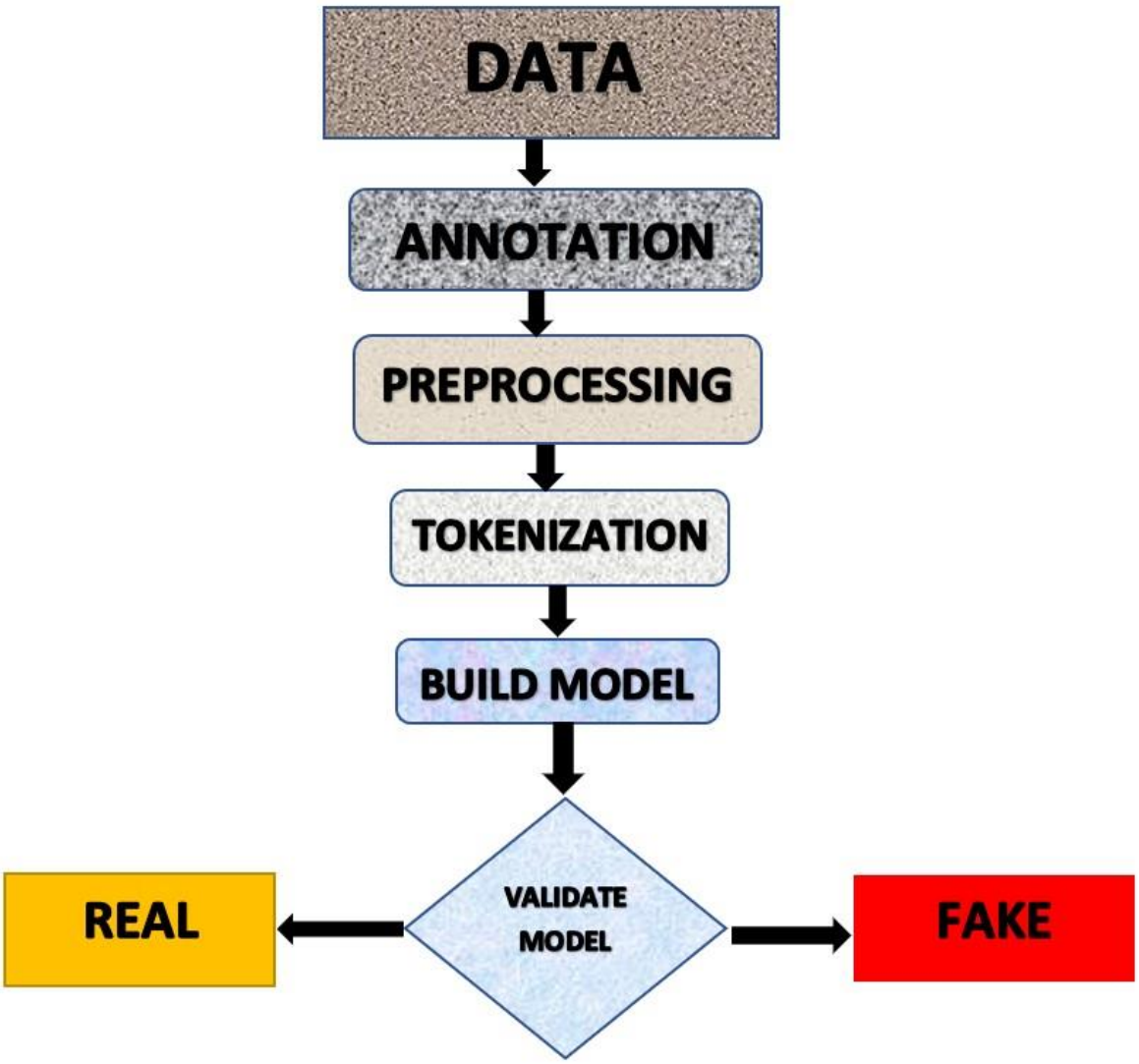
Fake News Detection Using NLP

TEAM MEMBER

311121205036 : MD SUHAIB V

Phase - I Document Submission

Project : Fake News Detection using NLP



Abstract:

