Sarcasm Detection in Twitter

Independent Project Report CS – 7650

Vivek Nabhi

Vinodh Krishnan

1. Introduction

Sarcasm is an example of what researchers call **'unplain speaking'**, a form of speech where what the speaker says is not the same as what he means. The message therefore is implicit and not entirely obvious. Sarcasm is often used as a means to humor or insult someone. A common form of a sarcastic statement consists of a positive sentiment generally followed by an undesirable activity or a negative situation [2]. Sarcasm is primarily a verbal function. The context of the speaker and the pitch and tone of his voice help us identify sarcasm while speaking. In text, sarcasm is much more difficult for even people to identify, and thus getting a system to do the same is expected to be a more challenging task.

Why detect sarcasm in text?

Detecting sarcasm is a very important field in Natural Language Processing. In areas such as sentiment analysis and text summarization taking sarcastic statements literally might result in a completely incorrect analysis of the data. For example, statements like "I love Justin Beiber" might mislead sentiment analyzers because of their use of strong positive words like 'love' but in actuality express a negative sentiment.

Sarcasm detection can therefore help improve NLP systems like product review summarization (text summarization), brand monitoring, dialogue systems and sentiment analysis. Sarcasm detection can also help in conflict resolution - Many conflicts arise between writers and politicians/celebrities due to misunderstanding of sarcasm in written text, which can be avoided through effective Sarcasm Detection.

2. Previous Work on sarcasm detection

"Sarcasm is a little bit like the weather; we all know it is there, but so far nobody has done much about it." *- Anonymous*

Although sarcasm has been an element in discourse and text matter for centuries, there has not been as many attempts to study it in Natural Language Processing because of its inherent ambiguity. Simply put, people themselves have trouble identifying sarcasm, so how would computers detect them? Some of the earlier work on automatic sarcasm detection has relied on Speech related cues like laughter and prosody. We are more interested in dealing with sarcasm identification in text for this project, specifically handling Twitter data. Two papers that are closely related to the problem we are looking to tackle are, Semi supervised Recognition of Sarcastic Sentences in Twitter and Amazon[1], and Sarcasm as Contrast between a Positive Sentiment and a Negative Situation[2]. Both these papers have one common theme – they assume or attempt to build a pattern that for sarcastic tweets on Twitter.

[1] deals with using Surface Patterns constructed based on combinations of High Frequency words and Content Words, and Punctuation Features to construct a feature vector. It then uses a training set with sentences labeled from 1 to 5 depending on the level of sarcasm (1- lowest, 5 – highest), with a k Nearest Neighbour classifier to learn the weights for the features. The problem however is, Sarcasm is primarily

signaled by contrast (of positive words in a negative sentiment) and while surface patterns do try to establish a pattern, they are not based on the sentiment of words used but on the frequency of words.
A more recent, relevant approach is [2] that deals with using contrast between positive and negative sentiment words to identify Sarcasm. They make an assumption that sarcastic sentences, especially in Twitter occur as a contrast of positive phrases and negative sentiment which they show, is a fair assumption to make. They use bootstrapping, intuitively building sets of positive and negative words from a seed word and the training data. In every step they consider 1 – 3 grams of words that occur immediately after positive words and before negative words and add them to the negative or positive word set respectively based on a score. These clusters are then used in conjunction with SVM classifiers built on unigram and bigram features to classify whether a sentence is sarcastic or not.

There are ways in which we can improve upon this. We can leverage the internet to obtain information about the sentiment of a phrase that we were otherwise not able to obtain in our training set. They have not considered Syntactic or Punctuation features either which have been proved to be good indications of sarcasm in text[1].

The rest of the report is structured as follows. In the next section we look at how we tackled the problem of Sarcasm Identification and built a baseline system. Then in Section 4, we describe the improvements we made to the baseline and why we believed it would work. In Section 5 we describe the Experiments we conducted and the Results we got and see how it fared against the baseline. Finally in Section 6 we conclude by summarizing our findings and suggest ways for improvement.

