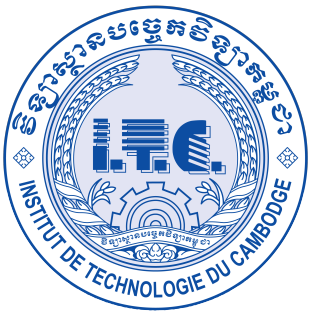
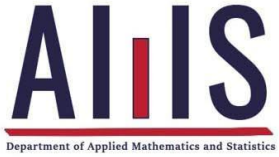




Institute of Technology of Cambodia

Data Ethics and Privacy

2024-2025

4th Year of Engineering Degree in Data Science

Department of Applied Mathematics and Statistics

Fake News Detection

Lecturers

Mr. Sok Kimheng (Course)
Dr. Neang Pheak (TP)
Dr. Phauk Sokkhey (TP)

Group Members::

ENG Sive Eu e20210914
EN Sreytoch e20210085
EN Sreythom e20210084

Contents

Abstract

The topic of fake news has drawn attention both from the public and the academic communities. Such misinformation has the potential of affecting public opinion, providing an opportunity for malicious parties to manipulate the outcomes of public events such as elections. Because such high stakes are at play, automatically detecting fake news is an important, yet challenging problem that is not yet well understood. Nevertheless, there are three generally agreed-upon characteristics of fake news: the text of an article, the user response it receives, and the source users promoting it. Existing work has largely focused on tailoring solutions to one particular characteristic, which has limited their success and generality.

In this work, we propose a model that combines all three characteristics for a more accurate and automated prediction. Specifically, we incorporate the behavior of both parties, users and articles, and the group behavior of users who propagate fake news. Motivated by the three characteristics, we propose a model called CSI, which is composed of three modules: Capture, Score, and Integrate. The first module is based on the response and text; it uses a Recurrent Neural Network to capture the temporal pattern of user activity on a given article. The second module learns the source characteristic based on the behavior of users, and the two are integrated with the third module to classify an article as fake or not. Experimental analysis on real-world data demonstrates that CSI achieves higher accuracy than existing models and extracts meaningful latent representations of both users and articles.
Key Words: Fake news detection, Neural networks, Deep learning, Social networks, Group anomaly detection, Temporal analysis.

1 Introduction

The rise of digital media and social platforms has transformed the way information is shared and consumed, leading to unprecedented access to news and opinions. However, alongside this vast flow of information has emerged the pervasive issue of fake news—false or misleading information presented as factual news. Fake news can be intentionally created for various reasons, including political manipulation, financial gain, or simply to attract attention.

The rapid spread of fake news, particularly on social media, poses significant challenges to individuals, organizations, and societies as a whole. It can shape public opinion, influence elections, and even incite violence or societal unrest. In the digital age, where misinformation can be disseminated in seconds and reach millions of people, it has become increasingly crucial to develop effective strategies for detecting and mitigating fake news.

Fake news detection involves the application of computational methods, often leveraging machine learning, natural language processing, and data mining techniques, to identify and flag content that is likely to be misleading or false. The goal of fake news detection is not only to identify these pieces of content but also to better understand the patterns and dynamics that allow misinformation to spread.

This report explores various approaches to fake news detection, focusing on the use of artificial intelligence and machine learning algorithms, as well as the challenges involved in distinguishing legitimate news from fake news. By identifying potential solutions, this work aims to contribute to the ongoing efforts to reduce the impact of misinformation and promote the dissemination of accurate, trustworthy information in the digital age.

1.1 Problem Statement

Although numerous approaches to detecting fake news exist, many models struggle with issues like domain adaptability, language complexity, and scalability. Traditional machine learning models may not capture the nuanced textual patterns associated with fake news, resulting in suboptimal accuracy and higher false detection rates.

Delimitations

- Our system does not guarantee 100%.
- The system is unable to test data that is unrelated to the training dataset.

1.2 Objectives

1. Develop a comprehensive dataset with labeled news articles for training and testing.
2. Explore advanced NLP techniques, such as word embeddings and transformers, for feature extraction.
3. Compare the performance of various machine learning models, including:
    - Logistic Regression,
    - Bayes Net
    - Naive Bayes
    - Support Vector Machines (SVM),
    - Deep learning architectures like Long Short-Term Memory (LSTM).
4. Address the problem of fake news detection as a classic text classification task.

2 Literature Reviews

2.1 Overview of Fake news

Fake news detection encompass a range of techniques, from traditional manual fact-checking to advanced machine learning and natural language processing (NLP) approaches. Manual fact-checking involves human experts verifying the accuracy of claims by cross-referencing information with reliable sources. While effective, this method is time-consuming and may not scale well for the vast amount of content online.

Machine learning techniques have been widely applied in fake news detection. These approaches involve training algorithms on labeled datasets of fake and real news to classify new articles. Supervised learning algorithms, such as Support Vector Machines (SVM), Random Forests, and Neural Networks, are commonly used for this purpose. Unsupervised learning methods like clustering and anomaly detection can also be employed to detect unusual patterns in news content.

NLP techniques play a crucial role in fake news detection by enabling systems to analyze and understand textual data. Sentiment analysis, topic modeling, and linguistic pattern recognition are commonly used NLP methods. Additionally, advanced NLP models like BERT have shown promise in capturing nuanced language cues that can indicate the credibility of news content.

Hybrid approaches combining machine learning and NLP techniques have also gained traction in fake news detection. By leveraging the strengths of both methodologies, these hybrid models aim to improve the accuracy and robustness of fake news detection systems.

2.2 Challenges Identified

Challenges identified in current approaches to fake news detection include:

1. Bias in Datasets: Many fake news detection models are trained on biased datasets, which can lead to skewed results. Biases in labeling, such as errors in identifying fake news or imbalanced distributions of fake and real news samples, can hinder the effectiveness of these models when applied to real-world scenarios.
2. Detecting Satire and Parody: Distinguishing between satirical content and actual fake news poses a significant challenge. Satirical news outlets often use irony, sarcasm, and humor to convey their message, making it difficult for algorithms to differentiate between satire and intentionally misleading information.
3. Rapid Evolution of Fake News Patterns: Fake news tactics are constantly evolving, adapting to circumvent detection methods. As creators of fake news become more sophisticated in their strategies, existing detection models may struggle to keep pace with the evolving landscape of misinformation.
4. Contextual Understanding: Fake news detection requires a deep understanding of context, nuances in language, and cultural references. Current models may struggle to grasp the subtleties of language and context, leading to misinterpretations and inaccurate classifications.
5. Generalization Across Languages and Cultures: Many fake news detection models are trained on English-language data, limiting their effectiveness in detecting misinformation in other languages or cultural contexts. Adapting models to different languages and cultural nuances remains a significant challenge.
6. Explainability and Interpretability: Interpreting the decisions made by fake news detection models can be challenging. Ensuring transparency and interpretability in the decision-making process of these models is crucial for building trust and understanding their limitations.
7. Scalability and Real-Time Detection: The sheer volume of online content makes real-time fake news detection a daunting task. Scalability issues arise when trying to process and analyze

large amounts of data quickly and accurately to identify and counteract fake news in a timely manner.

