW, CW, HFW. While this does prevent capture of very short expressions, such as “great!”, which may be sarcastic, at this moment in time I am not searching for such lexical features.

In a similar vein to how Davidov, Tsur and Rappoport (2010) visualised it, I make use of two counters, one for HFW and the other for CW, to allow patterns to be extracted by filling up the slots

available. Utilising my split corpus and String comparison, I search through every word in every sentence till a word is found that isn't a CW. Once a HFW word has been found, the rest of the words in the sentence from this particular word are checked to see if a valid pattern can be formed. If a pattern is formed, the pattern is added to the pattern corpus, before the program continues checking from this point to see if another pattern can be formed with the remaining HFW and CW slots.

Each of the patterns being added to the pattern corpus also have their respective sarcasm level added in a second dimension of the corpus ArrayList, for use later on in the program. This is achieved through a simple counter, as the split sentences are already separated by their sarcasm level they are read in as parameters to the method in their order of sarcasm, with a counter simply appending the respective level to the pattern corpus.

3.3.2.4.1 Pattern Extraction Flowchart



3.3.2.5 Pattern Filtering

Though aiming to gather general, surface patterns, filtering is needed to prevent the patterns getting too general and causing conflicts between the levels of classification. This filtering is done by removing the patterns that are extracted from the training set from both level 1 and 5 sentences.

As previously noted, each pattern has the sarcasm level of the sentence it was extracted from assigned to it. With this information, it is possible to search through the pattern corpus, using String comparisons to compare the patterns that are marked as level 1 against those marked as level 5. If any matches are found, the corpus can then be searched through using String comparisons to find all occurrences of the offending pattern, not just those marked as level 1 or 5, for removal from the pattern corpus.

3.3.3 Pattern Matching

Once patterns from both the training and testing sets have been gathered, and after the patterns from the training set have been filtered, the pattern matching procedure begins. This stage of the classifier attempts to match every pattern in the training and testing sets against every sentence in the training set to create a feature vector for each pattern. Each pattern feature is classified as being in one of four groups:

- Exact Match – An exact match is where all of the pattern components appear in the sentence in the correct order without any additional words between any of the components.
- Sparse Match – A sparse match is where all of the pattern components appear in the sentence in the correct order, but with additional non-matching words between pattern components.
- Incomplete Match – An incomplete match occurs when only 2 or more, but not all, components of the pattern appear in the sentence, with at least one of the components being a HFW.
- No Match – This match only occurs when even an incomplete match is not possible; when no components, no HFW components, or only one pattern component appear in the sentence.

The values of each feature are within the range of 0-1, and are calculated as follows:

- Exact Match – 1
- Sparse Match – α
- Incomplete Match – $\gamma * n/N$
- No Match – 0

With α and γ having the value of 0.1, n being the number of matched components, and N being the total number of components in the pattern. While α and γ can be set anywhere between 0 and 1, I use the same value as used by Davidov, Tsur and Rappoport (2010) to facilitate any comparisons I wish to do with their work.

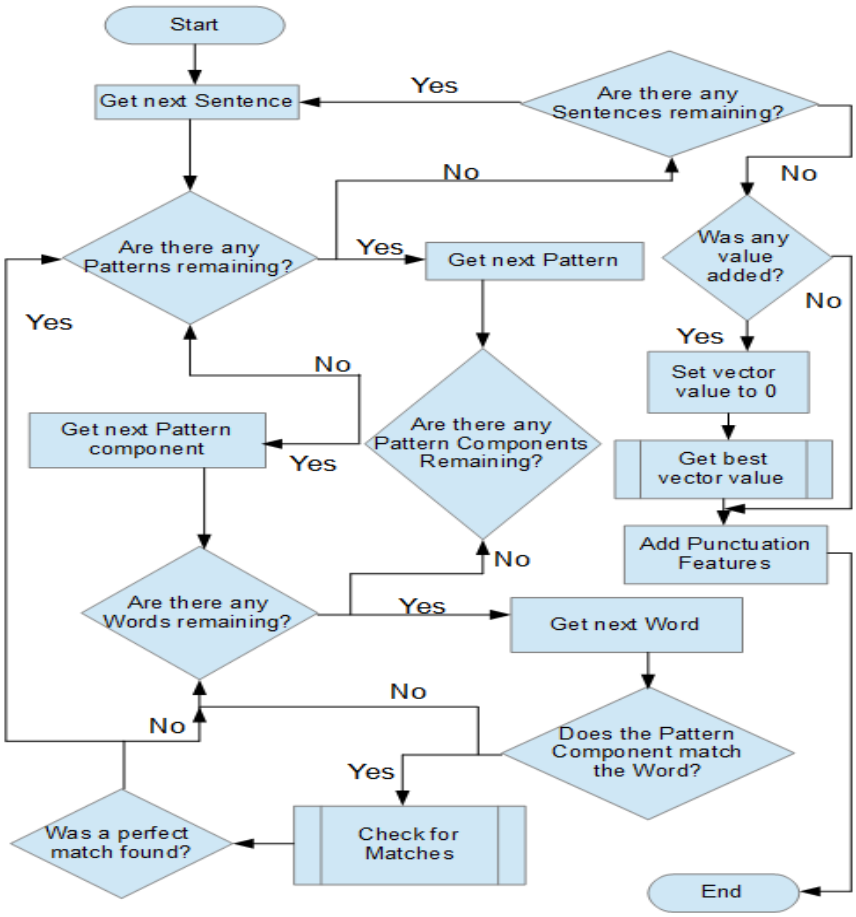
To facilitate String comparisons in a similar vein to how I have been performing comparisons so far, I split the patterns by their components. Each word in each sentence is then compared against each pattern component in each pattern to search for matches.

