University of Trieste
Department of Mathematics and Geosciences
Data Science and Scientific Computing

Project Report

Natural Language Processing Course

Text Classification : Sarcasm Detection

—

Academic year 2020/2021

Submitted by:

Azza M. N. Abdalghani

Issue date: 10.07.2021

1 Introduction

Sarcasm is an intricate form of speech, where meaning is conveyed implicitly and it could be both positively funny or negatively nasty meaning. In human social interaction detecting sarcasm plays an important rule in order to understand the actual sentiment and meaning of the discourse. But still it is not easy for all people to detect sarcasm, since it is largely dependent on context, thus this difficulty in recognition of sarcasm causes misunderstandings in everyday communication and poses problems to many Natural Language Processing (NLP) systems such as summarization systems and dialogue systems [1]. Thus, due to representation variations in the textual form sentences, sarcasm detection becomes one of the most difficult text classification tasks in data analysis and Natural Language Processing (NLP).

This study presents a NLP based approach to sarcasm detection on Reddit website, Reddit is an American social news aggregation, web content rating, and discussion website [2]. Kaggle provides a balanced dataset called Sarcasm on Reddit, which is a collection of 1.3 million sarcastic comments from the Internet commentary website Reddit. Mainly it is a labeled dataset and it was generated by scraping comments from Reddit.

The key aim of this work is build a model to predict whether a comment is sarcastic or non-sarcastic, Supervised classification algorithms are implied for the sake of this goal. In particular, a Logistic Regression model, and Convolutional Neural Network (CNN) and BERT model are implemented to achieve this goal. In order to evaluate the performance for each algorithm, the metrics *Precision*, *Recall*, *F-score* and *Accuracy* are used.

