Utilizing Maximum Entropy Classification for Sentiment Analysis in Twitter Messages

By

Douglas Anderson
0703557
dander01@uoguelph.ca

CIS*4900 - Undergraduate Research Project
August 7th, 2014

Supervisor:
Professor Fei Song

Second Reader:
Professor Xining Li

1. Introduction

Social media allows people to discuss topics and disseminate their opinions with a near negligible cost. Social media's low barrier of entry allows everyone to share their opinions as much as they please. The scale of textual data from social media is enormous. In order to gain insight into the 'overall' opinion on particular topics, the process of determining sentiment must be automated.

Performing sentiment analysis is becoming a very important field of study for many businesses. If a company can determine what consumers think of their products they can take actions such as improve their products or appease an unsatisfied customer.

Sentiment analysis is simply a classification task that attempts to correctly label fragments of text to appropriately summarizes the attitude of the text. Natural Language Processing (NLP) provides a number of method for performing sentiment analysis. Many of these methods, such as Naive Bayes and Maximum Entropy, take a probabilistic approach to performing the task. Another commonly used method of performing sentiment analysis is the uses of a support vector machine (SVM).

1.1. Specific Problem

The focus of this undergraduate research project has been to perform sentiment analysis in data from the microbloging website Twitter. All of the messages, known as Tweets, are constrained in length to 140 characters. This limit forces many users of Twitter to use abbreviations and contractions. This data presents some interesting problems to performing good sentiment analysis due to the short and informal nature of the text. Twitter makes it extremely easy to publish a Tweet and has no way to edit a Tweet once published. This results in more spelling mistakes and inconsistent grammar then other sources of text. Tweets often also contain URLs to other websites. Tweets also contain a certain amount of OOV (Out-Of-Vocabulary) words, such as Hashtags, a tagging system for topics allowing Tweets in a similar vein of conversation to be found. Other OOV words include mentions (e.g. '@BarackObama') which are a method to direct a Tweet to one or more users, as well as 'RT' which is an acronym for 'retweet' communicating that this particular Tweet is a reissuing of another users Tweet. Another issue that is out of the scope of this project is that Twitter currently supports 35+ languages, meaning that in order to determine the sentiment most systems must first determine which language the Tweet is in.

1.2. Task Description

Organizers of the SemEval-2013 workshop created various sentiment analysis tasks to be preformed. The task focused on for this project was task-2b: Message polarity classification. The organizers used the Amazon service "Mechanical Turk" to get many humans to annotate English language Tweets as 'positive', 'negative' or 'neutral'. In Tweets with mixed sentiments, the prevailing sentiment was chosen. Participants were tasked with creating a classifier to establish labels for Tweets not in the training dataset. The various submissions of the participants were then evaluated by the average of the F-positive and the F-negative (explained in depth in Section 4.3). The submissions were allowed categorized as 'unconstrained' or 'constrained' depending on if they did or did not use supplemental data, respectively.

2. Background

According to 'Opinion Mining and Sentiment Analysis' [1]: "The year 2001 or so seems to mark the beginning of widespread awareness of the research problems that sentiment analysis and opinion mining raise". This explosion in research has happened very recently, and as such the field is moving very quickly. Much of the earlier work in the field involved data from other contexts such as Movie reviews [2], and blog posts [3]. The challenges associated with classifying Twitter data has only attracted researchers in the past few years [4] [5] [6].

Sentiment analysis is a very specific type of **text classification**, a problem that is well summarized in chapter 6 of *Mining Text Data* by Aggarwal and Zhai [7] as:

> We have a set of training records $\mathcal{D} = \{X_1, ... , X_N\}$, such that each record is labeled with a class value drawn from a set of k different discrete values indexed by $\{1...k\}$. The training data is used in order to construct a classification model, which relates the features in the underlying record to one of the class labels.

In the case of sentiment analysis, the class labels being applied to the records denote the attitude or opinion in the text. The focus of this project is determining sentiment at the document level, which assumes that the document expresses a predominate sentiment. This is a somewhat reasonable assumption given the fact that Tweets are limited in size, making it difficult to express mixed sentiment. Sentiment analysis is very closely linked with opinion mining, which is the practice of extracting sentiment on a particular topic in order to gain insights into public opinion on the topic.