Where a matching word and component is found, a new loop begins from this point. This new loop attempts to find a match for the pattern by creating a secondary loop that searches for matches from the found component onwards. This secondary loop is used to ensure that all matches are found for a particular pattern.

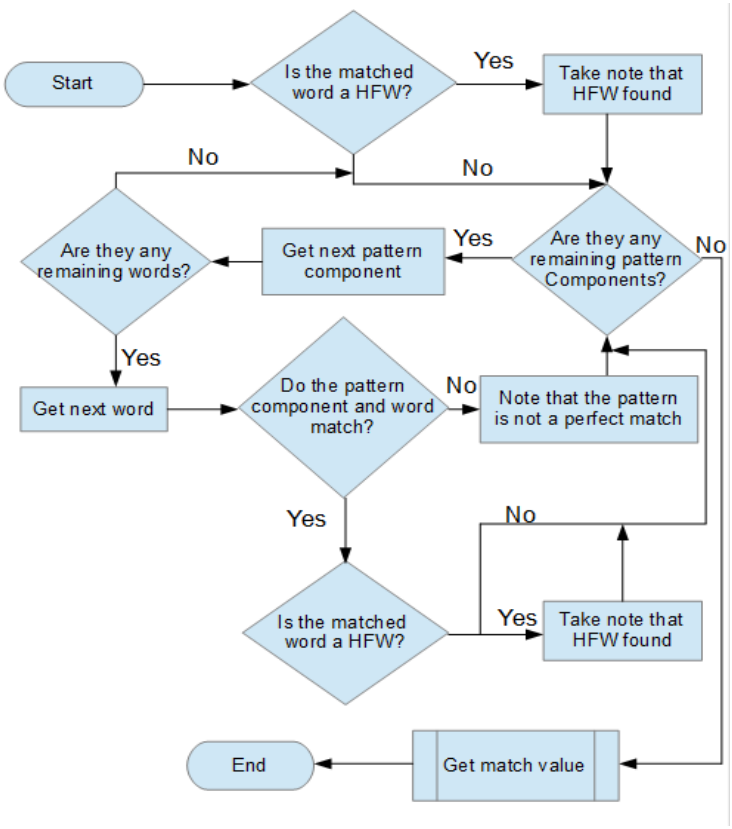
As each match is made, the feature value is calculated and added to an ArrayList of feature values for that particular sentence against that particular pattern. Unless a perfect match is found, this searching for matches continues until there are no more pattern components to run comparisons with.

Once a pattern has been compared in its entirety against a sentence for a match, the best match from the pool of matches is taken as the feature value. This is completed for each pattern for a single sentence before being concatenated together to create a feature vector. At this point the punctuation based feature values are calculated and appended to the feature vector (see section 3.3.4 below), to finalise the feature vector for that sentence.

