SARCASM DETECTION

VARUN MUTHANNA
Graduate Student
Computer Science Department
University of Texas at Dallas
vkm150030@utdallas.edu

ABSTRACT:

Sarcasm is a form of speech where message is conveyed in the implicit way and often it is difficult for a human being to detect the sarcasm. The purpose of the project is to detect the implicit presence of sarcasm in the tweets. I have used data from the Twitter and created feature sets using approaches like 1) N-grams 2) Sentiment analysis 3) POS tagging 4) Punctuation. Support Vector Machine (SVM) will be used to classify the sarcastic tweets based on the features provided. The performance of these approaches is measured individually and in combination and effect of each approach and possible improvements are discussed.

1. INTRODUCTION:

1.1 Motivation:

Sarcasm detection is a sophisticated form of speech act and its recognition is one of the difficult tasks in Natural language processing (NLP). It can benefit many NLP applications like review summarization, dialogue system, review ranking system. Hence working on this project will provide a great insight on the NLP techniques and the challenges that lies ahead.

1.2 Challenges:

Sarcasm, according to Merriam-Webster means, *use of words that mean opposite of the literal meaning intended to insult someone, to show irritation, or to be funny.* So, detecting sarcasm in the written form can be difficult due to the lack of intonations and absence of facial expression and the need to find that it was opposite of its literal meaning. Sarcasm can be used in almost all the topics and they can take variable grammatical structures. To understand sarcasm one must know the context of the sentence. For instance, “I am having fun!!” is not a sarcasm unless we know that the person is going through rough time. Therefore, for many of the instance we would need prior knowledge. So here we will try few techniques to predict the sarcasm with the existing knowledge and discuss the impact of each technique.

2. DATA:

2.1 Data Gathering:

Twitter is an online social networking service that allows its millions of users to post and read messages(tweets) that are at most 140 characters. And it is a common trend in the Twitter than one would add hashtags (#) in the tweet is convey the purpose or implicit meaning of the tweet. Thus, providing a huge advantage for the developers of NLP applications to get the required data which are labeled. Twitter provides API using which one can search for certain words or patterns. There are two main twitter APIs for retrieving data – Streaming API to retrieve live tweets which are being posted, REST API to retrieve tweets which are already been posted.

I have used the REST API to gather all the recent English tweets with keywords:
1) ‘#sarcasm’ - This will retrieve tweets which has implicit sarcasm in it.
2) ‘#fact’ - This will retrieve tweets which are considered facts and hence can be taken as non-sarcastic data

2.2 Data preprocessing:

One of the drawbacks of using tweets as the dataset is that it can be noisy as it can contain images, http links, and duplicates. So, during the preprocessing all the tweets which contain images, http link and the tweets starting with RT meaning it is a retweet (duplicate) are discarded. Also, all the hashtags, friend tags and mention of sarcasm and sarcastic are removed to give the classifier a cleaner dataset to classify. Emoticons are normal use in the tweets and it would be difficult for a machine to detect its implicit meaning, so I have replaced the most commonly used emoticons with either ‘happy’ or ‘sad’ based on what it signifies. After all these removal and replacement, if the string contains more than 3 words, then it is added to the dataset.

3. Feature Extraction:

Feature extraction is one of the most important part of sarcasm detection. Four different approaches for feature extraction were used to represent the tweet as a feature vector to train the classifier for sarcasm detection.

3.1 N-grams feature:

N-grams refers to sequence of token within a statement where N refers to number of words in the sequence. Here I have used unigrams (N=1) and bigrams (N=2). To use N-grams in the classifier the tweet is tokenized, lemmatized, uncapitalized and stored as binary feature. 1 if N-gram is present else 0. NLTK library was used to generate the bigrams.

3.2 Sentiment and Subjective feature:

One of the common structure in sarcastic statement is the presence of contrast in sentiments. In the sense that it may have positive sentiment in a negative activity or other way around. Here we will try to create a feature set which can harness this key structural feature. I have used three variations to get feature set based on the sentiments using a python library TextBlob which has a built-in sentiment score function.

1) The sentiment score of the whole tweet is taken as a feature.
2) The tweet is split into two and sentiment score of each half and the difference between two(contrast) is used as a feature.
3) The tweet is divided into three and sentiment score of each part and their contrast are taken into consideration.

TextBlob also provides the subjectivity score. So, subjectivity scores of the tweets are also taken as a feature.

3.2 POS-Tags feature:

The parts of speech of the tweet are counted and inserted into the features. Here I have used 4 tags – Noun, Verb, Adjective and Adverb. So, four different features are created using the POS-Tags.

3.3 Punctuation and special symbols feature:

It is considered that punctuation has lot of influence in the text classification especially in the area of sentiment analysis. I have considered three ways for feature extraction from punctuation and special symbols.
1) Capitals – If the number of capitals is greater than 4 than we make this feature 1 else 0.
2) Exclamation count – The number of exclamations in the statement is taken as another feature.
3) Emoticons count – The number of emoji used in the tweet is considered.