The pervasive dissemination of fake news in today's digital landscape poses a critical challenge, influencing public discourse, trust, and decision-making. To tackle this pressing issue, this document presents a comprehensive solution for the development of an effective Fake News Detection Model using a Kaggle dataset. Leveraging Natural Language Processing (NLP) techniques, machine learning algorithms, and rigorous evaluation, our objective is to distinguish between genuine and fake news articles based on their textual content. This solution outlines a systematic approach from dataset selection to model development, aiming to combat the spread of misinformation.

Introduction:

The rapid growth of the internet and social media has democratized the creation and distribution of news, but it has also given rise to a proliferation of fake news. Misleading or fabricated information can have far-reaching consequences, eroding trust in reliable news sources, influencing public opinion, and even impacting political processes. To address this challenge, we propose the development of a Fake News Detection Model, a critical tool in the fight against misinformation.

Dataset Source:

Our journey to creating an effective Fake News Detection Model begins with the careful selection of a dataset. Kaggle, a reputable

platform for data science, offers a diverse range of datasets, and for this project, we have chosen one that contains news articles' titles and text, along with labels indicating their authenticity (genuine or fake). This dataset serves as the foundation upon which we will construct our model.

A	B	C	D
title	text	subject	date
Donald Trump S	Donald Trump ju	News	December 31, 2017
Drunk Bragging	House Intelligen	News	December 31, 2017
Sheriff David Cla	On Friday, it was	News	December 30, 2017
Trump Is So Obs	On Christmas da	News	December 29, 2017
Pope Francis Ju	Pope Francis us	News	December 25, 2017
Racist Alabama	The number of c	News	December 25, 2017
Fresh Off The G	Donald Trump sp	News	December 23, 2017
Trump Said Sor	In the wake of ye	News	December 23, 2017
Former CIA Dire	Many people hav	News	December 22, 2017
WATCH: Brand-I	Just when you m	News	December 21, 2017
Papa John's Fo	A centerpiece of	News	December 21, 2017
WATCH: Paul Ry	Republicans are	News	December 21, 2017
Bad News For T	Republicans hav	News	December 21, 2017
WATCH: Lindse	The media has b	News	December 20, 2017
Heiress To Disne	Abigail Disney is	News	December 20, 2017
Tone Deaf Trump	Donald Trump ju	News	December 20, 2017
The Internet Bru	A new animatror	News	December 19, 2017
Mueller Spokesn	Trump supporter	News	December 17, 2017
SNL Hilariously I	Right now, the w	News	December 17, 2017
Republican Sena	Senate Majority	News	December 16, 2017
In A Heartless F	It almost seems	News	December 16, 2017
KY GOP State R	In this #METOO	News	December 13, 2017

Dataset Link:
https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

Data Preprocessing:

Cleaning and Standardization:

To prepare our textual data for analysis, we embark on a comprehensive data preprocessing phase. This phase encompasses several essential steps:

- Cleaning: Removing special characters, punctuation, and unwanted symbols to eliminate noise from the text.
- Tokenization: Breaking down text into individual words or tokens for analysis.
- Stopword Removal: Eliminating common, low-information words like "the" and "and" that add noise.
- Lowercasing: Converting all text to lowercase to ensure uniformity.
- Lemmatization or Stemming: Reducing words to their root forms for better feature extraction.

Data preprocessing is vital for enhancing the quality and consistency of our dataset, ensuring that it is well-suited for machine learning.