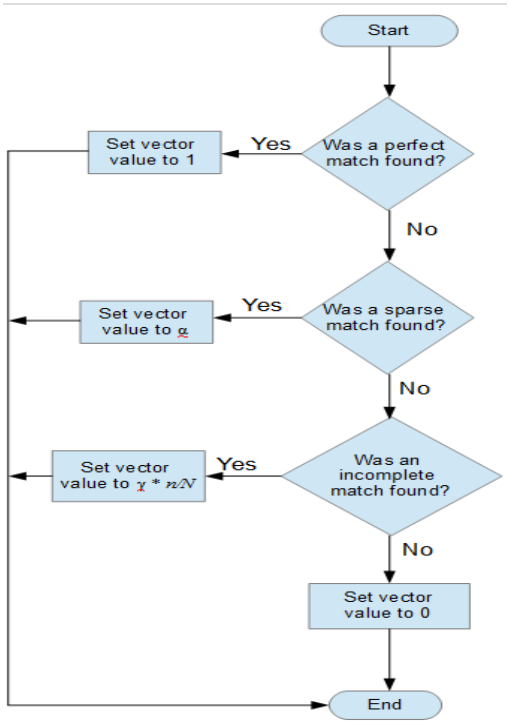
3.3.3.1 Pattern Matching Flowchart



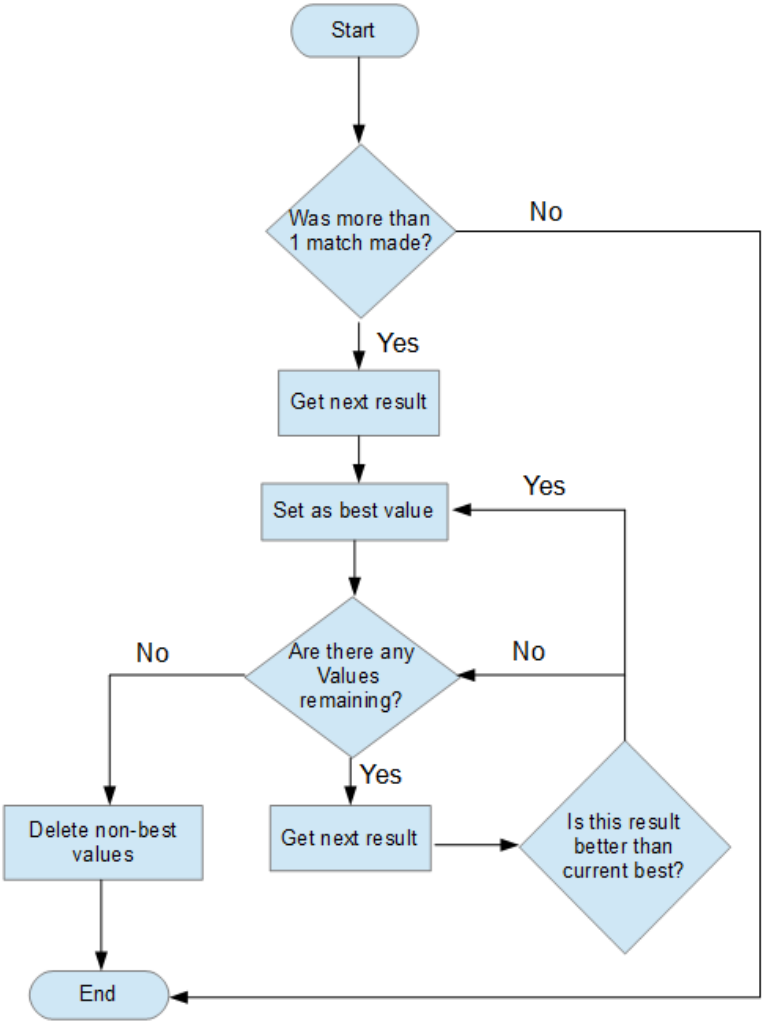
3.3.3.2 Check for Matches



3.3.3.2.1 Get Match Value



3.3.3.2.2 Get Best Vector Value



3.3.4 Punctuation Features

The punctuation-based features used by SASI carry the same weight as a single pattern feature when classifying a sentence as sarcastic or not. This is achieved by normalising each feature by the maximum observed value of that particular feature, placing it within the range of 0-1, the same as a pattern feature. Each punctuation feature is then appended to the feature vector for that sentence, thus allowing it to be used in the classification algorithm. The punctuation features searched for are:

- The length of the sentence in words.
- The number of exclamation points in the sentence.
- The number of question marks in the sentence.
- The number of quotes in the sentence.

- The number of capitalised words in the sentence (e.g. REALLY).

The punctuation features are counted on a sentence by sentence basis, with the method being called after the pattern features have been matched to a feature vector. Each of the punctuation features are counted using String comparison, before being normalised to give the feature value and added to the feature vector.

3.3.5 Classification Algorithm

As previously stated, the SASI classifier utilises a k-nearest neighbours like strategy. The 5 nearest matching, scarce, or incomplete vectors from each testing vector are calculated by euclidean distance, before being used in the classification algorithm. The classification algorithm returns a weighted average of the *k* closest matching vectors. The classification algorithm is as follows (Davidov, Tsur and Rappoport, 2010):

