Project: Fake News Detection using Natural Language Processing

Shubham Pradhan
2017eeb1168
Harsh Nara
2017eeb1143
Indian Institute of Technology,
Ropar, Punjab, India

Abstract

How might Natural Language Processing better inform us about how fake news filters get created? This case study attempts to distinguish between fake news and absurd news by building a classification model. We believe it's imperative to maintain a high level of accuracy when creating a fake news detector–where all fake news gets filtered out and authentic news will not be affected by a filter. The main objective is to detect the fake news, which is a classic text classification problem with a straight forward proposition. It is needed to build a model that can differentiate between "Real" news and "Fake" news.

1 Introduction

In this project we are going to use NLP. Natural Language Processing, usually shortened as NLP, is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the human languages in a manner that is valuable. Most NLP techniques rely on machine learning to derive meaning from human languages.

NLP entails applying algorithms to identify and extract the natural language rules such that the unstructured language data is converted into a form that computers can understand. When the text has been provided, the computer will utilize algorithms to extract meaning associated with every sentence and collect the essential data from them. Sometimes, the computer may fail to understand the meaning of a sentence well, leading to obscure results.

2 Literature Review

2.1 Data Cleaning

The data set which we use first requires cleaning. We first drop all the columns such as index and other useless columns which do not contribute in the classification and only kept the relevant columns. Then we make sure that every word in the news is lowercase if not then convert it to lower case because methods of classification may be case sensitive. Then we remove all non alphabet characters such as punctuation and numbers.

2.2 Lemmatization

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meaning to one word. It is used to decrease the number of predictor variables so as to reduce overfitting.

2.3 Binarizing Labels

Next step is to binarize the labels. We gave all the real news label 1 and all the fake news label 0. This is done so that our model can have the mathematical class and do the classification.

2.4 Text data Vectorization

2.4.1 Bag of Words

It is basic model used in natural language processing. Why it is called bag of words because any order of the words in the document is discarded it only tells us weather word is present in the document or not. Here each news is separate document, we make list of the word such that one word should occur only once.

2.4.2 CountVectorizer

So how a word can be converted to vector can be understood by simple word count example where we count occurrence of word in a document w.r.t list. By using CountVectorizer function we can convert text document to matrix of word count. Matrix which is produced here is sparse matrix. After applying the CountVectorizer we can map each word to feature indices.

2.4.3 TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency which basically tells importance of the word in the corpus or dataset. TF-IDF contain two concept Term Frequency(TF) and Inverse Document Frequency(IDF).

Term Frequency is defined as how frequently the word appear in the document or corpus. As each sentence is not the same length so it may be possible a word appears in long sentence occur more time as compared to word appear in shorter sentence.

Inverse Document frequency is another concept which is used for finding out importance of the word. It is based on the fact that less frequent words are more informative and important.

TF-IDF is basically a multiplication between TF table and IDF table . It basically reduces values of common word that are used in different document.

2.5 Logistic Regression

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values). Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Unlike linear regression which outputs continuous number values, logistic regression transforms its output using the logistic sigmoid function to return a probability value which can then be mapped to two or more discrete classes.

2.6 Naive Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem. Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

2.7 Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

3 Data Set

The first task to build a model to detect the fake news detector is to find the proper dataset for the model to train on. We found the dataset containing the labeled news. It have 6335 news articles with each of them labeled as either "Fake" or "Real". Dataset was taken from the kaggle.

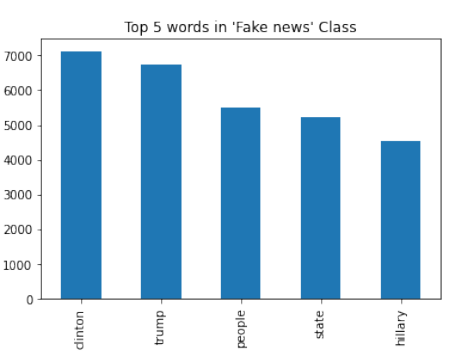
4 Data Exploration and Visualization

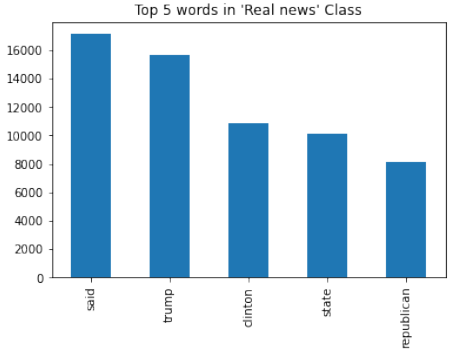
Exploratory Data Analysi refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. We plotted the top five word which occurs in each class. We claculated the percentage weight of each class in the data set. For proper training of the model it is important for the dataset to contain the comparable amount of datapoints from each of the classes.

