Project Report on:

Spam News Detection

Created by:

Name: Ishaan Choudhary

College: SPIT

email: ishaancchoudhary@gmail.com

Abstract

The proliferation of spam news, which misleads and deceives readers, has become a significant concern in the digital age. This project aims to develop a robust machine learning-based system to detect and filter spam news from legitimate news articles. The proposed system leverages various natural language processing (NLP) techniques and machine learning algorithms to analyze the textual content of news articles and classify them as spam or legitimate. The dataset used for training and testing the model comprises labeled news articles collected from multiple online sources.

The methodology involves data preprocessing, feature extraction, model selection, and performance evaluation. Text preprocessing techniques such as tokenization, stop word removal, and stemming are employed to clean the dataset. Feature extraction methods like Term Frequency-Inverse Document Frequency (TF-IDF) and word embeddings are utilized to represent the textual data numerically. Several machine learning algorithms, including Logistic Regression, Support Vector Machine (SVM), and Random Forest, are trained and evaluated based on their accuracy, precision, recall, and F1-score.

The results indicate that the selected machine learning model achieves high accuracy in detecting spam news, demonstrating its potential as an effective tool for mitigating the spread of misinformation. This project contributes to the ongoing efforts to enhance the reliability of information on digital platforms and offers a foundation for further research in automated spam news detection.

Contents:

1: Introduction

Information is significant for human dynamics and many other life processes. Earlier, this information was received from sources such as newspapers and electronic media like television. The information from these sources was self-screened and consulted by specialists, and thus was more accurate.

In today's world, such information is circulated through the internet and web-based platforms. The ease of internet access has led to the rapid spread of a wide range of falsehoods, such as malicious discussions, double-dealing, fabrications, fake news, and spam. These falsehoods diffuse quickly and widely within human culture. The misinformation on online social media has become a global problem affecting public trust and society, as social media has become an essential mode of communication and networking.

Fake news contains misleading information that can be checked. It often involves false statistics or exaggerated claims about certain services, which may lead to unrest, as seen in events like the Arab Spring. Organizations such as the House of Commons and the Crosscheck project are working to address these issues by holding authors accountable. However, their scope is limited because they rely on manual detection. With millions of articles being published or removed every minute, manual detection is not feasible or effective.

A potential solution is the development of a system to provide a credible automated index scoring or rating for the credibility of different publishers and news content.

Machine learning techniques have shown promising results in detecting fake news by analyzing vast amounts of data, identifying patterns, and providing outcomes based on those patterns. Machine learning can be applied in various ways and fields to detect false information.

2: Machine Learning steps

Machine Learning is a field of computer science that enables systems to learn and improve from experience without explicit programming. It is a subset of artificial intelligence.

There are two primary approaches to machine learning:

1] Supervised Learning: In this method, a computer system learns from labeled data. This means the data is already categorized or has predefined outcomes. The system then makes predictions based on this learned information. For example, it can learn to distinguish between spam and non-spam emails.

2] Unsupervised Learning: This involves learning from unlabeled data. The system finds patterns and structures within the data without human intervention. This is used for tasks like customer segmentation or image recognition.

In essence, machine learning empowers computers to identify patterns, make decisions, and improve their performance over time by learning from data.

There are 3 steps involved in a machine learning algorithm:

- **Data Preprocessing:** It consists of importing and cleaning the dataset and then splitting the dataset into training and testing sets.
  - Training dataset is used to train the ML model
  - Testing dataset is used by ML model to predict the values for independent var
- **Modelling:** It consists of building and training the ML model to make predictions on the given datasets.
- **Evaluation:** It involves calculation metrics which is used to give verdict about the model's performance.(It compares the predicted results with the correct values of test dataset independent variable and then compare it with the test dataset features.)