3 Methodology

3.1 Dataset

	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
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Figure 1: True Dataset

	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
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

Figure 2: Fake Dataset

Datasets Overview

1. Fake News Dataset

- **Size:** This dataset contains 23,481 rows and 4 columns.
- **Structure:**
  - **Title:** A brief headline summarizing the content of the article.
  - **Text:** The main body of the article, providing detailed information.
  - **Subject:** The category of the news (e.g., "politicsNews").
  - **Date:** The publication date of the article.
- **Source:** This dataset was obtained from Kaggle, a platform that provides curated datasets for data analysis and machine learning projects. It contains fabricated or misleading news articles, making it ideal for fake news detection tasks.

2. True News Dataset

- **Size:** This dataset contains 21,417 rows and 4 columns.
- **Structure:**

  - **Title:** A brief headline summarizing the content of the article.
  - **Text:** The main body of the article, providing detailed information.
  - **Subject:** The category of the news (e.g., "News").
  - **Date:** The publication date of the article.

- **Source:** This dataset was also sourced from Kaggle. It consists of legitimate, verified news articles, serving as a reliable counterpart to the Fake News Dataset.

Key Characteristics

- Both datasets are well-structured and contain no missing values, ensuring consistent and reliable analysis.
- The datasets provide a balanced representation of true and fake news articles, aiding in the development of robust machine learning models for fake news detection.

Updated Dataset with Labels

To facilitate supervised machine learning tasks, a new column, **label**, has been added to both the True News and Fake News datasets. This column serves as the classification label for the articles:

- **1:** Indicates true news (legitimate or verified articles).
- **0:** Indicates fake news (fabricated or misleading articles).

After adding the **label** column, both datasets now contain 5 columns each. The updated shapes of the datasets are:

- **True News Dataset:** (21,417 rows, 5 columns)
- **Fake News Dataset:** (23,481 rows, 5 columns)

The columns in the updated datasets are as follows:

- **Title:** A brief headline summarizing the content of the article.
- **Text:** The main body of the article, providing detailed information.
- **Subject:** The category of the news (e.g., "News" or "politicsNews").
- **Date:** The publication date of the article.
- **Label:** The classification label (1 for true news, 0 for fake news).

By including the **label** column, the datasets are now prepared for machine learning applications, such as training and evaluating fake news detection models.

3.2 Data Preprocessing

To prepare the dataset for machine learning, comprehensive preprocessing steps were applied to clean and transform the raw text data into a format suitable for analysis and modeling. These steps address issues such as noise, redundancy, and irrelevant information. Below are the detailed preprocessing techniques:

3.2.1 Cleaning the Data

Objective: Remove irrelevant, noisy, or redundant components from the text to focus on meaningful information and reduce computational overhead.

- **Lowercasing:** Convert all text to lowercase to ensure uniformity (e.g., "Hello" and "hello" are treated as the same word). **Tools:** `str.lower()` (Pandas), SpaCy, or NLTK. **Why?** Text data is often case-insensitive, so this step eliminates unnecessary distinctions.
- **Removing Punctuation:** Strip characters like `. , ; : ! ?` that don't add value to the analysis. **Tools:** Regular expressions (`re` module) or text preprocessing libraries. **Why?** Punctuation often adds noise without improving model performance.
- **Removing Numbers:** Delete standalone numbers unless they carry semantic value (e.g., "2017" in a news context). **Tools:** Regular expressions. **Why?** Numbers may not be informative unless they are contextually important.
- **Removing Special Characters:** Eliminate symbols such as `@, #, &,` and others. **Why?** These characters are usually metadata (e.g., hashtags or mentions in social media) and do not contribute to the core content of news articles.
- **Removing URLs:** Detect and remove web links (e.g., `https://example.com`). **Tools:** Regular expressions or `beautifulsoup` (for HTML-rich text). **Why?** URLs do not provide meaningful semantic content for fake news detection.
- **Correcting Encoding Issues:** Handle issues such as non-UTF-8 characters (e.g., `â€™` instead of `'`). **Tools:** Encoding libraries like `ftfy` or built-in Python utilities. **Why?** Encoding issues can disrupt downstream processing and model training.

3.2.2 Dropping Unnecessary Columns

Objective: Identify and remove columns in the dataset that do not contribute directly to the fake news detection task, reducing noise and computational overhead.

- **Columns to Drop:**
  - **Subject:** Categories or labels that may not align with the semantic analysis of the text.
  - **Date:** Timestamp information, which is often irrelevant for text analysis tasks.
- **Implementation:** Use Pandas to drop columns, for example: `df.drop(['Title', 'Subject', 'Date'], axis=1)`.
- **Why?** Dropping these columns ensures the focus remains on the core textual content, reducing noise and improving computational efficiency.

3.2.3 Tokenization

Objective: Split the text into smaller units (tokens), typically words or phrases, for further processing.

- Use tokenizers from libraries like NLTK (`word_tokenize`), SpaCy, or Hugging Face's transformers. For example, "The quick brown fox." becomes `["The", "quick", "brown", "fox"]`.
- **Challenges:** Tokenizing contractions (e.g., `"don't"` → `["do", "n't"]`) or handling domain-specific tokens like hashtags.
- **Why?** Tokens are the fundamental units for text analysis and feature extraction.

3.2.4 Removing Stopwords

Objective: Remove frequently occurring words (e.g., "the," "is," "and") that do not carry substantial meaning.

- Use predefined stopword lists from NLTK, SpaCy, or custom lists tailored to the dataset.
- Remove stopwords while retaining meaningful words (e.g., "fake news" → `["fake", "news"]`).
- **Challenges:** Over-removing words that might carry importance in certain contexts (e.g., "not" in "not fake").
- **Why?** Stopwords often dilute the significance of important terms and increase noise in the dataset.

3.2.5 Lemmatization

Objective: Reduce words to their root forms by stripping suffixes and prefixes.

