FAKE NEWS DETECTION USING NLP

TEAM MEMBER

810621104307:
MANOKARAN.M

Phase:4 submission document

Project title: Fake news detection using NLP

Phase 4: Development part 2

Topic: Continue building the fake news detection using NLP by feature engineering, model training, and evaluation.

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?

Everynews 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?

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.

READ DATASET:

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

df.head()

output:-

dataset | detecting fake news NLP

```
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4	4	Iranian woman jailed for fictional unpublished...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

DATA PRE-PROCESSING:

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.

MODEL SELECTION:

```
# 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_st

#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%

#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%

PASSIVE AGGRESSIVE:

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%

FEATURE SELECTION:

Feature selection is a crucial step in building effective machine learning models for fake news detection using Natural Language Processing (NLP). Fake news detection involves classifying text data into "fake" or "not fake" categories. Here are some steps and considerations for feature selection in NLP-based fake news detection:

Text Preprocessing:

Tokenization: Split the text into words or subword units.

Lowercasing: Convert all text to lowercase for uniformity.

Stop Word Removal: Remove common words like "and," "the," "in" that do not carry much information.

Stemming or Lemmatization: Reducing words to their root forms can reduce feature dimensionality.

Feature Extraction:

Bag of Words (BoW): Represent the text as a matrix of word frequencies.

TF-IDF (Term Frequency-Inverse Document Frequency): Weigh words based on their importance in a document relative to the entire corpus.

Word Embeddings (Word2Vec, GloVe, FastText): Transform words into dense vector representations.

n-grams: Consider word sequences of different lengths (bi-grams, tri-grams) as features.

Feature Selection Techniques:

Mutual Information: Measure the mutual dependence between each feature and the target variable (fake or not fake).

Chi-squared test: Assess the independence of categorical variables.

Information Gain: Quantify the reduction in entropy when a feature is used for splitting.

Feature Importance from Machine Learning Models: Use decision trees or random forests to evaluate feature importance.

Dimensionality Reduction:

Principal Component Analysis (PCA): Reduce dimensionality while preserving variance.

Singular Value Decomposition (SVD): Decompose the TF-IDF matrix.

Latent Dirichlet Allocation (LDA): Discover topic distributions within the corpus.

Feature Engineering:

Create custom features based on domain knowledge, such as the source of the news, sentiment analysis, or writing style.

Represent text as a graph and extract graph-based features.

Include features related to metadata, like publication date and authorship.

Feature Importance Analysis:

Use techniques like recursive feature elimination (RFE) or feature permutation to assess the importance of selected features.

Visualize feature importance using bar plots, word clouds, or correlation matrices.

Regularization:

If you're using machine learning models like logistic regression or linear SVM, consider L1 or L2 regularization to automatically select relevant features.

Cross-Validation:

Use cross-validation techniques to assess the model's performance with different feature subsets.

Ensemble Models:

Combine the predictions of multiple models, each trained on a different set of features.

Monitor Model Performance:

Continuously evaluate the model's performance and consider revising feature selection based on model results.

FEATURE SELECTION:

```
import numpy as np

from sklearn.feature_selection import SelectKBest, mutual_info_classif

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score
```

```
# Sample data (replace this with your dataset)
X = ["This is a real news article.", "Fake news is a problem.", "Unbiased reporting is essential.", "Fake news spreads rapidly."]
y = [0, 1, 0, 1]  # 0 for real news, 1 for fake news

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature extraction using TF-IDF
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Feature selection using Mutual Information
num_features_to_select = 2
selector = SelectKBest(score_func=mutual_info_classif, k=num_features_to_select)
X_train_selected = selector.fit_transform(X_train_tfidf, y_train)
X_test_selected = selector.transform(X_test_tfidf)
```

```
# Train a classifier (e.g., Naive Bayes) with the selected features
classifier = MultinomialNB()
classifier.fit(X_train_selected, y_train)

# Make predictions
y_pred = classifier.predict(X_test_selected)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Get the indices of the selected features
selected_feature_indices = selector.get_support(indices=True)
print("Selected feature indices:", selected_feature_indices)
```