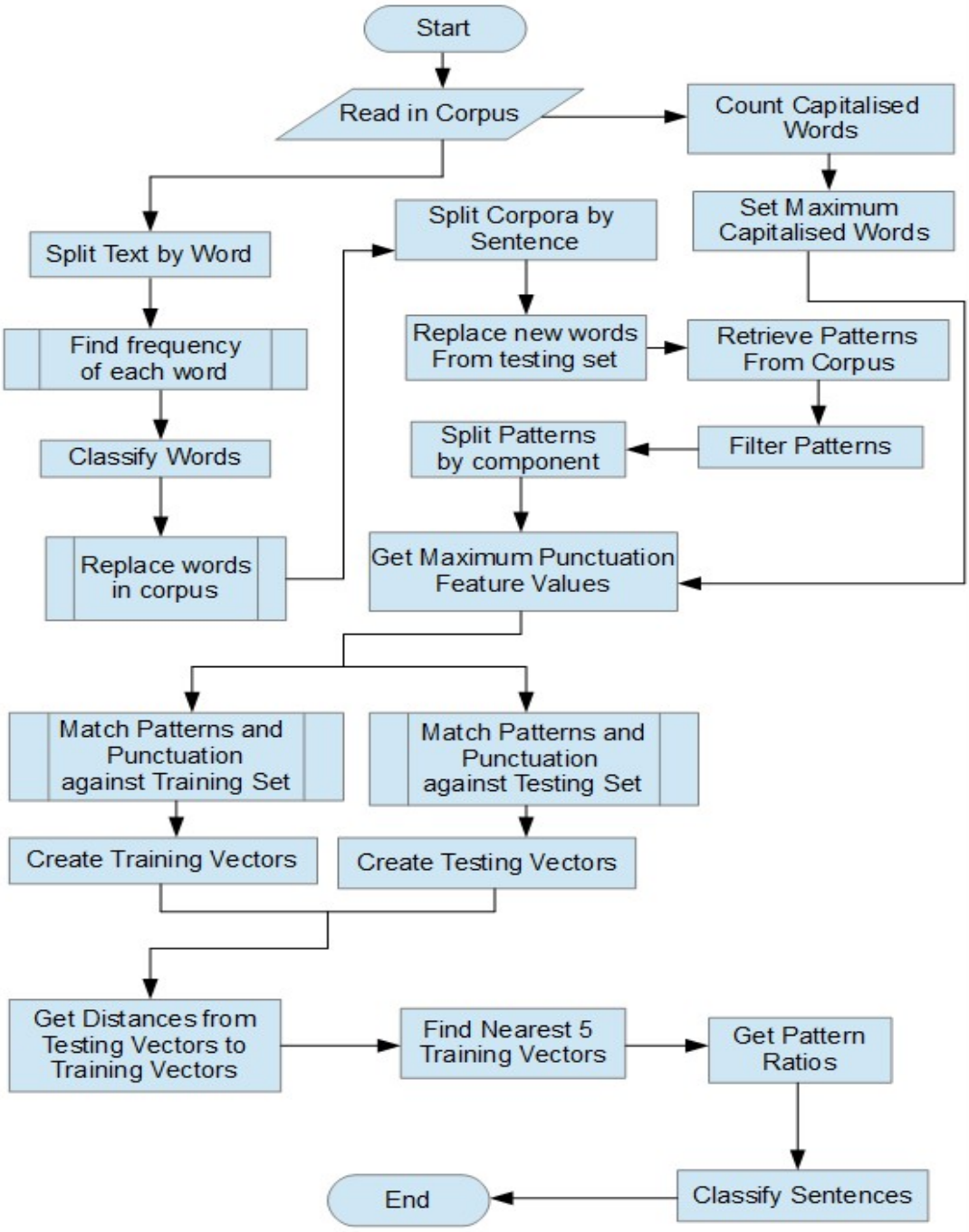
$$Count(l) = \text{Fraction of vectors in the training set with label } l$$

$$Label(v) = \left[\frac{1}{k}\sum_{i}\frac{Count(Label(t_i)) \cdot Label(t_i)}{\sum_j Count(label(t_j))}\right]$$

Where *i and j* are the nearest k vectors.

Where there are less than *k* matching patterns, *k,* instead of being equal to 5, is instead the number of matching patterns. Where there are no matches whatsoever, the sentence is given a label of "1" (clearly not-sarcastic) by default.

3.4 Flowchart of Classifier



3.5 Evaluation Method

I will evaluate the performance of my classifier algorithm against that of a human. Similar to the evaluation method used by González-Ibáñez, Muresan and Wacholder (2011), I will take a set of sentences from my corpus, González-Ibáñez, Muresan and Wacholder use 10% of their corpus (270 tweets), and then get two human judges to classify each as being either sarcastic or not sarcastic. I will take their average accuracy of classifying a sentence as sarcastic or not, and the accuracy of when all judges agree on a sentence being sarcastic or not, as the human baseline interval (HBI). I will then run my classifier on the same sentences and compare it against the HBI.

I will not be comparing my results of classification level against those of a human, as I expect there to be both a large disagreement between which of the five levels each sentence falls into between the judges, and also a large amount of error given the unclear notion of levels of sarcasm, which is likely subjective to the individual.

I will also evaluate my classifier algorithm by comparing my results against the results gained by Davidov, Tsur and Rappoport (2010) in their study. Though using a different baseline, I intend to only perform a results comparison to evaluate whether or not the SASI algorithm can be applied to the conversationally structured utterances found in literature.