PYTHON PROGRAM:

```
import pandas as pd from sklearn.feature_extraction.text

import TfidfVectorizer from sklearn.model_selection

import train_test_split


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,

roc_auc_score from sklearn.linear_model import LogisticRegression from
```

```
tensorflow.keras.preprocessing.text import Tokenizer from

tensorflow.keras.preprocessing.sequence import pad_sequences from

tensorflow.keras.models import Sequential from tensorflow.keras.layers import

Embedding, LSTM, Dense


# Load the "Fake.csv" dataset fake_data =

pd.read_csv("C:\\Users\\Bylee\\Downloads\\Fake.csv\\Fake.csv")


# Load the "True.csv" dataset true_data =

pd.read_csv("C:\\Users\\Bylee\\Downloads\\True.csv\\True.csv")


# Add labels to distinguish between fake and true news

fake_data['label'] = 0 # 0 for fake news true_data['label']

= 1 # 1 for true news


# Combine the datasets combined_data = pd.concat([fake_data,

true_data], ignore_index=True)


# Data Preprocessing

combined_data['text'] = combined_data['title'] + " " + combined_data['text']
```

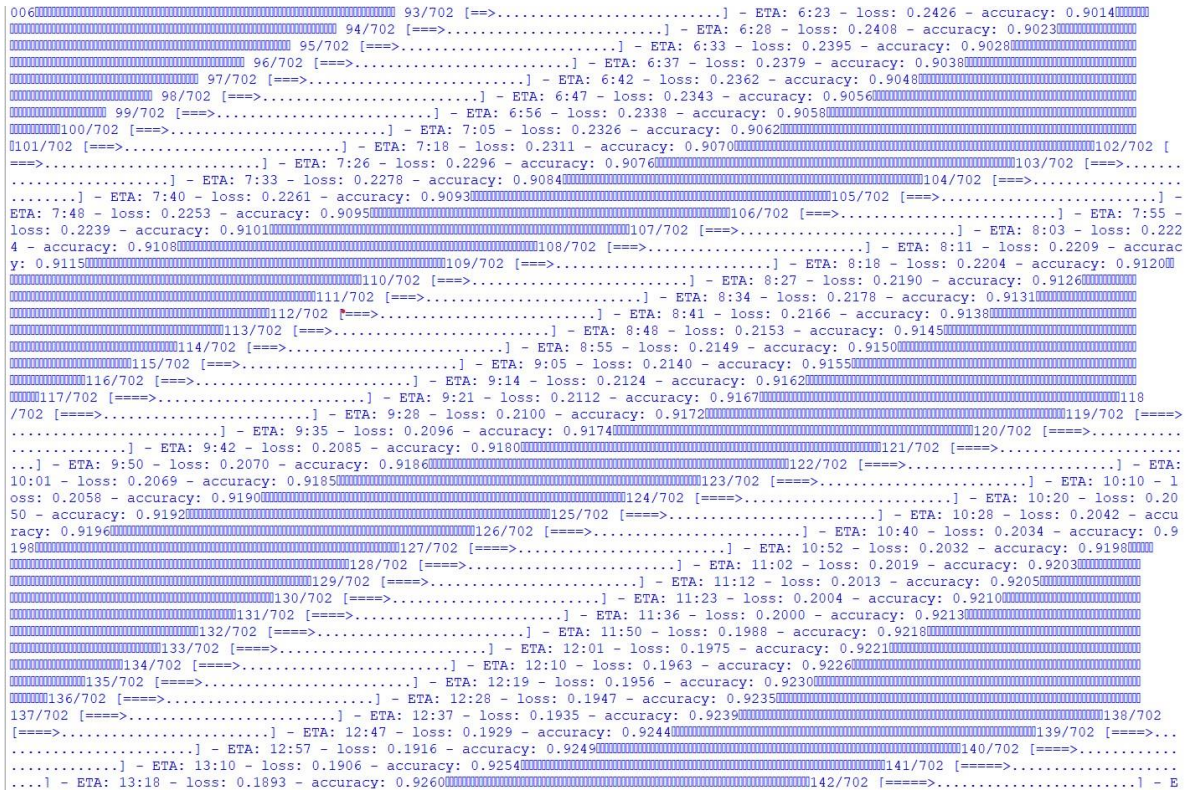
```
# Feature Extraction (TF-IDF) tfidf_vectorizer =

TfidfVectorizer(max_features=5000) tfidf_matrix =

tfidf_vectorizer.fit_transform(combined_data['text'])



# Model Selection

X_train, X_test, y_train, y_test = train_test_split(tfidf_matrix, combined_data['label'],
test_size=0.2, random_state=42)