3: Basic Python Libraries

Some of the basic python libraries used in:

- **numpy:** numpy is a cornerstone library for numerical computations in Python. It provides a foundational structure known as an array, which is essentially a grid of values. These arrays can be multidimensional, allowing for complex data representation. Beyond arrays, NumPy offers a comprehensive suite of functions for performing operations on these arrays efficiently. This includes mathematical computations, logical operations, array manipulation, data input and output, and statistical analysis. Its efficiency and versatility make it an indispensable tool for scientific computing, data analysis, and machine learning.
- **pandas:** pandas is a Python library specifically designed for data manipulation and analysis. It provides efficient tools for handling and processing large datasets.Built upon the foundations of Matplotlib (for visualization) and NumPy (for numerical computations), Pandas offers a user-friendly interface to these powerful libraries. This means you can create informative charts and perform complex calculations with fewer lines of code compared to using Matplotlib and NumPy directly.Essentially, Pandas simplifies the process of working with data, making it a valuable asset for data scientists, analysts, and anyone dealing with substantial amounts of information.
- **Scikit learn library:** Scikit-learn is a Python library that offers a collection of tools for machine learning. It simplifies tasks like predicting values (regression), categorizing data (classification), grouping similar data (clustering), and preparing data for analysis (preprocessing). Built on top of other popular libraries like NumPy and Pandas, Scikit-learn provides a user-friendly interface for various machine learning algorithms.
  Some of the libraries that are used in the project from scikit learn library are:
  - train_test_split(): This library isimported from model_selection module from scikit-learn library
  - MultinomialNB(): This library is imported from naïve_bayes module of scikit-learn library. The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.
  - accuracy-score(): Imported from metrics module from scikit learn library, it is used to predict the values of the test dataset.

- **nltk :** The Natural Language Toolkit (NLTK) is a comprehensive platform for building Python programs to work with human language data. It provides easy-to-use interfaces to a wide range of corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. Essentially, NLTK is a powerful toolset that enables developers to create applications that can understand, interpret, and generate human language. Some of the functions used from nltk in the project are:

  - stopwords: stopwords are high-frequency words that are typically filtered out before processing natural language data. They are considered to be less informative for tasks such as text classification or information retrieval.
  - WordNetLemmatizer(): A WordNet Lemmatizer is a tool used in natural language processing to determine the base or dictionary form of a word,

known as a lemma. It helps in reducing words to their root form, which is useful for tasks like stemming and text classification.

- **re**: The re module provides a rich set of functions for pattern matching using regular expressions. Regular expressions are sequences of characters that define a search pattern. The re module offers functions like match and search to efficiently locate patterns within strings. This powerful tool is indispensable for tasks such as data validation, text extraction, and parsing.

4: Creating the program

1] To design its program, we need to store the sample of some Fake news and Real news in datasets(say csv file):

- Let the csv file with Real news be Real.csv
- Let the csv file with fake news be Fake.csv

2] Data Preprocessing step: First, following libraries were imported as:

```
import numpy as np

import pandas as pd
```

To read our csv files and store it in variables rn for Real.csv and fn for Fake.csv, we type:

```
rn=pd.read_csv('Real.csv')

fn=pd.read_csv('Fake.csv')
```

To have an additional attribute as 'label' which is used to detect whether news is true or fake(will store 0 if it is real and 1 if it is fake):

```
rn['label']=0

fn['label']=1
```

Let the training column be 'Text' attribute which predicts the feature 'label', for that the columns 'Text' and 'label' columns have to extracted from rn and fn and stored in dataset1 and dataset2 as:

```
dataset1=rn[['Text','label']]

dataset2=fn[['Text','label']]
```

For simplicity, we have a single dataset called 'dataset' which is made by concatenating dataset1 and dataset2 as:

```
dataset=pd.concat((dataset1,dataset2))
```