3.6 Planned Extensions

3.6.1 Extended Corpora

As one of the extensions of my project is to run it on a larger, more diverse corpus, with available time I will use Twitter to get more unstructured, free form sentences to train and test on. I will gather the tweets utilising the #sarcasm and #sarcastic hash-tags in the same manner as Davidov, Tsur and Rappoport (201), and González-Ibáñez, Muresan and Wacholder (2011).

The tweets would be pre-processed in a similar manner to the corpus gathered from Google Books: names would be replaced with a [name] meta-tag; single quotes would be replaced with double quotes; changing italicised words for fully capitalised words; and getting a judge to classify each tweet based on its sarcasm level within the discrete range of 1-5. Additional pre-processing tasks specific to a Twitter corpus, as detailed by Davidov, Tsur and Rappoport (2010), would be to replace the *ToUser*, hash-tag, and URL link commands with the meta-tags [user], [hash-tag], and [link] respectively.

3.6.2 Additional Lexical and Pragmatic Features

Another of my extensions was to implement additional lexical and pragmatic features that have been shown to be good predictors of sarcastic intent. The features that I would have implemented, as stated previously, were emoticons, and laughter acronyms such as “lol”. Both of these are widely used on the

internet and through text messaging, and have been shown previously to be significant predictors of sarcasm.

An extra pragmatic feature that could be included would be to search for tweets utilising the *ToUser* command; studies have shown that this is an indicator of common ground, which, when combined with other features, is potentially a strong predictor that a tweet may be sarcastic.

A further lexical feature that could be included would be positive interjections, such as "great". Positive interjections have received mixed responses from the papers I studied. In my corpus however, I frequently came across single or two word utterances that consisted solely of a positive interjection (typically with the word "oh" preceding it) that were clearly sarcastic. This would whoever require that a lexicon of positive interjections be either incorporated into the classifier, or a study of the most frequently sarcastic interjections would need to be conducted.

These features could be included into my algorithm in the same way that the punctuation features currently are: each feature will be normalised to be within the range of 0-1, and utilised as a single feature of a vector with the same weight as a pattern.

3.6.3 Other Rhetorical Modes

This extension would be the most complex to implement, if it could be at all. As my classifier is primarily based upon patterns which are learnt from a corpus with a specific target in mind (e.g. irony), either the classifier would need to be separately trained and tested on a corpus with a different focus, or the way the sentences are classified would need an overhaul.

Rhetorical modes that are more explicit in their use would be possible to find, but given a pattern based classifier, a large extension would need to be implemented. Litotes as an example typically take the form of utilising a double negative to imply a positive. Similar to how positive interjections would need to be found, a lexicon of negatives, complimentary adjectives, and possessive determiners would need to be obtained and incorporated into the classifier. Typical patterns of litotes such as possessive determiner – negative - complimentary adjective, could then be searched for.

4 Results

My human baseline interval was created before analysing the results. Using two judges, both of which did not do the initial classification of the test sentences. They achieved a mean accuracy of 68.75%, and an accuracy of 58.33% where they both agreed when classifying as sarcastic or not.

Sentence Number	Human classified level	Classifier classified level
1	1	1
2	1	3
3	1	2
4	1	4
5	1	2
6	1	2
7	1	1
8	1	3
9	2	1
10	2	4
11	2	3
12	2	2
13	3	2
14	3	1
15	3	2
16	4	1
17	4	1
18	4	3
19	4	3
20	4	3
21	5	4
22	5	2
23	5	3
24	5	1

Using binary classification of whether each sentence is correctly classified as sarcastic (levels 3-5) or not (levels 1-2), the classifier achieved:

Accuracy	Precision	Recall	F-Score
0.5	0.5	0.417	0.455

4.1 Comments

These results were very disappointing, given the success Davidov, Tsur and Rappoport had with their algorithm which mine is based upon; they achieved results of 0.912, 0.756, 0.947, and 0.827 for precision, recall, accuracy, and F-score respectively, a far cry away from my own results. My own results seem to consist of primarily level 2's and 3's, over half in fact. This possibly hints at the fact that my classifier is simply unable to decide and is returning middle of the road results.

Given that my classifier is based upon their work, I was expecting similar results. With this in mind, I began investigating why my results were so far off what I predicted.

4.2 Additional Results

Given the disappointing performance I got in comparison to the solid results achieved by SASI (Davidov, Tsur and Rappoport, 2010) for which my classifier is based on, I decided to investigate what results I would get by changing the HFW threshold from my original value of greater than 3.

4.2.1 HFW > 2

Sentence Number	Human classified level	Classifier classified level
1	1	1
2	1	3
3	1	1
4	1	4
5	1	2
6	1	2
7	1	2
8	1	3
9	2	2
10	2	3
11	2	3
12	2	2
13	3	2
14	3	2
15	3	3
16	4	2
17	4	1
18	4	3
19	4	2
20	4	2
21	5	4

22	5	2
23	5	3
24	5	2

Accuracy	Precision	Recall	F-Score
0.458	0.444	0.333	0.381

Lowering the threshold so that more words were HFWs, increasing the number of patterns, gave poorer overall performance.

