COMP90042: Natural Language Processing Project 2020

CLIMATE CHANGE MISINFORMATION DETECTION

Gaurang Sharma
1041953

1 Introduction

With the advent of internet, followed by social media, the world has come closer and has bridged gaps in many aspects. The internet is one of the most powerful resource in today's world as it has contributed largely in building a strong knowledge base for every user through access to a wealth of information. The internet has revolutionized the lives of people in many ways as it is a key source of news and information, being used at a very large scale. But every story has a dark side to it. The internet, being easily accessible, has recently been overly abused to spread lies and misinformation. According to a study, there has been an exponential rise in the number of internet violators who have contributed in propagating false claims and misinformation over the social media. This illegitimate use of the internet has grown to such an extent that sometimes it becomes impossible to judge the reliability of a piece of news on the internet.

Having said that, over the past few years, there have been many sources on the internet which have given rise to false information on climate change. Climate Change denial is dangerous for the society as it refutes scientific evidences and facts, thereby misleading a large section of the people. This paper discusses about the design and implementation of a machine learning system, which aims at detecting whether a piece of information on climate change is misinformed or not.

2 DataSet

The data comprises of a large corpus of news articles from various unidentified sources. This dataset provided is as follows:

`train.json`	a set of 1168 training documents
`dev.json`	a set of development of documents with 100 articles
`test.json`	a set of test documents with 1410 articles

Table 1: Dataset

3 Problem Overview

The primary aim of the project lies in the design of a system that classifies a document as containing climate change misinformation or not. It could be thought of as a binary classification problem which can be trivialised by using underlying machine learning algorithms, but the peculiarity lies in the fact that the training dataset contains documents with only positive labels (*label 1*), i.e., documents containing only misinformation. There are different approaches that can be channelized to use for this project like which have been discussed further in the report.

4 Methodology

One-Class Classification algorithms are the first thing which would come up in mind for projects like this, but these algorithms would only be suitable if it's practically impossible to collect negative samples. For any machine learning task to be effective enough, where the data turns out to be unbalanced as in this case, the aim should be to collect some reliable negative samples which would be a true representation of the provided dataset as a whole. After a careful review of the the training data, a query for some genuine news articles was generated over the internet. A publicly available dataset on *Kaggle* comprising of a large number of news articles from media houses like *NewYork Times, WallStreetJournal, The Guardian* (to name a few) was finally downloaded to expand the training dataset.

This external data consisted documents with varying topics, ranging from sports and politics to health and science. The primary goal was to train a model which could classify a climate change misinformed article against any article(regardless of being related to climate and science). The next step would be to cleanse the data to make it more refined before vectorizing it into appropriate features. Finally, different approaches and techniques would be implemented to build the model and based on their performance on the development set, the best set of models would be used to predict the labels for the test set.

5 Design and Implementation

In this section, a detailed and analytical explanation of the techniques used for this project has been discussed.

5.1 Data Transformation

Transformation of data into a suitable format plays a significant role in enhancing the performance of supervised learning models. It is quite evident from past experiences that data from unreliable sources often produce erroneous results. Therefore, data collection and hence cleaning the data are the foremost important steps in building any classification system. The previous section talked about the collection of the external data (*negatively labelled*), and this section would discuss about the various techniques to preprocess the data.

5.1.1 External Data Creation

Since the corpus of documents downloaded for our external data was too large ($\approx$100,000), it had to be stripped in order to filter out the unwanted anomalies in the data. This is done by categorizing the external data into two parts, one of which contains factual information about climate change and related news, while the other set contains genuine news articles from other varying topics.

This is achieved by *word presence filteration techinque*, wherein the entire list of documents is parsed and only those documents are returned which contain certain words. The word set that is created to carry out the filteration contains 2 bi-grams: `['climate change', 'global warming']`. The rationale behind choosing these two phrases is rather straightforward as majority of the documents containing these phrases would be gravely related to information about climate. This technique yields about 900 articles which would be used as part of the training data. Finally, around 300 articles are selected at random from the original corpus without applying any filteration, which would contribute in making up the second part(news articles from various topics) of the final external data to be used. This leads to creation of an external data set consisting of negatively labelled documents, to be combined with the provided training data, which would finally be used to build the supervised models.

5.2 Data Preprocessing

Once the training data set is prepared with a good proportion of both positive and negative samples, there comes the task of standardizing it for making it ready for machine learning algorithms by removing noisy elements.