MODEL TRAINING:

```
import numpy as np

import pandas as pd

from sklearn.feature_extraction.text import
TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score,
classification_report

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

from nltk.stem import PorterStemmer


# Sample data (replace this with your dataset)

data = pd.DataFrame({

    'text': ["This is a real news article.", "Fake news is a
problem.", "Unbiased reporting is essential.", "Fake news
spreads rapidly."],

    'label': [0, 1, 0, 1]  # 0 for real news, 1 for fake news

})
```

```
# Data preprocessing
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def preprocess_text(text):
    words = word_tokenize(text)
    words = [stemmer.stem(word) for word in words if word.isalpha() and word not in stop_words]
    return ' '.join(words)

data['text'] = data['text'].apply(preprocess_text)

# Split the data into training and testing sets
X = data['text']
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature extraction using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)  # You can adjust the number of features as needed
```

```
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train a classifier (Multinomial Naive Bayes)
classifier = MultinomialNB()
classifier.fit(X_train_tfidf, y_train)

# Make predictions
y_pred = classifier.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred,
target_names=['Real News', 'Fake News'])

print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
```

LOGISTIC REGRESSIVE:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

# Load your dataset
data = pd.read_csv('your_dataset.csv')

# Data preprocessing
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def preprocess_text(text):
    words = word_tokenize(text)
```

```
    words = [stemmer.stem(word) for word in words if
word.isalpha() and word not in stop_words]
    return ' '.join(words)

data['text'] = data['text'].apply(preprocess_text)

# Split the data into training and testing sets
X = data['text']
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Feature extraction using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=1000)  #
You can adjust the number of features as needed
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Train a Logistic Regression classifier
classifier = LogisticRegression()
classifier.fit(X_train_tfidf, y_train)
```

```
# Make predictions
y_pred = classifier.predict(X_test_tfidf)


# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred,
target_names=['Real News', 'Fake News'])


