Automated Sarcasm Detection in Twitter tweets

CS291: Exploratory Project

Prayas Jain : 14075041
Pranav Goel : 14075039
Computer Science and Engineering
BTech Part II
IIT (BHU), Varanasi

Guide: Dr. A.K Singh

Area of focus: NLP, Machine Learning

1 Introduction

Sarcasm is defined as a cutting, often ironic remark intended to express contempt or ridicule. Automated sarcasm detection is the task of predicting a text as sarcastic or non-sarcastic by a machine, without human intervention. Recognizing sarcasm is important for natural language processing to avoid misinterpreting sarcastic statements as literal.

For example a tweet - " Yay, I am on call for work this entire weekend! So thrilled!" may be misleading with so much positive sentiment, but the actual sentiment is negative, cleverly implied using sarcasm. Hence, reliable sarcasm detection is imperative for better sentimental analysis and opinion mining. It may also contribute to better automated feedback systems in the context of e-commerce and other consumer based sites. With the usage of sarcasm highly popular across all forms of social media, this relatively new area of research rises in importance.

Twitter is a micro blogging platform used extensively by all kinds of people to express thoughts, opinions and convey information in short texts called tweets, which can also be labeled under various topics using hash tags. Since sarcasm can be difficult to interpret even by a human, we take the tweets marked by the author as #sarcasm as our sarcastic tweet corpus. This is the gold standard for sarcastically labeled tweets in modern research. Our work encompasses statistical and rule based approaches using unigrams and pragmatic features (like emoticons, punctuation, etc). Additionally, it also uses the linguistic theory of Context Incongruity. This theory uses the fact that a common form of sarcasm on Twitter consists of a positive sentiment contrasted with a negative situation. We use 'explicit' incongruity as yet another category of our features. The major motivation of doing this extra effort on top of the relatively simple, well defined and extensively used statistical models is that sarcasm detection research can push the frontiers by taking help of well-studied linguistic theories.

Rest of the paper is organized as follows. Section 2 expands on our corpus extraction (of the Twitter tweets) and the preprocessing or cleanup steps involved to improve upon the dataset. Section 3 explains all the 3 categories of our features in detail. We describe our experimental setup along with some technical details of our machine learning architecture in section 4. The quantitative evaluation of our system is presented in section 5. Section 6 presents some error analysis along with some ideas for possible future improvement. Section 7 concludes our work.

2 Corpus Extraction and Preprocessing

Sarcasm detection is in many ways a major enhancement over basic sentiment analysis, and to enforce reliability, tweets are taken to be sarcastic only when explicitly marked so by the author. We need to build 2 datasets one containing sarcastic tweets and other containing non sarcastic ones. We download tweets with hashtags #sarcasm and #sarcastic as sarcastic tweets, using the Twitter streaming API (https://dev.twitter.com/). We stream all the tweets with the mentioned tags, and set the language parameter to 'en' to gain only the tweets written in English. A variety of hashtags, with a preference for plain emotions (as sarcastic text tends to contain both polarities of emotions, and a happy or sad marked tweet is likely to contain only one kind of polarity, indicating absence of sarcasm), such as : #happy, #sad, #excited, #amazing, #education, etc are used to stream for the tweets that form our non- sarcastic dataset.

Tweets downloaded in such a way tend to contain a lot of meta-data alongside the actual text- like author information, temporal and location information,etc which are not relevant to this particular work. Hence only the text portion is first extracted. This preprocessing is done on the JSON file initially produced via twitter API streaming. The texts are moved to a text file for further processing.

Tweets in twitter can be 're-tweeted', that is, an user can share the tweet of some other user. This is indicated by 'RT' at the beginning of the text, and we use it to remove these 're-tweets', along with other duplicate tweets. Also, any name following RT is replaced by 'NAME' and all hyper-links within the text are replaced by 'HYPERLINK'.

Additionally, tweets containing less than 3 words are removed, as there are adjudged to be of insufficient length for containing any detectable incongruity. Then these unique set of tweets are also stripped off of the '#' character, so that tags like happy, sad, etc (tags apart from #sarcasm or #sarcastic) can be used as words. These may sometimes contain useful information.

The above steps are applied to the sarcastic and non-sarcastic data files separately. Further work is now down on this clean and optimized corpus.

Senti-Strength tool is applied on the cleaned up tweets. It is a tool which performs sentimental analysis on our data to produce another file with the results of its analysis. This tool is optimized for twitter tweets, covering a lot of popular 'lingo' used by the casual twitter user. Some basic preprocessing allows us to use this produced file for generation of features pertaining to explicit incongruity [explained further in section 3.3].

3 Feature Design

Features are of central importance and decide the quality of any classification task done via machine learning. Carefully designed and chosen features go a long way in determining the quantitative and qualitative results. Since, sarcasm detection is a relatively complex (linguistically) task in the domain of NLP, we argue (based on modern research directions, our knowledge and our intuition) that only rule based grammatical features are not sufficient.

