SAINEY MANGA
MASTER'S IN DATA SCIENCE AND ECONOMICS, UNIVERSITY OF MILAN

TEXT MINING AND SENTIMENT ANALYSIS PROJECT 2020/2021

Author's Note

This is an end of term paper submitted in partial fulfilment for the exam requirement of the course.

Course Coordinator: PROF. ALFIO FERRARA

1. INTRODUCTION

Sarcasm according to the oxford dictionary is a way of using words that are the opposite of what you mean in order to be unpleasant to somebody or to make fun of them. It is a linguistic tool that uses irony to express contempt (Devamanyu Hazarika et al.). Its figurative nature poses a great challenge for effective systems performing sentiment analysis (Cambria et al., 2017). Previous research in automated sarcasm detection has primarily focused on lexical and pragmatic cues found in sentences (Kreuz and Caucci, 2007). In the literature, interjections, punctuations, and sentimental shifts have been considered as major indicators of sarcasm (Joshi et al., 2017). When such lexical cues are present in sentences, sarcasm detection can achieve high accuracy. However, sarcasm is also expressed implicitly, i.e., without the presence of such lexical cues. This use of sarcasm also relies on context, which involves the presumption of commonsense and background knowledge of an event. When it comes to detecting sarcasm in a discussion forum, it may not only be required to understand the context of previous comments but also the necessary background knowledge about the topic of discussion. Examples of sarcastic sentences are **" I work 40 hours a week to be this poor (https://examples.yourdictionary.com/examples-of-sarcasm.html)"** and **"*Was there a lack of graves in Egypt, that you took us away to die in the wilderness? (Exodus 14:11)*."**

In this project, we used the ideas in the literature mentioned in the introduction as a guide to our choices of data cleaning and preparation. We basically applied two classes of models namely, traditional machine learning models and one model in deep learning model class. First, four different types of models of machine learning were applied just to explore the performances of the different models. As we would see later, each of these models has an accuracy better or worse than other corresponding models. Then we also compare that class of models to the neural network model. In a nut cell, the traditional machine learning models are Logistic Regression, Naïve Bayes Classifier, Support Vector Machine and Random Forest Classifier, and our deep learning model is Convolutional Neural Network (CNN). The former class of models outperformed the latter. Models were first used to detect sarcasm and then predicted the probability of each sentence in the corpus receiving a sarcastic comment, then we tested our models by applying a random sentence to produce the probability that the sentence will receive a sarcastic comment.

2. RESEARCH QUESTION AND METHODOLOGY

The main goal of this project is, given **only** the parent comment and the Reddit category (subreddit), to predict the probability of a parent comment to receive a sarcastic comment. This is indeed a step further to existing literature where most approaches only worked on the detection of sarcasm on a corpus. Therefore, the efforts put in this project are jeered towards answering two main research questions:

- What is the probability of a parent comment receiving a sarcastic reply given only Subreddit and parent comment attributes in the dataset?
- Which of the models used has a better performance based on the cross-validation accuracy?

We applied various machine learning models and tools to detect sarcastic comments and then predict the probability as per the research questions. Subsequently, we would provide a theoretical overview of all the methods used in this project.

- **Logistic Regression**: falls in a class of probabilistic models referred to as discriminative models. Such models assume that the dependent variable is an observed value generated from a probabilistic distribution defined by a function of the feature variables. The model was executed using the ready-made function in scikit learn with a maximum iteration of 1000 a solver 'lbfgs'.
- **Support Vector Machine**: creates two parallel hyperplanes symmetrically on each side of the decision boundary, so that most points lie on either side of these two margin hyperplanes on the correct side. It was also implemented using scikit learn with a regularization penalty of 12.
- **Naïve Bayes Classifier**: The naïve Bayes classifier uses a probabilistic generative model that is identical to the mixture model used for clustering. The model assumes that the corpus is generated from a mixture of different classes. The observed (training and test) data are assumed to be outcomes of this generative process, and the parameters of this generating process are estimated so that the log-likelihood of this data set being created by the generative process is maximized. Scikit learn provides an easy function for solving this model.
- **Random Forest Classifier**: falls under the broad umbrella of ensemble-based learning methods. The key principle underlying the random forest approach comprises the construction of many “simple” decision trees in the training stage and the majority vote (mode) across them in the classification stage. The random forest is a collection of decision trees that are associated with a set

of bootstrap samples that are generated from the original data set. The nodes are split based on the entropy (or Gini index) of a selected subset of the features. The default parameter in scikit learn implements a 100 number of decision trees.

