Sarcasm Detection – A Pipe Dream

Nandini S, Mihier RoyChaudhury, Mohammed Jawhar
Computer Science and Engineering, ***PESIT Bangalore South Campus.***

Abstract

Detection of sarcasm in a written form is a challenge due to the lack of facial expressions and the ability to stress on words in a sentence that indicate sarcasm. However, in a conversation, a person can detect sarcasm by relying on the speaker's facial expressions, and by the speaker stressing on certain words to convey that he/she is being sarcastic. Using Natural Language processing, we can help the machine understand us. However, figures of speech, verbal devices, phonetics, and other such forms of language devices which are used in day-to-day conversations are hard for machines to understand. In this paper, we pose a solution to the problem of Sarcasm Detection by machines, using a pipeline process that allows us to identify if a sentence is sarcastic or not. In the pipeline, we pass the sentence through three stages of testing, allowing us to classify these sentences as sarcastic or not sarcastic at the end of the pipeline.

Introduction

Sarcasm comes from the Greek word sarx(flesh) and azein(tear), an apt name for the use of irony to convey contempt.

Merriam-Webster defines sarcasm as the use of words that mean the opposite of what you really want to say especially in order to insult someone, to show irritation, or to be funny. The ability to identify sarcasm in a text can improve many of the NLP systems like opinion mining, text summarization and sentiment analysis.

Sarcasm detection is becoming one of the hot topics in research and analysis. It is one of the toughest problems in NLP due to its highly contextual nature. Sarcasm can be contextual i.e., it can only be perceived by the targeted audience.

For example, "What a fine musician you turned out to be!". The sentence can be perceived as sarcastic by the audience only if they know that the he(musician) was a terrible musician. Otherwise the sentences is just another statement.

Sometimes sarcasm is general, at this time sarcasm doesn't really have to be contextual.

For example, "I love being the second choice."

"Just love it when people don't text me back."

Since nobody actually like being the second choice, this sentence is sarcastic and since no context is given it is also not contextual. Similarly, no one likes it when people don't text them back making the sentence sarcastic and non-contextual.

In this proposed work we first cleaned the data that was obtained from various sources and it is passed to a sentiment analyzer the data is segregated as sarcastic and non-sarcastic. Since the sentiment-analyzer doesn't give any false-positives in the first process. Then the negative data set is passed into a text categorizer where feature extraction is done and the data is again segregated as sarcastic and non-sarcastic based on the important features. The negative data set obtained from this process is undergoes topic based classification where LDA is used to detect the sarcastic sentences from the corpus.

This process increases the accuracy of detecting sarcastic sentences since it undergoes different types of regression methods before getting the final output.

Related Works

Riloff et al. (2013) used a bootstrap-algorithm is presented that automatically learns phrases with negative situations, or events, combined with positive sentiments, from twitter data with the hashtag sarcasm. This algorithm relies on the assumption that negative situations often appear after positive situations in sarcastic texts. In addition, the negative phrase has to be in the vicinity of the positive phrase. Thus, the structure of the sentence is as follows:

[positive verb phrase] [negative situation phrase]

Our approach agrees with this assumption, implying importance of sentiment analysis in sarcasm detection.

Davidov et al. (2010) used sarcastic tweets and sarcastic Amazon product reviews to train a sarcasm classifier with syntactic and pattern-based features. They examined whether tweets with a sarcasm hashtag are reliable enough indicators of sarcasm to be used as a gold standard for evaluation, but found that sarcasm hashtags are noisy and possibly biased towards the hardest form of sarcasm (where even humans have difficulty).

Gonza´lez-Iba´n˜ez et al. (2011) explored the usefulness of lexical and pragmatic features for sarcasm detection in tweets. They used sarcasm hashtags as gold labels. They found positive and negative emotions in tweets, determined through fixed word dictionaries, to have a strong correlation with sarcasm.

Liebrecht et al. (2013) explored N-gram features from 1 to 3-grams to build a classifier to recognize sarcasm in Dutch tweets. They made an interesting observation from their most effective N-gram features that people tend to be more sarcastic towards specific topics such as school, homework, weather, returning from vacation, public transport, the church, the dentist, etc. This observation has some overlap with our observation that stereotypically negative situations often occur in sarcasm.

Mathieu Cliché from thesarcasmdetector.com explored sarcasm detection by taking in a set of features such as n-grams, topics, parts of speech, and capitalization. Using these features he tested the probability of successfully identifying a sentence as sarcastic or non-sarcastic by giving it a score. He displayed results of using a Naïve Bayes classifier, and of using a Support Vector Machine.

Dataset

The data was collected over a span of several months in 2017. The cleaning processed included removing all the hashtags, non-ASCII characters, and http links. In addition, each sentence from the dataset is tokenized, stemmed, and uncapitalized through the use of the Python NLTK library. The features that we assume to be important can be broadly stated as : sentiments, n-grams, topics, parts-of -speech.

The dataset consists of a sarcastic(positive) corpus as well as a non-sarcastic(negative) corpus. Using these together, we are able to train and test our classifiers.