I next tried raising the threshold to allow fewer HFWs, which gave fewer patterns.

4.2.2 HFW > 4

Sentence Number	Human classified level	Classifier classified level
1	1	1
2	1	3
3	1	1
4	1	4
5	1	2
6	1	1
7	1	3
8	1	1
9	2	2
10	2	4
11	2	4
12	2	2
13	3	2
14	3	1
15	3	2
16	4	2
17	4	2
18	4	2
19	4	3
20	4	2
21	5	4
22	5	2
23	5	3
24	5	2

Accuracy	Precision	Recall	F-Score
0.417	0.375	0.25	0.3

This gave even worse results than lowering the threshold, with a significant overall drop in performance from the original threshold. I decided to raise it higher again, to see if this decreasing trend continued.

4.2.3 HFW > 5

Sentence Number	Human classified level	Classifier classified level
1	1	1
2	1	3
3	1	1
4	1	4
5	1	2
6	1	2
7	1	2
8	1	4
9	2	2
10	2	3
11	2	3
12	2	2
13	3	2
14	3	2
15	3	2
16	4	2
17	4	3
18	4	3
19	4	4
20	4	2
21	5	4
22	5	2
23	5	4
24	5	2

Accuracy	Precision	Recall	F-Score
0.5	0.5	0.417	0.455

Surprisingly, despite having fewer patterns, the same overall results as my initial threshold value were achieved by raising the threshold again. With this result, I raised the threshold twice more. Unfortunately this did not improve the result any further, with it becoming more apparent that

my classifier returning results of 2 and 3 for almost all of the sentences.

4.3 Evaluation

As evidenced by my results, my classifier was not capable of distinguishing between sarcastic and non-sarcastic text gathered from Google Books. The results were worse than my HBI, and far worse than the original results achieved by Davidov, Tsur and Rappoport (2010). It's worth noting that my HBI comparison had quite a range to it, 10.42%, again demonstrating the difficulty of this task for humans, let alone computational methods. This range however is slightly biased in comparison to other, larger corpora however; the small size of the corpus means that even a single disagreement leads to a significant change in performance.

There are several possible reasons for the disparity present in the results. The first of which is the size of my corpus. The original corpus used by Davidov, Tsur and Rappoport (2010), before being data enriched, contained 80 clearly sarcastic sentences to 505 clearly not-sarcastic sentences. Mine in comparison is smaller, being 100 sarcastic sentences of three levels, and 100 non-sarcastic sentences of 2 levels. SO while my corpus is more balanced, their corpus is simply bigger, providing more patterns and a greater opportunity for trends to appear. The SASI classifier utilises a weighted k-nearest neighbours algorithm to classify sentences as sarcastic or not. The nearest neighbours, as noted in the methodology, are selected by the best matching patterns. Due to my small corpus, the number of patterns from each sarcasm level were relatively smaller in comparison to the likely larger number gained by Davidov, Tsur and Rappoport (2010). This can lead to the problem that even if there are three perfect matches for a vector from the same sarcasm level, it still has to fill out the other two slots.

This is leads to another possible reason for my results: insufficient and general patterns. Continuing with the previous example, the remaining two slots will be the next two closest vectors. The problem comes about that the next two closest, given my small corpus size, may be very general vectors from a very different sarcasm level. This alone will sway the result, but if the majority of patterns happen to come from one level, the extra weight from this vector can heavily sway the result towards it if the three matching vectors belong to a less populated class. Had a larger corpus been used, these general patterns may have lost out to more specific, matching patterns.

Observing how my results are produced reveals that though matches are typically being found of the correct level, there are either not enough similar patterns in that level for enough weight to be given, or that some sentences give very general patterns which match better than the less similar, specific ones of the correct level. This is somewhat apparent in my results, where the majority of the results are classified as being of class 2 or 3, which can be reads as the classifier being stating that the sentences are probably not or maybe sarcastic. As stated previously, this is likely the case; when the threshold for HFWs was raised, the classifier began returning predominantly 2's and 3's.

Observing how these results are being produced provides further credence to this theory; the

majority of patterns are in level 1, with another large chunk in level 2, thus giving vectors from these results a large weight, swaying results strongly if they are found nearest. This combined with the fact that even though one or two close matches are found for vectors at the correct level, the other nearest vectors are often not of the same level, the results typically get swayed to a middling level.

Another reason that my results may be so different is the difference in corpora used by myself and Davidov, Tsur and Rappoport (2010). One reason I chose Google Books as my corpus was to test the classifier on a corpus utilising a different style of text than what it was originally tested on. This is however rather unlikely, given the fact that the algorithm proved robust enough to provide solid results on both structured text from Amazon Reviews, and unstructured, free-form text from Twitter.

