COMP90042 Project 2020:

Climate Change Misinformation Detection

Amar Babuta (1031452)

amar.babuta@student.unimelb.edu.au

Abstract

There has been an exponential rise in the number of text documents that need a greater understanding of machine learning techniques to be able to correctly classify the texts in various applications. The achievement of applying various machine learning algorithms depends on how they can understand complex models and non-linear relationships within the data. This paper discusses the possible approaches that can be used for text classification of climate change misinformation. The challenge was to build a system that detects whether a document contains climate change misinformation. I was able to achieve 0.65 f1_score for label classification. TF-IDF model was used to create features for combined training and external data while for label classification pre trained models were used on the test dataset.

1. Introduction

In the world where information is available everywhere, it becomes very important to find whether the information provided is misclassified or not. This project is about building a machine learning approach that detects whether the climate change information is misclassified or not for the unlabelled data. Climate change misinformation task is to identify whether the climate change information is misclassified or not for the given dataset. The application of climate change misinformation is to identify the misinformation related that is regularly being distributed on mainstream and social media. In this project we have been given the data with their text and positive label (meaning information misclassified) as a training data and we need to predict the labels from the test data.

To predict the label of the text we implemented different Machine Learning classification models. Classification models are used to identify category of the new observations based on the training set of data containing observations whose category is known but since the training data had only one label, so we did some web scraping to find external data that will be included with the training data to make it a binary supervised classification problem. Classification models include linear models like SVM and non-linear ones like K nearest neighbour, Naive Bayes.

2. Data Description

Three datasets were given for this project that is train.json, dev.json, and test-unlabelled.json data.

a) **train.json:** The training data consists of articles with climate change misinformation i.e. only (positive labels).
b) **dev.json:** The dev.json file contains both labels (0 and 1) which will be used for measuring performance of the models and finding optimal hyperparameter.
c) **test-unlabelled.json;** The test.json file contains text for which we need to predict labels using our model.
d) **newtrain.json:** Since, training data contains only positive label and for binary classification multi labels are required so I read the dev.json file with (0 label) and searched for articles that are related to (0 label) and made an external_data.json file and combined it with the train.json to make a new json called newtrain.json which will be used for binary classification.

3. Background

In this section, we discuss the features and techniques that were adopted for the development of our system.

3.1 Feature Engineering

We used text data to extract basic features. The features were:

a) **Stop words Removal**: Stop Words(they are the most common words that doesn't provide context in the language) they are processed and filtered out from the given text as they are of no use. Stop words act like connectors to the sentence, for example, conjunctions like "and, or," articles like "a, an and the" and prepositions. Stop words like these are less important as they contribute to valuable processing time. NLTK library was used to remove stop words I performed tokenization, Lemmatization to remove that. The first thing when we do data preprocessing is to remove the stop words but sometimes it helps us in gaining extra knowledge.

b) **Removal of Punctuation**: Punctuation provides grammatical context to the sentence. Punctuations like comma might not add much value in understanding the sentence meaning. I removed all the punctuation from the text as it helps in reducing the size of training data.

c) **Converting all to lowercase Words**:

I converted each word to lowercase because lowercase conversion of word helps in classifying the words that are present in uppercase as well as lowercase forms which can result in having words that are similar in the feature space. For example, in an article word like "Climate" might be given in uppercase or in lowercase form like "climate" which will be present in feature space. Both of them means the same so conversion to lowercase was done so that we can classify these type of words as same.

d) **Lemmatization:**

It processes the text and groups together the inflected forms of the word for analysis as a single item and returns dictionary form of word. Simply, it can be said that lemmatization is conversion of words in the text to their base form that are present in the dictionary. Lemmatization was done so that I can see what the intended meaning of the word is rather than looking at literal meaning.

3.2 Bag of Words

It processes each text as document and calculate the count of each word which creates word count vector, also called vector features of fixed length. It is simplifying the representation used in NLP. We achieved Bag of Words by using a Library YAKE (Yet another keyword extractor).

YAKE: It is an unsupervised keyword extractor which helps us in extracting keywords and provide their significance in the text. It helped me in generating slightly better results than the normal feature selection.

3.3 TF-IDF Vectorizer

It stands for term-frequency and inverse document frequency for feature extraction. Term-frequency and Inverse Document Frequency are 2 components of TF-IDF. Term frequency is calculated using this methodology and this frequency of the term tells us about the local importance of that term in the text. IDF tells us about the signature words which is important for the document and has high frequency but is not a common word in the text. This is used for filtering the stop words and for the creation of the features.

Term Frequency: It measures the frequency of the term occurring in the document.

$$TF(t) = \frac{\text{Number of times t appears in a document}}{\text{Total number of terms in the dcument}}$$

Inverse Document Frequency:

It measures how important the term is in the document.

$$IDF(t) = Log_e \frac{\text{Total number of terms in the document}}{Number\ of\ documents\ with\ term\ t\ in\ it}$$

4. Approach

1)On studying carefully about the project guidelines and description, I was able to judge that the accuracy for the model depends upon the various features that we will select to train the model. If we have better features, then better will be the accuracy of the model. At first, the aim was to get started with the problem. So, I started searching for more articles for false misclassification by web scraping and then combining the true and false misclassification data and calculating the features using the most common approach which was normal feature selection i.e. removing stopwords, removing punctuation of each text, Lemmatizing and converting each word to lowercase. Using these features, I started applying different models on the same set of feature space with the Statistical Machine Learning Algorithm i.e. Bernoulli Naive Bayes Classifier, Multinomial Naive Bayes Classifier, Logistic Regression and Support Vector Machines. The accuracy of the model are as follows: Naïve Bayes 0.22 f1_score, Support Vector Machines 0.24 f1_score, Logistic Regression 0.28 f1_score.