- Use stemming algorithms like the Porter Stemmer or Snowball Stemmer from NLTK.
- **Examples:** "playing," "played," and "plays" → `"play"`.
- **Challenges:** Stemming can produce non-standard words (e.g., "better" → "bet").
- **Why?** Reduces vocabulary size and groups similar terms, simplifying the analysis.

3.2.6 Removing Duplicates

Objective: Eliminate redundant entries to reduce bias and improve computational efficiency.

- Identify duplicates based on key columns (e.g., Title and Text).
- Drop duplicate rows using `drop_duplicates()` in Pandas.
- **Why?** Duplicate entries can lead to overfitting and inflate the performance metrics of models.

3.2.7 Handling Missing Values

Objective: Address rows or fields with missing data to ensure consistency.

- Remove rows where critical fields like Title or Text are missing.
- Fill missing values (if applicable) using techniques like:
  - Imputation (e.g., replacing missing dates with the most frequent value).
  - Leaving missing data untouched for auxiliary fields like Subject.
- **Why?** Missing data in critical fields can distort results and reduce model accuracy.

3.2.8 Text Normalization

Objective: Standardize text for consistency across the dataset.

- Expand contractions (e.g., "don't" → "do not").
- Correct common spelling errors using spell-check libraries (e.g., `pyspellchecker`).
- Standardize domain-specific terms (e.g., "US" → "United States").
- **Why?** Normalization ensures uniformity, reducing variability caused by linguistic differences.

3.3 Feature Engineering

Feature engineering is a crucial step in building a machine learning model, especially for text data in a Fake News Detection project. For this project, we employed various natural language processing (NLP) techniques to extract relevant features from the dataset. These techniques include **TF-IDF**, **Bag of Words**, and **word embeddings**.

3.3.1 Text Preprocessing

Before feature extraction, the text data underwent preprocessing to ensure consistency and remove noise. The preprocessing steps included:

- **Lowercasing**: Converted all text to lowercase to eliminate case sensitivity.
- **Removing Punctuation and Special Characters**: Cleaned text to retain only alphanumeric words.
- **Tokenization**: Split sentences into individual words (tokens).
- **Stopword Removal**: Removed commonly used words (e.g., "the," "and") that do not add much meaning.
- **Stemming/Lemmatization**: Reduced words to their root forms (e.g., "running" → "run").

3.3.2 Feature Extraction Methods

Term Frequency-Inverse Document Frequency (TF-IDF) TF-IDF is a statistical measure that evaluates the importance of a word in a document relative to the entire corpus. It assigns higher weights to words that are frequent in a document but infrequent across the corpus, capturing the uniqueness of each term.

Why TF-IDF?

- Captures the significance of rare terms that are often good indicators of fake or real news.
- Reduces the influence of common terms that appear frequently in all documents.

Implementation:

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
X= tfidf_vectorizer.fit_transform(news_features['news'])
X.shape
```

Key Parameters:

- `max_features=5000`: Limits the feature space to the top 5000 most important terms.
- `ngram_range=(2, 2)`: Includes both unigrams (single words) and bigrams (two consecutive words).

Bag of Words (BoW) The Bag of Words model converts text into a sparse matrix, where each row represents a document, and each column represents the count of a unique word in that document.

Why Bag of Words?

- Simple and effective for capturing word frequency patterns in small datasets.
- Works well for models like Naive Bayes and Logistic Regression.

Implementation:

```
from sklearn.feature_extraction.text import CountVectorizer

bow_vectorizer = CountVectorizer(max_features=5000)
bow_features = bow_vectorizer.fit_transform(df['text'])
```

Key Parameters:

- `max_features=5000`: Reduces dimensionality by selecting the top 5000 most frequent words.

Word Embeddings (e.g., Word2Vec, GloVe) Word embeddings are dense vector representations of words in a continuous space, capturing semantic relationships between words. Unlike TF-IDF and BoW, embeddings preserve contextual information.

Why Word Embeddings?

- Captures complex relationships between words, such as synonyms and analogies.
- Improves model performance on deep learning models like LSTMs or Transformers.

Implementation (Using GloVe Pretrained Embeddings):

```
import numpy as np
from gensim.models import KeyedVectors

# Load GloVe embeddings
embeddings_index = {}
with open('glove.6B.100d.txt', 'r', encoding='utf-8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs

# Create embedding matrix
embedding_dim = 100
vocabulary = tfidf_vectorizer.get_feature_names_out()
embedding_matrix = np.zeros((len(vocabulary), embedding_dim))
for i, word in enumerate(vocabulary):
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

Use Case: Deep learning models like RNNs or CNNs use the `embedding_matrix` for text classification tasks.

3.3.3 Dimensionality Reduction (Optional)

To further reduce the feature space, techniques like **Principal Component Analysis (PCA)** or **Truncated Singular Value Decomposition (SVD)** were applied to the extracted features.

```
from sklearn.decomposition import TruncatedSVD

svd = TruncatedSVD(n_components=300)
reduced_features = svd.fit_transform(tfidf_features)
```

3.3.4 Summary of Feature Engineering

- **TF-IDF** and **Bag of Words** were used for traditional machine learning models.
- **Word Embeddings** (e.g., GloVe) were used for deep learning models.
- Dimensionality reduction techniques helped optimize computational efficiency.