3. Basic System

Twitter data makes it harder but at the same time easier to detect Sarcasm. Harder, because it is restricted to 140 characters, therefore there is not a lot of context that can be extracted and it relies a lot on the "world knowledge" of the reader to identify it. For instance, "I love waking up early for work in this cold weather" can be easily gauged as sarcastic by us because we know that very few people prefer waking up early in cold weather but this may not be known to the Natural Language Processor. Easier, because of the exact same reason. Since it is restricted to 140 characters, we know that if there is a contrast, it should be within these one or two lines in the tweet. This is the idea behind employing bootstrapping to this problem by [2], and it is this method that we implement as our baseline.

Our baseline system thus consists of 2 parts.

1. **Bootstrapping**

   We have already looked at the basic intuition behind employing bootstrapping. To implement bootstrapping, we assume that sarcastic tweets follow a specific pattern :
   [+ VERB PHRASE] [- SITUATION PHRASE]
   The structural assumption that positive verb phrase are immediately followed by negative situational phrases in sarcastic tweets, drives the bootstrapping algorithm. Although the contrast can also be achieved with a positive Noun Phrase and a negative situation phrase, we assume VPs as positive phrases, as NPs are more prone to ambiguity than VPs due to more possibilities. This contrast is also seen to be most common on twitter, therefore seems like a good pattern to assume. This contrast is thus chosen as the source of sarcasm in the tweet.

In the training step the algorithm builds 3 kinds of word sets, Positive words, Negative words and Positive Predicate words that are then used to make predictions. All sets are initially empty except the Positive word set that contains the seed word.

The algorithm starts with a positive seed word, in our case, "love". We extract ngrams (upto 3) that occur directly after the positive phrase from the positive instances of sarcastic tweets in our training set. These will be negative situation phrases based on our assumption of the sentence structure for positive instances of sarcasm. We then filter them based on POS tags (Verb Phrases and their derivatives) so that we can have a meaningful syntax structure where the negative situation represents some activity or state. We score them based on how powerful the word is in determining whether the tweet is sarcastic or not by using the frequency of its occurrence in the negative instances of sarcasm in our training set. The exact formula is

$$\text{Score} = N_s/N_t$$

N_s - Number of times the negative phrase follows a positive sentiment and is sarcastic (obtained from the positive instances of sarcasm in the training set).
N_t - Total number of times the negative phrase follows a positive sentiment in the entire training set (both positive and negative instances of sarcasm present in the training set).
Essentially this is the joint probability of a tweet being sarcastic and having a structure consisting of the positive word and the negative phrase in question.

We choose a certain threshold to compare the scores to, so that we ensure that we take only the ones that are discriminative enough in classifying tweets as sarcastic or not. This also ensures that the list does not grow endlessly and stops at a certain point (either when all the words that have a score greater than 0.7 or when the words obtained are already present in the training set). The threshold value of 0.7 is determined through experimentation and earlier work. We compare the scores of the top 20 n grams to the threshold and only take one that are higher than it.

To build the positive word set, we follow the same procedure but only in this case, we extract n grams (upto 2) occurring before each negative phrase from our negative words set built in the previous step. This is once again done to keep in line with our assumption that a positive verb phrase occurs before a negative situation. The scoring is done in a similar way and the top 10 words (again greater than the threshold) are added to our positive words set.

Based on the same assumption that sarcasm occurs as a contrast between positive phrases and negative situations we extract tweets that contain a negative situation and a positive phrase in close proximity. This is done to handle cases like "My iphone was stolen. This is great", where the positive part of the contrast occurs as a predicate to the negative situation. To handle this we extract n grams (upto 3) that occur within 5 words on either side of the negative word. We filter and score them as before and compare them to the threshold, and add them to our positive predicate set.

These 3 sets are built iteratively in the same manner until they reach convergence and no more words are added, as illustrated in figure 1. In most cases the list converges because the threshold score ensures that common words like "the", "and", "a", etc are not considered. Thus, words which co-occur with those selected by the bootstrapper more often in the positive instances of our training set than the negative instances, are generally selected. These sets are then employed in

the prediction task. If a positive or a positive predicate word from their corresponding wordsets occur in close proximity to the negative words and maintain the same structure, they are determined as a positive instance of sarcasm.