Our features can be divided into 3 broad categories
1. Lexical
2. Pragmatic
3. Explicit Incongruity (the linguistic theory we are using).

The individual features of these categories are described in detail in the next few sections.

3.1 Lexical Features

These features are designed to use the lexical information that the tweets contain. We use unigrams for this purpose. Certain words may be more particular of the sarcastic tweets, thus becoming a potentially important indicator.

We create a dictionary from our training corpus, with each unique word mapped onto a particular ID. These ID numbers are used as feature numbers, and their values are the number of times that particular word occurs in the particular tweet we are creating features of. Note that this will be a highly sparse matrix, as most of the IDs would have the value 0 (since vocabulary will be large, and the tweet would be containing only a very few words out of it). We do not need to keep the IDs with zero value in the file containing feature values.

3.2 Pragmatic Features

These are associated with the grammatical hints, which can always play a role in any kind of language analysis. In our case, this category contains the following features-

3.2.1 Number of capital letters in the tweet

This obviously does not include the letters in 'NAME' or 'HYPERLINK' that we formed in the preprocessing stage. This can sometimes we useful, as sarcastic portion can be highlighted by the author using capitalization for extra impact.

3.2.2 Number of emoticons in the tweet

These can be captured using UTF-8 encoding. Python is capable of exploiting these encodings (through the module 'codecs'), by opening and reading files such that the emoticons are present as utf-8 encoding, which can be captured easily by regular expressions ('regex'). These emoticons are again important indicators of a user's sentiment.

3.2.3 Number of laughter expressions

These include popular expressions like 'lol' (we capture modified versions like lol with multiple possible capitalizations, lol with multiple 'o' in between, and even variations like 'lololol', which we identified through extensive observation of our training corpus). Other expressions include 'rofl', and 'lmao', which we again found to be fairly well used in the Twitter realm. The total count of all these expressions constitute one feature in the pragmatic category. Since sarcasm is intended to be funny many times, higher value of this feature is a potential indicator of sarcasm.

3.2.4 Number of punctuation marks

Exclamation marks are useful to convey dismay, or add effect to the underlying emotion, and are sometimes exploited within sarcastic tweets. Specifically, we targeted '!', '. . . ', and '?' punctuation marks for this feature, and the value becomes the total count of these within the tweet.

3.3 Explicit Incongruity

An explicit incongruity giving rise to sarcasm bears resemblance to 'thwarted expectations' (another commonly known challenge to sentiment analysis). Consider this example: 'I love the color. The features are interesting. But a bad battery life ruins it'. The positive expectation in the first two sentences is thwarted by the last sentence. A similar incongruity is observed in the sarcastic tweet 'My tooth hurts! Yay!'. The negative word 'hurts' is incongruous with the positive 'Yay!'. So basically, explicit incongruity is an indication of contrasting emotions within the tweets, a hallmark of sarcasm.

This is a critical feature for our system, as we believe that linguistic concepts are best placed to encounter challenging natural language tasks like sarcasm detection.

The particular features used within this category -

3.3.1 Number of sentiment incongruities

This category corresponds to a single numeric feature which denotes the number of times a positive word is followed by a negative word, and vice versa.

3.3.2 Largest positive/negative subsequence

The length of the longest series of contiguous positive/negative words is included as a single numeric feature for our analysis.

3.3.3 Number of words with positive and negative polarity

This category includes two features that are generated with the help of senti-strength tool. Senti-strength assigned a number between [-5,5] for each word of tweet. Words with value greater than 0 are of positive sentiment and vice-versa. Count of both positive and negative words are included as two features.

3.3.4 Lexical Polarity

The overall sentiment, or polarity, or the positive/negative score of the tweet or text. An important observation is that a tweet that is strongly positive on the surface is more likely to be sarcastic than a tweet that seems to be negative. This is because sarcasm, by definition, tends to be caustic/hurtful. This also helps against humble bragging. (as in case of the tweet 'so i have to be up at 5am to autograph 7,000 pics of myself? Sounds like just about the worst Wednesday morning I could ever imagine').

Using the SentiStrength tool, we get the sentiment rating (-1 to -5 indicating negative, 0 for neutral, +1 to +5 for positive) for each word as well as for the overall tweet. Thus, once we get the output file from this tool, some simple preprocessing and regex allowed us to generate all the above mentioned features.

Lexical	
Unigrams	Unigrams in Training Corpus
Pragmatic	
Capitalization	Numeric feature indicating presence of capital letters
Emoticons and laughter expressions	Numeric feature indicating presence of emoticons and lol's
Punctuation Marks	Numeric feature indicating presence of punctuation marks
Explicit Incongruity	
#Explicit Incongruity	Number of times a word is followed by a word of opposite polarity
Largest positive/negative Subsequence	Length of largest series of words with polarities unchanged
#Positive Words	Number of positive words
#Negative Words	Number of negative words
Lexical Polarity	Polarity of tweets based on words present

Table 1: Feature Set

4 Experimental Setup and Architecture

After applying all the cleanup steps, we have a total of 3753 annotated tweets, out of which 2353 are sarcastic and 1400 are non sarcastic. The corresponding feature vector for each tweet is created using the 6 explicit incongruity features, 4 pragmatic features and n unigrams where is n is the length of the tweet. We train our system using LibSVM with RBF kernel (Chang and Lin, 2011).The radial basis function kernel is generally the best for language analysis purposes, while LibSVM is a highly convenient tool for all kinds of SVM classification.