2 Data

2.1 Data Overview

Sarcasm detection dataset on Kaggle is balanced, and basically it contains comments with their sarcastic label with other meta-data such as the author of the comment, the parent comment, the category of subreddit, the published date and others as shown in figure 1.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1010826 entries, 0 to 1010825
Data columns (total 10 columns):
 #   Column          Non-Null Count    Dtype
---  ------          --------------    -----
 0   label           1010826 non-null  int64
 1   comment         1010773 non-null  object
 2   author          1010826 non-null  object
 3   subreddit       1010826 non-null  object
 4   score           1010826 non-null  int64
 5   ups             1010826 non-null  int64
 6   downs           1010826 non-null  int64
 7   date            1010826 non-null  object
 8   created_utc     1010826 non-null  object
 9   parent_comment  1010826 non-null  object
dtypes: int64(4), object(6)
memory usage: 77.1+ MB
```

Figure 1: Info for Sarcasm Detection Dataset

As shown in Figure 1, the dataset contains 1010826 observations, and 10 columns. In this work, we are going to consider only *comment* and *label* variables, we are going to focus on predicting the label of a given comment depending on the context only and not taking into account other features. Since the provided dataset is large for local memory and processors, we sampled two datasets containing an equal number of sarcastic and non-sarcastic comments, first dataset of size 100000 used for training part (saved in *train_ data.csv* file), and the other dataset of size 50000 for testing part (saved in *test_ data.csv* file).

	Training	Testing
sarcastic	50000	25000
non-sarcastic	50000	25000

Table 1: Total number of data used for training and testing

As we are going to focus on the context of the comments, we visualize the most important words have been used by users in Reddit for each label. Figure 2 and Figure 3 below show word-clouds for each class, which show the importance of words for each class.

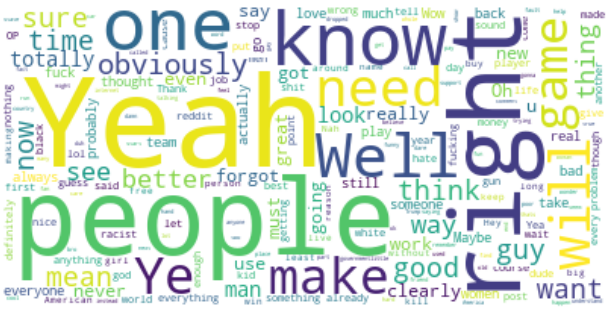

Figure 2: Most popular words in sarcastic comments

Figure 3: Most popular words in non sarcastic comments

It is noticed from the figures that the most used words in sarcastic comments are ("Yeah", "people", "right", "Well", "know", "one", ...) and for non-sarcastic comments there are ("one", "think", "people", "will", "well", "time", ...). We can observe that there is no specific words that can define sarcasm, for this reason identifying sarcasm is difficult task since it depends on the context as mentioned before.

2.2 Data Preprocessing

In this part, we applied two methods to filter the datasets used in this work. Firstly, we identified and removed "URL", "Punctuation Marks", "Numbers" and "User names" references present in the comments by using efficient regular expressions. Secondly, we applied lemmatization over the comments to extract the *Lemma* (base form) for each word in the comment, to reduce variation as much as possible. Finally, we dropped any word that has length less than 2, Figure 4 shows a sample of comments before preprocessing and their corresponding clean comments.

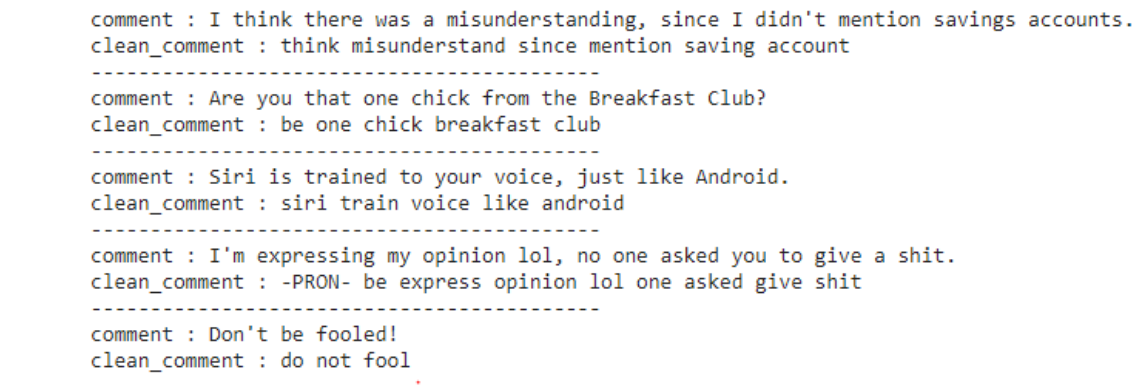

Figure 4: Sample of comments and their corresponding cleaned comments

2.3 Data Representation

Before implementing the proposed models, our text data should be represented in numerical values to be used as input to our predictive models, this operation is called *vectorization* (or feature extraction). In this work, *TfidfVectorizer* is used to achieve this purpose, It converts a collection of raw documents to matrix of TF-IDF features [3]. It stands for "Term Frequency-Inverse Document Frequency", it is a score applied to each word in every comment. For this representation, we also take into account $ngram_range = (1, 2)$,meaning that unigrams and bigrams are extracted.