Fake News	Real News
50.67%	49.33%

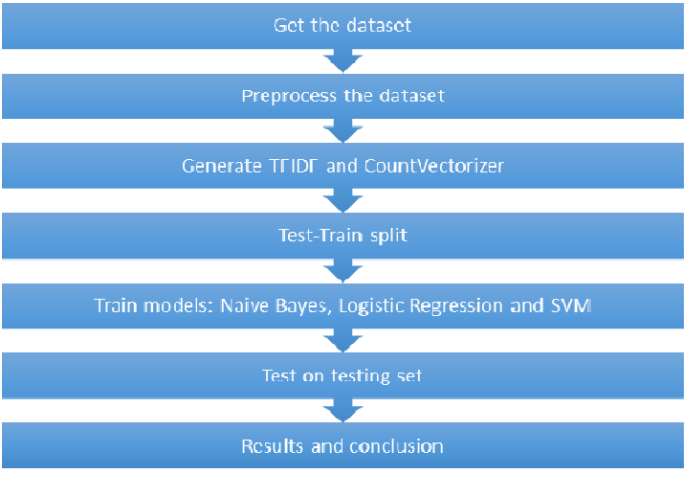
Class Distribution

By using the Bag of Words and count vectorizer we also found the the top five words which occured in both "Fake news" class and "Real news" class. Plots are shown below





5 Model Pipeline



6 Data preprocessing

Now working with the text data can be tricky as our model cannot understand the natural language, so we have to convert the data to numerical form but before that we need to clean and preprocess the data.

First we'll convert all text to lowercase and strip all punctuation. We'll also convert any numbers to the word 'num'. Since this can often be more informative than the numbers themselves, which can vary quite a lot. Then remove all the duplicate rows.

Now we have articles that are a bit easier to work with, though they still contain a lot of whitespace. They also contain a lot of nuisance words that do not add to the analysis, such as the, to, and. To get around this we'll split the document at any whitespace into a list of words. We'll then remove any words flagged as 'nuisance words'.

Finally we apply something called lemmetizing. Many words with similar semantic meanings have different endings depending on the context.

7 Converting to Numeric Data

Now that we've done all the preprocessing required, we'll convert the data to numeric data. The bag-of-words model just counts the number of times a word appears in each document. We used two methods to get the vectors from the text and these are CountVectorizer and TFIDF Vectorizer. We implemented both the techniques and implemented different classifying algorithms on each of the vectorizer to get the best result.

8 Pipeline

In this step we used the pipeline function in sklearn to get the most optimum parameters for each our algorithms and vectorizer to get the best result they can provide. The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters. We have 6 combinations of algorithms which are:

- Naive Bayes with CountVectorizer
- Logistic Regression with CountVectorizer
- Support Vector Machine with CountVectorizer
- Naive Bayes with TFIDF Vectorizer
- Logistic Regression with TFIDF Vectorizer
- Support Vector Machine with TFIDF Vectorizer

9 Applying Machine Learning algorithms

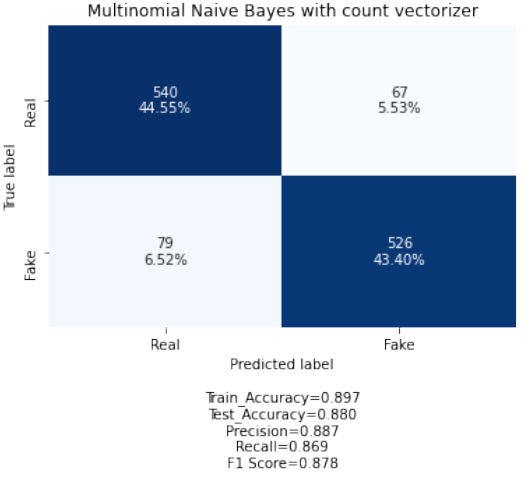
Each of the above algorithm is implemented with the optimum parameter obtained by the pipeline function. The Vetcorized data is first split into train(0.8) test(0.2) set and then machine learning algorithm is trained on the training set and then tested on the testing set.

9.1 Model application

We implemented all the six models. To implement we had to reduce our feature space dimension since in pipeline step it was observed that the best results were obtained with 3 n_grams that is considering 3 words at a time. This gave us lots of feature which could cause overfitting this we reduced our feature space to 8000. It was observed that above 8000 almost every model showed overfitting. The classification report for all of the six combination are shown.

