# FAKE NEWS DETECTION USING NLP

## PHASE 5: DOCUMENTATION

## PROBLEM STATEMENT

The rapid proliferation of fake news and misinformation in the digital age poses a significant threat to society, democracy, and public trust. Addressing this problem requires the development of robust and efficient tools for detecting fake news using Natural Language Processing (NLP) techniques. The problem can be defined as follows:

"Develop an NLP-based system for the automatic detection of fake news and misinformation in textual content. This system should be capable of analyzing and classifying news articles, social media posts, and other textual sources into two categories: 'Real News' and 'Fake News.' The system's objective is to help individuals, organizations, and platforms to identify and filter out false or misleading information, thereby mitigating the negative impact of fake news on public discourse, decision-making, and trust in media sources."

This problem statement can be further refined by specifying the desired goals and outcomes of the fake news detection system, the scope of sources and languages it should cover, the evaluation metrics for its performance, and any potential constraints or considerations related to data privacy and ethical considerations.

## DESIGN THINKING

Design thinking is a problem-solving approach that focuses on user-centered design and iterative development. When applied to the problem of fake news detection using NLP, the process can be broken down into several stages:

1. **Empathize**
   - Understand the users' needs and challenges related to fake news.
   - Conduct interviews, surveys, and observations to gain insights into how people encounter and handle fake news.
   - Identify the specific pain points and information gaps in the current fake news detection process.

2. **Define**
   - Clearly define the problem you are trying to solve, considering user insights.
   - Set specific goals for the NLP-based fake news detection system, such as accuracy, scalability, and usability.
   - Identify the target audience (e.g., individuals, news organizations, social media platforms) for your solution.

3. **Ideate**
    - Brainstorm potential solutions and approaches for fake news detection using NLP.
    - Encourage a diverse team to generate a wide range of ideas, from algorithmic methods to user interfaces.
    - Focus on creative and innovative approaches that can address the identified challenges.

4. **Prototype**
    - Create a low-fidelity prototype of the fake news detection system. This could be a simple NLP model or a basic user interface.
    - Test the prototype with a small group of users to gather feedback and refine the design.
    - Iterate on the prototype based on user feedback and insights.
5. **Test**
    - Implement a more advanced NLP-based fake news detection model.
    - Test the system with a larger dataset of fake and real news articles and social media posts.
    - Evaluate the system's performance using appropriate metrics (e.g., precision, recall, F1-score) and user feedback.
    - Identify areas where the system may make false positives or false negatives and iterate on the model to improve its accuracy.

6. **Implement**
    - Develop a user-friendly interface for the fake news detection system.
    - Ensure scalability and efficiency for processing a large volume of textual data.
    - Integrate the NLP model into the interface and deploy the system in a real-world environment.

7. **Monitor**
    - Continuously monitor the system's performance and gather user feedback after deployment.
    - Implement automated data collection and feedback loops to adapt to evolving fake news tactics.
    - Be prepared to update the system and algorithms as needed to maintain effectiveness.

8. **Feedback**
    - Regularly gather feedback from users, including end-users, fact-checkers, and other stakeholders.
    - Use this feedback to make ongoing improvements to the fake news detection system and its user interface.

9. **Iterate**
    - Continue the design thinking process by revisiting the previous stages to refine and enhance the system based on changing needs and emerging challenges related to fake news.

The design thinking process is iterative, and it places a strong emphasis on empathy and user-centric design, ensuring that the fake news detection solution remains effective and adaptable to the evolving landscape of misinformation and disinformation.

## PHASES OF DEVELOPMENT

**Phase1:** Problem Definition

**Phase2:** Innovation

**Phase3:** Development Part1

**Phase4:** Development Part2

## DATASET USED

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

The dataset "Fake and Real News" contains two separate datasets of news articles, one for real news and one for fake news. These datasets are valuable for research and machine learning projects related to fake news detection. Here are the details about these datasets:

**1. Real News Dataset:**

This dataset includes news articles that are considered "real" or "genuine." These articles are typically sourced from reputable news outlets and are expected to contain accurate and reliable information.

The articles in this dataset are typically factual and are not intended to deceive or mislead readers.

Each entry in the dataset contains the following information:

**Title:** The headline or title of the news article.

**Text:** The content of the news article.

**Subject:** The subject or topic of the news article (e.g., politics, world news).

**Date:** The publication date of the news article.

**2. Fake News Dataset:**

This dataset includes news articles that are intentionally fabricated or misleading. The articles are often created to deceive readers, spread false information, or promote a specific agenda.