- **Convolutional Neural Network (CNN):** CNNs were largely known for the major breakthroughs in Image Classification and are the core of most Computer Vision systems today. More recently, we've also started to see the application of CNNs to problems in Natural Language Processing and gotten some interesting results. It is composed of convolutional layers and down sampling or pooling layers. Convolutional layers comprise neurons that scan their input for patterns and pooling layers are often placed after convolutional layers in a CNN, mainly to reduce the feature map dimensionality for computational efficiency, which can in turn improve actual performance. We implemented the CNN model using TensorFlow with 20 epochs, 16 feature maps, kernel shape of 3, maximum pooling kernel of 2, "relu" as the activation function and binary cross entropy as loss.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** it's a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

3. EXPERIMENTAL RESULTS

3.1. Data Description

The dataset "Sarcasm on reddit" was downloaded from Kaggle. It contains 1.3 million Sarcastic comments from the Internet commentary website Reddit. The dataset was generated by scraping comments from Reddit containing the sarcasm tag \s. This tag is often used by Redditors to indicate that their comment is in jest and not meant to be taken seriously, and is generally a reliable indicator of sarcastic comment content. The dataset contains 1010826 rows and about 10 columns, but our interest lies on only three columns defined below:

- Label: shows whether the comment is sarcastic (1) or not (0)
- Subreddit: tells the category in which the comment belongs or the topic if you like.
- Parent-comment is the original comment posted with tagging.

As we can see, the dataset contains a balance between sarcastic and non-sarcastic comments. In other words, sarcastic and non-sarcastic comments are evenly distributed. We noticed that some comments in the dataset had URLs in them and we had to remove such rows from the dataset. There were about 119 rows with URLs. We decided to go ahead with our tasks without much preprocessing on the data as we have seen earlier that such could affect the performances of our models.

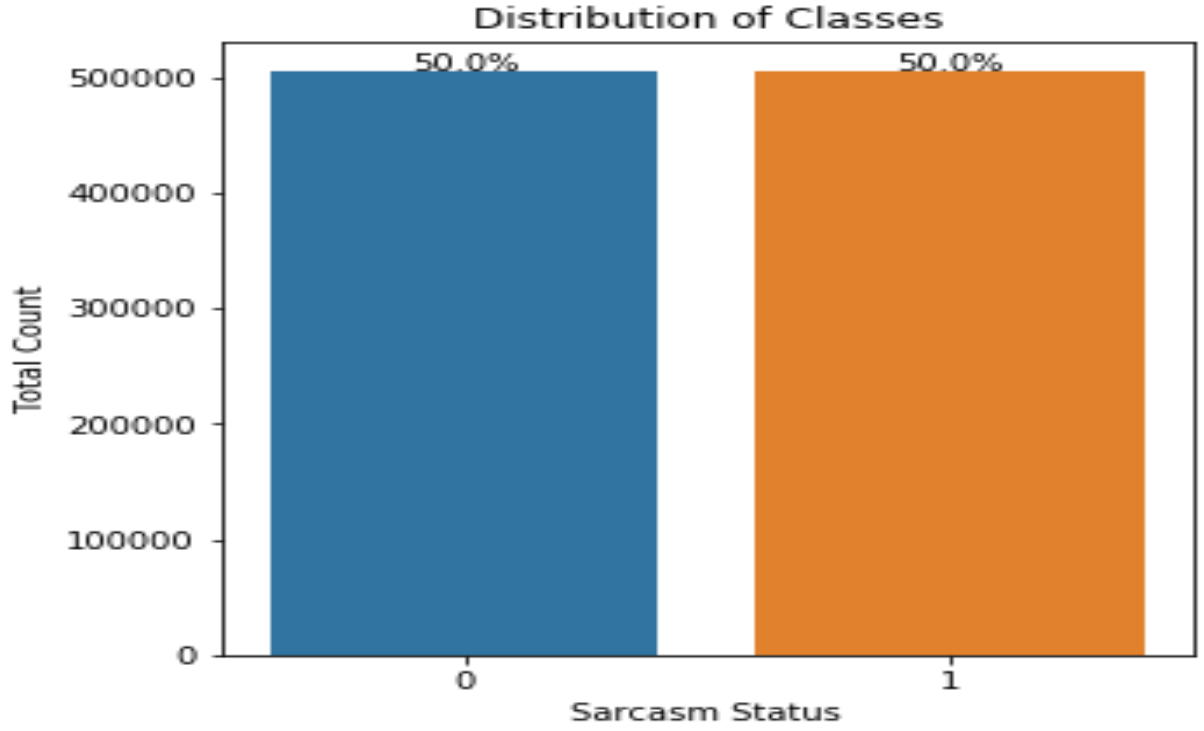

Figure 1: Data Distribution

Figure 1 shows that the dataset is evenly distributed with 50% to 50% of sarcastic and non-sarcastic comments denoted by 1 and 0 respectively. This indicates that the dataset is balanced as mentioned earlier.