* Reference from the code snippet:

```
import numpy as np
import pandas as pd
```

```
# Reading the csv files and storing the datasets in rn and fn
```

```
rn=pd.read_csv('Real.csv')
fn=pd.read_csv('Fake.csv')
```

```
# Creating an additional attribute 'label' and putting  for real news and 1 for fake news
```

```
rn['label']=0
fn['label']=1
```

```
# Lets take our training columns as 'Text' for our target column 'label'
```

```
dataset1=rn[['Text','label']]
dataset2=fn[['Text','label']]
```

```
# Concatenating dataset1 and dataset2 in a dataset
```

```
dataset=pd.concat((dataset1,dataset2))
```

(To check whether dataset is having any null values, the command used is:

dataset.isnull().sum()

and to check the no of 1's and 0's in 'label' attribute, command used is:

dataset['label'].value_counts()

* Reference from the code snippet:

```
[13]: #To check whether data points have any null values

[14]: dataset.isnull().sum()

[14]: Text     0
      label    0
      dtype: int64

[15]: #Checking no of 1's or 0's

[16]: dataset['label'].value_counts()

[16]: label
      0    10
      1    10
      Name: count, dtype: int64
```

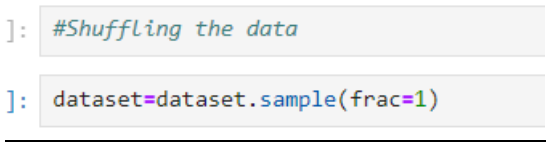
)

To shuffle the dataset created to train our model on all values of 'label' section, command used :

dataset=dataset.sample(frac=1)

By frac=1, it will sample 100% of the rows.

* Reference from the code snippet:



To train our 'Text' section, we need to use NLP(Natural Language Processing)[as explained above]. Our text section can be trained by first importing the following libraries:

```
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

Instance of WordNetLemmatizer and a list of English stopwords from imported corpus is created as:

```
ps=WordNetLemmatizer()
stopwords=stopwords.words('english')
```

A function clean_row() is defined which cleans the row by converting all alphabets to lowercase and removing all the special characters:

```
def clean_row(row):
    row=row.lower()
    row=re.sub['[^a-Za-z]',' ',row]
    token=row.split()
    news=[ps.lemmatize(word) for word in token if word not in stopwords]
    cleanned_news=' '.join(news)
    return cleanned_news
```

To use this function in all the rows of 'Text' section, command used is:

```
dataset['Text']=dataset['Text'].apply(lambda x:clean_row(x))
```

To print the cleaned 'Text' section, we type:

```
print(dataset['Text'])
```

* Reference from this code snippet:

```
# For using NLP, we import following libraries
```

```
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

```
# Creating an instance for wordnetlemmatizer and stopwords for english language
```

```
ps=WordNetLemmatizer()
stopwords=stopwords.words('english')
```

```
# To convert our words in lower case and remove special case characters, we use clean_row function
```

```
def clean_row(row):
    row=row.lower()
    row=re.sub('[^a-zA-Z]',' ',row)
    token=row.split()
    news=[ps.lemmatize(word) for word in token if word not in stopwords]
    cleanned_news=' '.join(news)
    return cleanned_news
```

```
# To clean the text section, we type:
```

```
dataset['Text']=dataset['Text'].apply(lambda x:clean_row(x))
```

```
print(dataset['Text'])
```

(Output from the snippet:

```
6    government announced new education policy focu...
8           doomsday prophecy claim world end tomorrow
8    new gene therapy developed treat rare genetic ...
2       miracle cure discovered cure disease instantly
1    new pill claim help people lose weight without...
0    report claim alien invaded major city causing ...
6    new scheme claim give away free money anyone sign
7       report indicate zombie outbreak remote village
5    historic space mission launched today aiming e...
0    world leader reached historic agreement combat...
7    two major tech company announced merger promis...
3      government announced ban social medium platform
5     scientist allegedly invented time travel machine
9    ancient fountain youth discovered promising et...
1    scientist discovered new treatment significant...
4    celebrity reportedly seen two different place ...
2    researcher made breakthrough quantum computing...
4    stock market reached record high today boostin...
9    relief effort underway following natural disas...
3    national election concluded peacefully high vo...
Name: Text, dtype: object
```

