# FAKE NEWS DETECTION USING NLP

**Team Leader**

**422521104037:SYED JALEEL S**

## Phase-3: Development Part 1

**Project**: Fake News Detection


## Introduction:

- Fake news detection with NLP is like teaching a computer to spot lies. NLP helps the computer understand language and find strange things in news articles that might mean they're fake. It checks how words are used and the feeling of the text.

- Computers use what they learn to tell us if a news story is real or not. This is really important in today's world where wrong information spreads fast. NLP, along with smart machines, like GPT-3 and BERT, can help stop fake news by looking at the words and how they fit together.

- Using NLP to fight fake news helps make sure we get true information, making us smarter news readers.

**Content:**

- Phase 3 involves building the project, which includes obtaining the fake and real news datasets from the Kaggle website.
- The Kaggle dataset comprises two files: one for real news and another for fake news.
- During this phase, after loading the dataset, we will carry out preprocessing steps.

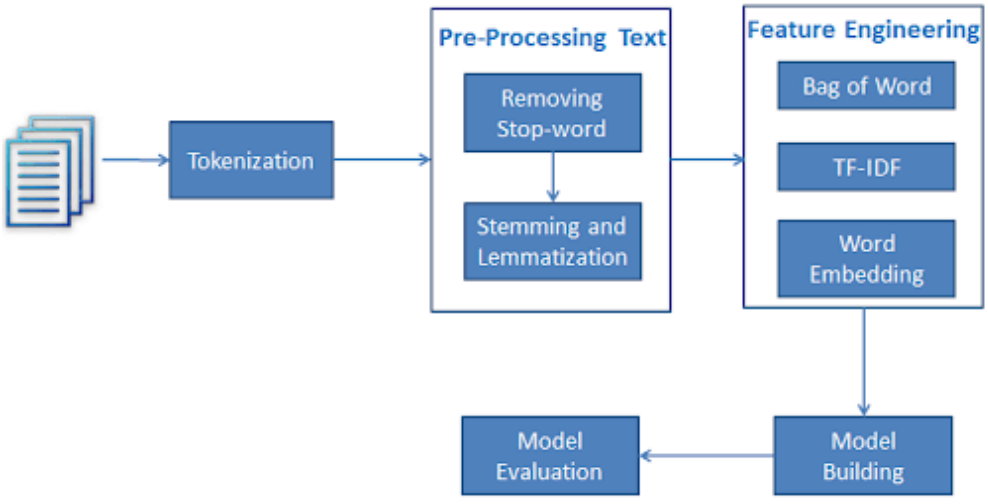
**Phase 3 steps:**

- Loading
- Preprocessing

**Dataset Link:**

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

**Overview:**




## Preprocessing :

- In any machine learning task, getting the data ready (cleaning and organizing) is just as important, if not more so, than creating the actual model. This is especially critical when working with messy, unstructured text data.

# Preprocessing Steps:

- Lower casing
- Tokenization
- Removal of Punctuations
- Removal of Stopwords
- HTML Tag Removal
- Stemming
- Removing Non-Alphanumeric Characters
- Vectorization
- Splitting the Dataset

# Program:

### Import necessary libraries:

```
import pandas as pd
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from sklearn.feature_extraction.text import
TfidfVectorizer

# Download NLTK resources
nltk.download('punkt')
nltk.download('stopwords')
```

### Load the 'Fake.csv' and 'True.csv' datasets:

```
fake_df = pd.read_csv('Fake.csv')
real_df = pd.read_csv('True.csv')
fake_df.head(2)
real_df.head(2)
```

Out[22]:

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 |

Out[11]:

| | 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 |

## 1. Lowercasing:

i. Lowercasing involves converting all letters in text to lowercase.
ii. It ensures uniformity and case insensitivity in text data.
iii. Lowercasing simplifies text analysis and comparisons by normalizing the text.