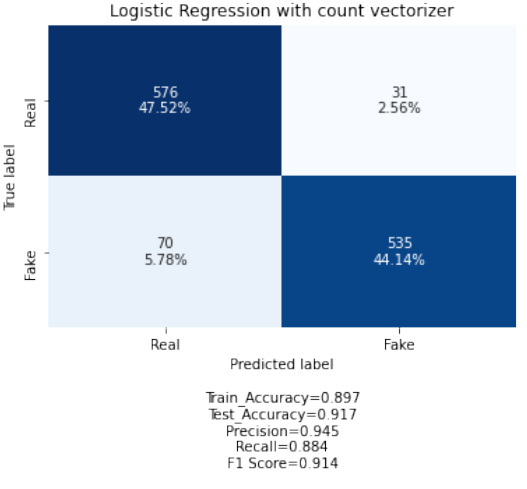
9.1.1 Naive Bayes with CountVectorizer

This combination was the fastest among all the algorithms. This is due to the fast approach of Naive Bayes and on top of that CountVectorizer is also the fast model. This model has the little overfitting compared to other as it can be seen that train accuracy is more than test accuracy. Also it have the worst F1 score.



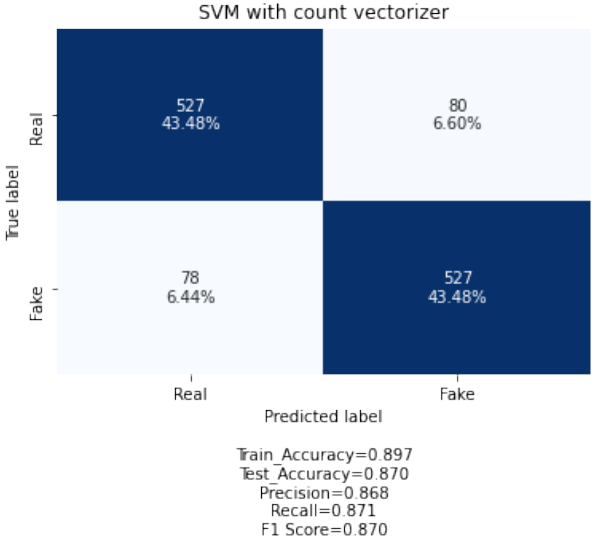
9.1.2 Logistic Regression with CountVectorizer

This combination was slower than the above combination but still faster than rest of the combination. This model shows no overfitting. It has the highest precision among all the other models.



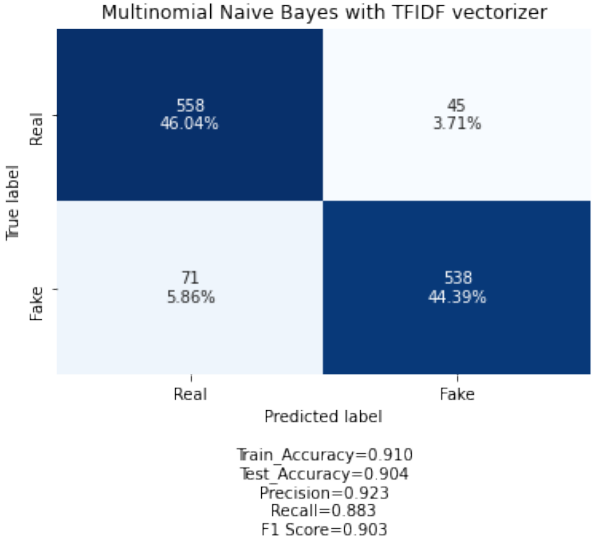
9.1.3 Support Vector Machine with CountVectorizer

It was observed that the SVM is was the slowest among all the three algorithms. This combination also showed little overfitting. It also have the lowest precision and Recall.