Much of the previous research make use of Naive Bayes, Maximum Entropy, and SVM classifiers. The Naive Bayes classifier uses the occurrences of features in the *training set* to calculate the probability that an example in the *test set* belongs in a given class. This method has the benefit of linear time training since there is no iterative process, but it makes the assumption that all features are independent, which is not the case in natural language. The Maximum Entropy approach is explained in detail in section 3.1. While Maximum Entropy and Naive Bayes are both probabilistic approaches the to problem, Support Vector Machine (SVM) classifiers take a different approach to the problem. "SVM Classifiers attempt partition the data space with the use of linear or non-linear delineations between the different classifiers."[7].

In order to perform text classification of any kind one of the most important aspects is deciding how to represent a document and how to select representative features. A document is often represented as a *bag-of-words*, which is defined a set of strings with no apparent order. The *bag-of-words* method of representation was selected for this project.

There are many established methods of selecting features such as document frequency, which is simply the number of documents that a word occurs at least once in. Document frequency has the added benefit of not requiring a class labels to select features. While document frequency is the only method employed for this project, many other methods exist, such as Gini Index, Information Gain, and χ^2-statistic [7]. Each of these more complex methods rely on the class label in the *training set* in order to determine how well the feature discriminates between classes.

3. Our Implementation

3.1. Summary of Maximum Entropy

The Maximum Entropy classification method was focused on due to the limited duration of the project, however a Naive Bayes classifier was also implemented to provide a point of comparison. This supervised machine learning method calculates the probability that a given document d belongs to the class c. The Maximum Entropy approach is summarized below (for a more detailed description, see: Nigam et al. [8]).

The training data is used to establish constraints in the following fashion:

$$P(c|d) = \frac{1}{Z(d)}\exp(\sum_i \lambda_{i,c} F_{i,c}(d,c)) \tag{1}$$

where $Z(d)$ is a normalization function, $F_{i,c}$ is a feature function for feature f_i and class c, and λ_i is a feature parameter.

In order to establish reasonable λ_i for each feature the Improved Iterative Scaling (IIS) is employed. Given a set of training data $\mathcal{D}$, IIS trains a model Λ by performing a hillclimbing procedure with the function $l(\Lambda|\mathcal{D})$, where:

$$l(\Lambda|\mathcal{D}) = \sum_{d \in \mathcal{D}} \sum_{i} \lambda_i f_i(d, c(d)) - \sum_{d \in \mathcal{D}} \log \sum_{c} \exp \sum_{i} \lambda_i f_i(d, c) \tag{2}$$

Starting with any initial vector of parameters Λ, at each iteration we improve the parameters by setting $\Lambda = \Lambda + \Delta$.

3.2. Preprocessing and Feature Selection

The classifier produced for this project is implemented in *python* and make extensive use of the *nltk* (Natural Language Toolkit) library. As well the implementation makes use of the *flex* tool to create a lexer for tokenization. The classifier does not make uses of any supplemental datasets and therefore should be considered constrained within the context of the task. The classifier adheres to the following procedure:

1. **Tokenization** - The text is read into memory and separated into tokens. Each token represents an atom of text. Token types include word, number, user, hashtag, url, emoticon and punctuation.

2. **Normalizing** - Each token is simplified to assist in classification by reducing the number of tokens.

    - **Words** are coerced into lowercase
    - **Numbers** are replaced with "#NUM"
    - **User** are replaced with "@USER"
    - **Hashtags** are replaced with "#OCTOTHORPE"
    - **URLs** are replaced with "@URL"
    - **Emoticon** are grouped into four different groups based on their intended meaning. Each group is then replaced with a representation. [e.g. ':)' → 'EM_HAPPY' and ';-p' → 'EM_WINK']
    - most **Punctuation** is replaced with "PUNCT" however question marks, punctuation marks, and ellipsis are each replaced with their own representation.

3. **Feature Selection** - The number of tokens retained as features is further reduced.

- By **Removing Stopwords**, words that do not discriminate between classes are eliminated. Because of the limited size of the dataset only 28 stopwords are used. For a full list of stopwords see section 7.2.
- Then **Uncommon words** are removed. Words that do not meet a certain document frequency threshold are eliminated. Many of these words are proper nouns that, if retained are likely to cause the classifier to overfit to the training set. A threshold of one was used throughout the testing process.