Process – An Overview

The model we attempt to implement is a pipeline that consists of 3 stages. The first stage is the Sentiment based classification stage. The second stage is a text categorization stage and the last is the topic based stage where we use Linear Dirichlet Allocation. The sentences that are classified as sarcastic at dropped at every stage and the remainder of data is sent to the next stage(s). The following diagram illustrates the structure of the pipeline.

Stage 1: Sentiment Analysis

In this stage, we use a simple logic that many sarcastic sentences begin with one sentiment and end with the opposite sentiment. For example, consider the sentence – 'I love being poor.'. The sentence starts off with a positive sentiment and ends with a negative sentiment. So, we use this property of contrasting sentiments to classify sentences as sarcastic.

We use Vader for sentiment analysis. VADER belongs to a type of sentiment analysis that is based on lexicons of sentiment-related words. In this approach, each of the words in the lexicon is rated as to whether it is positive or negative, and in many cases, **how** positive or negative. when VADER analyses a piece of text it checks to see if any of the words in the text are present in the lexicon. For example, the sentence "The food is good and the atmosphere is nice" has two words in the lexicon (good and nice) with ratings of 1.9 and 1.8 respectively.

VADER produces four sentiment metrics from these word ratings. The first three, positive, neutral and negative, represent the proportion of the text that falls into those categories. The final metric, the compound score, is the sum of all of the lexicon ratings (1.9 and 1.8 in this case) which have been standardized to range between -1 and 1. In this case, our example sentence has a rating of 0.69, which is pretty strongly positive.

In our case, for the detection of sarcasm, we use VADER sentiment analysis on two halves of the sentence and compare their sentiments. If they are contrasting, we say that the sentence is sarcastic.

Stage 2: Text Categorization

Text categorization is an area that overlaps both with Natural Language Processing (NLP) and Machine learning. The task is to automatically sort documents under some predefined categories. Text in itself is not amenable for classification, and the first step in text categorization is to process and extract features from the text.

In this stage, the features extracted from the sentences in this stage include : n-grams, POS tags, and sentiments of each sentence.

We selected these features because we assume that these are the features that will provide us the most amount of information on whether the sentence is sarcastic or not.

N-grams: Individual tokens (i.e. unigrams) and bigrams are placed into a binary feature dictionary. Bigrams are extracted using the same library and are defined as pairs of words that typically go together. Examples include artificial intelligence, peanut butter, etc.

Parts of Speech: The parts of speech tags for each word in the sentence are generated and counted. We hypothesize that the structure of sarcastic sentences is similar for all sarcastic sentences. Thus, by obtaining the POS structure of the sentence, we are able to compare the structures of various sentences.

Sentiments: The sentiment of the entire sentence and the subjectivity of the sentence are calculated using the sentiment analyzer provided by TextBlob. We hypothesize that each of these values is similar in all sarcastic sentences.

Once the features are extracted, the important features are selected through a *Chi-Squared* test.

A *Chi-Squared* test, is any statistical hypothesis test wherein the sampling distribution of the test statistic is a Chi-Squared distribution when the null hypothesis is true.

In statistics, the Chi-Squared test is applied to test the independence of two events, where two events A and B are defined to be independent if **P(AB) = P(A)P(B)** or equivalently **P(A|B) = P(A) and P(B|A) = P(B).**

In feature selection, the two events are occurrence of the term and occurrence of the class. We then rank terms with respect to the following quantity:

$$X^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}}$$

Thus, after Chi-Squared test, we have a set of features which are more important than the others. These selected features are then fed to the classifier and the results are evaluated.

Stage 3: Topic Based Classification

(*David M.; Ng, Andrew Y.; Jordan, Michael I (January 2003). Lafferty, John, ed. "Latent Dirichlet Allocation")*

In natural language processing, latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.

For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. In LDA, each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. This is identical to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a sparse Dirichlet prior. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently.

In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics.

In this stage of the pipeline, we perform LDA on the sarcastic part of our corpus to generate a list of topics. We infer that, in a sarcastic sentence, a sentence that contains words related to the top *n* topics from the list are more likely to be sarcastic than other sentences.
We perform LDA for the non-sarcastic(negative) part of our corpus to find the topics that are more likely to be non-sarcastic. Thus, we have a complete list of topics which are passed to the machine to help it determine whether a sentence is sarcastic or non-sarcastic based on the topic of that particular sentence.

Classification

When building a model for text classification, we must take into consideration he amount of data we have. One of the most challenging parts of classification is gathering enough training data. Thus, depending on the amount of data we have, we choose the correct classifier. We have decided upon using a *Gaussian Naïve Bayes* classifier and a *Support Vector Machine.*

1. Naïve Bayes Classifier

The Naive Bayes classifier is a popular machine learning classifier, because of its simplicity and is often used as baseline for text categorization. The classifier makes the ”naive” assumption that independence occurs between all the features. The classifier is applied from Bayes' theorem.

$$p(C_k|x) = \frac{p(C_k) \cdot p(x|C_k)}{p(x)}$$

One downside with Naïve Bayes is that it is a bad estimator so probability outputs are not very trustworthy.