2. **Support Vector Machines**

   For our baseline, we employ SVM to classify sarcastic and non - sarcastic sentences using unigram and bigram features. We employ 10 fold cross validation using the RBF Kernel for accurately determining the hyper parameters. The unigram and bigram features have binary values indicating their presence and absence. This is then trained on our training set.

   During testing, each one of our components makes a prediction on whether a tweet from the test set is sarcastic or not. Then if either one of our components says it is sarcastic, the tweet is marked as a positive case of sarcasm. The reason for doing this is simple. While bootstrapping is very effective in determining sarcasm in terms of precision, its recall is low as you might expect because of our relatively strong assumptions. SVM on the other hand has a high recall but a relatively lower precision value since we use unigrams and bigrams from positive instances of sarcasm, but no other feature derived for the specific case of sarcasm. We observe that the sarcastic tweets that SVM fails to recognize, are recognized by the bootstrapping method because of its more stringent assumptions to fit the sarcasm case. So augmenting bootstrapping with SVM helps in improving the recall significantly, while not affecting precision at the same time.

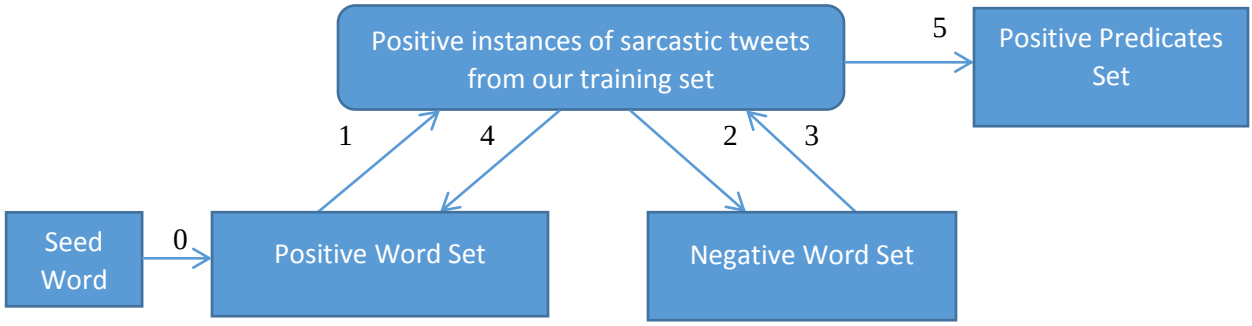

Figure 1: Steps at every iteration of the bootstrap algorithm. The order followed is 1 – 2, 3 – 4, 3 – 5. There is no return arrow from positive predicates set as that is not employed to get negative or positive words. All sets are initially empty except Positive word set that contains the seed word ("love") that it gets from 0.

4. Our Improvements on Baseline

While the baseline seems to perform well for sarcasm detection, giving a recall of 44%, a precision of 62% and F measure of 51%, we can think of other features that can be added to the SVM or bootstrapping to improve the F measure. Our improvements can be broadly divided to 3 classes: Improvements to SVM (Syntactic and Twitter Based Features), Improvements to bootstrapping and Introduction of knowledge through google search augmentation.

a. Improvements To SVM

The SVM of the baseline takes into account only unigram and bigram features. There are several other features that can be employed to improve this

Syntactic Features – We employed syntactic features like Sentence Length (in characters/in words), stemming and lower case unigrams and bigrams. Stemming caused a drop in F measure so it was removed from the final list. Sentence length and lower case caused significant improvement in the F measure. The Sentence Length can be expected to be a key feature since sarcastic tweets can be observed to be smaller than most other tweets.

Punctuation features – Punctuation features like ellipsis and exclamation marks were employed in the form of binary features. Ellipsis and multiple exclamation marks are indicative of sarcastic tweets.

Twitter Specific Features – Twitter specific features like @mentions where also considered. It can be seen that more people pass sarcastic remarks in conversation than in a general tweet, and thus, this feature might be discriminative between sarcastic and non-sarcastic tweets.

Features based on Smilies could not be implemented as they were not UTF-8 encoded and therefore it could not be read properly. This can be a pretty useful feature and can be implemented as a future improvement.

