Sarcasm Detection Analysis

Sanjana Kaza, Nikhil Sarecha, Rehmat Kang

Abstract

Sarcasm refers to the use of words that mean the opposite of what you really want to say, especially in order to insult someone, or to show irritation, or just to be funny. It is also considered to be one of the more challenging features to interpret through sentiment analysis. Although humans have developed the linguistic know-how to interpret sarcasm based on tonality and socio-cultural norms, machine learning algorithms are still in their infancy to successfully comprehend sarcasm in a sentence. In this project, we explore some data-mining techniques to extract objective features of sarcasm and extract those from a dataset of tweets to train several different classifiers which analyze, compare and contrast their accuracy results. Special attention was paid to classifiers already covered in this course to help the team understand the technical working of such algorithms better.

Introduction

Due to the increase in the access of social media to a wide spectrum of people, sarcasm has become a powerful form of expression. This particular form of expression is also growing increasingly relevant for data gathering which aids applications in domains of media marketing, consumer satisfaction, classifying news etc. However, the international contrast in the literal and implied idea makes it rather difficult to comprehend sarcasm by machine learning algorithms. Detecting sarcasm automatically is particularly useful for opinion comprehension and data gathering and classification purposes, and has received growing interest from the natural language processing community. One particular sarcasm-rich source of data is tweets from Twitter's database, which also form the basis of this project.

For the purposes of this project, we work with a dataset of approximately 40,000 tweets, each of which has been classified as sarcastic or not. We then studied this data looking for common features which allowed us to implement our feature extraction models. We then trained a set of classification models, namely Logistic Regression, LinearSVC, Gaussian Naive Bayes, Decision Trees and Random Forests to test for the accuracy of our models against smaller test data sets. The primary focus of this project was to analyze how different features affect training models, and how each training model compares with another. With this, the team not only gained a deeper understanding into machine learning applications of classification techniques and feature extraction, but also got to work on a project that employs real-world data-mining techniques to extract objective features that can be analysed from an algorithm's perspective.

Related Work

In Deep learning sphere using Neural Networks in Zhang and Zhang (2016), although a limited amount of work has been done using neural networks for sarcasm detection and doesn't really push the envelope forward in that field, these models are very effective and have had increasing applications in sentiment analysis. These two tasks are quite similar as they work hand in hand when training and testing models. It does however give highly competitive results as demonstrated in the paper, with one reason being the power of neural networks in automatic feature induction, which helps capture patterns not really possible to catch using manual feature extraction processes.

This paper talked about a systematic approach to multimodal learning for sarcasm detection Castro, Hazarika (2019), which leveraged three different modalities which included use of text, speech and visual signals. Although tweets can feature multimedia and this approach can reduce rates of upto

12.9%, a very high quality dataset is required with rich annotations which sacrifice corpus size. A much balanced version of a dataset is needed which would need a lot of data processing overhead, not ideal if we need a lean but effective model as the one used in our case.

Dataset and Features

A **dataset** of 39780 tweets was used. The dataset contains the sarcastic/non-sarcastic labels of the tweets. The dataset is balanced containing 18k sarcastic data and 21k non-sarcastic data.

Extracted Features:

1. **Pragmatic Features:**
   a. **POS Tags**: POS Tags are useful to NLP tasks since they give a report on how a word is being used in the context of the given sentence. We have used Nltk library to collect the POS tags. 5 different tag sets have been used: Nouns, Verbs, Interjections, Adverbs and Adjectives. The POS tags of tweets were counted and inserted into the relevant set.
   b. **Punctuations:** Punctuations such as exclamation points and continuous use of question marks are usually used in sarcastic data as a way to lay emphasis on the emotion of the sentence. We have counted the occurrences of exclamation points and the occurrences of 2 consecutive question marks.
   c. **Capital Letters:** Capitalized letters are often seen in sarcastic data during the shift of the sentiment or when the person wants to highlight the sarcastic component of the text. We have counted the occurrences of capital letters to add into our features set.
   d. **Scoring of Emoticons:** Emoticons give an insight into the overall sentiment of the text. We have used a scoring system where positive emoticons in a tweet are assigned positive weights and the negative emoticons are given negative weights. The weights are given a spectrum, so the most positive emoticon gets the largest weight and the most negative emoticon gets the least weight. We collected the net score of each tweet.
2. **Linguistic Features:**
   a. **Overall Sentiment:** Getting the overall sentiment score of the tweet would help us get an idea of the range of the sentiment scores associated with sarcastic tweets. TextBlob was used to get the overall sentiment score of the tweet.
   b. **Contrast:** Sarcastic data usually contains a contrast of sentiments, the sentence goes from being positive to negative or vice versa. The tokens of tweets were divided into two halves and the sentiment of each half was calculated. The difference between the two sentiment scores was recorded.
   c. **Subjectivity:** Sarcasm tends to be mostly subjective so the subjective scores of the tweets used were also collected.
3. **Lexical Features:**
   a. **N-grams:** N-grams are commonly used for a lot of sentiment analysis tasks in order to get the relation between tokens and also in order to get the accurate sentiment of the text by looking at the two or three consecutive words together. We have used different types of n-grams: Unigrams, Bigrams and Trigrams in combination with term frequency. The term frequency is intended to show how important that word or combinations of words are in context of the document provided.

Methods

For the purposes of this project, we processed the data to extract the feature sets mentioned above and used a set of 5 distinct classifiers to obtain accuracy scores. These 5 classifiers are listed as follows:

1. **Logistic Regression:** Logistic regression is among the most common algorithms used to solve classification problems. As such, this particular classifier describes data and finds a relationship between variables through a hypothesis dependent on the input, resulting in a binary classification of 0 (negative) or 1 (positive).

Input -> X	Hypothesis -> Z = WX + B	theta(X) = sigmoid(Z)

   Here, the hypothesis results in a probabilistic determination of the classification of a particular input, which is run against a sigmoid function to obtain a 1 for Z -> infinity and 0 for Z -> -infinity.
2. **Linear Support Vector Classifier (LinearSVC):** LinearSVC is yet another type of a classification problem, which in our case allows us to make binary classifications implying whether or not our data is sarcastic. Unlike logistic regression which aims at maximizing the probability of the input data, based on the distance of the data from the separating hyperplane, LinearSVC attempts at finding an optimal separating hyperplane that maximizes the distance of the closest points to the support vector margins. Such a margin further reduces the risk of error on the data.

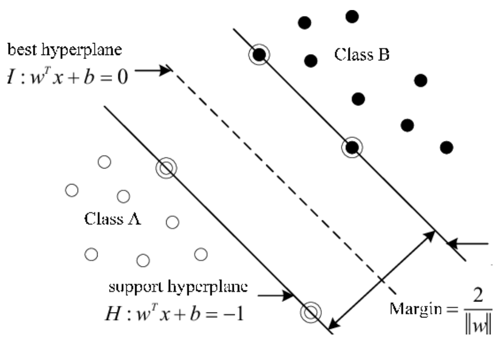

Figure 1. SVM Representation

3. **Decision Tree:** Decision Tree is a power classification algorithm that depends on a series of decisions to deduce a binary result. Its flowchart represents that of a tree wherein the internal nodes represent a test on an attribute, branches represent the outcome of such a test and leaf nodes represent the classification label.
4. **Random Forest:** The underlying working principle of random forest remains the same as that of decision trees. However, while decision trees work on the entire dataset, random forests randomly selects specific features from the dataset to build multiple trees upon and then averages the results.
5. **Gaussian Naive Bayesian:** Gaussian Naive Bayesian (GNB) is an advanced variation of Naive Bayes Classification based on Bayes' probability theorem. GNB supports continuous data on top of the Naive Bayesian assumption that each feature within a dataset is independent of another. Naive Bayesian further employs a family of algorithms based on the same principle which make this model highly efficient.

