Fake News Source Detection using NLP Techniques

Author – Kunwarvir Singh

Problem Overview

Fake news refers to misinformation in the country which is spread through word of mouth and traditional media and more recently through digital forms of communication such as edited videos, memes, unverified advertisements and social media propagated rumours. Fake news spread through social media in the country has become a serious problem, with the potential of it resulting in mob violence, and hence it is necessary to stop the spread of such misinformation.

Besides Fake News, the sources responsible for spreading Fake News are much more dangerous. Hence such sources must be detected and prevent spread of Fake News. Ultimately Eliminating the source could be very helpful in preventing spread of Fake News.

Solution Overview

Using the techniques of machine learning and Natural Language Processing we have developed a machine learning model that could detect the source of News provided it is Fake/Real. Many machine learning systems were developed but following reasons would prove to be advantageous over machine learning models created so far: -

i. This machine learning model is designed based on Indian News. Until now very few machine learning models have been made that could make predictions on Indian News. Most of the Models are based on US News.
ii. The dataset used contains articles from over 900 Sources.
iii. Very few or no machine learning models have been built so far that could detect the source of Fake News.
iv. The news collected are not concentrated on one single field.

The details of the Project are as follows: -

Platform – Jupyter Notebooks

Language – Python 3.7

Important Libraries Used – NumPy, Matplotlib, Pandas, Scikit Learn, Natural Language Toolkit (nltk), Regular Expression (re)

Project Walkthrough

Dataset

The dataset used in designing the machine learning model is a self-prepared dataset. List of Fake and Real News along with their sources have been collected from 'Inshorts' - a mobile app that offers the latest news stories summarised in 60 words. The dataset consists of 58937 News Articles from over 900 (correction) sources including both Real and Fake News during the period 2016-18. The exact figures of entire dataset are mentioned below: -

- Dataset is divided into two CSV Files: Train_Set.csv and Test_Set.csv
- Train_Set.csv: Consists of 46387 Data Entries
- Test_Set.csv: Consists of 12550 Data Entries
- Each data entry is composed of 5 Variables:
  - *Headlines* (String) – News Headline
  - *Summary* (String) – Summary of the News Story
  - *Sources* (String) – Source of the News
  - *Date* (Integer dd/mm/yyyy Format) – Date of Publish
  - *Fake/Real* (String) – Whether the news is Fake/Real

In the above dataset we have to predict the Source of the News so '*Sources*' will the dependent variable and rest all others will be independent variable. Since '*Date*' is a non-essential feature so we shall not include it while training our model.

Implementation

The machine learning model is implemented using several NLP techniques and trained using Multiclass Classification. The details of all the steps involved in designing the machine learning model are mentioned below: -

Step 1 – Importing the Dataset

The dataset whose details are mentioned above are imported using '*Pandas*' library.

Step 2 – Combining Important Features

The variables '*Headline*', '*Summary*', '*Fake/Real*' are combined as a single string and the vector is named as '*total*'.

Step 3 – Applying NLP Techniques

- Importing Natural Language toolkit library (nltk) for cleaning the text. The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
- Downloading Package for tokenizing the sentence "*Punkt*"- It divides a text into a list of sentences, by using an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences.\
- Importing *Stopwords* - Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence.

Creating Lemmatizer (using nltk library)

Lemmatization - Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Basically, Lemmatization is the process of converting a word into its proper root word. Below example would illustrate what exactly is lemmatization.

original_word	lemmatized_word
trouble	trouble
troubling	trouble
troubled	trouble
troubles	trouble

There is always a confusion between stemming and Lemmatization. The difference is that stemmed word might not be an actual word whereas, lemmatized word is an actual language word. Stemming follows an algorithm with steps to perform on the words which makes it faster but may result in undesirable results. Illustration below clearly depicts the difference between Stemming and Lemmatization.

Applying Text Cleaning techniques into the Training Dataset – The '*total*' variable created needs to be cleaned before undergoing training. Important steps involved are as follows: -

- Creating a lemmatizer that will lemmatize the training dataset.
- Removing Punctuation from the Text Data using *Regular Expression library (re).*
- Tokenizing all the sentences using the tokenizer created using *nltk* library.
- Removing the stopwords that do not change the meaning of the sentence even when they are removed. This makes training easier.
- Finally, cleaned text is lemmatized and is ready for training.

Step 4 – Encoding the Categorical Data

The dependent variable that has to be predicted is in the form of string and hence needs to be encoded for pre-processing and training. Also, it has to be further decoded back to obtain the desired results. For this I have used**. factorize *()*** method for encoding. The code involved in encoding the dependent variable is available in the attached code file.