10 fold cross validation with RBF kernel was also performed with the training set to determine the right set of hyper parameters. The 10 fold cross validation for the RBF kernel using syntactic features along with the unigrams yielded a value of C= 2 and gamma = 0.02125 for the hyperparameters.

b. Improvements to bootstrapping

The baseline implementation of bootstrapping had a good precision (64%) but a very low recall (10%). Thus for improving baseline we mainly focused on improvements that improved recall, without affecting the precision. We improved performance of bootstrapping by adding 2 key steps in word extraction and filtering.

Lemmatization – Lemmatization deals with converting verb tenses to their base forms. Since bootstrapping deals mainly with extraction of verb phrases while building the positive and negative word set, it can be reasoned that this would be more suited to bootstrapping than stemming which operates on all words (including nouns). Lemmatization increases recall ensuring that more tweets that are sarcastic are classified as sarcastic, but it also increases false positives, decreasing precision marginally.

Leniency towards trigrams – In the baseline method, all words obtained were filtered based on frequency (only if it occurred greater than 3 times it was considered) and score before adding to the respective (positive or negative) sets. This resulted in very few trigrams being included in the negative and positive word set. In our case, we reduced the limit for frequency and score to ensure more trigrams got into the set. This decreased the recall because trigrams occur more rarely than the corresponding unigrams or bigrams, but it increased the precision.

c. Using Search engine for additional knowledge

While improvements to bootstrapping and SVM caused improvements in recall and F measure, we were not able to improve precision. To do this, we decided to add another method to the pipeline that would increase precision, but at the same time not reduce recall.

The idea was to incorporate knowledge about whether a phrase was negative or positive through the internet. Since tweets are small, it is impossible to find out if a phrase is of negative or positive sentiment, just by looking at the tweet context. So instead, we passed this phrase to a search engine, searching for occurrences of the word in blog posts (since it is a subjective source). We took the first n results and did sentiment analysis on them to get the sentiment of the phrase. In our case, we performed this on the test set by extracting 4 grams occurring immediately after a positive word and determining the sentiment of the phrase. If it was negative, we would classify it as sarcastic, since it is an instance of a negative phrase occurring after a positive word.

It's easy to see why we believed this would work. This module tries to improve precision while not affecting the recall by leveraging 4-grams. While bootstrapping and SVM consider n grams upto 3, it does not consider higher n grams due to sparsity and unavailability of data. This method also makes up for a reduced training set by considering phrases that may actually be positive or negative but may not be seen in our training set.

We took 4 grams so that we could get a reasonable sentiment for the phrase (unigrams like "walking" or "talking" can't really have a sentiment, whereas phrases like "talking loudly on the phone" has a negative sentiment). This method was added in conjunction with the bootstrapping and SVM and was used only when both the methods marked a tweet as a negative instance of sarcasm to ensure that it did not affect the recall while contributing to the precision (it can reduce false positives). The method is illustrated in Figure 2.