**Code:**

```
# 1. Lowercasing
fake_df['title'] = fake_df['title'].str.lower()
fake_df['text'] = fake_df['text'].str.lower()

real_df['title'] = real_df['title'].str.lower()
real_df['text'] = real_df['text'].str.lower()

print("1. Lowercasing:")
print("Fake news")
print(fake_df.head(2))

print("Real news")
print(real_df.head(2))
```

**Output:**

```
1. Lowercasing:
Fake news
                                               title  \
0   donald trump sends out embarrassing new year'...
1   drunk bragging trump staffer started russian ...

                                                text subject  \
0  donald trump just couldn t wish all americans ...    News
1  house intelligence committee chairman devin nu...    News

               date
0  December 31, 2017
1  December 31, 2017
Real news
                                               title  \
0  as u.s. budget fight looms, republicans flip t...
1  u.s. military to accept transgender recruits o...

                                                text       subject  \
0  washington (reuters) - the head of a conservat...  politicsNews
1  washington (reuters) - transgender people will...  politicsNews

                date
0  December 31, 2017
1  December 29, 2017
```

## 2.Tokenization:

- Tokenization is the process of breaking down a text into smaller units, typically words or subwords, referred to as "tokens." These tokens are the fundamental building blocks for various NLP tasks.

- Tokenization is a crucial preprocessing step because it helps convert unstructured text into a structured format, making it easier for machines to understand and process. It's essential for tasks like language modeling, text classification, and machine translation.

**Code:**

```
# Download NLTK resources
import nltk
nltk.download('punkt')

# Load the 'Fake.csv' and 'True.csv' datasets
import pandas as pd

fake_df = pd.read_csv('Fake.csv')
real_df = pd.read_csv('True.csv')

# Apply tokenization to the 'text' and 'title' columns
fake_df['text'] = fake_df['text'].apply(lambda x: nltk.word_tokenize(x))
fake_df['title'] = fake_df['title'].apply(lambda x: nltk.word_tokenize(x))
real_df['text'] = real_df['text'].apply(lambda x: nltk.word_tokenize(x))
real_df['title'] = real_df['title'].apply(lambda x: nltk.word_tokenize(x))

# Display the first few rows with tokenization applied to both 'text' and
'title' columns
print("Fake News DataFrame (with tokenization):")
print(fake_df.head())

print("\nReal News DataFrame (with tokenization):")
print(real_df.head())
```

**Output:**

```
Fake News DataFrame (with tokenization):
                                               title  \
0  [Donald, Trump, Sends, Out, Embarrassing, New,...
1  [Drunk, Bragging, Trump, Staffer, Started, Rus...
2  [Sheriff, David, Clarke, Becomes, An, Internet...
3  [Trump, Is, So, Obsessed, He, Even, Has, Obama...
4  [Pope, Francis, Just, Called, Out, Donald, Tru...

                                                text subject  \
0  [Donald, Trump, just, couldn, t, wish, all, Am...    News
1  [House, Intelligence, Committee, Chairman, Dev...    News
2  [On, Friday, ,, it, was, revealed, that, forme...    News
3  [On, Christmas, day, ,, Donald, Trump, announc...    News
4  [Pope, Francis, used, his, annual, Christmas, ...    News

               date
0  December 31, 2017
1  December 31, 2017
2  December 30, 2017
3  December 29, 2017
4  December 25, 2017
```

## 3.Removal of Stopwords:

- Stop words are common words (e.g., "the," "and," "in") that occur frequently in a language but often carry little meaningful information in a given context. Stop words removal is the process of eliminating these words from text data.

- The removal of stop words is performed to reduce the dimensionality of text data and focus on more informative words. This can improve the efficiency of text analysis, such as text classification or sentiment analysis, by reducing noise in the data.

- **Examples:** Common examples of stop words in English include "a," "an," "the," "of," and "in." By removing these words, the remaining words in the text are often more meaningful for understanding the content and context of the text.

