FAKE NEWS DETECTOR USING PYTHON AND MACHINE LEARNING DOCUMENTATION

By Nurudeen. R.A

11/12/2021

Introduction

Fake news is a kind of yellow journalism that incorporates pieces of news that may be hoaxes and is widely disseminated via social media and other internet platforms. This is frequently done to promote or enforce specific views, and it is frequently accomplished through political agendas. The assertions made in such news articles may be inaccurate or overstated. This documentation describes the formulation and testing of a fake news detector algorithm using Python and Machine Learning. The ML method used is the TfidfVectorizer and the PassiveAggresiveClassifier.

Definition of terms

Term Frequency (TF): The Term Frequency is the number of times a term appears in a document. When a phrase appears more frequently than others, a greater score indicates that the document is a good match when the term is part of the search criteria.

IDF (Inverse Document Frequency): Words that appear often in one document but not in others may be irrelevant. The IDF is a metric for determining how important a phrase is over the whole corpus. The TfidfVectorizer transforms a set of raw documents into a TF-IDF feature matrix.

Passive Aggressive algorithms are online learning algorithms. Such an algorithm remains passive for a correct classification outcome, and turns aggressive in the event of a miscalculation, updating and adjusting. Unlike most other algorithms, it does not converge. Its purpose is to make updates that correct the loss, causing very little change in the norm of the weight vector.

Methodology

This advanced python project for identifying false news is concerned with the distinction between fake and true news. On our dataset, we create a TfidfVectorizer with sklearn. The model is then fitted using a PassiveAggressive Classifier. The accuracy score and the confusion matrix, in the end, tell us how well our model performs. We'll call the dataset we'll use for this Python project news.csv. The shape of this dataset is 77964. The news is identified in the first column, the title and content are in the second and third columns, and the fourth column has labels indicating if the news is REAL or FAKE. The dataset is 29.6 Megabytes in size and was downloaded here. The entire algorithm ran on a Google Colab instance. The link is attached at the end of the documentation.

1. The following libraries were installed using the !pip command:

```
pip install numpy pandas sklearn
```

2. Then the necessary imports were made:

```
[ ]  import numpy as np
     import pandas as pd
     import itertools
     from sklearn.model_selection import train_test_split
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.linear_model import PassiveAggressiveClassifier
     from sklearn.metrics import accuracy_score, confusion_matrix
```

3. The Google Drive folder containing the dataset was mounted:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

4. And the data imported from the file directory

```
#Read the data
df=pd.read_csv('/content/drive/MyDrive/data/news.csv')
```

5. The dataset was split into training and test sets:

```
[ ]  #Split the dataset
     x_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.3, random_state=10)
```

6. Initialized a TfidfVectorizer with English stop words and a maximum document frequency of 0.7 (terms with a greater document frequency will be rejected). Stop words

are the most prevalent terms in a language that must be removed before the natural language data can be processed. A TfidfVectorizer converts a matrix of TF-IDF features from a set of raw documents:

```
[ ]  #Initialize a TfidfVectorizer
     tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.8)

     #Fit and transform train set, transform test set
     tfidf_train=tfidf_vectorizer.fit_transform(x_train)
     tfidf_test=tfidf_vectorizer.transform(x_test)
```

7. We'll then create a PassiveAggressiveClassifier. This is the case. We'll use tfidf train and y train for this. Then, using accuracy score () from sklearn.metrics, we'll predict on the test set from the TfidfVectorizer and calculate the accuracy:

```
[ ]  #Initialize a PassiveAggressiveClassifier
     pac=PassiveAggressiveClassifier(max_iter=50)
     pac.fit(tfidf_train,y_train)

     #Predict on the test set and calculate accuracy
     y_pred=pac.predict(tfidf_test)
     score=accuracy_score(y_test,y_pred)
     print(f'Accuracy: {round(score*100,2)}%')

     Accuracy: 94.32%
```

8. With this model, we were able to achieve an accuracy of 94.32 percent. Finally, we'll print a confusion matrix to see how many false and true negatives and positives there are:

```
[ ]  #Build confusion matrix
     confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])

     array([[876,  44],
            [ 64, 917]])
```

We have 876 true positives, 917 true negatives, 64 false positives, and 44 false negatives using this model.

The entire code can be found on my Google Colab instance. Just connect to a runtime and run each cell of the notebook.