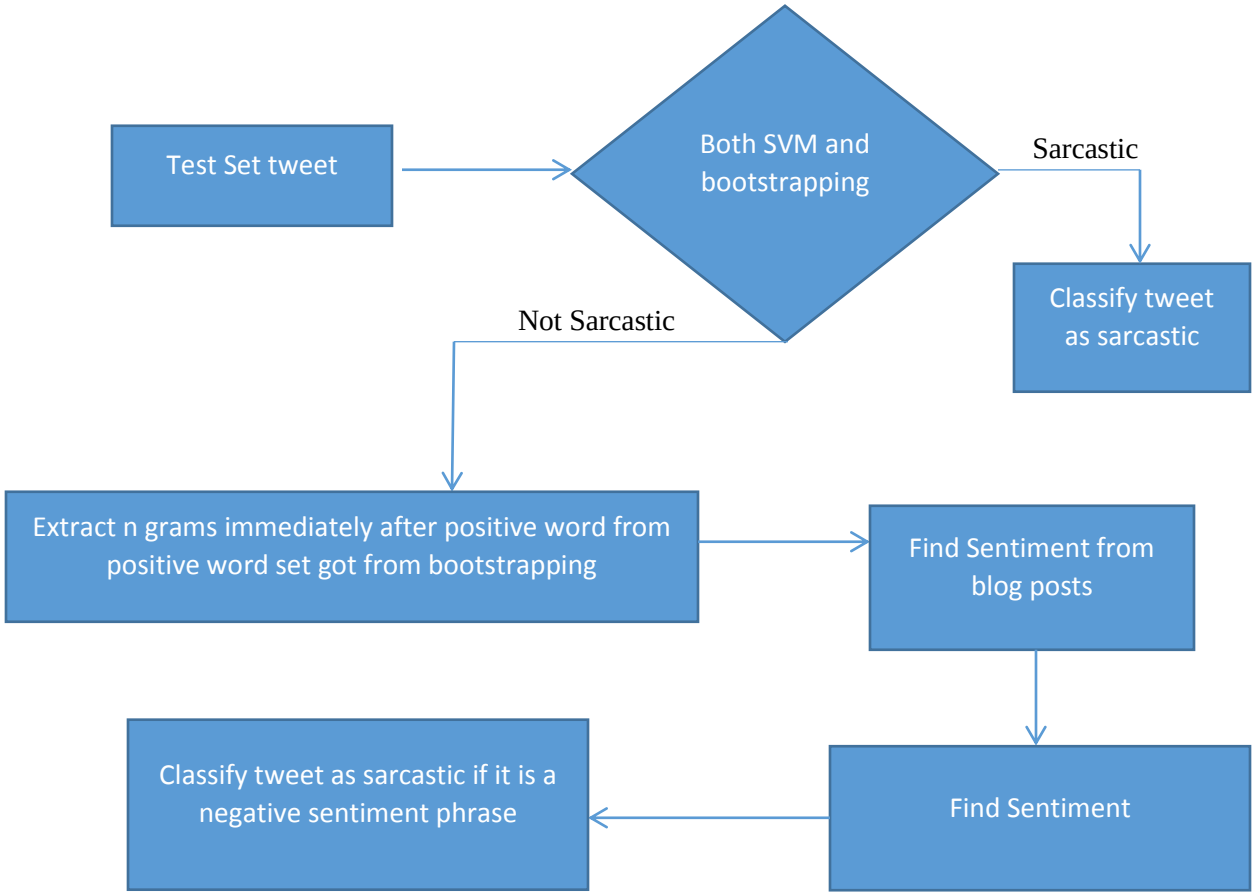

Figure 2: Flow chart for Search engine augmentation for increased knowledge. It extracts 4 gram phrases and finds sentiment of its occurrences online. Thus for example, the phrase "going to the dentist" was classified as negative as lots of people have blogged about it in a negative context. Thus, "Yay! I love going to the dentist " will be identified as sarcastic by this, whereas bootstrapping and SVM will not be able to recognize this if it is absent in the training set.

5. Experiments and Results

Data

We used Twitter API to extract data for training and test sets. Positive instances of sarcasm for training set were extracted by considering tweets that had the hashtag, #sarcasm or #sarcastic. These are shown to be a reasonably good set to train upon. We extracted negative instances of sarcasm from the twitter stream api after filtering based on language and hashtag. We collected a total of 23853 tweets, out of which 3920 were positive instances of sarcasm (containing #sarcasm or #sarcastic). This was our training set.

We obtained the gold standard test set from Prof Ellen Riloff from the University of Utah, through Prof Jacob Eisenstein. These were annotated by 3 experts with an IAA of 0.80 according to Riloff et al[2]. It consisted of 2232 tweets, out of which 505 were positive instances of sarcasm. This is lower than the original test set of 3000 tweets with 693 positive instances, as some tweet ids in the test data set had expired. If we consider the baseline to classify every tweet as sarcastic, we would achieve an accuracy of 22.5%.

Experiments

Several experiments were conducted based on different features. We categorize them as experiments on bootstrapping and experiments on SVM.