3 Approach

In this section we will discuss the experimental setup for the implemented algorithms for sarcasm detection task. There are 3 main algorithms are implemented for this work as follows :

1. Logistic Regression Logistic Regression is a classification algorithm used to solve binary classification problem. Logistic regression classifier uses the weighted combination of the input features and passes them through a sigmoid function. Logistic Regression classifier provided from *sklearn.linear_model* has been used as a first classification algorithm, we specified to parameters for this classifier, *n_jobs = -1* to indicate for using all processors, and *class_weight = balanced* mode which gives equal weights for both classes.

2. Convolutional Neural Network

CNN basically is a set of layers of convolutions with non-linear activation functions applied to the results, and CNN can be applied to text classification problems by feeding the network by the data matrix that represents our comments. The architecture of CNN that we applied for this task is shown below :

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               120832
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 131,201
Trainable params: 131,201
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: The architecture of CNN model

After trying multiple architectures, this architecture gives better performance. The input is the data tf-idf representation of the training data comments, and then the network contains 3 hidden dense layers with different number of neurons for each one (128, 64, 32 respectively) and one dropout layer applied to the first dense layer. The output of the network contains one neuron that represent the prediction either 0 (non-sarcastic) or 1 (sarcastic) of the input comment.

3. BERT Model

BERT stands for Bidirectional Encoder Representations from Transformers, it is a transformer-based machine learning technique for NLP pre-training developed by Google [4]. It provides dense vector representations by using deep, pre-trained neural network with the Transformer architecture [5].

There are originally two Bert models, Bert_Base (12 Encoders with 12 bidirectional self-attention heads) and Bert_Large (24 Encoders with 12 bidirectional self-attention heads). In this project, we used *bert_ en_ uncased_ L-12_ H-768_ A-12*, which is Bert_base model that has L = 12-hidden-layers, a hidden size of H = 768, A = 12 attention heads and 110M parameters. This model has been pre-trained on lower-case English text, meaning that the text has been lower-cased before tokenizationinto words.

Some tokenization steps were applied on comments before passing it to Bert model, first we tokenized each comment into word pieces by using a tokenizer from Bert layer and

we added a two tokens for each comment, "CLS" (stands for classification) at the start of the each sequence and "SEP" that separates two parts of the comment. Then each token in each comment is converted to id as present in the tokenizer vocabulary, next all comment are padded to have the same size, 100 was chosen to be the maximum length. Finally a Bert model has been created with 3 Dense layers and 2 dropout, the architecture is shown in Figure 6.

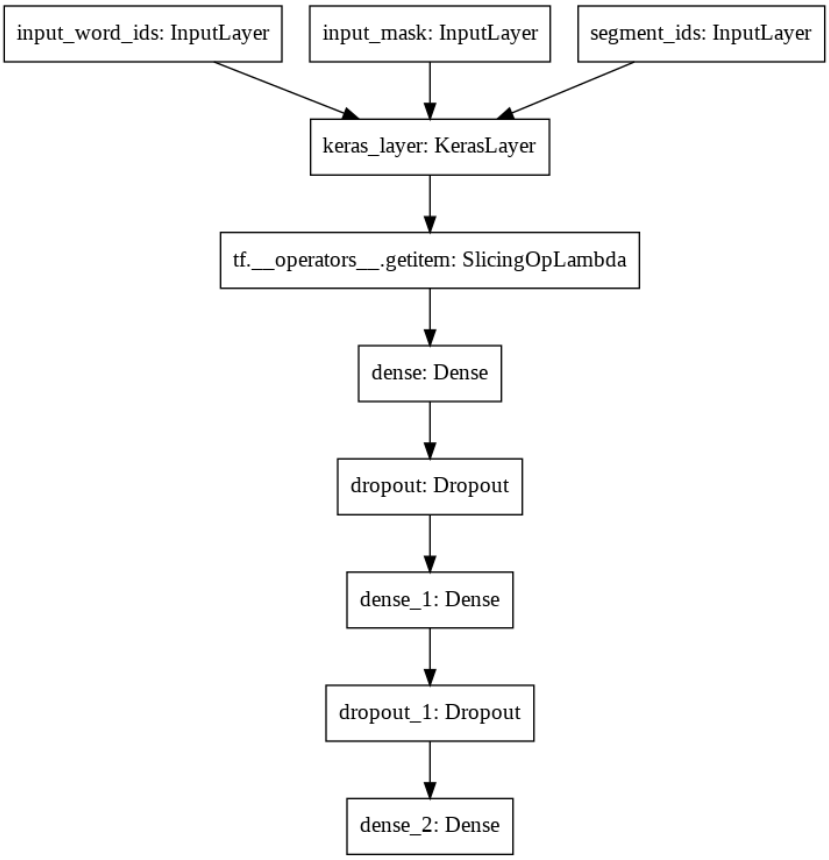

Figure 6: Architecture of Bert model

4 Experimental Results

