



Internship Project Report

Fake News Detection

A PROJECT REPORT

Submitted by

KASU NAGESWARA REDDY

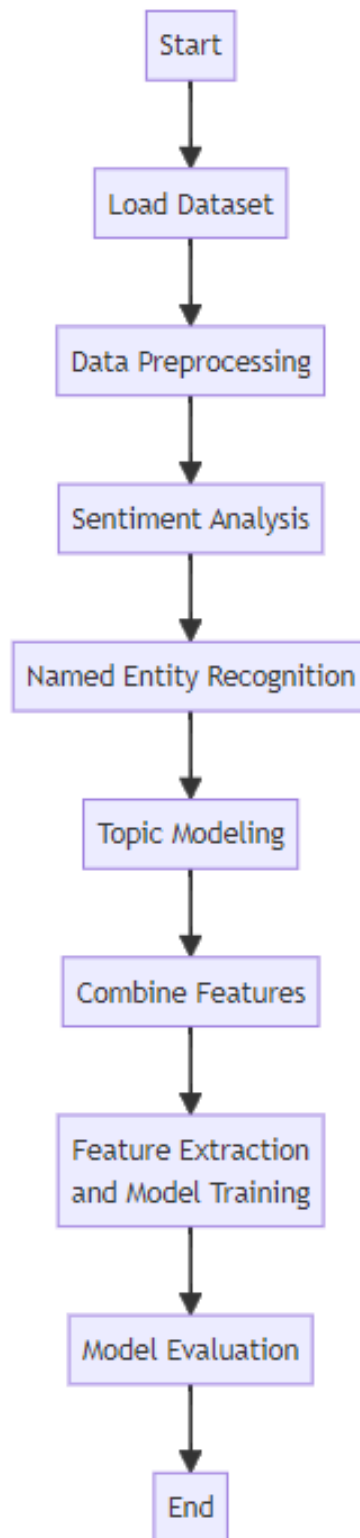
Introduction

In the information age, the internet is flooded with news articles, not all credible. The spread of fake news can have dire consequences, including misinformation, panic, and societal harm. I developed a fake news detection system using various Natural Language Processing (NLP) techniques and Machine Learning (ML) algorithms to address this issue. This system identifies the authenticity of news articles by leveraging Sentiment Analysis, Named Entity Recognition (NER), Topic Modeling, and other NLP tools.

Dataset

The dataset consists of news articles labeled as "REAL" or "FAKE." Each article includes text content and a corresponding label.

Methodologies



➤ Setup Environment and Import Libraries

Setup the Environment

```
✓ [29] pip install pandas numpy scikit-learn nltk spacy gensim  
6s
```

Import Libraries

```
[30] import pandas as pd  
import numpy as np  
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  
import nltk  
import spacy  
from gensim import corpora  
from gensim.models.ldamodel import LdaModel  
from textblob import TextBlob
```

Download necessary NLP resources

```
▶ nltk.download('punkt')  
spacy.cli.download('en_core_web_sm')
```

```
⇒ [nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data] Package punkt is already up-to-date!  
✓ Download and installation successful  
You can now load the package via spacy.load('en_core_web_sm')  
⚠ Restart to reload dependencies  
If you are in a Jupyter or Colab notebook, you may need to restart Python in  
order to load all the package's dependencies. You can do this by selecting the  
'Restart kernel' or 'Restart runtime' option.
```

Setting up the environment involves importing essential libraries such as pandas for data handling, numpy for numerical operations, scikit-learn for machine learning tools, nltk for natural language processing tasks, spacy for advanced NLP tasks, gensim for topic modeling, and TextBlob for

sentiment analysis. Downloading NLTK and SpaCy resources ensures that we have necessary models for tokenization, named entity recognition, and sentiment analysis.

➤ Loading the dataset

Loading the Dataset

```
[19] # Load the dataset
df = pd.read_csv('/content/news.csv') # Adjust the filename if necessary

# Check for missing values and drop them
df = df.dropna()

# Inspect the first few rows of the dataset
print(df.head())
df = df.dropna()
```

```
↔ Unnamed: 0 title \
0      8476      You Can Smell Hillary's Fear
1    10294 Watch The Exact Moment Paul Ryan Committed Pol...
2     3608      Kerry to go to Paris in gesture of sympathy
3    10142 Bernie supporters on Twitter erupt in anger ag...
4      875   The Battle of New York: Why This Primary Matters

      text label
0 Daniel Greenfield, a Shillman Journalism Fello... FAKE
1 Google Pinterest Digg Linkedin Reddit Stumbleu... FAKE
2 U.S. Secretary of State John F. Kerry said Mon... REAL
3 - Kaydee King (@KaydeeKing) November 9, 2016 T... FAKE
4 It's primary day in New York and front-runners... REAL
```

Loading the dataset is the first step in data preparation. This dataset contains news articles labelled as 'REAL' or 'FAKE'. Each article is represented by text content and a corresponding label indicating its authenticity.