2) Since the accuracy was too low, I decided to change the method to calculate the features. I started searching about Bag of Words. This led me to search about something called YAKE (Yet Another Keyword Extractor). It is an unsupervised automatic keyword extraction method. It helps in calculating the keywords with more importance in the sentences with their importance value. The most important feature of YAKE was that it does not need data to be cleaned. It just takes the data and start calculating features and since it is domain and language-independent, it helped in calculating way more better features. After getting the features, each text has its own sets of features and the number which tells us the importance of those words in the sentence. I applied the same three models on these set of features and the results were: Naïve Bayes 0.42 f1_score, Support Vector Machines 0.43 f1_score, Logistic Regression 0.44 f1_score.

3) To improve the model accuracy, I was encouraged to search for new methods that will improve my features. Then I got to know about TF-IDF Vectorizer. It is a model used for learning vector representations of words called “word embeddings”. Through this, I got word vectors which are fed into the model to do analysis. I performed all the above models as discussed. Then to improve the results again I performed same models but after performing hyperparameter tuning of the models on the development data then training it on train data. The accuracy achieved was: Naïve Bayes 0.49 f1_score, Support Vector Machines 0.59 f1_score, Logistic Regression 0.65 f1_score.

3) To improve the accuracy, I searched on internet on how I can combine 2 models then I found about ensemble approach and implemented it models by taking two classifiers at a time. The model accuracy achieved was:

4. Error analysis

This section discusses the challenges that were faced and the different approaches that were taken to perform the analysis.

Naive Bayes Classifier: The accuracy achieved through this model was not good because the Naive Bayes approach implicitly assumes that all attributes are mutually independent. It was not producing good results because the data became imbalanced due to categorical variable not observed in training dataset.

Support Vector Machines: Since the features are many, I used SVM. It is not affected by any outliers when detected. This is because the algorithm works on Hyperplanes concept that are affected by support vectors. The other reason why I used this algorithm was it works well with data in text files. And scales up well when we have large amount of data. The reason why SVM didn’t perform well was because of hyperplane selection which reduces efficiency and accuracy of the model.

Logistic Regression:
This algorithm works based on the probability that it assigns observations based on discrete class. This algorithm performed well as compared to others as the data was linearly separable. This algorithm overfits when we are having large dimension data.

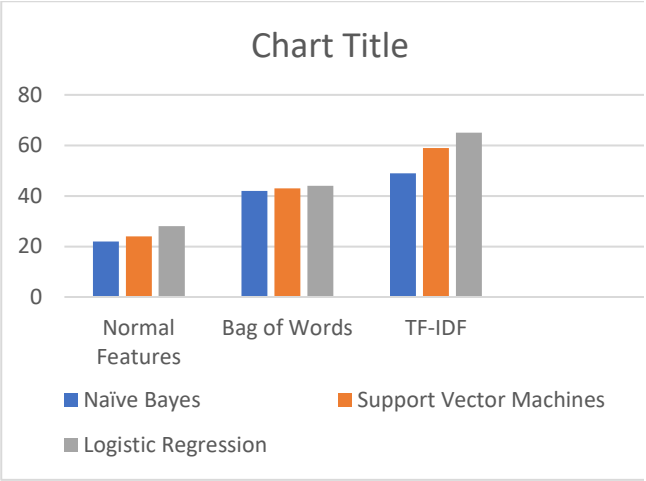

Fig1: Accuracy with different models for various features

```
Confusion Matrix :

[[13  2]
 [ 4 11]]

Performance on the positive class (documents with misinformation):
Precision = 0.7647058823529411
Recall    = 0.8666666666666667
F1        = 0.8125
```

5. Final Model Selected

The accuracy of the model depends on how well it performs on training data with the features generated. The evaluation metric for the model was the accuracy of each model. A great Machine Learning model depends on how well it performs during training of the data and as well as on the test data. After the thorough hyperparameter tuning on the best performing model on test data. We know that model performance depends upon overfitting and underfitting. Underfitting as well as overfitting can lead to poor model performances.

Overfitting happens when the evaluation of train data and test data is different. To reduce overfitting of the model either we can use a resampling technique for estimating the accuracy of the model or we can hold a validation dataset.

The most common sampling method is K-Fold cross validation. It helps us in training and testing the model k times. This is achieved by dividing the data in k different parts and taking k-1 for the test parts and rest for the training set.

6. Future Enhancements

I would like to try LSTM algorithm for future purpose for future purpose. According to me this algorithm would have performed better and would give better results on such data as “LSTM networks suits well for classifying, predicting and processing the data based on time series. Due to halts while performing this on important events between time series for unknown duration. LSTM will be developed to deal with vanishing gradient that is encountered when training with traditional RNN’s. Relative insensitivity to gap length is an advantage of LSTM over RNNs.

8. Conclusion:

I was able to arrive at the conclusion that Logistic Regression with TF-IDF features performed better. The reason behind this is given in error analysis.

9. Final Evaluation

The result I achieved earlier was 0.65 using the TF-IDF vectorizer on Hyperparameter tuned parameters but after the final evaluation the result got changed and my final f1_score decreased by 0.2 this can be due to model overfitting on the data. And other reason can be that we had to extract data from the web so the f1_score would have reduced

9. References

[1] Langin, K.: http://www.sciencemag.org. Fake news spreads faster than true news on Twitter—thanks to people, not bots (2018).

[2] https://www.kaggle.com/c/learn-ai-bbc/data for data collection

[3] https://www.kaggle.com/shineucc/bbc-news-dataset

[4] https://www.kaggle.com/bbose71/bbc-news-classification