Fake news articles can be written to mimic the style and format of real news articles, making them more challenging to detect.

Each entry in the dataset typically contains the following information:

**Title:** The headline or title of the fake news article.

**Text:** The content of the fake news article.

**Subject:** Some fake news articles may claim to cover legitimate news subjects to appear more credible.

**Date:** The publication date, which may be fabricated or altered.

## DATA PREPROCESSING STEPS

Data preprocessing is a critical step in fake news detection using NLP. Properly processed data improves the accuracy and effectiveness of the NLP model. Here are the essential data preprocessing steps for fake news detection:

1. **Text Cleaning**
   - Remove HTML tags, special characters, punctuation, and non-alphanumeric characters.
   - Convert text to lowercase to ensure uniformity in the data.
2. **Handling Numerical Data**
   - Convert numerical values to a standardized format or replace them with placeholders to maintain text data consistency.

3. **Data Visualization and Exploratory Data Analysis (EDA):**
    - Generate visualizations and summary statistics to better understand the data and identify patterns or anomalies.
4. **Data Encoding and Scaling**
    - Scale or normalize features to ensure they have the same scale, which is essential for algorithms like gradient descent.
    - Encode target variables (labels) if they are categorical using techniques like label encoding.
5. **Data Splitting:**
    - Divide the dataset into training, validation, and test sets to facilitate model development and evaluation.
    - Consider cross-validation for robust model assessment.

## IMPORTING LIBRARIES

Importing libraries is a fundamental step when working on data analysis, machine learning, or any programming task in Python. You import libraries to access their functions, classes, and modules that provide various capabilities.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf

import re

from tensorflow.keras.preprocessing.text import Tokenizer

import tensorflow as tf

from sklearn.metrics import accuracy_score

from sklearn.model_selection import train_test_split

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

import seaborn as sns

plt.style.use('ggplot')
```

**LOADING DATA :**

To load data into your Python script or Jupyter Notebook, you'll typically use libraries like Pandas or NumPy for structured data, or built-in functions for reading various file formats.

**CODE:**

```
fake_df = pd.read_csv('../input/fake-and-real-news-dataset/Fake.csv')

real_df = pd.read_csv('../input/fake-and-real-news-dataset/True.csv')
```

**CODE:**

```
fake_df.head(10)
```

**OUTPUT:**

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | Donald Trump Sends Out Embarrassing New Year'... | Donald Trump just couldn t wish all Americans ... | News | December 31, 2017 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | News | December 31, 2017 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwauk... | News | December 30, 2017 |

| title | text | subject | date | |
|---|---|---|---|---|
| 3 | Trump Is So Obsessed He Even Has Obama’s Name... | On Christmas day, Donald Trump announced that ... | News | December 29, 2017 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | News | December 25, 2017 |
| 5 | Racist Alabama Cops Brutalize Black Boy While... | The number of cases of cops brutalizing and ki... | News | December 25, 2017 |
| 6 | Fresh Off The Golf Course, Trump Lashes Out A... | Donald Trump spent a good portion of his day a... | News | December 23, 2017 |
| 7 | Trump Said Some INSANELY Racist Stuff Inside ... | In the wake of yet another court decision that... | News | December 23, 2017 |
| 8 | Former CIA Director Slams Trump Over UN Bully... | Many people have raised the alarm regarding th... | News | December 22, 2017 |
| 9 | WATCH: Brand-New Pro-Trump Ad Features So Muc... | Just when you might have thought we d get a br... | News | December 21, 2017 |

**CHECKING FOR NULL VALUES:**

**CODE:**

```
fake_df.isnull().sum()
```

**OUTPUT:**

```
title      0
text       0
subject    0
date       0
dtype: int64
```

**CODE:**

```
real_df.isnull().sum()
```

**OUTPUT:**

```
title      0
text       0
subject    0
date       0
dtype: int64
```

**CHECKING FOR UNIQUE VALUES OF SUBJECT:**

Both dataframes must be of similar distribution.

**CODE:**

```
fake_df.subject.unique()
```

**OUTPUT:**

```
array(['News', 'politics', 'Government News', 'left-news', 'US_News',
```

'Middle-east'], dtype=object)

**CODE:**