3.4 Exploratory Data Analysis

3.4.1 Count of News Type

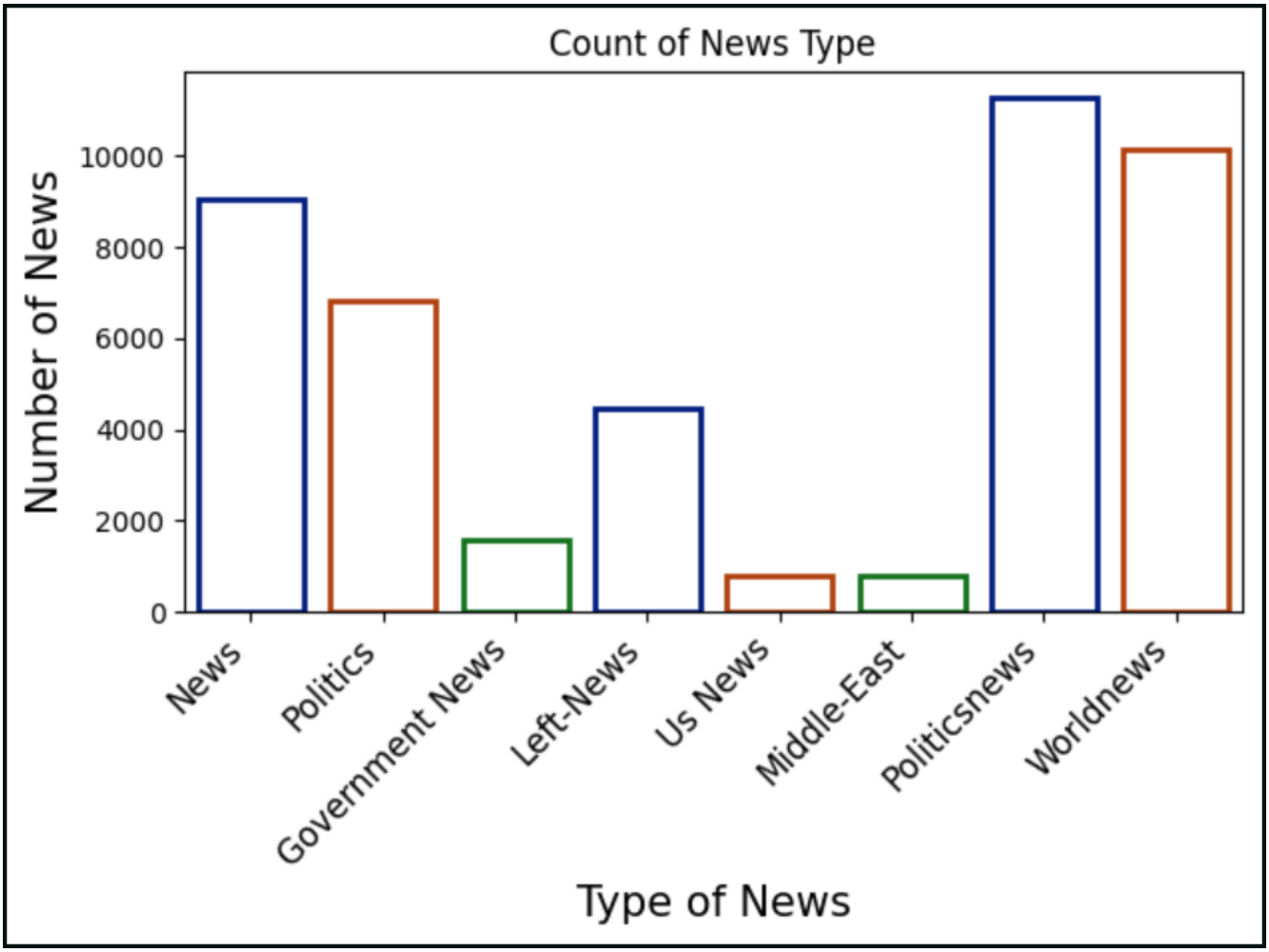


Figure 3

The bar chart titled **Count of News Type** depicts the number of articles or reports categorized under various news types.

Axes:

- **X-Axis (Type of News):** This axis lists the different categories of news:
  - News
  - Politics

  - Government News
  - Left-News
  - US News
  - Middle-East
  - Politicsnews
  - Worldnews
- **Y-Axis (Number of News):** This axis represents the count of news articles or reports, ranging from 0 to a maximum (in this case, around 10,000).

Bars: Each bar represents the count of news articles for its respective category. The height of each bar indicates the number of articles, with taller bars corresponding to a higher count.

Observations:

- **Worldnews:** This category has the highest count, with just over 10,000 articles. This suggests that world news is a major focus in the dataset.
- **News:** The general "News" category also shows a significant amount, indicating a broad range of topics covered under this label.
- **Politics:** The count of political news is lower than the general news category but still noteworthy.
- **Government News, Left-News, US News, Middle-East, Politicsnews:** These categories show a much lower count compared to the top categories. This could imply that these specific segments either have fewer articles published or are less emphasized in the overall reporting.

Analysis: The data suggests a strong interest in world news, which may reflect global events or issues that are more commonly reported. The lower counts in specialized categories (like Government News and Middle-East) might indicate niche reporting or limited coverage in those areas. If the report aims to explore trends in news reporting, the dominance of Worldnews could be a focal point, potentially linking it to current events or audience preferences.

3.4.2 Comparison of News Outputs

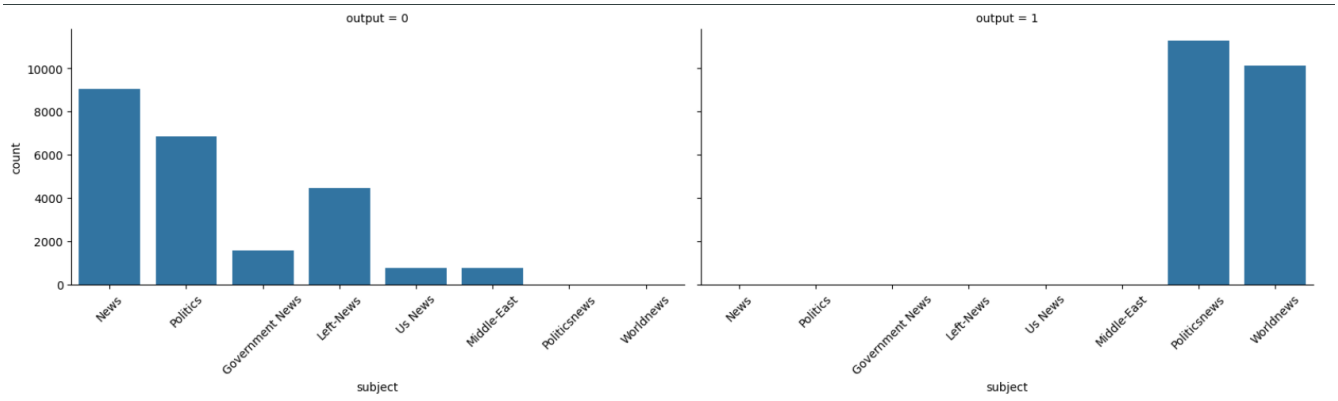


Figure 4

This output consists of two bar charts labeled as **output = 0** and **output = 1**, each representing the count of articles categorized by subject.

Layout: The charts are displayed side by side, allowing for direct comparison between the two outputs. Each chart has a similar structure, with the x-axis representing different news subjects and the y-axis indicating the count of articles.

Axes:

- **X-Axis (Subject):** The subjects represented in both charts include:
  - News
  - Politics
  - Government News
  - Left-News
  - US News
  - Middle-East
  - Politicsnews
  - Worldnews
- **Y-Axis (Count):** This axis indicates the number of articles, with values ranging from 0 to a maximum (the highest count is around 10,000).