What is more likely is that either the excerpts I extracted from Google Books are not good representations of sarcasm, or the initial classification of sentences was error prone. There is some evidence to support this theory; the classifier consistently gets bad results for the sentences classified as level 2. This is likely either a case of the sentences being initially classified, or that the patterns extracted from the utterances classified as level 2 bear little similarity to each other, thus causing other sentences to be better matches.

Given this speculation, it is possible that my classifier itself could accurately distinguish between sarcastic and non-sarcastic sentences, potentially achieving results similar to those achieved by Davidov, Tsur and Rappoport (2010). The flaw may only lie with the fact that the classifier requires a large corpus to function effectively.

4.4 Conclusion

I can conclude from this project, that while it is seemingly possible to distinguish between sarcastic and non-sarcastic utterances in text, it is a difficult problem, with my own attempt being insufficient to provide accurate classification. Pattern-based features utilised with clustering appear to be a very strong predictor of sarcasm, with my own falling short due to a weakness in the design. It would appear from my evaluation that the failure of my classifier came simply from my small corpus, though given the chance I would definitely like to run the classifier with a larger corpus to test this theory.

I have met all four of the minimum requirements of the project, with varying degrees of success: Analysis of textual characteristics that correlate with sarcasm; comparison of different approaches; A classifier algorithm which given English text input will classify a sentence as being sarcastic or not; and Evaluation of results from a corpus.

Researching past work by others provided me with both a good comprehension of textual characteristics associated with sarcasm, and also a comparison of different approaches. The research done on the lexical and pragmatic aspects of sarcasm, provides both a classification approach, and useful information about which textual features are effective at distinguishing sarcastic sentences.

After comparing the results from the pattern-based approach against the lexical and pragmatic based approach, would lead to the conclusion that the classification of sarcastic sentences is best served with a primarily surface-pattern-based approach. An approach that fully merges the solid results achieved by the surface pattern based classification method, and the strong lexical and pragmatic predictors, would likely provide the best results at this time.

Though I designed and created a classifier to distinguish between sarcastic and non-sarcastic sentences, and evaluated it on natural text gathered from a corpus, it proved to be unsuccessful, likely due to the small size of the corpus. The classifier itself works, with my own testing of dummy sentences proving that it will correctly label a sentence as sarcastic if there are enough similar patterns present within the desired level.

4.4.1 Future Improvements

The first obvious improvement would be to run the classifier on a larger corpus, to investigate whether it was just the size of my corpus, the corpus itself, or something unseen that was to blame for the failure of my classifier. Either gathering a larger corpus from Google Books, performing my extension of utilising multiple corpora to gather a diverse range of sentences, or simply using a different, larger corpus would all likely lead to improved results. The data enrichment utilised by Davidov, Tsur and Rappoport (2010), given that it did not negatively affect their results, would also likely be a good way to quickly increase the size of a corpus.

Implementing my other extensions, particularly the extension of searching for smileys and laughter acronyms given their success, would likely lead to an improvement in the classification, given a larger and better corpus first of course. Implementing them as described in my methodology, into the feature vector, would provide each sentence with another feature to look for, potentially overcoming the generalisation of patterns that my own project would appear to have suffered from.

5 Project Reflection

Overall I enjoyed the project experience. I have found it very interesting to research a problem that has not yet been fully solved, on a topic that is largely an unknown field. It has highlighted to myself that plans are always rough guidelines, due to it being impossible to predict how a project will evolve over a period of time, or how long certain tasks can take. My background reading is a good example of this, from my Gantt chart it can be seen that I intended for it to be finished in about 2 weeks time, instead it took me far longer due to complexity of the problem. This is where utilising an iterative project life cycle as I did comes in great use; planning for things to be changed is a very good strategy, as it allows you to realise quickly when you are falling behind schedule and need to take measures to correct.

Despite my implementation of an iterative life cycle, I did fall behind and could not catch back up to my planned schedule, having to particularly rush the last few sections due to me largely ignoring my own schedule. Given a second chance, I would ensure that I begin early and keep a strong work pace throughout and keep to the schedule, rather than working at a fairly leisurely pace, and picking up the pace when a deadline begins to loom. I would also not only add to the report in small chunks, instead filling it out as I go. This would facilitate the option of using my supervisor to the fullest, asking for opinions and how to improve each step of the way, instead of gambling on the presumption that I have already been doing it correctly.

6 Appendices

6.1 References

02d9656.netsoljsp.com. 2008. *SarcMark - Home*. [online] Available at: http://02d9656.netsoljsp.com/SarcMark/modules/user/commonfiles/loadhome.do [Accessed: 26 Aug 2013].

Bogard, D. 2008. *Living A Dream*. AuthorHouse, p. 16.