To apply LibSVM, we need to have our data (label and feature values) in a defined format. The label for that particular tweet should be at the very beginning followed by $\langle n : m \rangle$ pairs where n is the feature number and m is its value.

In our system, we have the following features (the feature number in brackets) - positive word count (1), negative word count (2), overall polarity(3), no.of times polarity changes (4), largest occurrence of consecutive positive words (5), largest occurrence of consecutive negative words (6), number of capital letters (7), number of punctuation (8), laughter expressions (9) , number of emojis (10), and the unigrams (from ID 11 onwards).

The size of our vocabulary is 10017 words. Hence, for one tweet, the structure of our data for LibSVM may be as follows -

1 1:2 2:1 3:2 4:0 5:0 6:0 7:5 8:1 9:1 10:0 17:1 19:2 1034:1 5555:1

Features	**P**	**R**	**F**
Lexical	0.6657	0.8861	0.7602
Lexical + Pragmatic	0.6772	0.8864	0.7708
Lexical + Explicit	0.7112	0.9293	0.8067
All features	**0.7216**	**0.9345**	**0.8144**

Table 2: Results for various feature combinations

Sarcastic tweets are labeled as 1, while non sarcastic are labeled 0.

Now LibSVM with RBF kernel is applied, and the results are reported in the next section.

5 Evaluation

The precision, recall and F-score for our dataset for different combinations of feature categories are shown in Table 2. Clearly, all categories of features contribute incrementally to the system. The highlight of our system is the final F-score of 0.8144 when all the features are applied.

Application of explicit incongruity feature set increases our score by roughly 5%, which validates our claim that well defined linguistic theories are actually promising, and could result in further improvements in the future.

Our lexical features (unigrams) give us a respectable baseline to work with in the first place. We also perform classification using Linear Kernel instead of RBF (not depicted in the table), but RBF is definitely better for our purpose.

Our work matches very well with the very best works out there. The first paper to incorporate incongruity for Sarcasm detection by Riloff et al(2013), reported F-scores of up to 75%. Our value of 81.44% is also 15% higher than the work done by Maynard and Greenwood (2014) that uses a hashtag retokenizer and rule- based algorithm.

'Harnessing Context Incongruity for Sarcasm Detection' by Joshi et al(2015) is the main inspiration behind our work, and our system is loosely based on theirs. They manage to achieve 85% F-score, although we believe some more subtle applications of the incongruity theory can help us reach their scores in near future. The most significant point of difference lies in our dataset, with their work using twice the amount of sarcastic data that we use. We believe this impacts our results in a very significant manner.

6 Error Analysis

Some possible sources of error, and potential areas for improvement with respect to our system are discussed in the following subsections.

6.1 Subjective polarity

The tweet 'Yay for 3 hour Chem labs' is tagged by the author as sarcastic, which may not be common perception.

6.2 Lack of context / Incongruity 'outside' the text

This system fails to capture incongruity which is not present inside the specific tweet. For example, the tweet may be in reply to another tweet, and only on having the knowledge of both the tweets can we have enough context to make an accurate prediction. This problem of lack of context is quite general, and since sarcasm is one of the most subtle forms of expression, gaining more and more outside information could be extremely important. One possible solution is to take into account the 'history' of the user, or all the tweets he has made on that topic in the past. Another idea is to study the tweet and also the tweet to which it is 'in-reply-to' .

6.3 Incongruity due to numbers

Our system could not detect incongruity arising due to numbers as in 'Going in to work for 2 hours was totally worth the 35 minute drive.'

6.4 Lack of data

Our sarcastic data should ideally be at least two times our current total. More training data can be a big help, especially since learning sarcasm is not a straightforward task.

7 Conclusion

Our system uses the linguistic relationship between context incongruity and sarcasm as a basis for sarcasm detection. Our sarcasm classifier uses three kinds of features: lexical, pragmatic, and explicit incongruity features. We evaluate our system on 3753 tweets streamed from Twitter API, using SVM classification with RBF kernel, and produce a decent F-score of 81.44, with visibly high recall.

The most important future direction is to incorporate more and more context, possibly using historical tweets or tweets which are in reply to our target tweet, for better predictions. Another important idea is to extend the application of the linguistic theory we have used, and explore its various other forms, like implicit incongruity and inter-sentential incongruity. Detection of subjective sentiment and figuring out the incorporation of number-based incongruity as pointed out in our error analysis are some other possible lines, and thus, this field has great scope, with significant amount of research and extension expected in coming years.