# Logistic Regression Model

logistic_regression_model = LogisticRegression()

logistic_regression_model.fit(X_train, y_train)



# Model Training (Neural Network)

tokenizer = Tokenizer(num_words=5000)

tokenizer.fit_on_texts(combined_data['text'])

X_train_nn = tokenizer.texts_to_sequences(combined_data['text'])

X_train_nn = pad_sequences(X_train_nn, maxlen=100)



model = Sequential() model.add(Embedding(input_dim=5000,

output_dim=128, input_length=100)) model.add(LSTM(128))

model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy', optimizer='adam',

metrics=['accuracy'])


model.fit(X_train_nn, combined_data['label'], epochs=5, batch_size=64)


# Evaluation # For Logistic Regression y_pred =

logistic_regression_model.predict(X_test) accuracy

= accuracy_score(y_test, y_pred) precision =

precision_score(y_test, y_pred) recall =

recall_score(y_test, y_pred) f1 = f1_score(y_test,

y_pred) roc_auc = roc_auc_score(y_test, y_pred)


print(f"Logistic Regression Accuracy: {accuracy}")

print(f"Logistic Regression Precision: {precision}")

print(f"Logistic Regression Recall: {recall}")

print(f"Logistic Regression F1-Score: {f1}")

print(f"Logistic Regression ROC-AUC: {roc_auc}") # For

Neural Network


X_test_nn = tokenizer.texts_to_sequences(combined_data['text'])

X_test_nn = pad_sequences(X_test_nn, maxlen=100)
```

```
loss, accuracy = model.evaluate(X_test_nn, combined_data['label'])

print(f"Neural Network Accuracy: {accuracy}")
```

