FAKE NEWS DETECTION USING NLP

Team member

810621104005:E.ARJUNAN

BEGINNER CLASSIFICATION MACHINE LEARNING NLP PROJECT PYTHON

This article was published as a part of the Data Science Blogathon

1. Introduction

We consume news through several mediums throughout the day in our daily routine, but sometimes it becomes difficult to decide which one is fake and which one is authentic.

Do you trust all the news you consume from online media?

Every news that we consume is not real. If you listen to fake news it means you are collecting the wrong information from the world which can affect society because a person's views or thoughts can change after consuming fake news which the user perceives to be true.

Since all the news we encounter in our day-to-day life is not authentic, how do we categorize if the news is fake or real?

In this article, we will focus on text-based news and try to build a model that will help us to identify if a piece of given news is fake or real.

Before moving to the practical things let's get aware of few terminologies.

2. Terminologies

2.1 Fake News

A sort of sensationalist reporting, counterfeit news embodies bits of information that might be lies and is, for the most part, spread through web-based media and other online media.

This is regularly done to further or force certain kinds of thoughts or for false promotion of products and is frequently accomplished with political plans.

Such news things may contain bogus and additionally misrepresented cases and may wind up being virtualized by calculations, and clients may wind up in a channel bubble.

2.2 Tfidf Vectorizer

TF (Term Frequency): In the document, words are present so many times that is called term frequency. In this section, if you get the largest values it means that word is present so many times with respect to other words. when you get word is parts of speech word that means the document is a very nice match.

IDF (Inverse Document Frequency): in a single document, words are present so many times, but also available

so many times in another document also which is not relevant. IDF is a proportion of how critical

a term is in the whole corpus.

collection of word Documents will convert into the matrix which contains TF-IDF features using TfidfVectorizer.

3. Project

To get the accurately classified collection of news as real or fake we have to build a machine learning model.

To deals with the detection of fake or real news, we will develop the project in python with the help of 'sklearn', we will use 'TfidfVectorizer' in our news data which we will gather from online media.

After the first step is done, we will initialize the classifier, transform and fit the model. In the end, we will calculate the performance of the model using the appropriate performance matrix/matrices. Once will calculate the performance matrices we will be able to see how well our model performs.

The practical implementation of these tools is very simple and will be explained step by step in this article. Let's start.

3.1 Data Analysis

Here I will explain the dataset.

In this python project, we have used the CSV dataset. The dataset contains 7796 rows and 4 columns.This dataset has four columns,

1. **title**: this represents the title of the news.
2. **author**: this represents the name of the author who has written the news.
3. **text**: this column has the news itself.
4. **label**: this is a binary column representing if the news is fake (1) or real (0).

The dataset is open-sourced and can be found here.

3.2 Libraries

The very basic data science libraries are sklearn, pandas, NumPy e.t.c and some specific libraries such as transformers.

```
import pandas as pd from nltk.corpus import stopwords from nltk.stem.porter import PorterStemmer import re import nltk from sklearn.feature_extraction.text import CountVectorizer from sklearn.feature_extraction.text import HashingVectorizer import matplotlib.pyplot as plt from sklearn.model_selection import train_test_split from sklearn.feature_extraction.text import TfidfVectorizer
```

3.3 Read dataset from CSV File

```
df=pd.read_csv('fake-news/train.csv')
```

```
df.head()
```

output:-

```
df.head()
```
executed in 16ms, finished 09:37:16 2021-06-07

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Before proceeding, we need to check whether a null value is present in our dataset or not.

```
df = df.isnull()
```

There is no null value in this dataset. But if you have null values present in your dataset then you can fill it. In the code given below, I will tell you how you can replace the null values.

```
df = df.fillna(' ')
```

3.4 Data Preprocessing

In data processing, we will focus on the text column on this data which actually contains the news part. We will modify this text column to extract more information to make the model more predictable. To extract information from the text column, we will use a library, which we know by the name of **'nltk'**.

Here we will use functionalities of the **'nltk'** library named Removing Stopwords, Tokenization, and Lemmatization. So we will see these functionalities one by one with these three examples. Hope you will have a better understanding of extracting information from the text column after this.

3.4.1 Removing Stopwords:-

These are the words that are used in any language used to connect words or used to declare the tense of sentences. This means that if we use these words in any sentence they do not add much meaning to the context of the sentence so even after removing the stopwords we can understand the context.
For more details click on this link.

3.4.2 Tokenization:-

Tokenization is the process of breaking text into smaller pieces which we know as tokens.
Each word, special character, or number in a sentence can be depicted as a token in NLP.

Tokenization is the process of breaking down a piece of code into smaller units called tokens.

```
from nltk.tokenize import word_tokenize


text = "Hello everyone. Welcome to Analytics Vidhya. You are studying NLP article" word_tokenize(text)
```

The output looked like this:

```
['Hello everyone.','Welcome to Analytics Vidhya.','You are studying NLP article']
```

3.5 CONVERTING LABELS:-

The dataset has a Label column whose datatype is Text Category. The Label column in the dataset is classified into two parts, which are denoted as Fake and Real. To train the model, we need to convert the label column to a numerical one.

```
**********************************************

df.label = df.label.astype(str) df.label = df.label.str.strip() dict = { 'REAL' : '1' , 'FAKE' : '0'}
df['label'] = df['label'].map(dict)df.head()
```

To proceed further, we separate our dataset into features(x_df) and targets(y_df).

```
x_df = df['total'] y_df = df['label']
```

3.6. VECTORIZATION

Vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which is used to find word predictions, word similarities/semantics.