5.2.1 Tokenization

The list of documents are parsed and every article is processed to split into individual words called tokens. The process mainly involves lowercasing and punctaution removal from the text corpora. Here, the `gensim.utils.simple_preprocess()` function from the *Gensim* library of Python is used to perform the respective task.

5.2.2 Stopwords Removal

Tokenization is followed by the filtering of some common 'function' words amongst all the documents which are highly unlikely to give any significant pattern for the classification system. These set of words are called *Stopwords* and are removed using the *NLTK* library of Python which has an in-built set of English stopwords.

5.2.3 Lemmatization and POS Tagging

Normalization is a technique which which converts a set of words into their root words. The two most widely used algorithmic methods to perform normalization are *Stemming* and *Lemmatization.* Stemming involves the removal of suffixes and affixes of words depending on the root words, while lemmatization involves the morphological analysis of words and returning the dictionary form of a word called lemma. Lemmatization, being an intelligent process and always returning valid words, becomes the first choice of normalization for this project. Moreover, lemmatization is enhanced by preceding it with *Part-Of-Speech tagging*, which disambiguates the word by giving it a proper POS tag based on its context and definition.

5.3 Feature Extraction

An essential technique after the data gets preprocessed is to reduce dimensionality by extracting important features from the corpora, and encoding the tokens into numerical values to be used as an input for the machine learning algorithms. For this project, the term frequency-inverse document frequency (TF-IDF) technique has been used as it is very useful in highlighting important words. This technique gives a higher weightage to the important term, giving a value of 1 to the relevant terms and 0 to the contextually unimportant words. For calculating the TF-IDF values the document is converted into inverted text file wherein for all documents words are extracted and corresponding to each word a weight is assigned i.e. the term occurrence value and document occurrences using TF-IDF value is calculated.

5.4 Model-Building Architecture

In this section, an overview of the various supervised learning models has been provided upon which the training data is fit. The classifier algorithms used for this project are discussed below.

5.4.1 Naive Bayes Classifier

A probabilistic machine learning algorithm, Naive Bayes technique is purely based on the *Bayes Theorem.* For this particular task, the Multinomial Naive Bayes classifier is used. These algorithms are fast and simple to process, but their only drawback lies in their assumption of predictors being independent.

5.4.2 K Nearest Neighbors

This is one of the simplest yet effective supervised learning algorithm which classifies a document depending on the classes of the K nearest neighbours of the unseen data point. It is a non-parametric classification method and works significantly well with a finite amount of data.

5.4.3 Support Vector Machine

Support Vector machines are supervised learning models which make use of decision boundaries, also called hyperplanes, which help in classifying the data points. So the primary objective of this algorithm is to figure out a *Hyperplane* in an *n-dimensional space* that would distinctly predict the labels for the unseen data points.

5.4.4 Logistic Regression

This classifier is based on the underlying principles of statistical learning, and is a special case of classical linear regression where the target variable is categorical in nature. Being a probabilistic method, in due course it will be seen how this statistical classifier turns out be a significant performer for this particular project.

5.4.5 Random Forests

The random decision forests or random forests are an object learning technique for used for classification. It creates a multitude of decision trees during training of the data. It then gives the output class that is the classification of individual trees.

5.4.6 Stacking Classifier

Taking motivation from a rather baseline ensemble learning technique like Random Forest, a proper stacked classifier is used as one of the ensemble learning techniques for combining a couple or more of basic level-one classifiers and then finally training a meta classifier using the new features obtained from the multiple stacked estimators. Ensemble learning methods have always been essential in producing significant results in classification problems and this has been discussed further in the report.

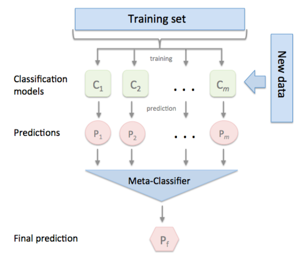

Figure 1: Stacking Classifier Illustration.