4. Classification Algorithm and Evaluation:

4.1 Support Vector Machine(SVM) Linear kernel:

SVM is a supervised learning algorithm that can be used both for classification and regression. Here we will use it for classification. SVM is known for its simplicity and effectiveness in binary classification. Also, the linear kernel will not consume as much time and resources on large data compared to polynomial kernel. Sklearn library was used in training. The best performance was seen when C (cost of classification or penalty parameter of error) value was given as 0.1.

4.2 Evaluation:

To evaluate the performance of the algorithm for the given feature set the metrics used are Precision, Recall, F-score and Accuracy. In the next section, we can see the values of the above metrics for each of the different feature sets and the combination.

4.3 Scoring:

For a new statement to be detected as a sarcasm or not, a score is deduced in terms of percentage. This was done using the margin (distance from the boundary) of the new statement which is calculated from the classifier and it is used in a sigmoid function to obtain value between 0 and 1. This value is scaled to a value between -1 and +1 and multiplied by 100 to obtain the sarcasm score in terms of percentage. So, if the score obtained is positive it is considered as sarcastic and higher the score it is more sarcastic.

5. Results and Discussion:

5.1 Validation Results:

Using the above-mentioned methods, the data from the twitter was gathered and preprocessed and a total of 1770 sarcastic tweets and 2370 non-sarcastic tweets were obtained. Features were extracted from this data to form the feature set. This was shuffled up and divided into 70-30 ratio (training - test) and 70% of the data was fed to the SVM classifier for training. The other 30% of the data was used as a test set and the performance of each of the features was calculated. Table shows the performance measure of the test data.

Method	Precision	Recall	F-Score	Accuracy
Sentiment	57%	59%	52%	59%
N-grams	72%	73%	73%	73%
N-grams + sentiment	75%	74%	74%	75%
N-grams + sentiment + POS tags	77%	77%	77%	77%
N-grams + sentiment + POS tags + Punctuation	76%	76%	76%	76%

Table 1 Performance Score in percentage

5.2 Validation Result Discussion:

5.2.1 N-grams:

Unigrams and bigrams provide a major chunk of accuracy. The accuracy can be improved if we could use larger dataset and by including higher N value features.

5.2.1 Sentiments:

Sentiment features has a relatively low performance mostly due to inconsistency of the tweets. It is expected that in many case user would have some hidden background knowledge which is not seen by the classifier and hence many might be classify erroneously.

5.2.2 POS-Tags:

Using POS-Tags count has provided some improvement in the accuracy and hence can be considered as an important feature. Using all the tags or by considering the POS N-grams as the feature could improve the performance.

5.2.3 Punctuation:

Surprisingly, the Punctuation and special characters' feature gives out bad performance. One reason could be these were incorporated in the unigram feature set and it could be partially duplicated. Other reason could be we have might have some wrong usage of these symbols and since the dataset is small it is showing up some effect in the output. Having a larger dataset might alleviate this issue.

5.3 Example Results:

I have randomly chosen few sentences not from my data set and below are the results obtained:
1) sentence = "Messi is the best footballer in the world"
   score = -70

2) sentence = "Oh how I love being ignored"
score = 67
3) sentence = "Absolutely adore it when my bus is late"
score = 48
4) sentence = "I never miss the lecture of Dan Moldovan"
score = -6
5) sentence = "Donald trump will make America great again"
score = 18

6. Future Improvements:

In the above section I have mentioned few improvements that can be incorporated in the feature set that I have used. Apart from these we can include topic based feature set. Other major improvement that can be done is in data processing. Spell checks, word sense disambiguation, slang detection can make the data more cleaner and can help in better classification. Also, the ratio of the sarcastic to non-sarcastic data is quiet high which is not the case in the real word hence we need to gather more data with lower ratio to get the real performance measure of our system.

7. Conclusion:

Sarcasm is a complex phenomenon. In this project, we could notice how a social networking service like twitter can act as huge asset in terms of data gathering and how a few basic features like N-grams, sentiment contrast, POS-Tags and punctuations can be powerful in the detection (accuracy of 78%) of sophisticated language form like that of sarcasm. Data preprocessing and feature engineering would be one of the most important tasks in terms of improving accuracy and more in-depth analysis in these domains will help in improving the accuracy considerably.

Reference:

[1] Cliche, M. The sarcasm detector, 2014. URL http://www.thesarcasmdetector.com/about/
[2] Piyoros Tungthamthiti, Kiyoaki Shirai, Masnizah Mohd. Recognition of Sarcasm in Tweets Based on Concept Level Sentiment Analysis and Supervised Learning Approaches
[3] Dmitry Davidov, Oren Tsur and Ari Rappoport. Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon
[4] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert and Ruihong Huang, Sarcasm as Contrast between a Positive Sentiment and Negative Situation
[5] scikit-learn. Support vector machines, URL http://scikit-learn.org/stable/ modules/svm.html