Observations:

- **Output = 0:**
  - **Worldnews:** The category shows the highest count, with over 10,000 articles, indicating a strong emphasis on global events or issues.
  - **News:** This category also has a significant number of articles, though less than Worldnews.
  - **Politics:** The political news count is moderate but lower than the general news category.
  - **Government News, Left-News, US News, Middle-East, Politicsnews:** These categories have relatively lower counts, suggesting less focus or fewer articles published in these areas.
- **Output = 1:**
  - **Politicsnews and Middle-East:** These two categories have notably higher counts compared to the same output in output = 0. This could indicate a shift in reporting focus or a specific surge in articles related to these subjects.
  - **Worldnews:** The count remains high, maintaining its position as a critical category.
  - **Other subjects:** The counts for subjects like Government News, US News, and Left-News appear similar to those in output = 0, indicating consistent reporting levels across these categories.

Analysis: The comparison between the two outputs reveals a shift in focus, particularly with increased attention on Politicsnews and Middle-East in output = 1. The presence of consistent high counts for Worldnews across both outputs suggests that global news remains a priority for reporting. The lower counts in certain categories (like Government News and Left-News) in both outputs may indicate niche topics or less frequent coverage.

3.4.3 Count of Fake and True News

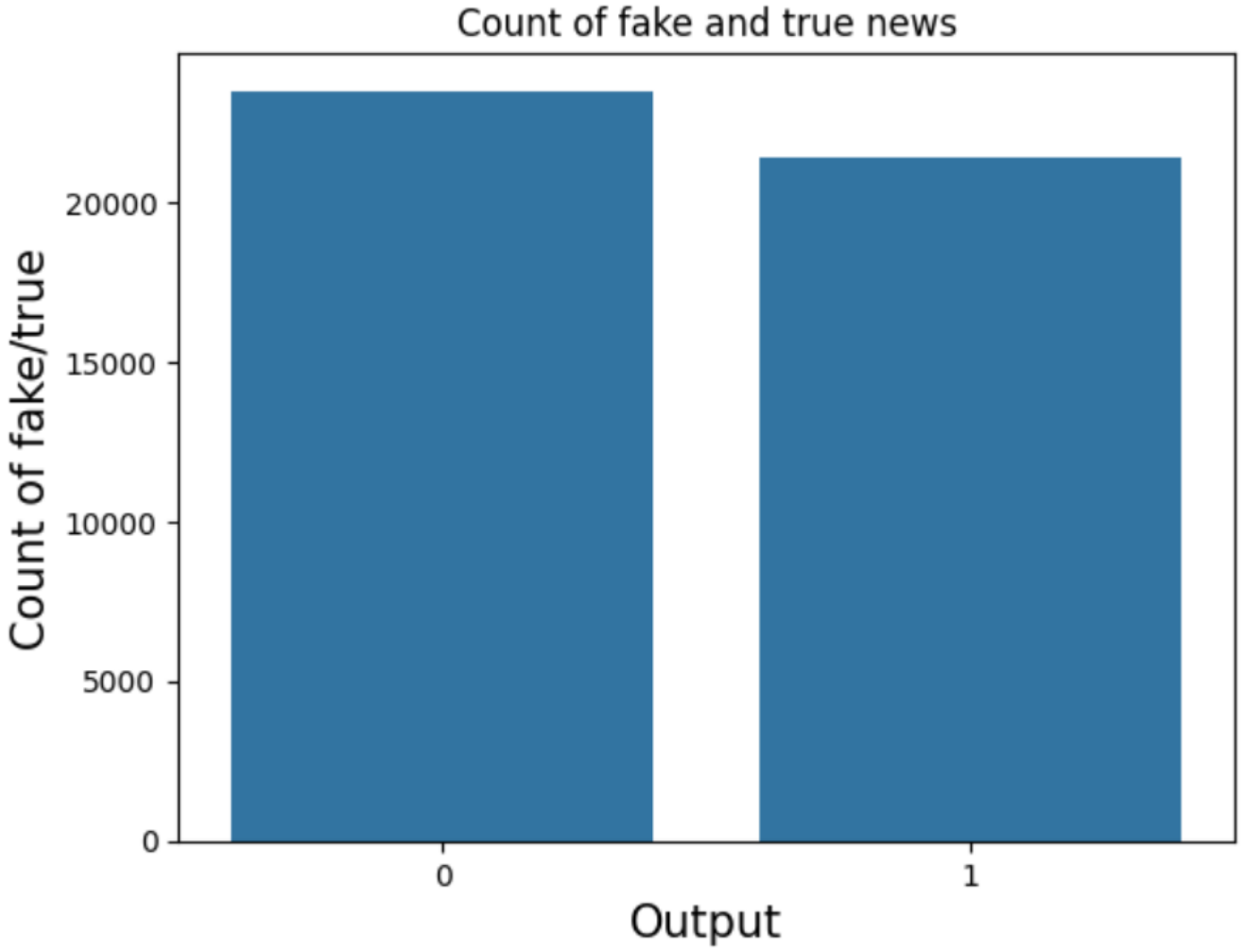


Figure 5

The output consists of a bar chart that visually represents the counts of fake and true news, categorized by two outputs labeled as **0** and **1**.

Chart Description:

- **Title:** The chart is titled *Count of Fake and True News*, indicating the subject matter of the data.
- **Axes:**
  - **X-Axis (Output):** The x-axis consists of two categories:
    - Output = 0
    - Output = 1
  - **Y-Axis (Count):** The y-axis represents the count of news articles, with a scale ranging from 0 to over 20,000.

Observations:

- **Output = 0:** The bar corresponding to output = 0 indicates a count of approximately 20,000 articles. This suggests that a significant number of articles are categorized as either fake or true news under this output.
- **Output = 1:** The bar for output = 1 shows a slightly lower count, just under 20,000 articles. This indicates a marginal reduction in the number of articles compared to output = 0.

Analysis: The chart indicates a relatively balanced distribution of articles between the two outputs, with both categories having counts close to 20,000. The slight decrease in the count for output = 1 could suggest that the dataset might have fewer articles classified under this category compared to output = 0. This distribution may reflect the nature of the dataset or the classification criteria used for determining whether news is categorized as fake or true.