Evaluation Measure

The evaluation of the current baseline and our methods are done based on Recall, Precision and F-measure. Recall is essentially what fraction of the sarcastic tweets in the test set, the system was able to identify (Recall = $t_p/(t_p+f_n)$, where t_p – true positive, f_n – false negative). Precision is how many of the tweets that the system classified as sarcastic are actually sarcastic (Precision = $t_p/(t_p+f_p)$, where t_p – true positive, f_p – false positive). F-measure is calculated as the harmonic mean of the Recall and Precision.

Bootstrapping

Seed Word – Different seed words were tried, as this is the initial parameter to construct the negative word set which is employed to construct positive sets. 'Love' gave us the best words, and was thus chosen to be the seed word. The total number of positive, negative and positive predicate words obtained was 19, 55 and 40 respectively. Some of the words are shown in figure 3.

As you can see, the words in each cluster make sense. Also words like Black Friday are present in the set which show that sarcasm is a function of current events as well.

Bootstrapping was tried with stemming, lemmatization and trigram emphasis. The test data Recall and Precision are shown in figure 4.

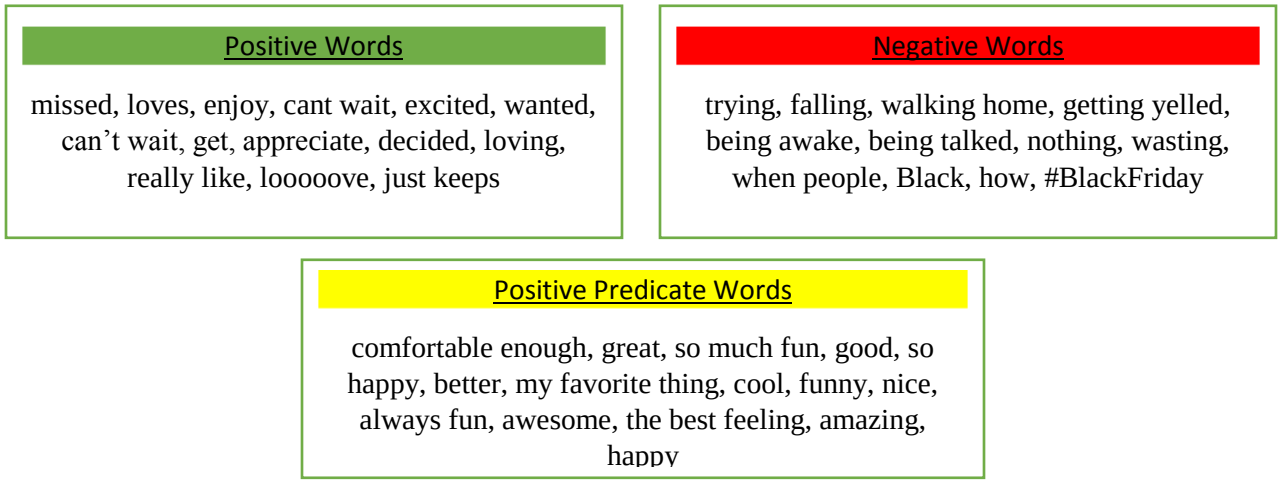

Figure 3 – Positive, negative and Positive Predicate words obtained by using love as seed word in the baseline case.