4. **Splitting** - The corpus of documents are random separated into three sets: the *training set*, the *validation set*, and the *testing set*. Respectively, each dataset is 60%, 20%, and 20% of the corpus.

5. **Training** - The classifier is trained using the *training set*. The classifier uses the *validation set* to evaluate the training process and attempt to curtail overfitting.

6. **Testing** - The performance of the dataset is evaluated against the *testing set*.

3.3. Training Optimization

The implementation created for this project used a *validation set* during training as the primary mechanism to avoid overfitting. While training, the effectiveness of the classifier was evaluated after every iteration using the accuracy or F-positive measure. During the training process the following procedure was followed:

1. The classifier was trained for 5 iterations to establish reasonable initial weights.

2. Evaluate the classifier using the validation set. Like the *testing set* that will evaluate the classifiers final performance, the *validation set* shares no common examples with the *training set* and will therefore give a reasonable indication of how the classifier will perform with unseen examples.

3. Repeat the following until performance degrades for a specified number of iterations

   (a) Train the classifier for another iteration.

   (b) Evaluate performance with the *validation set*.

4. Restore the Λ value of the best performing iteration.

4. Experiments and Discussion

4.1. Dataset

The dataset for the task consists of 6225 labelled Tweets. 36.66% of the corpus has 'positive' label, 14.04% has a 'negative' label, 49.30% has a label that conveys neutral sentiment. This neutral label is one of 'neutral', 'objective', or 'objective-OR-neutral'. The following is an example of a positive Tweet:

```
"positive", "Gas by my house hit $3.39!!!! I'm going to Chapel Hill on Sat. :)"
```

4.2. Evaluation Measures

After the SemEval-2013 workshop the organizers released a paper discussing the proceedings and the results of the task [9]. The paper establishes baseline in terms of F. F can be calculated as:

$$F = \frac{2\frac{P_{pos}R_{pos}}{P_{pos}+R_{pos}} + 2\frac{P_{neg}R_{neg}}{P_{neg}+R_{neg}}}{2} \tag{3}$$

Which can be simplified to:

$$F = \frac{P_{pos}R_{pos}}{P_{pos}+R_{pos}} + \frac{P_{neg}R_{neg}}{P_{neg}+R_{neg}} \tag{4}$$

The *Precision* measure for the positive label is $P_{pos} = \frac{\#\text{of true positive}}{\#\text{of true positive}+\#\text{of false positive}}$. The *Recall* measure for the positive label is defined as $R_{pos} = \frac{\#\text{of true positive}}{\#\text{of true positive}+\#\text{of false negative}}$. P_{neg} and R_{neg} can be calculated in a similar fashion.

Accuracy is used as a secondary performance measure. Accuracy, henceforth referred to as *Acc*, can be calculate with the following formula:

$$Acc = \frac{\#\text{of true positive} + \#\text{of true negative}}{N} \tag{5}$$

Where N is equal to the total number of documents in the *test set*. While more intuitive, it is not used as the primary evaluation measure due to it's tendency to be biased by uneven numbers of examples of particular labels in the dataset.

4.3. Improving Performance Measurement

Testing the performance of the system with a single run can produce skewed results since whether 'easy' Tweets end up in *training set* or the *test set* will affect performance. To combat this the following **exhaustive cross-validation** method is used:

1. **Partitioning** - The corpus of documents are shuffled and separated into 5 or 10 sections.

2. **Folding** - The training and testing process are run multiple times with sections used for various roles. The roles are permuted so that the classifier is trained and tested with the sections in all possible configurations. In each fold 60% of the partitions are designated as part of the *training set*, 20% as the *validation set*, and 20% as the *test set*. Since the Naive Bayes classifier does not use a *validation set*, 40% of the partitions are designated as part of the *test set*

   (a) **Training** - The classifier is trained with the partitions designated as part of the *training set* for the fold. The Maximum Entropy classifier also uses the *validation set* to determine when to stop the training process.

   (b) **Testing** - The performance of the classifier is evaluated using the *test set* and the results are recorded.

3. **Aggregating** - The results from all folds are averaged.