```
real_df.subject.unique()
```

**OUTPUT:**

```
array(['politicsNews', 'worldnews'], dtype=object)
```

**CODE:**

```
fake_df.drop(['date', 'subject'], axis=1, inplace=True)
real_df.drop(['date', 'subject'], axis=1, inplace=True)
```

**CODE:**

```
fake_df['class'] = 0
real_df['class'] = 1
```

**CODE:**

```
data = pd.concat([false_df, true_df], ignore_index = True)
data
```

**VISUALISATION:**

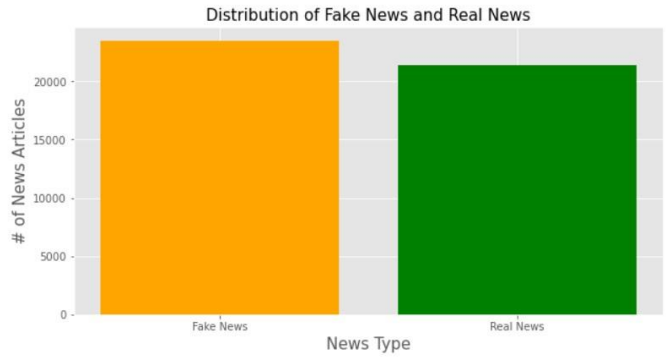
**CODE:**

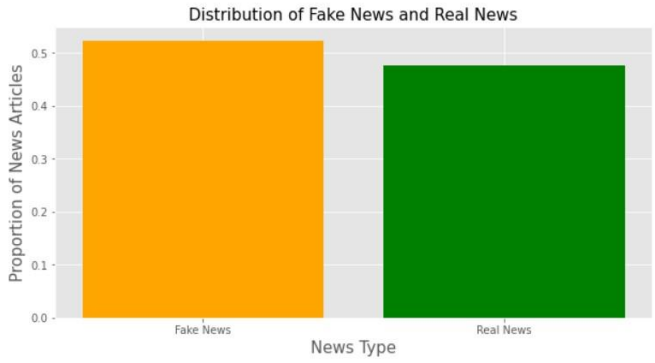
```
plt.figure(figsize=(10, 5))
plt.bar('Fake News', len(fake_df), color='orange')
plt.bar('Real News', len(real_df), color='green')
plt.title('Distribution of Fake News and Real News', size=15)
plt.xlabel('News Type', size=15)
plt.ylabel('# of News Articles', size=15)
```

```
total_len = len(fake_df) + len(real_df)
plt.figure(figsize=(10, 5))
plt.bar('Fake News', len(fake_df) / total_len, color='orange')
plt.bar('Real News', len(real_df) / total_len, color='green')
plt.title('Distribution of Fake News and Real News', size=15)
plt.xlabel('News Type', size=15)
plt.ylabel('Proportion of News Articles', size=15)
```

**OUTPUT:**

```
Text(0, 0.5, 'Proportion of News Articles')
```




**CODE:**

```
print('Difference in news articles:',len(fake_df)-len(real_df))
```

**OUTPUT:**

Difference in news articles: 2064

**CODE:**

```
news_df = pd.concat([fake_df, real_df], ignore_index=True, sort=False)
news_df
```

**OUTPUT:**

| title | text | class | |
|---|---|---|---|
| 0 | Donald Trump Sends Out Embarrassing New Year'... | Donald Trump just couldn t wish all Americans ... | 0 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | 0 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwauk... | 0 |
| 3 | Trump Is So Obsessed He Even Has Obama's Name... | On Christmas day, Donald Trump announced that ... | 0 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | 0 |
| ... | ... | ... | ... |

| title | text | class | |
|---|---|---|---|
| 44893 | 'Fully committed' NATO backs new U.S. approach... | BRUSSELS (Reuters) - NATO allies on Tuesday we... | 1 |
| 44894 | LexisNexis withdrew two products from Chinese ... | LONDON (Reuters) - LexisNexis, a provider of l... | 1 |
| 44895 | Minsk cultural hub becomes haven from authorities | MINSK (Reuters) - In the shadow of disused Sov... | 1 |
| 44896 | Vatican upbeat on possibility of Pope Francis ... | MOSCOW (Reuters) - Vatican Secretary of State ... | 1 |
| 44897 | Indonesia to buy $1.14 billion worth of Russia... | JAKARTA (Reuters) - Indonesia will buy 11 Sukh... | 1 |

**CODE:**