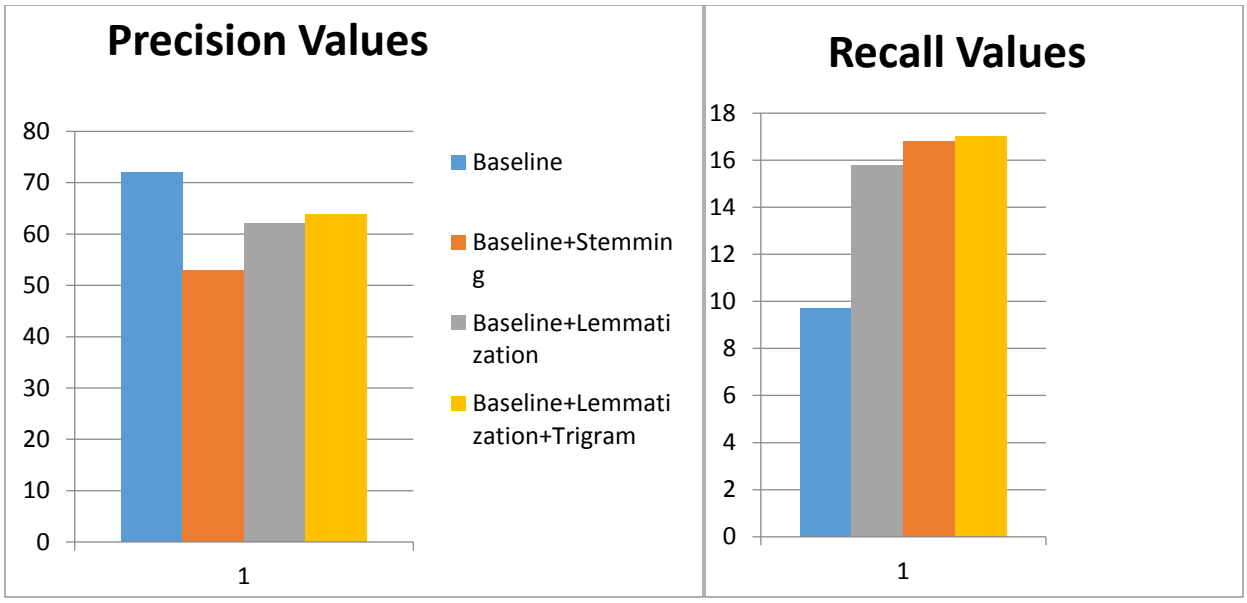

Figure 4 - The best precision is gotten by the baseline, but the recall is low. Baseline with stemming gives good recall, but reduces the precision significantly. Baseline with lemmatization and trigram leniency is therefore the best.

Support Vector Machines

As mentioned before we tried several lexical and syntactic features in addition to unigram and bigram features for SVM. The sentence length feature was the feature that contributed most significantly to improving recall and f-measure, without affecting precision. This may be attributed to the fact that sarcastic sentences on twitter generally tend to be short sentences. Syntactic features and punctuation also managed to slightly improve the recall and f-measure values. One surprising result was that stemming actually decreased the recall. This can be attributed to the small size of our training data. On larger datasets with a more diverse vocabulary, one can expect stemming to improve recall. The smaller size of data set can also be attributed to lower values of precision for the vanilla SVM we implemented in comparison to the values obtained in [2].

The following table summarizes the experiments conducted with SVM:

Features	Recall	Precision	f-Measure
Baseline SVM (Riloff et.al.)	0.39	**0.64**	0.48
Cross Validated + Unigrams + Bigrams	0.39	0.59	0.46
+Syntactic + Punctuation	0.42	0.58	0.49
+ Sentence Length	**0.49**	0.58	**0.53**

Table 1: comparison of evaluation metric with various features added in SVM

Final Results

The SVM and bootstrapping were combined in the final Hybrid Classifier. If either of these classifiers classified a tweet as sarcastic, the tweet was considered Sarcastic. Our improvements resulted in a 13% improvement in recall and 6% improvement in f measure, while the precision dropped by 5%.

We also tried augmenting our search assisted knowledge model to our improved hybrid. This unfortunately resulted in a drop in precision, while the recall was the same, since we are just using it if both the other classifiers classified a tweet as non-sarcastic. The results are summarized in Table 2 below.

Method	Recall	Precision	f-Measure
SVM + Bootstrapping (Hybrid[2])	0.44	**0.62**	0.51
Improved Hybrid	**0.57**	0.57	**0.57**
Improved Hybrid + Search assisted knowledge	0.57	0.55	0.56