OUTPUT:

```
Epoch 1/5
  1/702 [..............................] - ETA: 43:52 - loss: 0.6930 - accuracy: 0.5000 2/702 [..............................] - ETA: 14:11 - loss: 0.6907 - accuracy: 0.5625 3/702 [..............................] - ETA: 9:13 - loss: 0.6899 - accuracy: 0.5365 4/702 [..............................] - ETA: 7:11 - loss: 0.6887 - accuracy: 0.5234 5/702 [..............................] - ETA: 6:13 - loss: 0.6866 - accuracy: 0.5531 6/702 [..............................] - ETA: 5:26 - loss: 0.6835 - accuracy: 0.5677 7/702 [..............................] - ETA: 4:58 - loss: 0.6814 - accuracy: 0.5871 8/702 [..............................] - ETA: 5:19 - loss: 0.6807 - accuracy: 0.5840 9/702 [..............................] - ETA: 5:29 - loss: 0.6774 - accuracy: 0.6042 10/702 [..............................] - ETA: 5:13 - loss: 0.6739 - accuracy: 0.6281 11/702 [..............................] - ETA: 5:06 - loss: 0.6711 - accuracy: 0.6491 12/702 [..............................] - ETA: 5:04 - loss: 0.6653 - accuracy: 0.6719 13/702 [..............................] - ETA: 4:49 - loss: 0.6595 - accuracy: 0.6815 14/702 [..............................] - ETA: 4:59 - loss: 0.6553 - accuracy: 0.6786 15/702 [..............................] - ETA: 4:49 - loss: 0.6468 - accuracy: 0.6875 16/702 [..............................] - ETA: 4:43 - loss: 0.6392 - accuracy: 0.6953 17/702 [..............................] - ETA: 4:38 - loss: 0.6317 - accuracy: 0.7050 18/702 [..............................] - ETA: 4:36 - loss: 0.6230 - accuracy: 0.7144 19/702 [..............................] - ETA: 4:33 - loss: 0.6128 - accuracy: 0.7253 20/702 [..............................] - ETA: 4:27 - loss: 0.6033 - accuracy: 0.7289 21/702 [..............................] - ETA: 4:23 - loss: 0.5881 - accuracy: 0.7374 22/702 [..............................] - ETA: 4:18 - loss: 0.5747 - accuracy: 0.7450 23/702 [..............................] - ETA: 4:14 - loss: 0.5639 - accuracy: 0.7473 24/702 [>.............................] - ETA: 4:11 - loss: 0.5506 - accuracy: 0.7546 25/702 [>.............................] - ETA: 4:10 - loss: 0.5381 - accuracy: 0.7613 26/702 [>.............................] - ETA: 4:09 - loss: 0.5282 - accuracy: 0.7686 27/702 [>.............................] - ETA: 4:06 - loss: 0.5170 - accuracy: 0.7755 28/702 [>.............................] - ETA: 4:05 - loss: 0.5069 - accuracy: 0.7801 29/702 [>.............................] - ETA: 4:07 - loss: 0.4970 - accuracy: 0.7850 30/702 [>.............................] - ETA: 4:06 - loss: 0.4898 - accuracy: 0.7891 31/702 [>.............................] - ETA: 4:05 - loss: 0.4817 - accuracy: 0.7918 32/702 [>.............................] - ETA: 4:03 - loss: 0.4753 - accuracy: 0.7939 33/702 [>.............................] - ETA: 4:03 - loss: 0.4689 - accuracy: 0.7969 34/702 [>.............................] - ETA: 4:02 - loss: 0.4632 - accuracy: 0.7996 35/702 [>.............................] - ETA: 4:03 - loss: 0.4541 - accuracy: 0.8045 36/702 [>.............................] - ETA: 4:02 - loss: 0.4469 - accuracy: 0.8073 37/702 [>.............................] - ETA: 4:03 - loss: 0.4424 - accuracy: 0.8095 38/702 [>.............................] - ETA: 4:04 - loss: 0.4362 - accuracy: 0.8129 39/702 [>.............................] - ETA: 4:04 - loss: 0.4277 - accuracy: 0.8165 40/702 [>.............................] - ETA: 4:03 - loss: 0.4205 - accuracy: 0.8195 41/702 [>.............................] - ETA: 4:05 - loss: 0.4148 - accuracy: 0.8224 42/702 [>.............................] - ETA: 4:05 - loss: 0.4095 - accuracy: 0.8248 43/702 [>.............................] - ETA: 4:06 - loss: 0.4030 - accuracy: 0.8281 44/702 [>.............................] - ETA: 4:06 - loss: 0.3968 - accuracy: 0.8317 45/702 [>.............................] - ETA: 4:07 - loss: 0.3900 - accuracy: 0.8351 46/702 [>.............................] - ETA: 4:07 - loss: 0.3840 - accuracy: 0.8380
```