```
news_df['text'] = news_df['title'] + news_df['text']

news_df.drop('title', axis=1, inplace=True)
```

**CODE:**

```
features = news_df['text']

targets = news_df['class']


X_train, X_test, y_train, y_test = train_test_split(features, targets, test_size=0.20,
random_state=18
```

**NORMALISATION:**

Getting rid of extra spaces and url links.

**CODE:**

```
def normalize(data):
    normalized = []
    for i in data:
        i = i.lower()
        # get rid of urls
        i = re.sub('https?://\S+|www\.\S+', '', i)
        # get rid of non words and extra spaces
        i = re.sub('\\W', ' ', i)
        i = re.sub('\n', '', i)
        i = re.sub(' +', ' ', i)
        i = re.sub('^ ', '', i)
        i = re.sub(' $', '', i)
        normalized.append(i)
    return normalized

X_train = normalize(X_train)
X_test = normalize(X_test)
```

**VECTORISATION:**

**CODE:**

```
max_vocab = 10000

tokenizer = Tokenizer(num_words=max_vocab)

tokenizer.fit_on_texts(X_train)

#Convert text to vectors, our classifier only takes numerical data.

# tokenize the text into vectors


X_train = tokenizer.texts_to_sequences(X_train)

X_test = tokenizer.texts_to_sequences(X_test)


#Apply padding so we have the same length for each article

X_train = tf.keras.preprocessing.sequence.pad_sequences(X_train, padding='post',
maxlen=256)

X_test = tf.keras.preprocessing.sequence.pad_sequences(X_test, padding='post',
maxlen=256)

#Building the RNN.

model = tf.keras.Sequential([

tf.keras.layers.Embedding(max_vocab, 128),

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64,  return_sequences=True)),

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(16)),

tf.keras.layers.Dense(64, activation='relu'),

tf.keras.layers.Dropout(0.5),
```

```
    tf.keras.layers.Dense(1)
    ])

model.summary()
```

**OUTPUT:**

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 128)         1280000
_________________________________________________________________
bidirectional (Bidirectional (None, None, 128)         98816
_________________________________________________________________
bidirectional_1 (Bidirection (None, 32)                18560
_________________________________________________________________
dense (Dense)                (None, 64)                2112
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 1,399,553
Trainable params: 1,399,553
Non-trainable params: 0
_________________________________________________________________
```

**CODE:**

```
    early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2,
    restore_best_weights=True)

    model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),

        optimizer=tf.keras.optimizers.Adam(1e-4),
```

```
                metrics=['accuracy'])


        history = model.fit(X_train, y_train, epochs=10,validation_split=0.1,
        batch_size=30, shuffle=True, callbacks=[early_stop])
```

**OUTPUT:**

```
Epoch 1/10
1078/1078 [==============================] - 60s 56ms/step - loss: 0.2757 -
accuracy: 0.8577 - val_loss: 0.0839 - val_accuracy: 0.9755
Epoch 2/10
1078/1078 [==============================] - 59s 55ms/step - loss: 0.0759 -
accuracy: 0.9807 - val_loss: 0.0446 - val_accuracy: 0.9852
Epoch 3/10
1078/1078 [==============================] - 60s 55ms/step - loss: 0.0451 -
accuracy: 0.9896 - val_loss: 0.0441 - val_accuracy: 0.9869
Epoch 4/10
1078/1078 [==============================] - 60s 55ms/step - loss: 0.0793 -
accuracy: 0.9761 - val_loss: 0.0860 - val_accuracy: 0.9710
Epoch 5/10
1078/1078 [==============================] - 59s 55ms/step - loss: 0.0519 -
accuracy: 0.9855 - val_loss: 0.0456 - val_accuracy: 0.9839
```