4.4. Baseline

The baseline for this project is established by running the Maximum Entropy classifier without any stopword removal, rare word removal (words with a document frequency < 1), Or feature normalization. The F-score that is achieved is **50.72**.

4.5. Experimental Results

The following experiments examine the results of the Maximum Entropy classifier and the Naive Bayes classifier with various feature selection methods turned on or off. The 'Baseline' experiment refers to no feature selection. The 'Only Stopword Removal' experiment refers the only removing stopwords as feature selection. The 'Only Rare Word Removal' experiment refers to only removing words with a document frequency of 1. The 'Only Normalization' experiment refers to only applying feature normalization mechanisms. The 'All Feature Selection' experiment refers to testing the classifier with all of the previous feature selection methods turned on.

4.5.1. Number of Features

The number of features retained after feature selection varies from Tweet to Tweet. To summarize the number of features retained after different processes, Table 1 presents the mean number of features per Tweet for positive, neutral, negative examples as well as an overall average.

Description	Positive	Neutral	Negative	Overall
Baseline	20.65	20.45	20.96	20.58
Only Stopword Removal	17.13	16.86	17.41	17.04
Only Rare Word Removal	18.43	17.87	18.85	18.21
Only Normalization	19.65	19.16	20.06	19.46
All Feature Selections	14.85	14.06	15.10	14.49

Table 1: 'Positive', 'Neutral', 'Negative', and 'Overall' refer to the average number of features per Tweet in each experiment.

Interestingly 'negative' Tweets contain more features then 'positive' Tweets which, in turn, contain more features than 'neutral' tweets. Normalization does not appear to significantly reduce the number of features per Tweet. This makes sense since the number of features in the *bag-of-words* would only be reduced if two or more of the features are normalized to the same thing (e.g. A Tweet that contained '2' and '3' before normalization would only contain the feature '#NUM' after normalization).

4.5.2. Maximum Entropy Classifier

Description	μF	σF	$\delta\ \mu F$	μAcc	σAcc
Baseline	48.37	2.15	-	**63.89**	1.13
Only Stopword Removal	48.27	1.83	-0.10	62.07	1.69
Only Rare Word Removal	49.02	1.96	0.65	62.47	1.91
Only Normalization	49.55	2.10	1.18	63.57	1.75
All Feature Selections	**50.96**	1.93	2.59	63.82	0.91

Table 2: Performance of the Maximum Entropy classifier using exhaustive cross-validation with 5 partitions

As Table 2 shows, the 'All Feature Selections' experiment provides the best results over the baseline. The 'Only Stopword Removal' experiment actually provides slightly worse results then the baseline indicating that in context of Tweets with so few features in each example every feature counts. The 'Only Rare Words Removal ' experiment does not provide

very significant gains when used alone. This is potentially due to the fact that without normalization turned, features such as 'just' are 'Just' are determined to be different and may be removed if each only occur once.