2. Support Vector Machines

SVM are a set of un-/supervised learning methods for machine learning that can be used for classification, anomaly detection and regression. One of SVMs strong suits is that they are very efficient when your data is high dimensional and also effective in cases where number of dimensions are greater than the number of samples. SVMs are especially good to solve text and hypertext categorization since the application decreases the use of label training occasions Disadvantages with SVM are that they do not provide an estimate of the probability. To get these a calculation of an expensive cross-validation may be performed. In cases when the number of samples are significantly less than the number of features is likely that the method gives a poor result.

Results

1. Naïve Bayes

Our Nave Bayes classifier was built and the classification performance was also compared, all using the scikit-learn package for python.

The results we obtained at the end of the 2nd stage of the pipeline were:

	Precision	Recall	F1 Score
Non-Sarcastic	64%	57%	60%
Sarcastic	61%	67%	64%
Overall	62%	62%	62%

The results obtained at the end of the pipeline were:

	Precision	Recall	F1 Score
Non-Sarcastic	63%	57%	60%
Sarcastic	58%	64%	61%
Overall	61%	61%	61%

This shows us that when topics are included as features, the performance of the classifier drops in the case of Naïve Bayes.

The precision rate in the table above shows that there is a larger number of false positives (58%) in sarcastic sentences than the sarcastic sentences (63%). Naive Bayes also classifies more false negatives in non-sarcastic sentences than the sarcastic ones, as suggested by the reported recall values. Using the reported Precision and Recall values, the classifier demonstrates higher F1 Score in sarcastic sentences than the non-sarcastic ones.

Thus, we can see that the Naïve Bayes classifier is not suitable for our problem.

2. Support Vector Machines

We built a one class SVM using scikit-learn package for python.

The results we obtained at the end of the 2nd stage of the pipeline were:

	Precision	Recall	F1 Score
Non-Sarcastic	64%	63%	73%
Sarcastic	77%	69%	68%
Overall	76%	66%	66%

The results we obtained at the end of the pipeline were:

	Precision	Recall	F1 Score
Non-Sarcastic	65%	79%	71%
Sarcastic	76%	60%	67%
Overall	71%	69%	69%

From the results we can see that upon taking the Topic features into consideration, the performance of the SVM improves. Once the topic features are added, the classifier is much more accurately able to classify if a given sentence is sarcastic or not.

The precision rate in the table above shows that there is a larger number of false positives (65%) in the non-sarcastic sentences than in the sarcastic sentences(76%). The SVM classifies a larger number of false-negatives in the sarcastic sentences as compared to the non-sarcastic sentences. Thus, the F1 Score of non-sarcastic sentences is higher than the sarcastic sentences.

This shows us that the SVM is a more suitable classifier for our problem.

Conclusion and Future Work

Sarcasm can be present in a large number of topics and is highly contextual. Therefore, if we want to build a classifier that detects sarcasm in all topics, we need to sample much larger data. The use of unigrams and bigrams alone is not sufficient. Finding new features relevant to sarcasm is a critical step for improving sarcasm detection.

The poorer results obtained using Naive Bayes were not a surprise. Since we know that sarcasm is highly contextual, we expect the context of the sentence to play a big role in the sarcastic nature of a sentence. As implemented, the Naïve Bayes approach does not take any of that into account. We did find that in both of the classifiers, the misclassification of sarcastic data was a problem.

The accuracy of both classifiers also depended on the mixture of feature types that were present. We saw that when the topic features were added, the performance of the Naive Bayes classifier decreased whereas the performance of the Support Vector Machine improved. This highlights the importance of feature engineering in sarcasm detection.

Another approach to sarcasm detection is to see how sarcastic a particular sentence is. For example, this sentence is 56% sarcastic. This would allow us to classify a sentence more accurately.

To find a general solution is hard, but we can build a model which allows to detect sarcasm based on the type of text might be easier.

We conclude by saying that a lot more research must be put into sarcasm detection if a completely automated detector which is extremely accurate is to be made.

References

1. *David M.; Ng, Andrew Y.; Jordan, Michael I (January 2003). Lafferty, John, ed. "Latent Dirichlet Allocation")*
2. *Chun-Che Peng; Mohammed Lakis, Jan Wei Pan : Detecting Sarcasm in Text: An Obvious Solution to a Trivial Problem (2016)*
3. *Erik Forslid; Niklas Wikén: Automatic irony- and sarcasm detection in Social media (August 2015)*
4. *David Bamman; Noah A. Smith: Contextualized Sarcasm Detection on Twitter. (2015)*
5. *Cliche, M. The sarcasm detector, 2014. URL http://* www.thesarcasmdetector.com/about/
6. *scikit-learn. Support fector machines, 2014. URL http://scikit-learn.org/stable/ modules/svm.html.*
7. *Ellen Riloff; Ashequl Qadir; Prafulla Surve; Lalinder De Silva; Nathan Gilbert; Ruihong Huang: Sarcasm as a Contrast between a positive sentiment and a Negative situation.*
8. *Tsur, O., Davidov, D., and Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in twitter and amazon*
9. *Roberto González-Ibánez; Smaranda Muresan; Nina Wacholder: Identifying sarcasm in Twitter : a closer look.*