In this section, we will present the results obtained by the proposed approach on labeled test dataset, and we run all models on Test dataset and then we compare the results with a baseline model based on f1-score and accuracy. The considered baseline model is **Dummy-Classifier** provided by *sklearn*, which is completely independent of the training data as the trends in the training data are completely ignored and instead uses

a specific strategy to predict the class label. In this work, we will use **Most Frequent** strategy, which always predicts the most frequent class label in the training data.
Since, the training data set is balanced, then the accuracy obtained by *Most Frequent Dummy Classifier* is 0.5, based on this result we will compare other classifiers.
First, Logistic Regression algorithm gives 66% accuracy in predicting the class for each comment in Test data, The table below shows the results of this LR classifier :

	precision	recall	f1-score	support
0	0.63	0.73	0.68	25000
1	0.68	0.58	0.63	25000
accuracy			0.66	50000
marco avg	0.66	0.66	0.65	50000
weighted avg	0.66	0.66	0.65	50000

Table 2: classification report obtained from Logistic regression model

From the table, we can observe that LR model has an accuracy equal to 0.66, which makes LR better than baseline model. Moreover, for class **0**, which indicates non-sarcastic class, the recall is 0.73 which means that 73% of the all elements of class 0 were found, and the precision tells that 63% of these found elements were correct, which leads to f1-score = 0.68 (which is the harmonic mean between precision and recall). This means that LR classifier is better in predicting class 0 over class 1.
Secondly, CNN models has a very close accuracy to LR model, it performs 67% accuracy. The results of CNN model are shown in Table 3 :

	precision	recall	f1-score	support
0	0.64	0.76	0.70	25000
1	0.71	0.58	0.64	25000
accuracy			0.67	50000
marco avg	0.68	0.67	0.67	50000
weighted avg	0.68	0.67	0.67	50000

Table 3: classification report obtained from CNN

We can tell that CNN gives almost as the same results of LR model, It also better in predicting class 0 as observed from f1-score for each class.
Finally, we get a better performance with BERT model, it achieves 71% accuracy. The results for BERT are :

	precision	recall	f1-score	support
0	0.69	0.76	0.72	25000
1	0.73	0.66	0.69	25000
accuracy			0.71	50000
marco avg	0.71	0.71	0.71	50000
weighted avg	0.71	0.71	0.71	50000

Table 4: classification report obtained from Bert model

We can observe that all models perform better than baseline model, and BERT model outperforms all of them be achieving better accuracy and f1-score.

5 Conclusion

In this project, we experimented with different classification models (LR , CNN and BERT) on Sarcasm Detection task and analyzed their performances. The results show that BERT model generated better results than LR, CNN and baseline models, It outperforms CNN model by 5%. We can conclude that Sarcasm Identification is a very complex task, and it is fully dependent in the context .

References

[1] D. Davidov, O. Tsur, A. Rappoport, Semi-supervised recognition of sarcastic sentences in twitter and amazon, in: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, CoNLL '10, Association for Computational Linguistics, USA, 2010, p. 107–116.

[2] Wikipedia contributors, Reddit — Wikipedia, The Free Encyclopediadoi:https://en.wikipedia.org/w/index.php?title=Reddit&oldid=1028963157.

[3] scikit-learn developers, sklearn.feature_extraction.text.TfidfVectorizerdoi:{https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html}.

[4] Wikipedia contributors, Bert (language model) — Wikipedia, The Free Encyclopediadoi:{https://en.wikipedia.org/w/index.php?title=BERT_(language_model)&oldid=1029082047}.

[5] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805. arXiv:1810.04805.
URL http://arxiv.org/abs/1810.04805