3.4.4 Top 20 Words in News

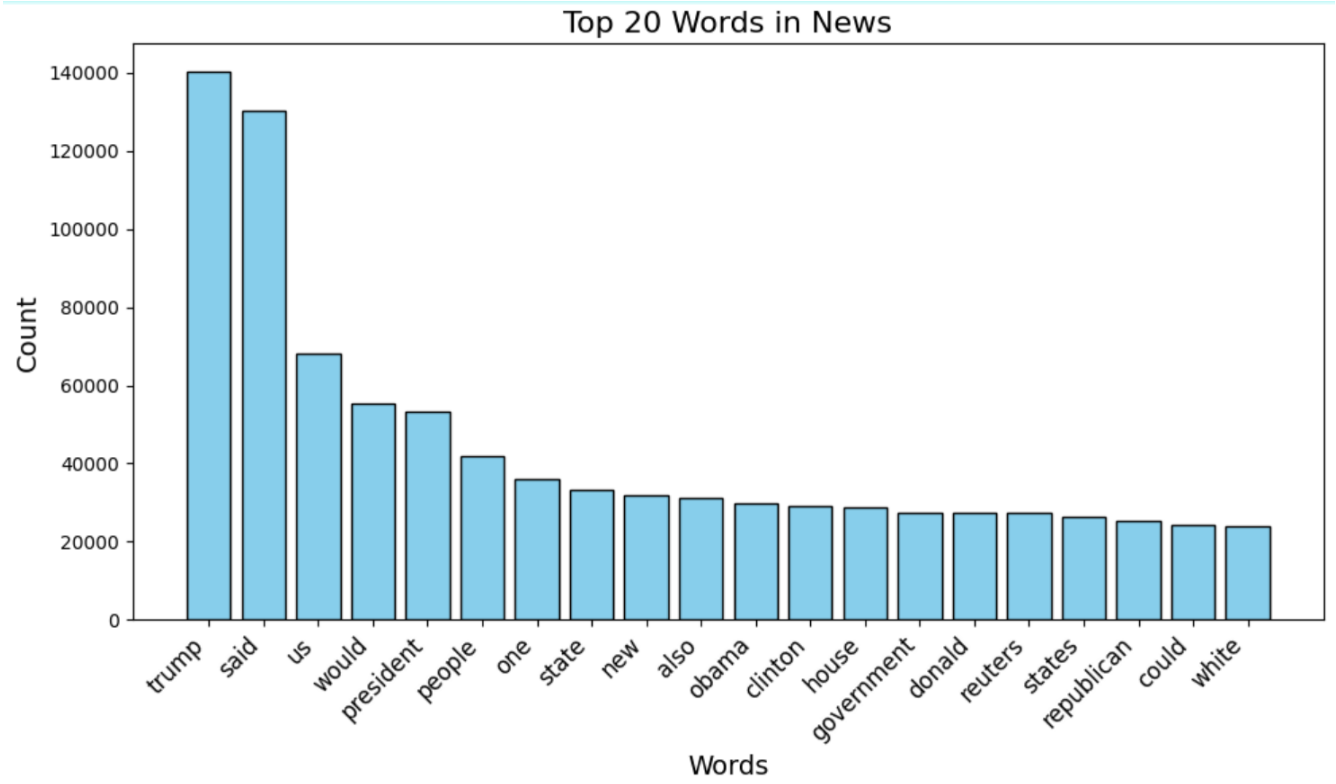


Figure 6

The output consists of a bar chart titled *Top 20 Words in News*, which visually represents the frequency of the most common words found in news articles. This chart provides insight into the key topics and themes being reported on in the media.

Chart Description:

- **Title:** The chart is titled *Top 20 Words in News*, indicating that it focuses on the most frequently used words in news content.
- **Axes:**

- **X-Axis (Words):** The x-axis lists the top 20 words, which include notable terms such as *trump*, *said*, *world*, and *president*.
- **Y-Axis (Count):** The y-axis represents the count of occurrences for each word, with values indicating how many times each word has been mentioned in the dataset.

Observations:

- The word *trump* appears to dominate the chart, with a count exceeding 40,000. This suggests that news articles heavily feature discussions related to Trump, highlighting his significant presence in current events.
- The word *said* also has a high frequency, indicating that quotations or reported speech are common in news articles.
- Other notable words include *world*, *president*, and *people*, which suggest a focus on global events, leadership, and public interest.

Analysis: The dominance of specific words like *trump* reflects the political landscape, particularly in contexts where Trump is a central figure. This could indicate the media's focus on political news or controversies surrounding him. The presence of common words like *said* shows the journalistic practice of quoting sources, which is fundamental in news reporting. The distribution of the other words can provide insights into the themes that are prevalent in the news, such as international affairs (*world*), governance (*president*, *government*), and social issues (*people*).