print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", report)
```

FEATURE ENGINEERING:

Text-based Features:

a. Text Length: The length of the news article or the number of words can be an indicative feature. Fake news might be excessively short or excessively long.

b. Punctuation and Special Characters: Count the number of punctuation marks, exclamation points, or question marks. Fake news might contain more exaggerated punctuation.

c. **Capitalization**: Count the number of capitalized words or sentences. Fake news may use capital letters for emphasis.

d. Readability: Use metrics like Flesch-Kincaid Grade Level or Gunning Fog Index to measure the readability of the text. Fake news might have a different readability pattern.

e. Sentiment Analysis: Analyze the sentiment of the text. Fake news might be more emotionally charged.

f. **Named Entities**: Count the number of named entities (e.g., people, organizations, locations) mentioned in the text. Fake news might involve fabricated or exaggerated entities.

Text Structure and Style:

a. Paragraph and Sentence Structure: Analyze the structure of paragraphs and sentences. Fake news might have a different structure than real news.

b. Writing Style: Examine the writing style, such as the use of passive voice, excessive use of adjectives, or frequent use of sensationalist language.

c. Repetition: Identify repeated phrases or sentences within the text. Fake news may use repetition to emphasize false claims.

Metadata Features:

a. Source Reliability: Include features about the credibility of the news source. Check for the reputation, bias, or history of the source.

b. Publication Date: Consider the publication date, and create features that capture the recency of the news. Fake news may be outdated or posted at suspicious times.

c. Authorship: Investigate the author's credentials and previous works. Some fake news articles use pseudonyms or unreliable authors.

NLP Techniques:

a. Topic Modeling: Use techniques like Latent Dirichlet Allocation (LDA) to identify the main topics in the text. Fake news might focus on different topics than real news.

b. Named Entity Recognition (NER): Extract named entities and classify them into categories. Fake news may include fake entities or affiliations.

c. Dependency Parsing: Analyze the grammatical structure of sentences. Fake news might have different linguistic patterns.

User-Generated Content (UGC):

a. User Comments: If available, analyze user comments and their sentiment regarding the news article. Suspicious patterns or opinions may suggest fake news.

b. Social Media Engagement: Collect data on the article's social media engagement, such as likes, shares, and comments. Unusually high or low engagement could be a sign of fake news.

Cross-Referencing and Fact-Checking:

a. External Fact-Checking: Use external fact-checking sources to verify the claims made in the news. Create features based on the results of fact-checking.

b. Cross-Referencing with Trusted Sources: Compare the information in the news article with information from reputable sources. Discrepancies can be indicative of fake news.

Network Analysis:

a. Propagation Analysis: Analyze how the news spreads through social networks. Fake news might exhibit different propagation patterns.

b. Source Network Analysis: Examine the network of sources and their relationships. Fake news may originate from less reputable sources.

Combining Features:

Consider creating composite features or combining multiple features to capture complex patterns related to fake news.

EVALUATION:

Evaluating a fake news detection system that uses NLP (Natural Language Processing) involves assessing its performance, including its accuracy, precision, recall, F1 score, and other relevant metrics. Here's a brief overview

of the key evaluation metrics and considerations for fake news detection:

Confusion Matrix:

A confusion matrix is a useful starting point for evaluating your fake news detection model. It provides a breakdown of the model's predictions:

True Positives (TP): The number of correctly classified fake news articles.

True Negatives (TN): The number of correctly classified real news articles.

False Positives (FP): The number of real news articles incorrectly classified as fake.

False Negatives (FN): The number of fake news articles incorrectly classified as real.

Accuracy:

Accuracy is the most straightforward metric and measures the overall correctness of predictions:

Accuracy

=

+

Accuracy=

TP+TN+FP+FN/TP+TN

While accuracy is essential, it may not be sufficient for imbalanced datasets, where one class significantly outnumbers the other.

Precision:

Precision measures the proportion of correctly classified fake news out of all the articles classified as fake:

Precision

=

+

Precision= TP+FP/TP

High precision indicates that when the model predicts an article as fake, it is often correct.

Recall (Sensitivity or True Positive Rate):

Recall measures the proportion of correctly classified fake news out of all actual fake news articles:

Recall

=

+

Recall=

TP+FN/TP

High recall indicates that the model is good at capturing fake news articles.

F1 Score:

The F1 score is the harmonic mean of precision and recall, which provides a balanced measure of model performance:

F1 Score

=

2

.

Precision

.

Recall

Precision

+

Recall

F1 Score=

Precision+Recall

2·Precision·Recall

It's especially useful when there's an imbalance between the classes.

Specificity (True Negative Rate):

Specificity measures the proportion of correctly classified real news out of all actual real news articles:

Specificity

=

+

Specificity= TN+FP/TN

High specificity indicates that the model is good at correctly classifying real news articles.

AUC-ROC (Area Under the Receiver Operating Characteristic Curve):

The ROC curve plots the true positive rate (recall) against the false positive rate for different classification thresholds. A model with a higher AUC-ROC score is generally better at distinguishing between the two classes.

Cross-Validation:

Use cross-validation techniques (e.g., k-fold cross-validation) to assess the model's performance on different subsets of the data. This helps evaluate the model's generalization capability.

Bias and Fairness:

Evaluate the model for potential biases, such as demographic or political bias. Ensure that the model doesn't unfairly discriminate against certain groups.

External Evaluation:

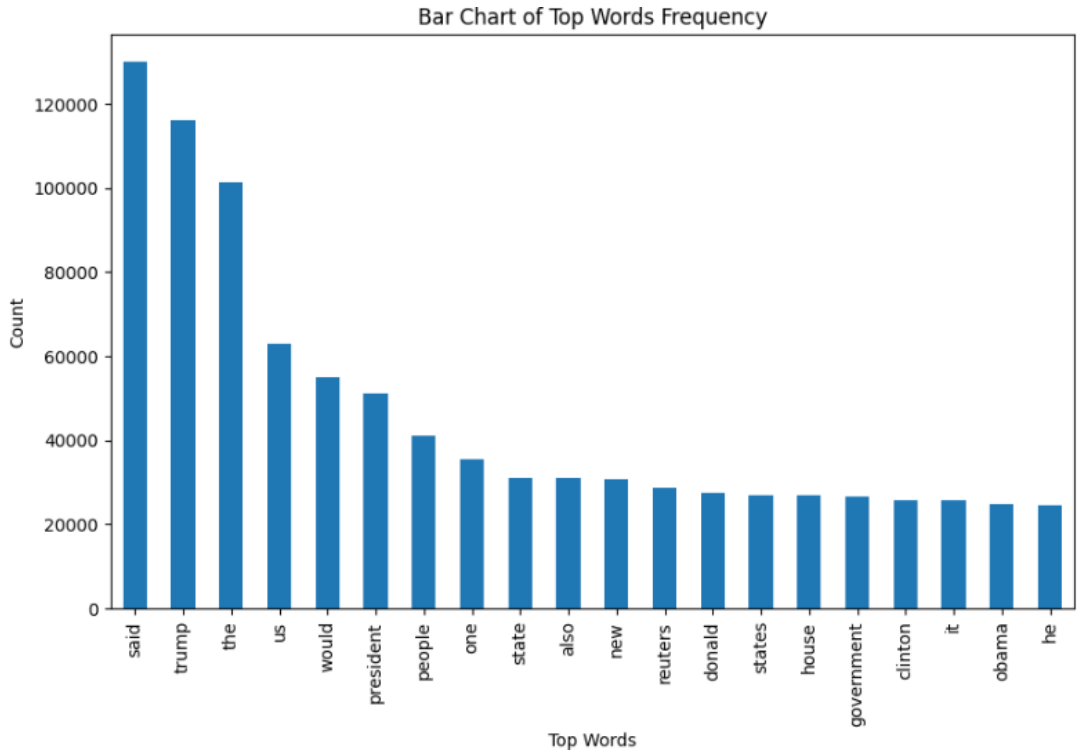
Consider external evaluation by comparing your model's performance with human fact-checkers or other established benchmarks for fake news detection.

Real-world Testing:

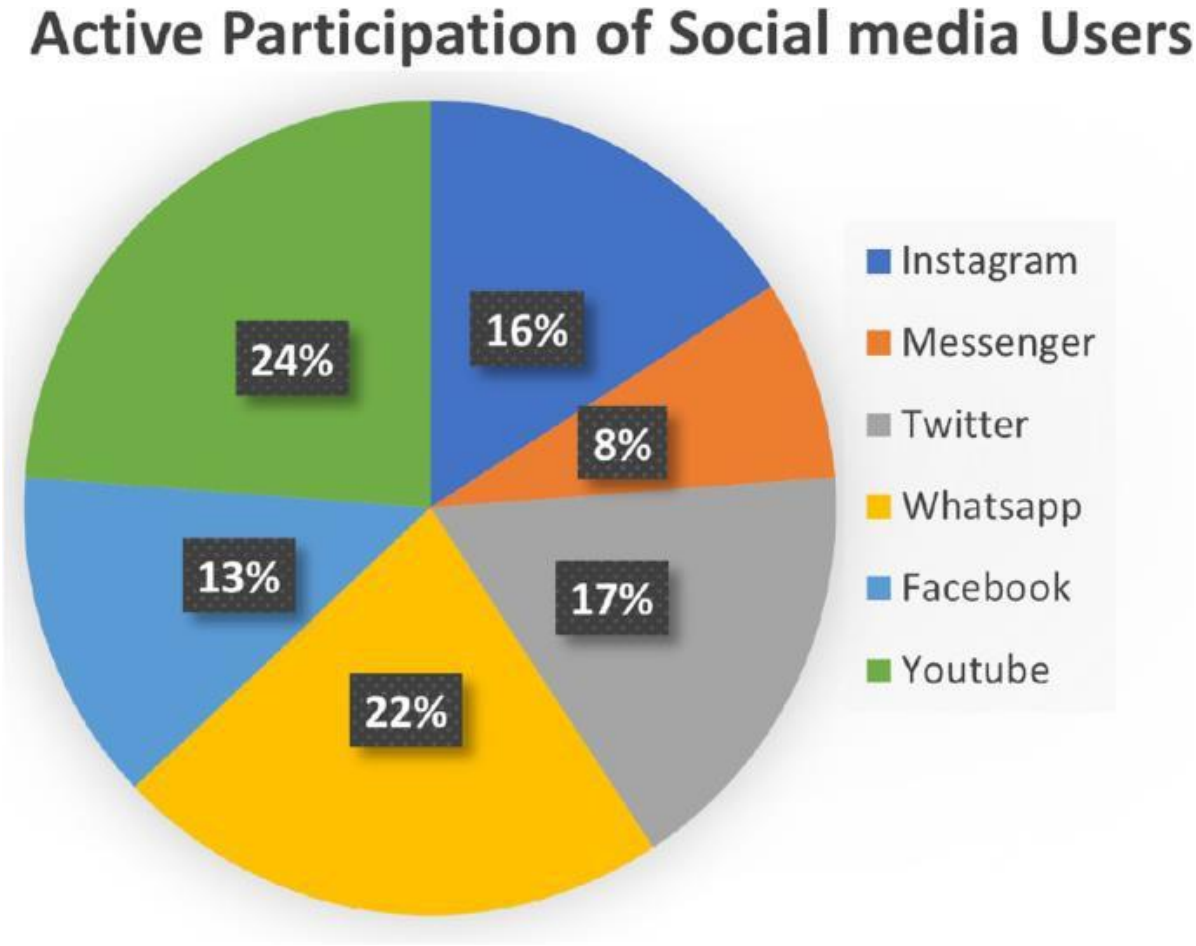
Finally, conduct real-world testing, where your model is applied to a continuous stream of news articles. Monitor

its performance over time and make adjustments as needed.

BARCHART:


Bar Chart of Top Words Frequency
Count
120000
100000
80000
60000
40000
20000
0
said
trump
the
us
would
president
people
one
state
also
new
reuters
donald
states
house
government
clinton
it
obama
he
Top Words

PIE CHART:


Active Participation of Social media Users
16%
8%
17%
22%
13%
24%
Instagram
Messenger
Twitter
Whatsapp
Facebook
Youtube