**Code:**

```
# Download NLTK resources (stopwords)
nltk.download('stopwords')

# Get and join the list of English stopwords
stop_words = ",
".join(nltk.corpus.stopwords.words('english'))

# Display the list of English stopwords
print("List of English Stopwords:")
print(stop_words)
```

```
List of English Stopwords:
i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves,
he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselve
s, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, h
ad, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, a
bout, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, o
ver, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most,
other, some, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, don't, should, s
hould've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, had
n't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't,
shouldn, shouldn't, wasn, wasn't, weren, weren't, won, won't, wouldn, wouldn't
```

**Code:**

```
STOPWORDS = set(stopwords.words('english'))

def remove_stopwords(text):
    return " ".join([word for word in str(text).split() if word.lower() not in STOPWORDS])

# Apply stopword removal to the 'text' and 'title' columns for both datasets
fake_df["text_wo_stop"] = fake_df["text"].apply(lambda text: remove_stopwords(text))
fake_df["title_wo_stop"] = fake_df["title"].apply(lambda text: remove_stopwords(text))
real_df["text_wo_stop"] = real_df["text"].apply(lambda text: remove_stopwords(text))
real_df["title_wo_stop"] = real_df["title"].apply(lambda text: remove_stopwords(text))

# Display the first few rows with stopwords removed for both 'text' and 'title' columns
print("Fake News DataFrame (with stopwords removed):")
print(fake_df.head(2))

print("\nReal News DataFrame (with stopwords removed):")
print(real_df.head(2))
```

**Output:**

```
Fake News DataFrame (with stopwords removed):
                                                title  \
0   Donald Trump Sends Out Embarrassing New Year'...
1   Drunk Bragging Trump Staffer Started Russian ...

                                                text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News

                date                                          text_wo_stop  \
0  December 31, 2017  Donald Trump wish Americans Happy New Year lea...
1  December 31, 2017  House Intelligence Committee Chairman Devin Nu...

                                      title_wo_stop
0  Donald Trump Sends Embarrassing New Year’s Eve...
1  Drunk Bragging Trump Staffer Started Russian C...
```

## 4. Stemming:

- Stemming is the process of reducing words to their base or root form by removing suffixes. For example, "jumping" is stemmed to "jump," and "happily" becomes "happi." This is done to group similar words together and simplify text analysis.

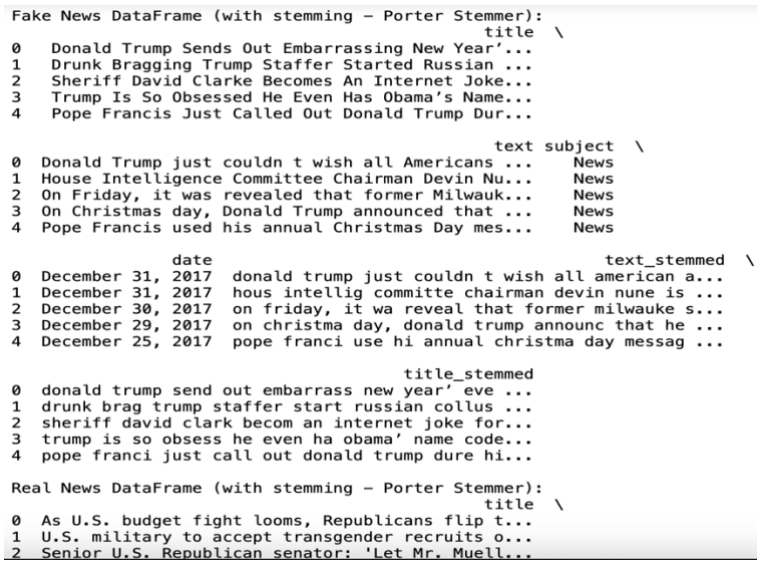
- Stemming helps in text simplification, as it reduces various word forms to a common representation. This can improve the efficiency and effectiveness of natural language processing tasks like information retrieval or document classification.

- While stemming can be beneficial, it has limitations. Sometimes, it may produce incorrect or non-dictionary words as stems. Additionally, the choice of a stemming algorithm can impact the results, so selecting the appropriate one for a specific task is essential.