3.4.5 Top 20 Bigrams in News

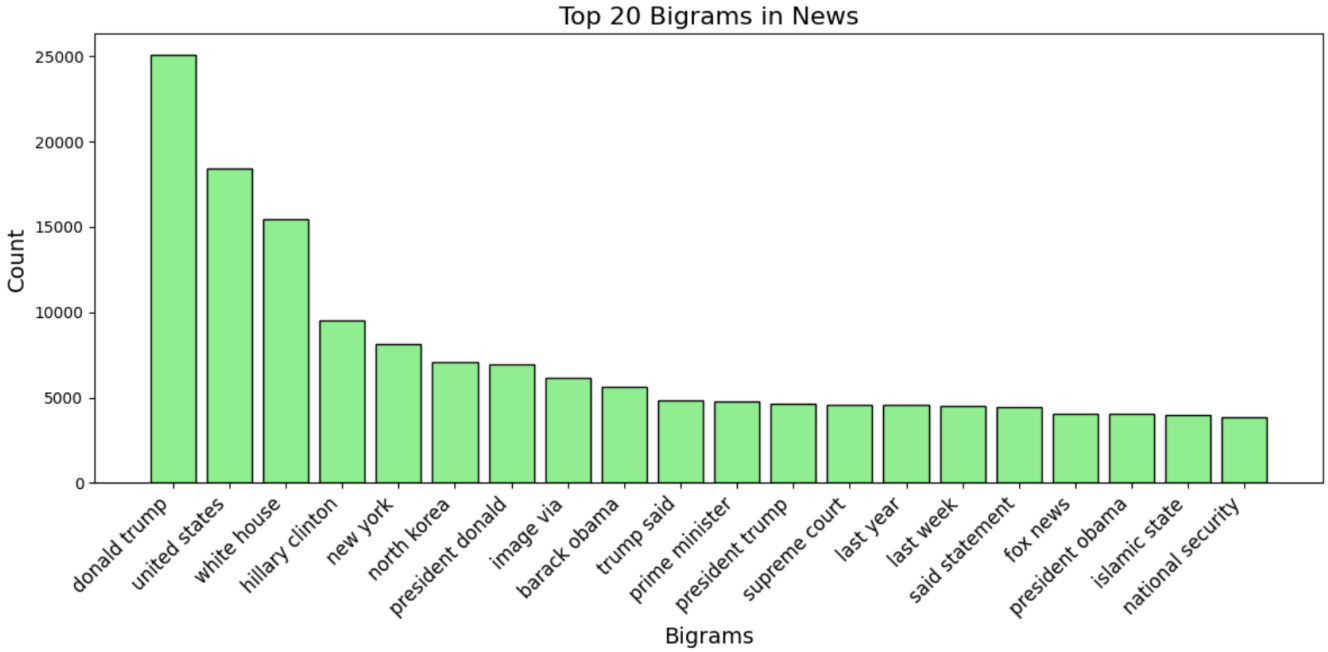


Figure 7

The output consists of a bar chart titled *Top 20 Bigrams in News*, which visually represents the frequency of the most common bigrams—pairs of consecutive words—found in news articles. This chart provides insight into the key phrases and topics being reported on in the media.

Chart Description:

- **Title:** The chart is titled *Top 20 Bigrams in News*, indicating that it focuses on the most frequently occurring pairs of words in news content.
- **Axes:**
  - **X-Axis (Bigrams):** The x-axis lists the top 20 bigrams, which include notable phrases such as *donald trump*, *united states*, *hillary clinton*, and *president obama*.
  - **Y-Axis (Count):** The y-axis represents the count of occurrences for each bigram, with values indicating how many times each pair has been mentioned in the dataset.

Observations:

- The bigram *donald trump* appears to dominate the chart, with a count exceeding 25,000. This suggests that discussions related to Trump are prevalent in news articles, reflecting his significant role in current events.
- The bigram *united states* also has a high frequency, indicating a focus on national issues, policies, and international relations.
- Other notable bigrams include *hillary clinton*, *president obama*, and *last week*, showcasing important political figures and temporal references relevant to current affairs.

Analysis: The dominance of specific bigrams like *donald trump* reflects the political climate, particularly times when Trump is a central figure in the news cycle. This could indicate heightened media coverage of political controversies or events involving him. The presence of phrases like *united states* suggests a focus on national issues and policies that affect the country. This is essential for understanding the context of discussions in the media. The bigrams *president clinton* and *president obama* indicate the historical relevance of these figures in ongoing political discussions, possibly in relation to current policies or comparisons with present leadership.

3.4.6 Top 20 Trigrams in News

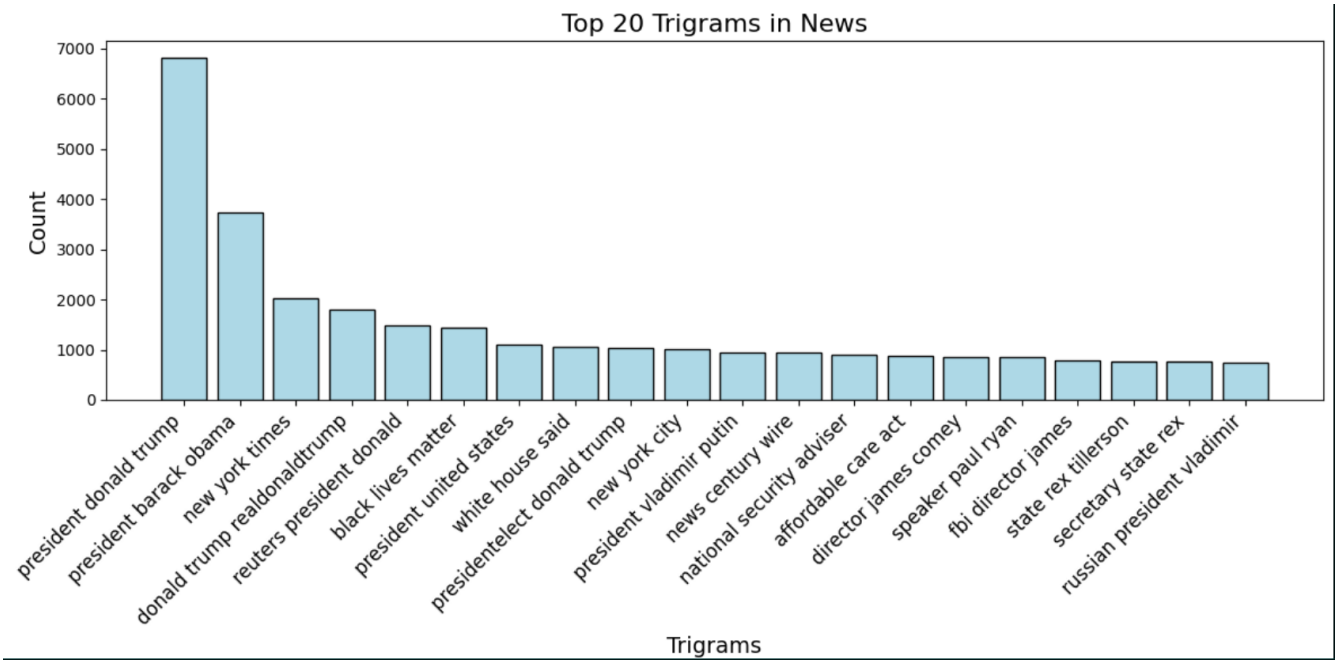


Figure 8

The output consists of a bar chart titled *Top 20 Trigrams in News*, which visually represents the frequency of the most common trigrams—sets of three consecutive words—found in news articles. This chart provides insight into the key phrases and topics being reported on in the media.

Chart Description:

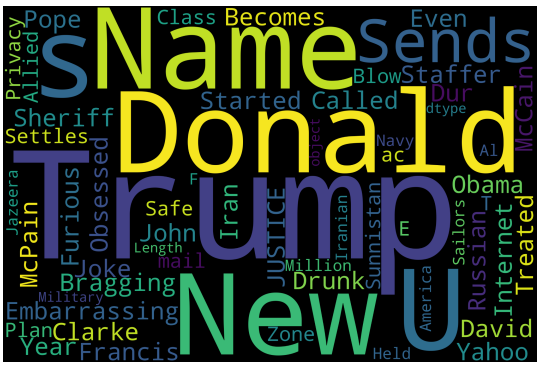
- **Title:** The chart is titled *Top 20 Trigrams in News*, indicating that it focuses on the most frequently occurring sets of three words in news content.
- **Axes:**
  - **X-Axis (Trigrams):** The x-axis lists the top 20 trigrams, which include notable phrases such as *president donald trump*, *new york times*, *president obama*, and *white house said*.
  - **Y-Axis (Count):** The y-axis represents the count of occurrences for each trigram, with values indicating how many times each set of words has been mentioned in the dataset.