➤ Data Preprocessing

Preprocess the Data

```
[21] def preprocess_text(text):  
      # Tokenization  
      words = nltk.word_tokenize(text)  
      # Lowercasing  
      words = [word.lower() for word in words if word.isalpha()]  
      return ' '.join(words)  
  
      df['processed_text'] = df['text'].apply(preprocess_text)
```

Data preprocessing involves cleaning and transforming the text data into a format suitable for further analysis. Tokenization breaks down the text into individual words, and lowercasing ensures consistency. Removing non-alphabetical characters helps to focus on meaningful words.

➤ Sentiment Analysis

Sentiment Analysis

```
[22] def get_sentiment(text):  
      blob = TextBlob(text)  
      return blob.sentiment.polarity  
  
      df['sentiment'] = df['processed_text'].apply(get_sentiment)
```

Sentiment analysis determines the sentiment polarity (positive, negative, or neutral) of each news article. This feature helps to capture the emotional context of the text, which can be indicative of subjective information.

➤ Named Entity Recognition (NER)

Named Entity Recognition

```
m [▶] nlp = spacy.load('en_core_web_sm')

def extract_entities(text):
    doc = nlp(text)
    entities = [ent.text for ent in doc.ents]
    return ' '.join(entities)

df['entities'] = df['processed_text'].apply(extract_entities)
```

Named Entity Recognition (NER) identifies and categorizes named entities such as names of people, organizations, and locations within the text. This helps to extract key entities that might indicate the subject or context of the news article.

➤ Topic Modeling

Topic Modelling

```
[24] # Tokenization for topic modeling
def tokenize(text):
    return [word for word in nltk.word_tokenize(text) if word.isalpha()]

df['tokens'] = df['processed_text'].apply(tokenize)

# Create a dictionary and corpus for LDA
dictionary = corpora.Dictionary(df['tokens'])
corpus = [dictionary.doc2bow(tokens) for tokens in df['tokens']]

# Train LDA model
lda_model = LdaModel(corpus, num_topics=10, id2word=dictionary, passes=10)

def get_topics(text):
    tokens = tokenize(text)
    bow = dictionary.doc2bow(tokens)
    topics = lda_model.get_document_topics(bow)
    return topics

df['topics'] = df['processed_text'].apply(get_topics)
```

Topic modeling identifies underlying topics within the news articles. Latent Dirichlet Allocation (LDA) is used to discover topics based on the distribution of words across documents. This helps to understand the main themes and content of the articles.

➤ Combine Features

Combining features

```
[25] def combine_features(row):  
      return f"{row['processed_text']} {row['entities']} {row['sentiment']} {row['topics']}"  
  
      df['combined_features'] = df.apply(combine_features, axis=1)
```

Combined features integrate the processed text, named entities, sentiment, and topics into a single feature set. This aggregated feature set will be used as input for the machine learning model.

➤ Feature Extraction and Model Training

Feature Extraction and Model Training

```
✓ [27] X = df['combined_features']  
js      y = df['label']  
  
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=7)  
  
      tfidf_vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')  
      tfidf_train = tfidf_vectorizer.fit_transform(X_train)  
      tfidf_test = tfidf_vectorizer.transform(X_test)  
  
      pac = PassiveAggressiveClassifier(max_iter=50)  
      pac.fit(tfidf_train, y_train)  
  
      y_pred = pac.predict(tfidf_test)  
  
      score = accuracy_score(y_test, y_pred)  
      print(f'Accuracy: {round(score*100, 2)}%')
```

➡ Accuracy: 90.84%

- **Feature Extraction:** The combined features are converted into numerical vectors using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. This transforms text into numerical features that can be used by machine learning algorithms.
- **Model Training:** The PassiveAggressiveClassifier is trained on the TF-IDF vectors to classify news articles as either 'FAKE' or 'REAL'.
- **Model Evaluation:** Accuracy and the confusion matrix are used to evaluate the performance of the model in predicting fake news.

➤ Confusion Matrix

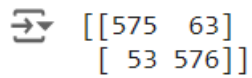
Confusion matrix

```
[28] cm = confusion_matrix(y_test, y_pred, labels=['FAKE', 'REAL'])  
      print(cm)
```

```
↔ [[575  63]  
   [ 53 576]]
```

A confusion matrix is a table that is often used to evaluate the performance of a classification model. It allows visualization of the performance of an algorithm by showing the counts of the actual and predicted classifications.

Results



```
[[575  63]
 [ 53 576]]
```

- **TP (True Positives):** This value (575) indicates the number of instances the model correctly classified as positive.
- **TN (True Negatives):** This value (576) shows the number of instances the model correctly classified as negative.
- **FP (False Positives):** This value (63) represents the number of instances the model incorrectly classified as positive (when they were actually negative).
- **FN (False Negatives):** This value (53) shows the number of instances the model incorrectly classified as negative (when they were actually positive).