3.2. **Evaluation Strategy**

Cross-validation: It is the evaluation strategy used to measure the performances of our individual classifier models. The scikit learn function makes it easy to implement this strategy, the dataset is split in to 5-folds of which the 4-folds are trained on by the model and the fifth fold is tested on by the model.

3.3. Experimental Methodology

Before implementing our models, we want to check and understand the nature of our dataset in respect with the subreddit attribute to see which categories or topics are

more likely to have sarcastic comments. We've checked all categories with more than 1000 entries and printed the first 5 rows to affirm our curiosity.

subreddit	size	mean	sum
creepyPMs	5466	0.784303	4287
MensRights	3356	0.680870	2285
ShitRedditSays	1283	0.661730	849
worldnews	26377	0.642529	16948
Libertarian	2562	0.640125	164

Table 1: sarcastic inclination

We also tried to sort the top 5 subreddit with the most sarcastic comments in it.

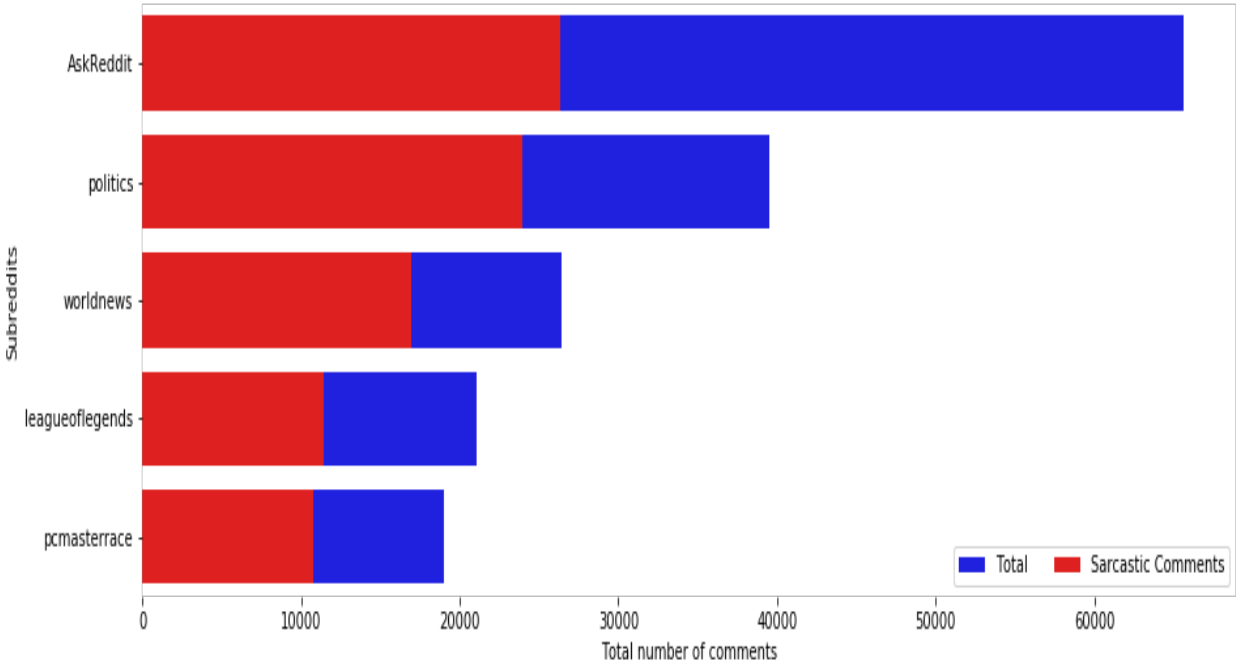

Figure 2: Top 5 subreddit with sarcasm

From Figure 2, we know that AskReddit has more sarcastic comments followed by politics. This could be as a result of the distribution of the dataset amongst the Subreddit. AskReddit as the figure shows, has more comments in entirety than other categories.

For the first class of models, we divided our dataset into 80% for training set and 20% for test set. We note that our input consists of two variables, and we should find a way to vectorized and represent the two as an input. We implemented a pipeline using scikit learn, where TF-IDF was implemented on parent-comment with 3-grams and 2-grams on subreddit. We called the pipeline "transformer" which is used to

transform the x-train set to be fed to our models. All the machine learning models were fed by the input prepared as mentioned above. We executed all the 4 models hoping to have an improved performance, but we would see what happens later.
For the CNN model, we have to change the shape of the input word vector which was implemented using a pipeline for the first class of models. First, we concatenated the two input variables separating them with a space and represented them as an input to be fed to the model.