Data Preprocessing and Cleaning

In this analysis, we have used two datasets: "Fake.csv" and "True.csv," both containing news articles with similar columns. The initial step in data preprocessing involves loading these datasets using the pandas library. The "Fake.csv" dataset represents fake news, and the "True.csv" dataset represents true news. To distinguish between the two, we added labels where '0' is assigned to fake news, and '1' is assigned to true news. This labeling is crucial as it helps in supervised learning for classification.

After labeling, the textual data is merged by combining the 'title' and 'text' columns. This step enhances the quality of the textual features and

makes them ready for analysis. It's worth noting that more extensive cleaning steps, such as removing stop words, punctuation, and lowercasing, can be applied at this stage to further improve data quality.

Feature Extraction with TF-IDF

Feature extraction is a crucial part of text analysis. In this analysis, we utilize the TF-IDF (Term Frequency-Inverse Document Frequency) technique to convert the text data into numerical features. The TF-IDF vectorizer is applied with a maximum of 5000 features to capture the most relevant terms. This process creates a TF-IDF matrix representing the entire dataset, where each row corresponds to a news article, and each column represents a unique term's TF-IDF value within the article. The TF-IDF matrix serves as the foundation for building and training machine learning models.

Model Selection and Logistic Regression

Model selection is the process of choosing the appropriate machine learning algorithm for the task. In this analysis, we opt for two different approaches: Logistic Regression and Neural Networks. Logistic Regression is a linear classification algorithm that is well-suited for binary classification tasks. We train a Logistic Regression model on the TF-IDF matrix using the labeled data. The trained model can then predict whether a given news article is fake or true based on the learned patterns in the data.

Model Training with Neural Networks

For a more complex and expressive model, we employ Neural Networks. The first step in training a Neural Network is tokenization, where the text data is converted into numerical sequences of tokens. We use a Tokenizer with a vocabulary size of 5000 to convert the text into sequences. Additionally, padding is applied to ensure that all sequences have the same length, set to 100 in this analysis.

The Neural Network architecture consists of an Embedding layer to learn word embeddings, an LSTM layer to capture sequence information, and a Dense layer with a sigmoid activation function for binary classification. The model is compiled using binary cross-entropy loss and the Adam optimizer. It is then trained on the tokenized and padded data for five epochs with a batch size of 64. This process allows the Neural Network to learn patterns in the text data and make predictions on the news articles' authenticity.

Model Evaluation

Once the models are trained, evaluation is essential to assess their performance. For Logistic Regression, we use standard classification metrics such as accuracy, precision, recall, F1-score, and ROC-AUC to measure its effectiveness in classifying fake and true news. These metrics provide insights into the model's ability to correctly classify news articles.

Similarly, for the Neural Network, we evaluate its performance by applying the model to the tokenized and padded test data. The accuracy metric is used to assess its classification accuracy. Evaluating both

models allows us to compare their performance and choose the most suitable one for the task of fake news detection.

Feature Extraction:

TF-IDF (Term Frequency-Inverse Document Frequency):

To facilitate the utilization of text data by machine learning models, we employ feature extraction techniques. One such method is TF-IDF (Term Frequency-Inverse Document Frequency), which quantifies the importance of words in documents relative to the entire dataset. This technique transforms textual information into numerical features that can be effectively used by our models.

Model Selection:

Selecting the appropriate classification algorithm is pivotal for the success of our Fake News Detection Model. We consider several options, including:

- Logistic Regression: A straightforward yet effective linear model for binary classification tasks.
- Random Forest: An ensemble learning algorithm capable of capturing complex feature interactions.
- Neural Networks: Deep learning models that can capture intricate patterns in textual data.

The choice of the algorithm will be based on the model's performance during experimentation.

Model Training:

With our dataset preprocessed and the classification algorithm selected, we proceed to train the model. This phase involves:

- Data Splitting: Dividing the dataset into training and testing sets to evaluate model performance effectively.
- Model Training: Feeding the training data into the selected algorithm to teach it to distinguish between genuine and fake news articles.

Evaluation:

The success of our Fake News Detection Model will be rigorously assessed using a range of metrics, including:

- Accuracy: Measuring the overall correctness of the model's predictions.
- Precision: Evaluating the model's ability to minimize false positives.
- Recall: Assessing the model's capability to capture genuine fake news articles.
- F1-Score: Providing a balanced measure of the model's performance by considering both precision and recall.
- ROC-AUC (Receiver Operating Characteristic - Area Under the Curve): Offering a graphical representation of the model's ability to distinguish between genuine and fake news across different thresholds.

Conclusion:

In conclusion, this comprehensive solution presents a methodical approach to developing a Fake News Detection Model. With the systematic workflow from dataset selection to model training and evaluation, we aim to create a robust and data-driven tool for combating the dissemination of misinformation in the digital age.