Table 2: Final result comparison with [2] and with search assisted knowledge

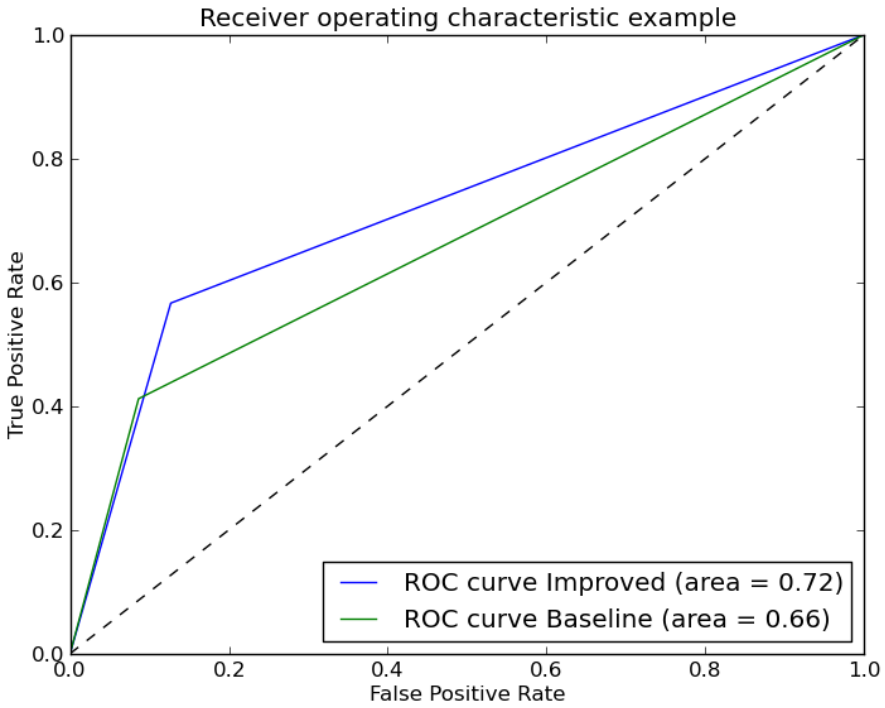

Figure 5 : ROC curve comparing our improved method to the baseline method

We also compared the performance of the baseline classifier to the improved classifier implemented by us by plotting a Receiver Operating Characteristic curve (ROC curve) with the help of sci-kit[4] . The ROC curve plots the true positives on the Y-axis and the false positives on the X-axis and represents the tradeoff between benefits and costs. Discrete classifiers generally produce a point in the ROC space, so we varied the classification threshold in order to generate the ROC curve. We can see from Figure 5 that for most threshold values our improved classifier performs better than the baseline classifier, This is also quantified in terms of the area under the ROC curve (AUC) which gives us a scalar value to represent expected performance of the classifier. The improved classifier gets an AUC value of 0.72 as compared to the baseline classifier which gets a score of 0.66.

Evaluation of Success Metric

In our proposal we hoped to first successfully implement the algorithm discussed in [2] and then add features to try and improve upon their recall, precision and f-measure scores. We managed to successfully implement the bootstrapping algorithm as well as the SVM classifier and out features managed to improve upon the baseline recall and the f-measure. We were unfortunately not able to improve precision, and the search assisted knowledge only slightly decreased the precision.

Our idea of improving the precision using search assisted knowledge also did not prove to be as effective as we hoped. This may be because some of the 4-grams we extracted, did not completely make sense and so putting it into our search assisted knowledge system resulted in false positives for negative sentiment. As a future task, we would like to add a filter step before passing it to the search engine so that we make sure only phrases that make partial sense at the very least, are passed.

Conclusion

In this project we implement the baseline algorithm discussed in [2] and then improve upon it adding several lexical and syntactic features to the SVM and the Bootstrapping steps. We also apply their suggestion of incorporating knowledge by leveraging google search and performing sentiment analysis. Although we were not able to achieve good results with Search assisted knowledge, we believe that with a filter step to check for meaning of the phrases, we could achieve better results. Another avenue of improvement would be considering other patterns for sarcasm matching, and extending positive word classes to Noun Phrases.

References

[1]: Davidov, Dmitry, Oren Tsur, and Ari Rappoport. "Semi-supervised recognition of sarcastic sentences in twitter and amazon." *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, 2010.

[2]: Riloff et al. “Sarcasm as a contrast between a positive situation and a negative Situation” . EMNLP, 2013.

[3]: González-Ibáñez, Roberto, Smaranda Muresan, and Nina Wacholder. "Identifying Sarcasm in Twitter: A Closer Look." *ACL (Short Papers)*. 2011.

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