6 Experimentation Results

The training data set is made to fit on the above specified classifiers and after a careful optimization of the hyper-parameters for each of the classification systems, this section discusses the analysis of results for all the models that have been trained. The results have been obtained by predicting the labels for the development set and then evaluating the results against the true labels. The metrics that have been used for analyzing the results are:

Accuracy	Fraction of correct responses among total no. of documents.
Precision	Ratio of correctly predicted labels to the total predicted positive labels.
Recall	Ratio of correctly predicted positive labels to all the observations in actual class.
F1 Score	Weighted average of Precision and Recall

Table 2: Classifier Performance Measures

The main metric for evaluating the performance of the models in this project is *F1-Score* as it is a better metric for imbalanced class labels by considering the distribution of the data set.

The chart below summarizes the performance of the classifiers on the development set, followed by an analysis on the same.

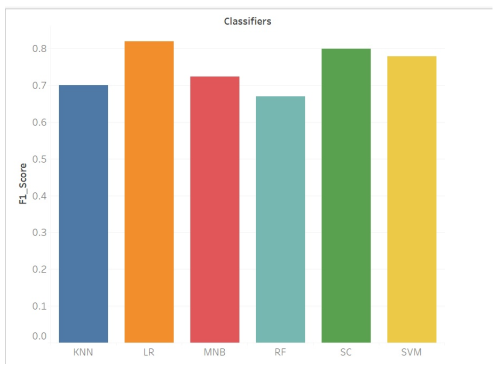

Figure 2: Performance Evaluation of Classifiers.

Figure 2 gives an overview of the performance of the models used. These F1-scores were produced after fine-tuning the hyper-parameters for every model, which helped in slight improvement for some of the models. K-Nearest Neighbors and Naive Bayes techniques come out as random baseline models, while Support Vector Machines enhances the performance with a better F1-score. But Logistic Regression, after a great deal of hyper-parameter optimization, outshines as the leading performer over the development set with a score of *0.81*. Bagging ensemble methods like Random Forest doesn't seem to impress much, with a relatively low score; but the Stacked Classifier, using Random Forest and Logistic regression as its base classifiers, enhances the performance with a good score(*0.80*) approximately at par with the Logistic Regression. The next section deals with a thorough error analysis of the models and their performance, along with any improvisation techniques that could possibly better the results for the system.

7 Error Analysis and Improvisation

Analyzing the errors on the training and development set after fitting the classification models has proved to be a beneficial practice in improving the performance on the unlabelled/unseen test sets. In this section, a thorough analysis of the development set is discussed along with the measures that have been taken to improve the performance of the overall system.

In the previous section, it is observed that Logistic Regression and Stacked Classifier happen to produce the best results, with most of the responses being overlapped. To improve upon these results, a deep inspection is carried out on the misclassified documents on the development set. In other words, the wrongly predicted documents are scrutinized in order to gain any insight into the functioning of the system. The steps taken to perform these measures are discussed below:

- Initially, out of the wrongly predicted documents from the development set, two separate lists were formed. One of the list contained documents that were wrongly classified as *label 1* and the other list naturally had the documents which were falsely classified as *label 0*. Approximately 75% of the documents lied in the former category of lists while the rest resided in the latter.

- A thorough study on the two lists helps in fetching out some common patterns amongst the errors committed by the model. The most common occurring words in both the lists were extracted to give a deeper understanding of the data being wrongly predicted. For example, the majority of the documents being falsely classified as label 1 talked a lot about *government of australia* and *scientific research and climate policies*. These sort of commonalities in both the lists of documents helps us in further improvising the classification system. Figure 3 and Figure 4 below illustrates the word clouds for both the lists.

- Having looked for some commonalities in both the lists, a small amount of relevant addition to the external data set was thought to be the correct approach. Consequently, approximately 50 documents were added to the external data set. A total of 32 articles representing factual information were scraped from the web through media houses like The Guardian and ABC News Network. On the other hand, after a lot of web surfing, some unreliable sources of information, propagating false news

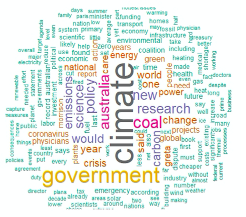

Figure 3: Wrongly classified as label 1

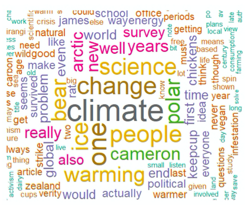

Figure 4: Wrongly classified as label 0

were found on https://mediabiasfactcheck.com/conspiracy/. From the long list of websites, a couple of them were selected to scrape news articles that supported *climate change denial*. A total of 16 relevant pieces of text were selected from https://beforeitsnews.com/ and https://climatechangedispatch.com/. These newly found documents were added to the external data set, which after combining with the training data set rejuvenated the distribution of the dataset as a whole.