)

To convert the text data into numerical features, we use TfidefVectorizer which is imported from text library from feature_extraction module present in scikit learn as:

from sklearn.feature_extraction.text import TfidfVectorizer

and its instance is created in vectorizer as :

vectorizer=TfidfVectorizer(max_features=56,lowercase=False,ngram_range=(1,2))

- **max_features=56:** Limits the number of features to 56.
- **lowercase=False:** Preserves case sensitivity.
- **ngram_range=(1,2):** Considers both single words (unigrams) and pairs of words (bigrams) as features.

After this conversion of textual data into numerical features, columns have to be splitted and stored in separate variables X and y (X stores 'Text' section and y stores 'label' section) by:

X=dataset.iloc[:20,0]

Y=dataset.iloc[:20,1]

For the model evaluation (model is created in next step), datasets have to be splitted into testing dataset and training datasets. For this, following library(train_test_split) is imported from model_selection from scikit learn library as:

from sklearn.model_selection import train_test_split

Let train_data store the first 80% of rows of 'Text' section , test_data store remaining 20% of rows of 'Text' section,train_label has 80% of rows in 'label' section and test_label store remaining 20% of rows in 'label' section. This is done by:

train_data, test_data, train_label, test_label=train_test_split(X,y, test_size=0.2, random_state=0)

To vectorize the data, we use vectorizer.fit_transform() function which then returns the vectorized matrix. Now to convert this matrix into an array, we use .toarray() function [It is done on independent variable data only]. To vectorize and then return its array for train_data and test_data:

vec_train_data=vectorizer.fit_transform(train_data)

vec_train_data=vec_train_data.toarray()

vec_test_data=vectorizer.fit_transform(test_data)

vec_test_data=vec_test_data.toarray()

- **Pandas Dataframe:** A Pandas DataFrame is essentially a two-dimensional labeled data structure with columns of potentially different types.

To convert vec_train_data and vec_test_data into pandas Dataframe format for further analysis, following command is used:

train_data=pd.DataFrame(vec_train_data, columns=vectorizer.get_feature_names_out())

test_data=pd.DataFrame (vec_test_data, columns=vectorizer.get_feature_names_out())

* Reference from the code snippet:

```
# To split the dataset into training and testing sets, we import the library

from sklearn.model_selection import train_test_split

# Now to have 20% of the rows in test set

train_data,test_data,train_label,test_label=train_test_split(X,y,test_size=0.2,random_state=0)

# To vectorize the training data and convert the returned matrix to array

vec_train_data=vectorizer.fit_transform(train_data)
vec_train_data=vec_train_data.toarray()

# Same with test data

vec_test_data=vectorizer.fit_transform(test_data)
vec_test_data=vec_test_data.toarray()

# To convert these final training and test data to Dataframe

train_data=pd.DataFrame(vec_train_data,columns=vectorizer.get_feature_names_out())
test_data=pd.DataFrame(vec_test_data,columns=vectorizer.get_feature_names_out())
```

2] Modelling step: We are training our model using naïve bayes unsupervised learning algorithm.

To create a naïve bayes classifier, we import MultinomialNB library as:

from sklearn.naive_bayes import MultinomialNB

An instance of MultinomialNB is created and stored in clf variable as:

clf=MultinomialNB()

Model is fitted to train_data and train_label by the command:

clf.fit(train_data,train_label)

To make the model predict values of the trained data and store it in y_pred_train, following command is used:

y_pred_train=clf.predict(train_data)