The encoded values of *Sources* are named as *Source ID.* Finally, we have the pre-processed training data with variables *'total', 'Sources', 'Source ID'.*

```
train.head()
```

	total	Sources	Source_ID
0	bungalow private island sale 46 crore a bunga...	Bloomberg	0
1	americans react bollywood song pinga a video ...	YouTube	1
2	world toilet day celebrated november 19 the u...	United Nations	2
3	over 157 lakh people affected assam flood ove...	Northeast Today	3
4	varun gandhi unhurt stage collapse bjp mp var...	India Today	4

Pre-Processed Trained Dataset

Step 5 – Encoding 'total' variable for Training (using Tf-Idf Vectorizer)

The '*total*' is still in the form of array of strings and it needs to be vectorized before fitting the training dataset into the training model. For this we are going to use special technique known as ***Tf-Idf Vectorizer***. TF-IDF is a short term for the term frequency-inverse document frequency formula that aims to define the importance of a keyword or phrase within a document or a web page.

- **Term Frequency (TF),** as implied by the name, indicates the frequency of a specific term within a document. It can be calculated by dividing the number of times the keyword is used in a document by the total number of words within the same document
- **Inverse document frequency (IDF)** shows how frequently a keyword is used within a collection of documents. This is a corrective metric that aims to level down the importance of articles and various function words and to give more prominence to meaningful words.
- The product of these two metrics is the TF-IDF formula that indicates the relevance of a keyword to the document. This formula is widely used by search engines to determine the corpus of web-pages that are relevant to a search query. The larger is the TF-IDF value - the more relevant (important) is the keyword to the document.

Reason for Using Tf-idf Vectorizer

The main idea behind creating a vectorizer is to create *Bag of words* so as to know the frequency of each word in the cleaned text and further generate an array of *Bag of Words* based on Position and Frequency. We could also use *Count Vectorizer* for this purpose but the main problem with *Count Vectorizer* is that it simply counts the words without given importance to words occurring in lower frequency. Such problem is solved by *Tf-idf vectorizer.* Let's explain the problem from one example.

Cleaned Text = "small businesses stopped due global pandemic", "small business may get affected"

- Count Vectorizer – gives equal importance to the words having same frequency.
- Tf-Idf Vectorizer – gives importance to words having less frequency like "*global*", "*pandemic*". Thus such words tell much more about what the text is about.

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Formula used by Tf-Idf Vectorizer to calculate Term-Frequency of words

Step 6 – Splitting the dataset into Sub-Train set and Sub-Test set.

In order to evaluate the performance of our model Main Train Dataset is divided into train dataset and test dataset.

- Independent variable (X) = *'total'*
- Dependent variable (y) = '*sources*'

Step 7 – Converting X_train to trainable form using Tf-Idf Vectorizer

The Tf-Idf vectorizer is used to convert X_train into X_train_tfidf so that it can fit and trained on the classification model.

Step 8 – Final Step – Fitting the features into the classification model

Since we have only one feature as independent variable so we will use ***Linear SVC (Support Vector Classifier)*** for classification. The vectorized training set (X_train_tfidf) and encoded sources (y_train) is fit into the classification model. Using Metrics package of Scikit Learn the final Accuracy Score and Classification Report is generated so as to evaluate the performance of model.

Final Classification Report

The scores generated are listed as follows: -

Score Type	*Score (Out of 1)*
Accuracy	0.346
Precision (Weighted Avg)	0.46
Recall (Weighted Avg)	0.47
F1 Score (Weighted Avg)	0.46

*The classification report for every 'Source' is available in the Code file.

Second Approach

The second way that I employed for making the machine learning model was very much similar to the first approach. The significant steps used in the approach are as follows: -

- '*Fake/Real*' Variable was converted to integer as '*1/0*'
- '*Fake/Real'* Variable was not included in the '*total*' Variable. 'total' variable only consisted of '*Headlines*' and '*Summary*'
- Similar steps were employed for Pre-Processing the '*total*' variable.
- After pre-processing the '*total*' variable, the '*Fake/Real*. Variable was appended with the '*total*' variable as one separate column using **DataFrameMapper** function. Thus, we have now two independent variables '*total*' and '*Fake/Real*' that will be used to predict the '*Sources*' Variable.

Code Involved

```
mapper = DataFrameMapper([('total', TfidfVectorizer()),
                          ('1/0', None)])
features = mapper.fit_transform(train) //X(Independent Variable)
categories = train['Sources'] // y(Dependent Variable)
```

- Rest all steps were similar to those used in First Approach

Demerit – After training the model using two features to predict 'Sources' the accuracy obtained was lower (27%) than the one obtained in the first approach. Hence this approach is not considered.

Conclusion-The reasons for low scores obtained in the final report needs to be improved. The following reasons may be accountable for low scores: -

- The dataset may contain undesirable text/words that gives poor results.
- Since it is a self-prepared dataset there might be uneven distribution of Sources producing Fake News and Sources producing real News.
- The '*Fake/Real*' variable was combined with the '*Headlines*' and '*Summary*' as Strings which could be one of the main causes of inaccurate results.

The model is still under the process of further improvements and shall be upgraded on the basis of accuracy and efficiency. I hope this model could be useful as a stepping stone to prevent the spread of fake news.

****