**Code:**

```
# Initialize the Porter Stemmer
stemmer = PorterStemmer()

# Define a function to stem words
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])

# Apply stemming to the 'text' and 'title' columns for both datasets
fake_df["text_stemmed"] = fake_df["text"].apply(lambda text: stem_words(text))
fake_df["title_stemmed"] = fake_df["title"].apply(lambda text: stem_words(text))
real_df["text_stemmed"] = real_df["text"].apply(lambda text: stem_words(text))
real_df["title_stemmed"] = real_df["title"].apply(lambda text: stem_words(text))

# Display the first few rows with stemming applied to both 'text' and 'title' columns
print("Fake News DataFrame (with stemming - Porter Stemmer):")
print(fake_df.head())

print("\nReal News DataFrame (with stemming - Porter Stemmer):")
print(real_df.head())
```

**Output:**



```
Fake News DataFrame (with stemming - Porter Stemmer):
                                                title  \
0   Donald Trump Sends Out Embarrassing New Year'...
1   Drunk Bragging Trump Staffer Started Russian ...
2   Sheriff David Clarke Becomes An Internet Joke...
3   Trump Is So Obsessed He Even Has Obama's Name...
4   Pope Francis Just Called Out Donald Trump Dur...

                                              text subject  \
0  Donald Trump just couldn t wish all Americans ...    News
1  House Intelligence Committee Chairman Devin Nu...    News
2  On Friday, it was revealed that former Milwauk...    News
3  On Christmas day, Donald Trump announced that ...    News
4  Pope Francis used his annual Christmas Day mes...    News

                date                                       text_stemmed  \
0  December 31, 2017  donald trump just couldn t wish all american a...
1  December 31, 2017  hous intellig committe chairman devin nune is ...
2  December 30, 2017  on friday, it wa reveal that former milwauke s...
3  December 29, 2017  on christma day, donald trump announc that he ...
4  December 25, 2017  pope franci use hi annual christma day messag ...

                                     title_stemmed
0  donald trump send out embarrass new year' eve ...
1  drunk brag trump staffer start russian collus ...
2  sheriff david clark becom an internet joke for...
3  trump is so obsess he even ha obama' name code...
4  pope franci just call out donald trump dure hi...

Real News DataFrame (with stemming - Porter Stemmer):
                                                title  \
0  As U.S. budget fight looms, Republicans flip t...
1  U.S. military to accept transgender recruits o...
2  Senior U.S. Republican senator: 'Let Mr. Muell...
```

## 5. Lemmatization:

- Lemmatization retains the base form of words, which can preserve the core meaning of a word. Removing lemmatization would result in various inflections of a word that might affect the understanding of the text.

- Lemmatization is valuable in linguistic analysis, as it reduces words to their canonical form, making it easier to identify relationships between words, understand context, and perform more accurate language processing tasks.

**Code:**

```
#Initialize the WordNet Lemmatizer
lemmatizer = WordNetLemmatizer()

# Define a function to lemmatize words
def lemmatize_words(text):
   return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

# Apply lemmatization to the 'text' and 'title' columns for both datasets
fake_df["text_lemmatized"] = fake_df["text"].apply(lambda text: lemmatize_words(text))
fake_df["title_lemmatized"] = fake_df["title"].apply(lambda text: lemmatize_words(text))
real_df["text_lemmatized"] = real_df["text"].apply(lambda text: lemmatize_words(text))
real_df["title_lemmatized"] = real_df["title"].apply(lambda text: lemmatize_words(text))

# Display the first few rows with lemmatization applied to both 'text' and 'title' columns
print("Fake News DataFrame (with lemmatization - WordNet Lemmatizer):")
print(fake_df.head())

print("\nReal News DataFrame (with lemmatization - WordNet Lemmatizer):")
print(real_df.head())
```

### 6. HTML Tag Removal:

- HTML (Hypertext Markup Language) tags are used to format and structure web content. HTML tag removal in text preprocessing involves stripping away these tags from text data, leaving only the textual content.

- Removing HTML tags is crucial to obtain clean and readable text data for natural language processing tasks. It ensures that the analysis focuses on the actual text content and not the formatting or structural elements used in web pages.

- HTML tag removal is essential when working with web scraping or extracting text from web pages, as it allows you to extract and process the relevant textual information while ignoring HTML-specific formatting or layout elements that have no linguistic meaning.