3.4. RESULTS

MODEL	CV-ACCURACY
Logistic Regression	0.5991021099297325
Support Vector Machine	0.573920460889811
Naïve Bayes Classifier	0.5956057889455704
Random Forest	0.512431283644087
CNN	0.5004

Table 2: Model Performance

Table 2 above shows that the Logistic Regression and Naïve Bayes models have the best performance with a cross-validation of about 60%, though the former is slightly higher. The deep learning model (CNN) performed worse than other models even though we implemented it to give a different flair of approach to our piece of work.

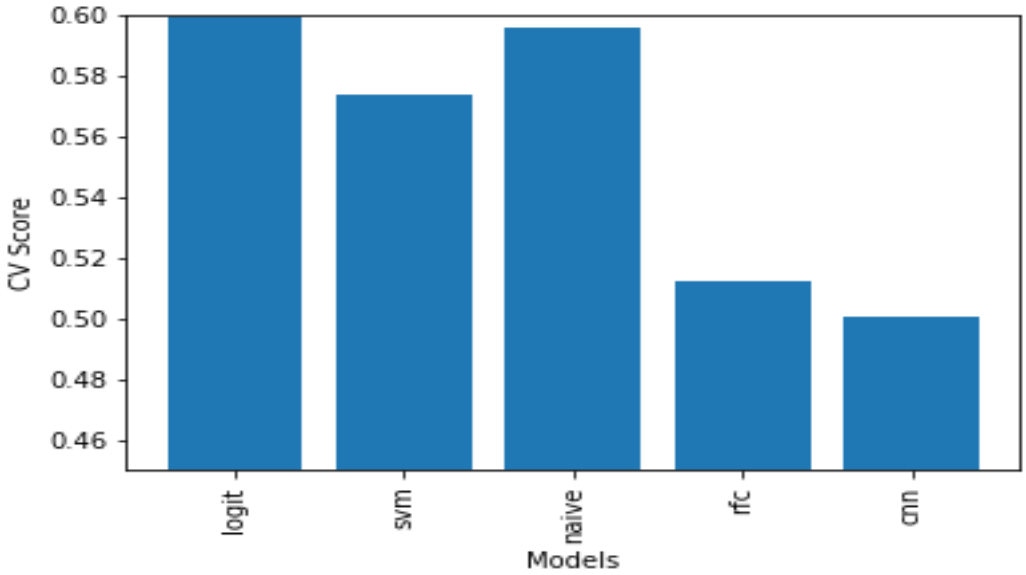

Figure 3: Model performance summary

Recall that our main task is to predict the probability of a parent comment receiving a sarcastic reply given only Subreddit and parent-comment attributes in the dataset. We used three models to predict the probability of the entire input array using the built-in functions in scikit learn.

To test our models, we took a random sentence from the Reddit platform and executed it on three of our trained models with the sentence as parent-comment, then predicted the probability of the sentence receiving a sarcastic reply.

"Who is Americas best president"?

subreddit	**Probability**		
	LR	NB	RFC
Politics	0.60159689	0.68345365	0.52068487

Table 3: Predicted probability

We first applied the sentence in AskReddit as subreddit, but the predicted probability under politics as subreddit is higher. This could be explained by the presence of the "president" in the sentence, which makes it more inclined to the chosen subreddit. However, being preview on the context on which the said question was also asked would be a great ingredient in predicting sarcasm. The fact that American politics is known to be a heated topic and captures the attention of the world, attracts a lot of discussions on various platforms. The sentence has a higher probability of being sarcastic under Naïve Bayes than Logistic Regression and Random Forest Classifier.

4. CONCLUSIONAL REMARKS

Analyzing sarcasm is one of the most challenging tasks in natural language processing. Understanding the characters that makes a comment sarcastic is one thing, it is also important to have knowledge about the general context on which the comment was made. We have so far seen that parent-comments in certain subreddits are more likely of being sarcastic than others.

We introduced two classes models, four under traditional machine learning and one under deep learning. Logistic Regression and Naïve Bayes models outperformed other models with about 60% cross-validation. We intended to improve the cross-validation accuracy and perhaps provide a different approach by introducing the convolutional neural network model, but the accuracy got worse.

Finally, the predicted probability of the Naïve Bayes and Logistic Regression models are higher with about 68% and 60% respectively.

This piece of work could not cover to some extent, a scope of issues. We could have tried a second deep learning model to compare it with the CNN model. Furthermore, pretrained models could have been utilized when implementing the neural network model. Feature work in this area could explore on the recommendations raised above.

References:

[1] Mikhail Khodak, Nikunj Saunshi, Kiran Vodrahalli (April 2017). A Large Self-Annotated Corpus for Sarcasm

[2] Adikya Joshi et al (Nov. 2017). Automatic Sarcasm Detection: A survey

[3] Devamanyu Hazarika et al (Aug. 2018). CASCADE: Contextual Sarcasm Detection in Online Discussion Forums

[4] Charu C. Aggarwal (2018). Machine Learning for text.