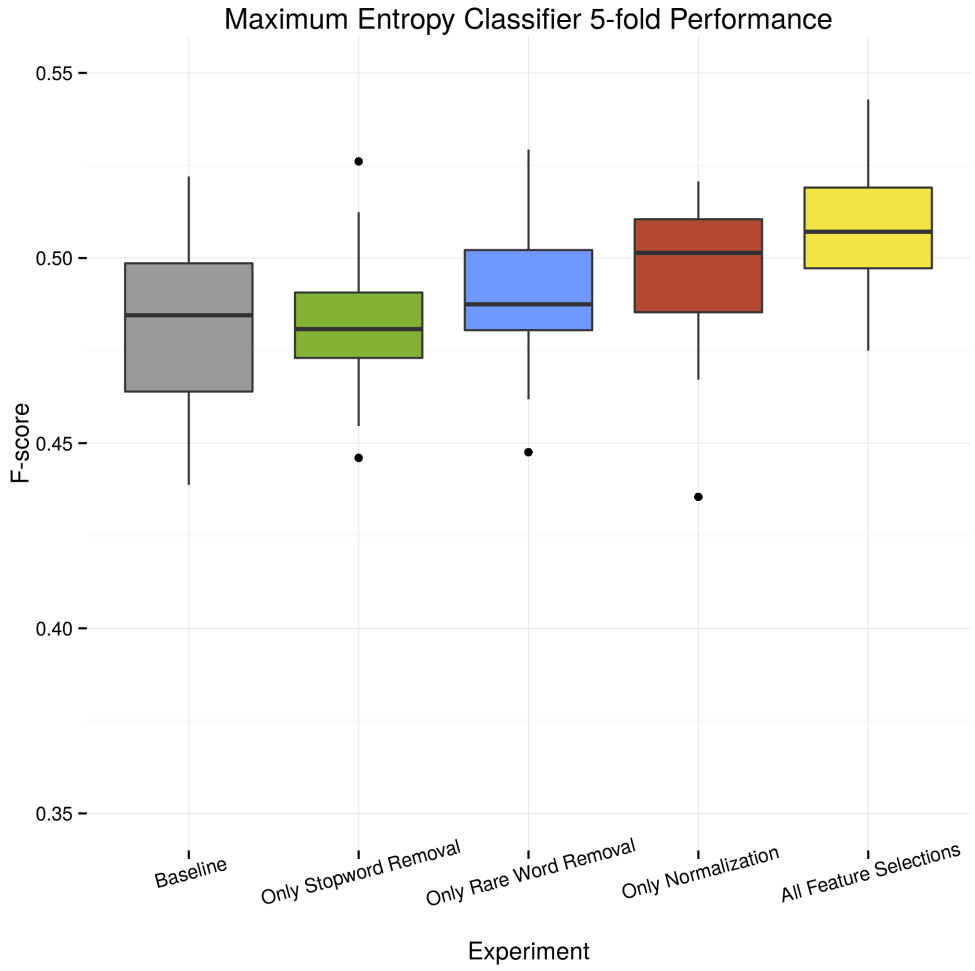

Figure 1: This boxplot shows the F-score of each of twenty folds in all five Maximum Entropy experiments

4.5.3. Naive Bayes Classifier

Description	μF	σF	$\delta\ \mu F$	μAcc	σAcc
Baseline	48.11	0.74	-	53.13	1.48
Only Stopword Removal	47.29	0.76	-0.82	52.71	1.16
Only Rare Word Removal	48.34	1.11	0.23	53.92	1.32
Only Normalization	**49.44**	1.00	1.33	**55.92**	0.99
All Feature Selections	49.28	0.81	1.17	54.91	1.05

Table 3: Performance of the Naive Bayes classifier using exhaustive cross-validation with 5 partitions