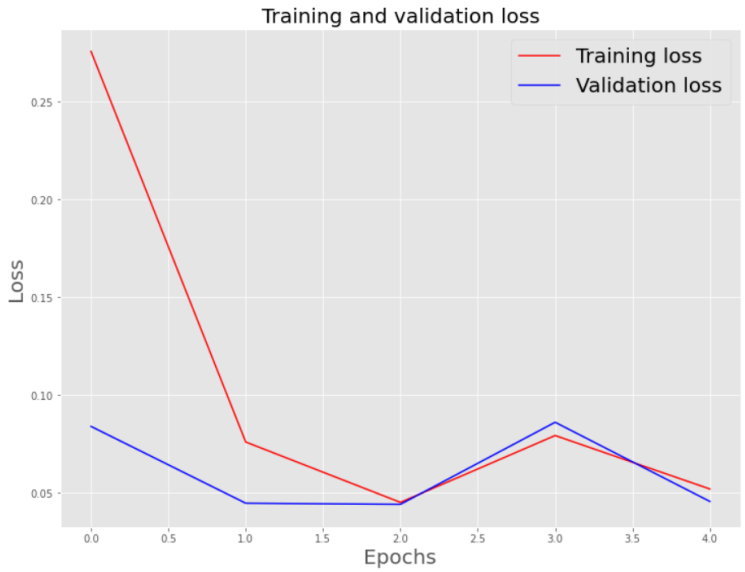
**VISUALISATION:**

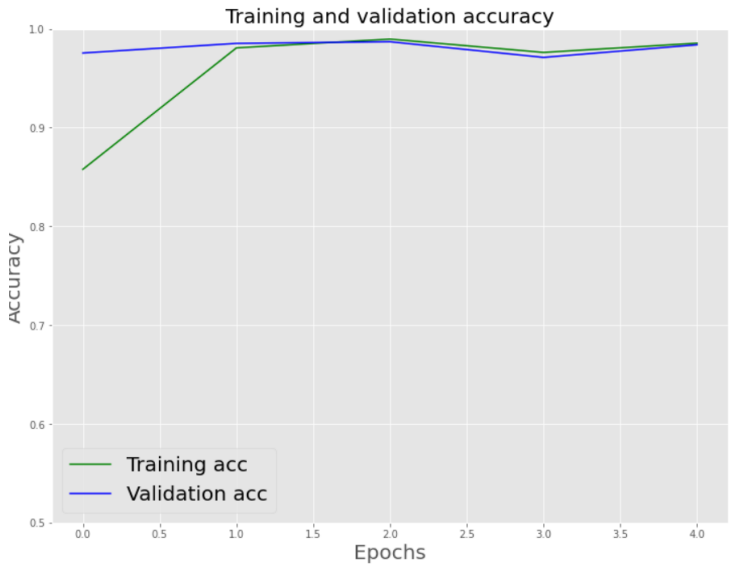
**CODE:**

```
        history_dict = history.history


        acc = history_dict['accuracy']

        val_acc = history_dict['val_accuracy']

        loss = history_dict['loss']
```

```
val_loss = history_dict['val_loss']
epochs = history.epoch

plt.figure(figsize=(12,9))
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss', size=20)
plt.xlabel('Epochs', size=20)
plt.ylabel('Loss', size=20)
plt.legend(prop={'size': 20})
plt.show()

plt.figure(figsize=(12,9))
plt.plot(epochs, acc, 'g', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy', size=20)
plt.xlabel('Epochs', size=20)
plt.ylabel('Accuracy', size=20)
plt.legend(prop={'size': 20})
plt.ylim((0.5,1))
plt.show()
```

**OUTPUT:**




**EVALUATING TEST SET:**

**CODE:**

```
model.evaluate(X_test, y_test)
```

**OUTPUT:**

```
[0.050108399242162704, 0.9860801696777344]
```

**CODE:**

```
pred = model.predict(X_test)


binary_predictions = []


for i in pred:
    if i >= 0.5:
        binary_predictions.append(1)
    else:
        binary_predictions.append(0)
print('Accuracy on testing set:', accuracy_score(binary_predictions, y_test))
print('Precision on testing set:', precision_score(binary_predictions, y_test))
print('Recall on testing set:', recall_score(binary_predictions, y_test))
```

**OUTPUT:**

```
Accuracy on testing set: 0.9860801781737194
Precision on testing set: 0.9812413154238073
Recall on testing set: 0.9897220275636534
```

**CONFUSION MATRIX:**

**CODE:**

```
matrix = confusion_matrix(binary_predictions, y_test, normalize='all')
plt.figure(figsize=(16, 10))
ax= plt.subplot()
sns.heatmap(matrix, annot=True, ax = ax)
```

```
# labels, title and ticks
ax.set_xlabel('Predicted Labels', size=20)
ax.set_ylabel('True Labels', size=20)
ax.set_title('Confusion Matrix', size=20)
ax.xaxis.set_ticklabels([0,1], size=15)
ax.yaxis.set_ticklabels([0,1], size=15)
```

**OUTPUT:**