Carvalho, P., Silva, M., Sarmento, L. and De Oliveira, E. 2009. *Clues for Detecting Irony in User-Generated Contents: Oh...!! It's "so easy" ;-)*. [pdf] Hong Kong, China: ACM. Available through: Google Scholar http://xldb.di.fc.ul.pt/xldb/publications/Carvalho09:Clues:Detecting:Irony_document.pdf [Accessed: 4 March 2013].

Clark, H. 1996. *Using Language*. [e-book] Cambridge University Press. Available through: Google Books http://books.google.co.uk/books?id=DiWBGOP-YnoC&printsec=frontcover&dq=using+language&hl=en&sa=X&ei=joLEUZruCIzasgagr4DwDw&ved=0CDMQ6AEwAA#v=onepage&q=using%20language&f=false [Accessed: 6 June 2013].

Clifford, S. 2002. *The Storm Before My Calm: Class of '98 Forever*. iUniverse, p. 500.

Davidov, D., Tsur, O. and Rappoport, A. 2010. "Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon", paper presented at *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, Uppsala, Sweden, 15-16 July. Association for Computational Linguistics, pp. 107-116.

Dictionary.com. 2013. *the definition of sarcasm*. [online] Available at: http://dictionary.reference.com/browse/sarcasm [Accessed: 8 Jun 2013].

Dictionary.com, 2013. *the definition of verbal irony*. [online] Available at:

http://dictionary.reference.com/browse/verbal+irony [Accessed: 8 Jun 2013].

Ellis, L. and Ellis, D. 2010. *The Adventures of the Teen Archaeologists: The Land of the Moepek*. AuthorHouse, p. 41.

Gibbs Jr., R. and O'brien, J. 1991. Psychological aspects of irony understanding. *Journal of Pragmatics*, 16 (6), pp. 523-530. [Accessed: 7 Jun 2013].

González-Ibáñez, R., Muresan, S. and Wacholder, N. 2011. "Identifying Sarcasm in Twitter: A Closer Look", paper presented at *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics:shortpapers*, Portland, Oregon, 19-24 June 2011. Association for Computational Linguistics, pp. 581-586.

Jurafsky, D. and Martin, J. 2009. *Speech and Language processing*. Upper Saddle River: Pearson Education (US).

Kreuz, R. and Caucci, G. 2007. "Lexical Influences on the Perception of Sarcasm", paper presented at *Proceedings of the Workshop on Computational Approaches to Figurative Language*, Rochester, New York, 26 April. Association for Computational Linguistics, pp. 1-4.

Kreuz, R. and Glucksberg, S. 1989. How to be Sarcastic: The Echoic Reminder Theory of Verbal Irony. *Journal of Experimental Psychology*, 118 (4), pp. 374-386. Available at: http://psycnet.apa.org/?fa=main.doiLanding&doi=10.1037/0096-3445.118.4.374 [Accessed: 7 Jun 2013].

Kreuz, R. and Link, K. 2002. Asymmetries in the use of Verbal Irony. *Journal of Language and Social Psychology*, 21 (2), pp. 127-148. Available at: http://www.ffri.hr/~ibrdar/komunikacija/seminari/Kreuz,%202002%20-%20Verbal%20irony%20and%20asymmetries.pdf [Accessed: 6 June 2013].

Merriam-webster.com. 2012. *Sarcasm - Definition and More from the Free Merriam-Webster Dictionary*. [online] Available at: http://www.merriam-webster.com/dictionary/sarcasm [Accessed: 10 Jun 2013].

Pennebaker, J., Booth, R. and Francis, M. 2007. *Linguistic Inquiry and Word Count*. [e-book] Austin, Texas: LIWC.net. Available through: Google Scholar http://homepage.psy.utexas.edu/homepage/faculty/pennebaker/reprints/LIWC2007_OperatorManual.pdf [Accessed: 10 Jun 2013].

Sarcasmsociety.com. n.d.. *Irony vs Sarcasm*. [online] Available at: http://www.sarcasmsociety.com/sarcasm-and-irony.html [Accessed: 10 Jun 2013].

Strapparava, C. and Valitutti, A. 2004. "WordNet-Affect: an Affective Extension of WordNet", paper presented at *Proceedings of LREC*, pp. 1083-1086.

Tsur, O., Davidov, D. and Rappoport, A. 2010. *ICWSM – A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews*. [pdf] Association for the Advancement of Artifical Intelligence. Available through: Google Scholar http://eprints.pascal-network.org/archive/00006940/01/sarcasmAmazonICWSM10.pdf [Accessed: 4 March 2013].

Utsumi, A. 2000. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics*, 32 pp. 1777-1806. Available at: http://www.utm.se.uec.ac.jp/~utsumi/paper/jop2000-utsumi.pdf [Accessed: 7 June 2013].

Yahoo! Contributor Network. 2008. *How to Tell If Someone is Being Sarcastic*. [online] Available at: http://voices.yahoo.com/how-tell-if-someone-being-sarcastic-2128736.html [Accessed: 9 Jul 2013].

6.2 Gantt Chart