Reused code: We have gotten the contrast feature score and capital letters feature score from a github [3]. We have modified the contrast feature score such that it also takes into account punctuations whereas their method doesn't. Furthermore, our capital letters scoring method does not have a threshold value while theirs does. These two methods are present in the features.py part of the file. We have implemented 12 other feature extraction methods apart from these two in our model. We have gotten a dictionary to use for emoticons scoring from a github account [4]. Our emoticon scoring method is different from theirs, only the scoring dictionary is the one we reused. This dictionary is present in the cleaning.py part of the file.

Experiments/Results/Discussions

The following tables tabulate the obtained results for the various classifiers and features employed in this project. In the first table, we see how the models perform against a bare dataset with all our combined features and n-grams included. The second table then tabulates the scores for the best model for each combination of features and n-grams. Finally, the third table tabulates the scores for the best model for several n-gram arrangements.

	Test Score (%)	**F1 Score (%)**
Logistic Regression	78.78	76.42
LinearSVC	79.36	77.28
Decision Trees	61.46	62.97
Random Forest	71.79	62.92
Gaussian Naive Bayes	66.71	65.23

Table 1. Model Accuracy

Analysis: Fairly stable estimate of the model's accuracy based on the F1 score. Both the linear modelling algorithms perform well, while the decision tree model resulted in relatively lower accuracy, along with a general observation of overfitting. Increasing the max-depth hyperparameter reduced the overall overfitting. One fix to this could be to employ larger datasets.

	Test Score (%)	**F1 Score (%)**
Pragmatic and Linguistic Features (RF)	60.61	56.6
Lexical Features (N-Grams) (LinearSVC)	79.1	76.7
Pragmatic Features + Linguistic Features + Lexical Features (N-Grams) (SVC)	79.4	77.3

Table 2. Different Feature Combinations

Analysis: The accuracy and F1 score for different feature combinations is shown in Table 2. Although the pragmatic + linguistic features accuracy score is not high, we can see that these features still are good indicators (60%). The lexical features are a significant improvement over the other two features. The highest accuracy is obtained when all three feature sets are used for the classification.

	Test Score (%)	**F1 Score (%)**
Unigrams (SVC)	77.7	75.14
Bigrams (SVC)	70.16	64.78
Trigrams (Decision Trees)	58.97	29.77
Unigrams + Bigrams (SVC)	79.1	76.7
Unigrams + Bigrams + Trigrams	79.26	77.12

Table 3. Different N-Gram Combinations

Analysis: Table 3 shows the highest test accuracy score and its corresponding F1 score while using different lexical features. The accuracy when only one type n-gram was used decreases with the increase in the number of tokens taken and we can see a similar decrease in the F1 score. There is a significant drop in the scores for trigrams, this might be because trigrams are more unique since there are looking at three tokens at a time and the dataset we have is not big enough for trigrams to get any useful information. Whereas the accuracy when different types of n-grams are combined increases with the increase in the number of token sets considered and we can see a similar increase in the F1 score.