- The new training data was made to undergo the same preprocessing steps as defined earlier in the report. The TF-IDF vectorizer was used to create the features, and the same pair of best models from the previous evaluation were fitted on this data. A graphical illustration depicting the comparison of the performances is given below:

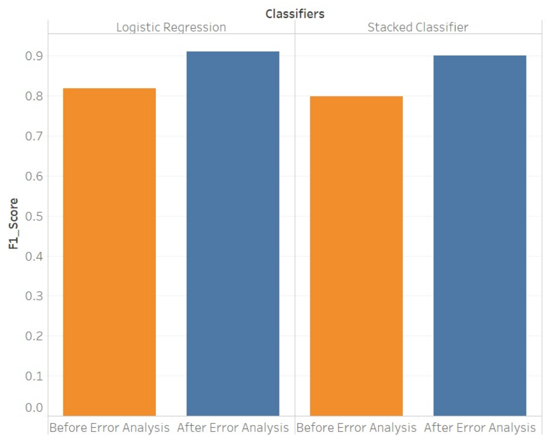

Figure 5: Performance Comparison Before and After Error Analysis.

A substantial improvement in the results is observed after the scrutinization of the errors followed by some data manipulation and a slight hyper-parameter tuning. The smallest yet vital ingredients for enhancement of any classification model lies in the data and this is pretty evident from the essential growth of our classier.

8 Conclusion

In this paper, a binary classification system is designed to predict the authenticity of climate change related information. The various techniques involved in data pre-processing helps in creating meaningful vector space models using the TF-IDF concept. After the features are fed into the supervised classifiers, this paper goes on to discuss about the results produced by each one of them, and thus help distinguish the baseline models from the significant ones. Once the best model(s)(*Logistic Regression and Stacked Classifier*) are selected after fine-tuning the hyper parameters, the same model is used to predict the unseen test data. Though Logistic Regression has a very slight improvement($\approx$0.1) over Stacked Classifier, yet the latter is used as the final classifier for the test data as it has a better recall and using Logistic Regression as a final estimator under the Stack Classifier makes it justifiable to use it as the final classification system.

A substantial growth in the output is observed after the error analysis, as discussed earlier in the paper. Optimizing the hyper-parameters again at this stage do not show any effective changes in the performance, so the same set of hyper-parameters are used in this final model. The classifier is then used to predict the unseen test data and the performance is assessed after submitting the results on the *Codalab Competition* page, https://competitions.codalab.org/competitions/24205?secret_key=37c3468d-1b4f-4a10-ac60-38bba553d8ee. The F1-scores for the same are given below:

Ongoing Evaluation	0.71
Final Evaluation	0.68

Table 3: Codalab Performance on Test Data

The results for the 'Ongoing Evaluation' were submitted during the competition, while the score for the 'Final Evaluation' was evaluated using a subset of the test data. Without any large difference between the two, the model tends to perform significantly well on the test data, with a recall of 0.87. This metric is suggestive of the fact that the classifier has a good rate in correctly predicting the negatively labelled (*label 0*) documents. The misinformed articles are majorly the ones being wrongly predicted, as depicted by a low precision.

Therefore, taking note of all the results and assessing the performance of the models, there are still some unexplored areas and techniques which, if applied, could have had made impactful changes in the performance of our system. Using some semantically-driven features called *Word Embeddings*, and training them on different layered neural networks would have proved vital in enhancing the system. Selecting features efficiently using dimensionality-reduction techniques is another area which could be applied while creating the vector space models. Be it data transformation, or application of deep learning models, a lot of interesting approaches could have had been applied for this project had time permitted. A deeper scrutiny of the dataset and a relative distribution amongst the class labels in the train data would have surely boosted the performance of the classifier.

9 Bibliography

1. Waykole, R. and Thakare, A., 2018. A Review of Feature Extraction Methods For Tex Classification. International Journal of Advance Engineering and Research Development, 05(04).

2. Kalra, V. and Aggarwal, D., 2018. Importance of Text Data Preprocessing Implementation in RapidMiner.

3. Chen, X. and Zou, L., 2018. Detecting Climate Change Deniers on Twitter Using a Deep Neural Network.