For curiosity, you surely want to check out this article on ' Why data are represented as vectors in Data Science Problems'.

To make documents' corpora more relatable for computers, they must first be converted into some numerical structure. There are few techniques that are used to achieve this such as 'Bag of Words'.

Here, we are using vectorizer objects provided by Scikit-Learn which are quite reliable right out of the box.

```
from sklearn.feature_extraction.text import TfidfTransformer


from sklearn.feature_extraction.text import CountVectorizer


from sklearn.feature_extraction.text import TfidfVectorizer


count_vectorizer = CountVectorizer()
```

```
count_vectorizer.fit_transform(x_df)


freq_term_matrix = count_vectorizer.transform(x_df)


tfidf = TfidfTransformer(norm = "l2") tfidf.fit(freq_term_matrix) tf_idf_matrix =
tfidf.fit_transform(freq_term_matrix)


print(tf_idf_matrix)
```

Here, with 'Tfidftransformer' we are computing word counts using 'CountVectorizer' and then computing the IDF values and after that the Tf-IDF scores. With 'Tfidfvectorizer' we can do all three steps at once.

The code written above will provide with you a matrix representing your text. It will be a sparse matrix with a large number of elements in a Compressed Sparse Row format.

The most used vectorizers are:

Count Vectorizer: The most straightforward one, it counts the number of times a token shows up in the document and uses this value as its weight.
Hash Vectorizer: This one is designed to be as memory efficient as possible. Instead of storing the tokens as strings, the vectorizer applies the hashing trick to encode them as numerical indexes. The downside of this method is that once vectorized, the features' names can no longer be retrieved.
TF-IDF Vectorizer: TF-IDF stands for "term frequency-inverse document frequency", meaning the weight assigned to each token not only depends on its frequency in a document but also how recurrent that term is in the entire corpora. More on that here.

3.7. MODELING

After Vectorization, we split the data into test and train data.

```
# Splitting the data into test data and train data


x_train, x_test, y_train, y_test = train_test_split(tf_idf_matrix,y_df, random_state=0)
```

I fit four ML models to the data,

Logistic Regression, Naive-Bayes, Decision Tree, and Passive-Aggressive Classifier.

After that, predicted on the test set from the TfidfVectorizer and calculated the accuracy with accuracy_score() from sklearn. metrics.

3.7.1. Logistic Regression

#LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
```

```
logreg = LogisticRegression() logreg.fit(x_train, y_train) Accuracy = logreg.score(x_test, y_test)

print(Accuracy*100)
```

Accuracy: 91.73%

3.7.2. Naive-Bayes

#NAIVE BAYES

```
from sklearn.naive_bayes import MultinomialNB

NB = MultinomialNB() NB.fit(x_train, y_train) Accuracy = NB.score(x_test, y_test)

print(Accuracy*100)
```

Accuracy: 82.32 %

3.7.3. Decision Tree

DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier() clf.fit(x_train, y_train) Accuracy = clf.score(x_test, y_test)

print(Accuracy*100)
```

Accuracy: 80.49%

3.7.4. Passive-Aggressive Classifier

Passive Aggressive is considered algorithms that perform online learning (with for example Twitter data). Their characteristic is that they remain passive when dealing with an outcome that has been correctly classified, and become aggressive when a miscalculation takes place, thus constantly self-updating and adjusting.

PASSIVE-AGGRESSIVE CLASSIFIER

```
from sklearn.metrics import accuracy_score

from sklearn.linear_model import PassiveAggressiveClassifier
```

```
pac=PassiveAggressiveClassifier(max_iter=50)


pac.fit(x_train,y_train)


#Predict on the test set and calculate accuracy


y_pred=pac.predict(x_test)


score=accuracy_score(y_test,y_pred)


print(f'Accuracy: {round(score*100,2)}%')
```

Output:

Accuracy: 93.12%

4. CONCLUSION

The passive-aggressive classifier performed the best here and gave an accuracy of 93.12%.

We can print a confusion matrix to gain insight into the number of false and true negatives and positives

Fake news detection techniques can be divided into those based on style and those based on content, or fact-checking. Too often it is assumed that bad style (bad spelling, bad punctuation, limited vocabulary, using terms of abuse, ungrammaticality, etc.) is a safe indicator of fake news.

More than ever, this is a case where the machine's opinion must be backed up by clear and fully verifiable indications for the basis of its decision, in terms of the facts checked and the authority by which the truth of each fact was determined.

Collecting the data once isn't going to cut it given how quickly information spreads in today's connected world and the number of articles being churned out.

I hope you might find this helpful. You can comment down in the comment sections for any queries.

kajal56