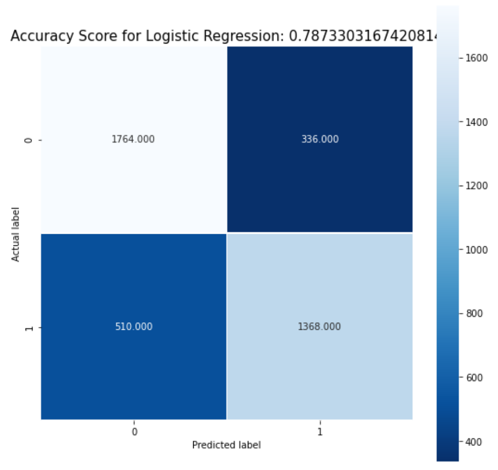

Figure 2. Confusion matrix for Logistic Regression scores

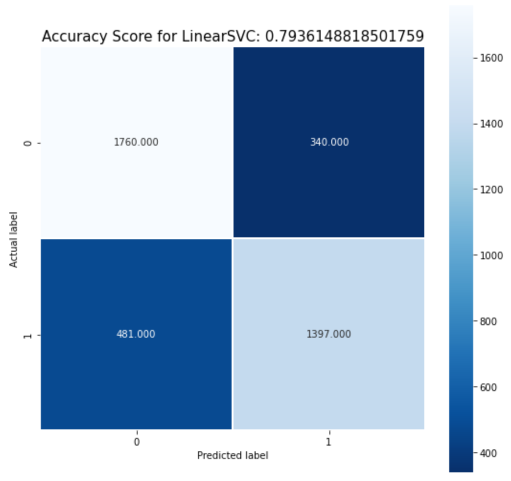

Figure 3. Confusion matrix for LinearSVC scores

Conclusion/Future Work

In this project, we have shown how important features such as POS tagging, punctuations, n-grams etc. are in detecting the sarcasm of a text. It has been shown that the pragmatic and linguistic features alone are not enough to accurately classify sarcasm. Linguistic features put the words into context with the respect to the rest of the document and as seen have caused a significant increase in accuracy. We have also seen how different classifiers perform with respect to this dataset and the given features, and what the drawbacks were of some of the classifiers when dealing with this data.

Extensive future work is possible going forward with sarcasm detection implementations through machine learning. Going forward, better algorithms could be employed to decipher the meaning of each individual hashtag instead of removing them, which could greatly affect the polarity of the input tweet. More attention could also be given to incorporation of an exhaustive list of emojis with a wider score spectrum which could be a better estimate of sarcasm. Given more time, fine-tuning of hyperparameters where possible could also affect our models' behavior. Finally, more sophisticated machine learning applications can also enable monitoring of the user's previous tweet behavior which could affect the interpretation of the input data

Contributions

The paper was a culmination of the efforts of Sanjana, Nikhil and Rehmat, where Sanjana's major contributions were the implementation of a base model, and worked with logistic regression and linear SVC parameters to get an optimally functioning ML algorithm and extraction of features. Nikhil was responsible for making specific changes to the dataset such as extracting some basic features from the data in the effort of cleaning it and preprocessing it, thereby improving the score. Rehmat's contribution was researching existing models, features, and implementing additional features which featured improvements of the accuracy score of the base model and found optimal datasets to be used to get the desired results, with changes and tweaks to the model to prevent over and under fitting.

References

[1] Singapore University of Technology and Design, "Tweet Sarcasm Detection Using Deep Neural Network," 2016. [Online]. Available: https://www.aclweb.org/anthology/C16-1231.pdf. [Accessed: 23-March-2021].

[2] S. Castro, D. Hazarika, V. Pérez-Rosas, R. Zimmermann, R. Mihalcea, and S. Poria, "Towards Multimodal Sarcasm Detection," arXiv.org, 05-Jun-2019. [Online]. Available: https://arxiv.org/abs/1906.01815. [Accessed: 01-May-2021].

[3]Varunmuthanna,"varunmuthanna/sarcasm-detection." [Online]. Available: https://github.com/varunmuthanna/Sarcasm-Detection. [Accessed: 07-May-2021].

[4] Jbnerd, "jbnerd/Sarcasm_Detection_Twitter," *GitHub*. [Online]. Available: https://github.com/jbnerd/Sarcasm_Detection_Twitter. [Accessed: 07-May-2021].

[5] T. Jain and N. Agarwal, "Sarcasm detection of tweets: A comparative study," IEEE Xplore, 06-Mar-2017. [Online]. Available: https://ieeexplore.ieee.org/document/8284317. [Accessed: 03-May-2021].

[6] G, Dr. Vadivu & Chandra Sekharan, Sindhu. (2018). "A comprehensive Study on sarcasm detection techniques in sentiment analysis." ResearchGate,01-June-2018. [Online]. Available: https://www.researchgate.net/publication/325843750_A_COMPREHENSIVE_STUDY_ON_SARCASM_DETECTION_TECHNIQUES_IN_SENTIMENT_ANALYSIS
[Accessed: 03-May-2021].