**Code:**

```
# Define a function to remove HTML tags
def remove_html_tags(text):
    return [re.sub(r'<[^>]+>', '', word) for word in text]

# Apply HTML tag removal to the 'text' and 'title' columns for both
datasets
fake_df["text"] = fake_df["text"].apply(lambda tokens:
remove_html_tags(tokens))
fake_df["title"] = fake_df["title"].apply(lambda tokens:
remove_html_tags(tokens))
```

```
real_df["text"] = real_df["text"].apply(lambda tokens:
remove_html_tags(tokens))
real_df["title"] = real_df["title"].apply(lambda tokens:
remove_html_tags(tokens))

# Display the first few rows with HTML tag removal applied to both 'text'
and 'title' columns
print("Fake News DataFrame (with HTML Tag Removal):")
print(fake_df.head(2))

print("\nReal News DataFrame (with HTML Tag Removal):")
print(real_df.head(2))
```

### 7. TF-IDF vectorization:

- TF-IDF is a numerical representation of text data used in natural language processing. It assigns a weight to each word in a document, considering both its frequency within the document (TF) and its importance across the entire collection of documents (IDF). Words that are frequent in a document but rare in the corpus receive higher TF-IDF scores.

- TF-IDF helps identify words or terms that are distinctive and important for a particular document. It downweights common words like "the" and "and" and highlights unique terms that carry more meaning and context. This weighting scheme is valuable for information retrieval, document classification, and text analysis.

- TF-IDF vectorization is widely used in various natural language processing tasks, such as information retrieval, document similarity, text classification, and topic modeling. It converts text data into numerical vectors, making it suitable for machine learning algorithms to process and analyze textual content effectively.

**Code:**

```
#Feature Engineering (TF-IDF vectorization as an example)
tfidf_vectorizer = TfidfVectorizer()
fake_df['text'] = fake_df['text'].apply(lambda tokens: '
'.join(tokens))
real_df['text'] = real_df['text'].apply(lambda tokens: ' '.join(tokens))
fake_df['text'] = tfidf_vectorizer.fit_transform(fake_df['text'])
real_df['text'] = tfidf_vectorizer.fit_transform(real_df['text'])

print("\n10. Feature Engineering (TF-IDF Vectorization):")
print(fake_df.head(2))
print(real_df.head(2))
```

**8.Split the data into training and testing sets (80% training, 20% testing):**

- Splitting a dataset involves dividing it into two subsets: a training set and a testing set. The training set is used to train a machine learning model, while the testing set is used to evaluate the model's performance. This separation helps in assessing how well the model generalizes to unseen data.

- To ensure fairness and reliability, it's common to randomly split the dataset. Randomization helps prevent any bias in the data and ensures that the model's performance is not dependent on the order of the data points.

- The most common split ratios are 70-30 or 80-20, where a majority of the data is used for training (70% or 80%), and the remaining portion is used for testing (30% or 20%). The exact split ratio may vary depending on the specific problem and dataset size, but maintaining a sufficient amount of data for testing is crucial for reliable model evaluation.

**Code:**

```
# Define a label for each dataset (0 for fake, 1 for true)
fake_df['label'] = 0
true_df['label'] = 1

# Combine the datasets into one, assuming the same columns
combined_df = pd.concat([fake_df, true_df], ignore_index=True)

# Split the dataset into features (X) and labels (y)
X = combined_df['text']
y = combined_df['label']

# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

## Conclusion:

- The eight preprocessing steps mentioned above help identify and handle data quality issues, such as missing values, outliers, and noise, which can negatively impact the accuracy and reliability of analysis and modeling.

- In summary, loading and preprocessing the dataset are essential steps in developing an effective fake news detection system.

- The preprocessing steps are designed to clean and prepare the textual data for machine learning, ensuring that the model can effectively distinguish between fake and real news articles.

- Properly preprocessed data is key to building accurate and reliable fake news classification models, which play a crucial role in safeguarding users from misinformation and potentially harmful content."