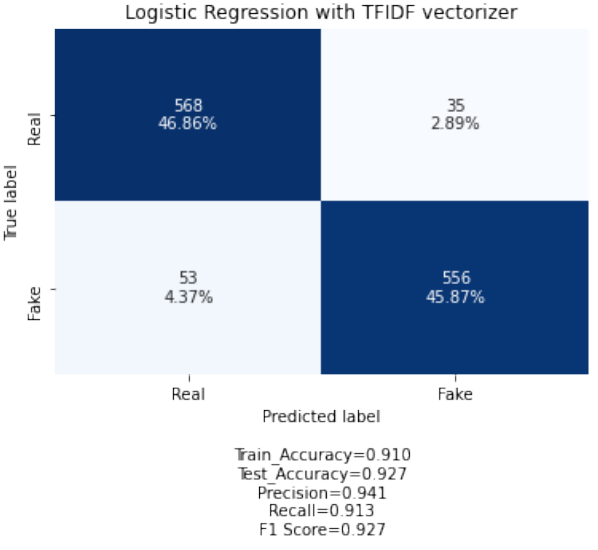
9.1.4 Naive Bayes with TFIDF Vectorizer

This model showed very little to no overfitting and good precision but very less F1 score.



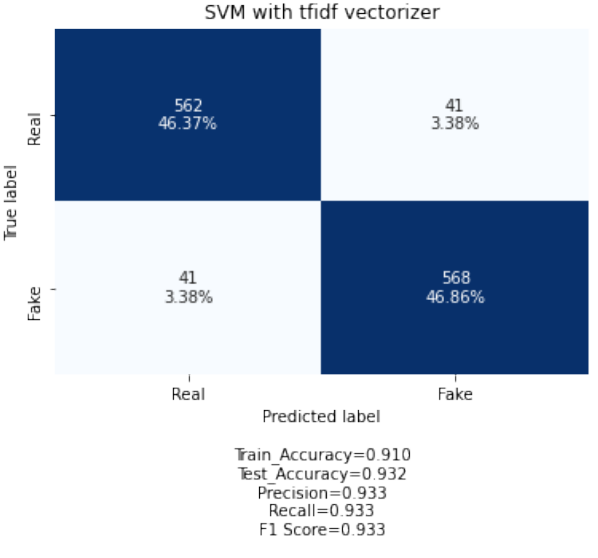
9.1.5 Logistic Regression with TFIDF Vectorizer

This model showed no overfitting and have the second highest precision.



9.1.6 Support Vector Machine with TFIDF Vectorizer

This model is the slowest among all other models. It gave the best results with highest F1 score and Recall. Also the Test accuracy was highest in this model.



9.2 Model comparision

Here we can compare the performance of each of the model with different metric.

Model	Train accuracy	Test accuracy	Precision	F1 Score	Recall
Naive Bayes with CountVectorizer	89.7%	88%	88.7%	86.9%	87.8%
Logistic Regressio n with CountVectorizer	89.7%	91.7%	94.5%	88.4%	91.4%
SVM with CountVectorizer	89.7%	87.0%	86.8%	87.1%	87.0%
Naive Bayes with TFIDF Vectorizer	91.0%	90.4%	92.3%	86.3%	90.3%
Logistic Regression with TFIDF Vectorizer	91.0%	92.7%	94.1%	91.3%	92.7%
SVM with TFIDF Vectorizer	91.0%	93.2%	93.3%	93.3%	93.3%

Scores of each model

10 Observations

The following observations were made:

- It was observed that each algorithm perform comparatively better with TFIDF Vectorizer than CountVectorizer. This may be due to the reason that In TfidfVectorizer we consider overall document weightage of a word. It helps us in dealing with most frequent words. Using it we can penalize them. TfidfVectorizer weights the word counts by a measure of how often they appear in the documents but in CountVectorizer we only count the number of times a word appears in the document which results in biasing in favour of most frequent words. this ends up in ignoring rare words which could have helped is in processing our data more efficiently.
- Worst result were shown by the SVM with CountVectorizer. This may be because the features obtained may not be linearly seperable as we used linear kernel in this case. SVM may not have been able to create a good hyperplane for seperation.
- Naive bayes algorithm is the fastes among all the three algorithms and the SVM is the slowest. This is because Naive Bayes just need to calculate the joint probability distribution and no optimization takes place but in SVM optimization of hyperplane and even the distance between hyperplane and each class is done.
- TFIDF was also slower compared to CountVectorizer because in Countvectorizer we just need to count the number of words but in TFIDF we have to calculate the word frequency and document frequency.
- Naive Bayes performed worse among all since because it assumes all the features to be independent but the classification of news is dependent on the order of the words. Although other algortihms also dont consider the ordering but still Naive Bayes showed the worst performance. The best perfrmance was shown by the SVM with TFIDF.

11 Conclusion

In this project we created the various machine learning model to classify the news as "Fake" or "Real". We implemented 6 different models optimized each model for best result. We implemented different Natural Language Processing techniques to prepare the data for the algorithms .We compared the result of each model and found that SVM with TFIDF gives the best result among all.