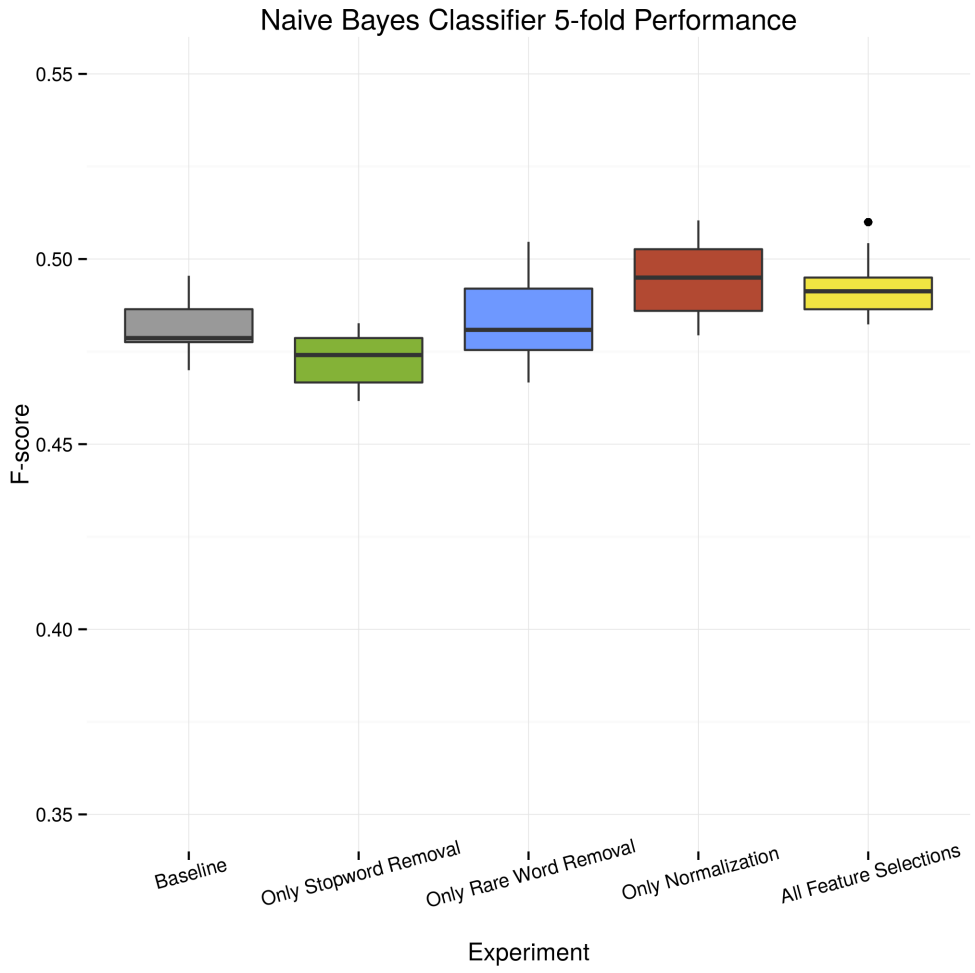

Figure 2: This boxplot shows the F-score of each of ten folds in all five Naive Bayes experiments

As Table 3 and Figure 2 indicates both removing stopwords and removing rare words appear to negatively affect performance of the naive bayes classifier.

4.6. Most Informative Features

The implementation created for this project has the ability to show which features are the most discriminatory. The following is a list of the top 30 most informative features from one run of the Maximum Entropy classifier (note that some of the weights are negative):

```
-3.176 excited==True and label is 'neutral'
 2.328 lied==True and label is 'negative'
 2.279 availability==True and label is 'negative'
-2.258 excited==True and label is 'negative'
-2.180 EM_HAPPY==True and label is 'negative'
-2.171 happy==True and label is 'neutral'
-2.098 birthday==True and label is 'negative'
 2.016 shitty==True and label is 'negative'
-1.836 8th==True and label is 'positive'
-1.801 international==True and label is 'positive'
 1.789 embarrassing==True and label is 'negative'
-1.762 went==True and label is 'negative'
 1.702 releases==True and label is 'negative'
-1.672 love==True and label is 'negative'
 1.650 lack==True and label is 'negative'
 1.650 oneil==True and label is 'neutral'
-1.638 fun==True and label is 'neutral'
 1.599 replay==True and label is 'negative'
-1.585 great==True and label is 'neutral'
 1.577 asses==True and label is 'positive'
 1.570 garlic==True and label is 'negative'
-1.564 pass==True and label is 'negative'
 1.560 den==True and label is 'negative'
 1.560 accepting==True and label is 'negative'
-1.557 sure==True and label is 'negative'
 1.551 insensitive==True and label is 'negative'
 1.547 calvin==True and label is 'negative'
 1.532 jailed==True and label is 'negative'
 1.520 lookin==True and label is 'neutral'
 1.515 2c==True and label is 'negative'
```

Some of the features listed above make perfect sense, such as 'excited' not being associated with the neutral class and 'embarrassing' being associated with the negative class.

However some of the most discriminating features appear in the dataset infrequently, such as 'calvin' and 'oneil'.

4.7. Comparison to Other Systems

Table 4 shows the performance of the classifier produced for this project compared to results of the workshop participants. The full table (with out this project's results) can be found in Nakov et al. as Table 9 [9].

Team	F
NRC-Canada	69.02
GU-MLT-LT	65.27
teragram	64.86
BOUNCE*	63.53
KLUE*	63.06
AMI&ERIC	62.55
FBM	61.17
AVAYA	60.84
SAIL*	60.14
UT-DB	59.87
FBK-irst	59.76
nlp.cs.aueb.gr	58.91
UNITOR	58.27
LVIC-LIMSI	57.14
Unigon	56.96
NILC_USP	56.31
DataMining	55.52
ECNUCS	55.05
nlp.cs.aueb.gr	54.73
ASVUniOfLeipzig	54.56
SZTE-NLP	54.33
CodeX	53.89
Oasis	53.84
NTNU	53.23
UoM	51.81
MaxEnt	**50.96**
SSA-UO	50.17
SenselyticTeam	50.10
Naive Bayes	**49.44**
9 more teams	< 49