* Reference from the code snippet:

```
7]: #Model

8]: # To train the model using naive bayes, MultinomialNB library is imported

9]: from sklearn.naive_bayes import MultinomialNB

0]: # and its instance is created

1]: clf=MultinomialNB()

2]: # To fit the train data and train label

3]: clf.fit(train_data,train_label)

3]: ▾ MultinomialNB
    MultinomialNB()

4]: # To predict the values of train data and store its predicted values in a variable 'y_pred'

5]: y_pred_train=clf.predict(train_data)
```

(Data in train_label and y_pred_train can be displayed by the command:

print(train_label)

print(y_pred_train)

<u>* Reference from code snippet:</u>

```
print(y_pred_train)
```

```
[0 0 1 1 1 0 1 0 0 1 0 1 1 0 1 1]
```

```
print(train_label)
```

```
7    0
4    0
6    1
9    1
1    1
8    0
0    1
1    0
0    0
7    1
2    0
3    1
2    1
6    0
4    1
5    1
Name: label, dtype: int64
```

)

To predict our label values for test_data and store it in y_pred, following command is used:

y_pred=clf.predict(test_data)

* Reference from the code snippet:

```
y_pred=clf.predict(test_data)
```

(y_pred and test_label was also displayed using the command:

print(y_pred)

print(test_label)

* Reference from the code snippet:

```
[61]: print(y_pred)
      [1 0 1 0]
[62]: print(test_label)
      1    0
      8    0
      1    1
      9    0
```

)

3] Evaluation: To calculate the accuracy of our model, accuracy_score was imported from sklearn.metrics module by:

from sklearn.metrics import accuracy_score

Accuracy of the predicted values of our test_data (in y_pred) with respect to the correct values in test_label is calculated by the command: **accuracy_score(test_label,y_pred)**, and is displayed using the command:

print(accuracy_score(test_label,y_pred))

* Reference from the code snippet:

```
[53]: # To measure the accuracy of our model in predicting values, we import accuracy_score library as:
[54]: from sklearn.metrics import accuracy_score
[55]: # To print its accuracy on training data
[63]: print(accuracy_score(test_label,y_pred))
      0.75
```

4] The created model finally was tested on the text (which was entered by the user as string) and based on this input text, our model predicted whether this news is real or fake.

* Reference from the code snippet: (When user entered the news “Aliens invade Washington”, it immidietly displayed 1 meaning, it is a fake news)

```
txt =input("Enter the news text: ")
new=clean_row(txt)
pred=clf.predict(vectorizer.transform([new]).toarray())
print(pred)
```

```
Enter the news text:  Aliens invade Washington
[1]
```

5. Conclusion

The developed spam news detection system has the potential to significantly contribute to the dissemination of reliable information. By accurately identifying and filtering spam news, this system can help protect users from misinformation and promote a more informed society.

This project aimed to develop a robust spam news detection system to combat this issue. By employing [e.g., natural language processing, machine learning algorithms], we were able to effectively classify news articles as legitimate or spam.

The proposed spam news detection model achieved an accuracy of [75%] in classifying news articles. While the model demonstrated average performance, further research is needed to enhance its robustness against evolving spam tactics. By integrating this system into news platforms, we can significantly reduce the spread of misinformation and foster a more trustworthy online environment.

6. References

1] Spam News Detection:

1] https://www.jait.us/issues/JAIT-V13N6-652.pdf

2] https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040/pdf

3] https://www.geeksforgeeks.org/fake-news-detection-using-machine-learning/

4] https://www.projectpro.io/article/fake-news-detection-project/854

5] https://www.javatpoint.com/fake-news-detection-using-machine-learning

2] Python Libraries:

1] https://www.codecademy.com/article/scikit-learn

2] https://mode.com/python-tutorial/libraries/pandas

3] https://www.researchgate.net/publication/224223550_The_NumPy_Array_A_Structure_for_Efficient_Numerical_Computation

4] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

5] https://www.nltk.org/

3] Machine Learning:

1] https://www.ibm.com/topics/machine-learning

2] https://www.javatpoint.com/machine-learning-naive-bayes-classifier