Table 4: Comparison of constrained classifiers submitted to the SemEval-2013 workshop. Sorted by F-score. * denotes other Maximum Entropy classifiers

5. Conclusions and Future Work

5.1. Conclusions

As shown in Table 4, there is a lot of room for improvement before this system is a completive solution. However, only simple methods have been employed including: stopword removal of only 28 words, words with a document frequency lower than one removed, and normalization of words, emoticons, and OOV words. These simple preprocessing and feature selection methods in combination with a Maximum Entropy classifier have demonstrated reasonable results.

5.2. Future Work

There is room for a number of improvement in the implementation of the system. The following sections are a number of improvements as future work. Many of the ideas are from the other submissions to the SemEval-2013 workshop.

5.2.1. Hashtags

Hashtags are a way that Twitter users tag a Tweet as part of a particular conversation. Hashtags refer to proper nouns such as events, places, or people. Hashtags can also denote modifiers (such as '#sarcasm') to the sentiment of the Tweet, or even raw sentiment (e.g. '#yay').

While it is likely infeasible to retain Hashtags about proper nouns, classification performance could be improved by including modifiers and Hashtags with raw sentiment. The following Tweet (not from the dataset) is a great example of a Tweet that Hashtags could help improve classification accuracy.

```
Well I have an ear infection #good #sarcasm
```

The words in the previous Tweet carry no inherent sentiment, however the message is clearly negative. While a simple approach could be to strip the octothorpe from the front of the Hashtag (e.g. '#good' $\rightarrow$ 'good'), the semantic meaning of the two are different. Hashtags often have message level meaning while simple words have local context withing the sentence. Since the dataset created for the SemEval-2013 workshop mostly contained examples of Hashtags used as proper nouns, more annotated Tweets would be needed for training and testing.

Hashtags sometimes use CamelCase to create multi-word tags (e.g. '#SorryNotSorry'). It could prove worthwhile to attempt the decompose CamelCase Hashtags into its component words during feature selection.

The top solution from the National Research Council of Canada made use of a Hashtag lexicon that they developed between April of 2012 and December 2012 [10]. The final *NRC Hashtag Sentiment Lexicon* has more than 350,00 entries and is available at Saif Mohammad's website [1].

5.2.2. Users

Nearly all Twitter users have published more than one Tweet. A more advanced classifier could establish a particular users propensity to publishing a prevailing sentiment in all of their Tweets.

As well, some Twitter users may be frequently be referred to with a particular sentiment. For a hypothetical example, Tweets addressed to Emma Watson ('@EmWatson') are more likely to have a positive sentiment, while Tweets to Barack Obama ('@BarackObama') may contain extremely mixed sentiment.

A challenge that this method could introduce is the introduction of a temporal element since people's outlooks, and therefore sentiment's in Tweets, will change over time. As well as public perception of a individual changes, the prevailing sentiment may change as well.

5.2.3. Negation

One problem with the current implementation is that it lack support for negation. In it's current form the classifier would likely misunderstand the sequence "not perfect" since 'not' is a vaguely negative word and 'perfect' is a very positive word. It would likely be classified as positive when in fact the sentiment is quite clearly negative. A common approach to this problem is to add a suffix (such as "_NEG") to word that appear between a negation word and a section delimiting punctuation mark (such as ',', '.', ':', ';', '!', '?'). By separating negated features from their non-negated instances, features can do a better job of distinguishing classes.

5.2.4. N-gram Features

One way that many of the other *bag-of-words* solutions capture word order is by creating features consisting of sequences of words called bigrams. Adding bigram features adds a large number of features which can add complexity to training. This concept can also be extended to sequences of length n, called ngrams.

[1] http://www.saifmohammad.com/WebDocs/NRC-Hashtag-Sentiment-Lexicon-v0.1.zip

The added complexity associated with ngrams did not deter the National Research Council of Canada from using ngrams of length 1, 2, 3, and 4 as well as 'non-contiguous' ngrams which are described as 'ngrams with one token replaced by *' [10].

5.2.5. Employing Sentiment Lexicon

Most of the more complex solutions submitted to the SemEval-2013 workshop made uses of sentiment lexicons that capture a term's polarity. These lexicons, such as SentiWordNet [2], express the positive, neutral, and negative sentiment of each homonym. An example of this from SentiWordNet, is 'good' meaning 'morally admirable' having an entirely positive sentiment versus 'good' meaning 'having the normally expected amount' having an entirely neutral sentiment [3].

5.2.6. Support Vector Machine

Many of the top solutions submitted to the workshop, such as NRC-Canada made use of a Support Vector Machine for classification [10]. This indicates that when used effectively a SVM solution can out-perform a Maximum Entropy solution. The KLUE team, on the other hand, found that their Maximum Entropy solution outperform their linear SVM solution [11].

6. References

[1] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundation and Trends in Information Retrieval*, vol. 2, pp. 1–135, 2008.

[2] B. Pang, L. Lee, and S. Vaithyanahan, "Thumbs up? sentiment classification using machine learning techniques," *Proceeding of EMNLP*, 2002.

[3] P. Melville, W. Gryc, and R. D. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," *KDD '09*, 2009.

[4] J. Si, A. Mukherjee, B. Liu, Q. Li, H. Li, and X. Deng, "Exploiting topic based twitter sentiment for stock prediction," *Association for Computational Linguistics - Short Papers*, 2013.

[5] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," *Coling 2010: Poster Volume*, 2010.

[6] B. Gokulakrishnan, P. Priyanthan, T. Ragavan, N. Prasath, and A. Perera, "Opinion mining and sentiment analysis on a twitter data stream," *The International Conference on Advances in ICT for Emerging Regions*, 2012.

[7] C. Aggarwal and C. Zhai, *Mining Text Data*, ch. 6. Springer, 2012.

[8] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," 1999.

[2] http://sentiwordnet.isti.cnr.it/

[3] http://sentiwordnet.isti.cnr.it/search.php?q=good

[9] *SemEval-2013 Task 2: Sentiment Analysis in Twitter*, vol. 2, June 2013.

[10] S. M. Mohammed, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," 2013.

[11] T. Proisl, P. Greiner, S. Evert, and B. Kabashi, "Klue: Simple and robust methods for polarity classification," 2013.

7. Appendices

7.1. Running the Implementation

To acquire your own copy of the implementation discussed above. The following instructions are written for Ubuntu 14.04 GNU/Linux, however it should be able to run on any Unix-like system.

1. **Download Source** - The source code is hosted on github.com and can be viewed there [4]. To download a zip file and open it:

   ```
   wget https://github.com/hockeybuggy/twitter-sentiment/archive/dist.zip
   unzip dist.zip
   cd twitter-sentiment-dist
   ```

2. **Install Prerequisites** - If *python 2.7*, *pip*, and *flex* are not installed, they can be installed easily via a package manger:

   ```
   sudo apt-get install python2.7 pip flex
   ```

   Then various required python packages can be installed with:

   ```
   sudo pip install numpy nltk
   ```

3. **Compile Tokenizer** - The tokenizer is written in C for performance reasons and can be compiled with:

   ```
   make
   ```

4. **Run the Program** - The classifier can be executed in a number of ways. There are two python programs that are executable: 'conductor.py' and 'crossval.py'. These two programs run the classifier in a simple fashion or with cross validation. There are a large number of command line arguments that can be viewed with:

[4] https://github.com/hockeybuggy/twitter-sentiment/tree/dist

```
python conductor -h
```

Since the command line arguments can be cumbersome many of the experiments can be run via the make file:

```
make tiny            # Quick running with only 100 examples
make crossfolds-m000 # The MaxEnt baseline with cross-validation
make crossfolds-m111 # MaxEnt with cross-validation & feature selection
make crossfolds-b000 # The Naive Bayes baseline with cross-validation
make crossfolds-b111 # Naive Bayes with cross-val & feature selection
make crossfolds      # Run all of the cross-validation experiments
```

7.2. Stopword list

The list of stopwords was acquired online[5] and is as follows:

```
a                   of
about               on
an                  or
are                 that
as                  the
at                  this
be                  to
by                  was
com                 what
for                 when
from                where
how                 who
in                  will
is                  with
it
```

[5] www.ranks.nl/stopwords