Observations:

- The trigram *president donald trump* appears to dominate the chart, with a count exceeding 7,000. This suggests that discussions related to Trump are prevalent in news articles, reflecting his significant role in current events.
- The trigram *new york times* also has a high frequency, indicating a focus on this prominent news outlet, possibly reflecting its influence on public discourse.
- Other notable trigrams include *president obama*, *white house said*, and *national security wise*, showcasing important political figures and significant policy discussions.

Analysis: The dominance of specific trigrams like *president donald trump* reflects the political landscape, particularly times when Trump is a central figure in the news cycle. This could indicate heightened media coverage of political controversies or events involving him. The presence of phrases like *new york times* suggests a connection between the news outlet and the topics being covered, underscoring the importance of media sources in shaping public perception. Other trigrams such as *white house said* and *national security wise* indicate ongoing discussions related to governance and national issues, highlighting the media's focus on political statements and policies.

3.4.7 Analysis of Word Cloud Outputs



Overview: The two word clouds visually represent the frequency of terms within their respective datasets, with larger words indicating higher frequency and significance.

3.4.8 First Word Cloud Analysis

Key Terms:

- **U.S. and Russia:** These terms suggest a focus on geopolitical relations, particularly discussions surrounding international policies and tensions.
- **Budget:** The prominence of this word implies significant discussions about financial matters, possibly related to military or foreign policy budgets.
- **Republican:** Indicates a political focus on the Republican Party's stance on various issues, particularly in relation to military and budgetary concerns.

Additional Observations:

- Terms like *military*, *probe*, and *senator* suggest governmental activities or investigations related to national security.
- The presence of international terms such as *Vatican*, *NATO*, and *Chinese* indicates a broader discussion on international relations and alliances.

Tone and Themes: The word cloud reflects a serious, political tone, focusing on budgetary issues, military readiness, and international diplomacy.

3.4.9 Second Word Cloud Analysis

Key Terms:

- **Donald and Trump:** These terms dominate the cloud, indicating a significant focus on Donald Trump, likely in the context of his policies or public perception.
- **New:** This suggests discussions around new policies or changes associated with Trump.

Additional Observations:

- Terms like *Obama*, *McCain*, and *Iran* highlight political discourse that involves comparisons of political figures and their policies.
- The presence of critical terms such as *Bragging*, *Drunk*, and *Cultural* suggests a sensational tone regarding discussions about Trump.

Tone and Themes: The emotional tone indicates discussions may involve controversies or public reactions to Trump's actions and statements, reflecting a critical perspective.

3.5 Training and Testing

- **Method:** The `train_test_split` function was used to divide the data into training and testing sets.
- **Parameters:**
  - `test_size=0.25`: 25% of the data was used for testing, while 75% was used for training.
  - `random_state=0`: Ensures reproducibility by fixing the randomness of the split.
- **Purpose:** This split ensures that the model is trained on 75% of the data and evaluated on unseen 25% to test its performance on new data, simulating real-world scenarios.

Evaluation Metrics

The following metrics were used to evaluate the models:

1. Accuracy

- **Definition:** The proportion of correctly predicted samples out of the total samples.
- **Importance:** Useful as a general performance indicator but may be misleading in imbalanced datasets.

2. Precision

- **Definition:** The proportion of true positive predictions out of all positive predictions.
- **Formula:**

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Importance:** Measures the model's ability to avoid false positives. This is essential when false positives have high costs.

3. Recall

- **Definition:** The proportion of true positive predictions out of all actual positive cases.
- **Formula:**

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Importance:** Indicates the model's ability to detect true positives. This is critical in scenarios where false negatives are costly.

4. F1-Score

- **Definition:** The harmonic mean of precision and recall.
- **Formula:**

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Importance:** Balances precision and recall, especially useful for imbalanced datasets.

5. Confusion Matrix

- **Definition:** A table that summarizes the model's performance by showing counts of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).
- **Importance:** Offers a clear breakdown of model predictions and errors.

Hyperparameter Tuning

Each model's performance could be further improved with hyperparameter tuning:

- **Logistic Regression:**
  - **Key Parameters:** `penalty` (e.g., L1, L2), `C` (regularization strength).
  - **Tuning:** `GridSearchCV` can be used to find the optimal penalty and regularization values.
- **Decision Tree:**
  - **Key Parameters:** `max_depth`, `min_samples_split`, `min_samples_leaf`, and `criterion` (e.g., Gini or entropy).

- **Tuning:** Pruning can be applied to prevent overfitting.

- **Support Vector Machine (SVM):**
  - **Key Parameters:** `C` (regularization), `kernel` (linear, RBF, polynomial), `gamma` (for RBF kernel).
  - **Tuning:** `GridSearchCV` can identify the best kernel, regularization, and gamma values.

- **Naive Bayes:**
  - **Key Parameters:** None for Gaussian Naive Bayes; for other types (e.g., MultinomialNB), the `alpha` parameter (Laplace smoothing) can be tuned.
  - **Tuning:** Limited due to the simplicity of the algorithm.

3.6 Model Selection

3.6.1 Machine Learning Models Used

In our fake news detection system, we utilized several machine learning models to compare their performance. Below are the models implemented along with their respective test accuracies:

1. **Support Vector Machines (SVM):** SVM is a powerful algorithm for classification tasks. The SVM model in our evaluation achieved an impressive test accuracy of **97%**, showcasing its robust predictive capabilities and suitability for fake news detection tasks.

2. **Logistic Regression:** Logistic Regression is a statistical model used for binary classification. It estimates the probability that a given input point belongs to a certain class. In our tests, Logistic Regression achieved a test accuracy of approximately **96.52%**.

3. **Decision Tree:** Decision Trees are a non-parametric supervised learning method used for classification and regression. They model decisions and their possible consequences as a tree structure. The Decision Tree model yielded a test accuracy of around **93.53%**.

4. **Naive Bayes:** Naive Bayes is a family of probabilistic algorithms based on Bayes' theorem, assuming independence among predictors. It is particularly effective for large datasets and text classification tasks. In our implementation, the Naive Bayes model reached a test accuracy of approximately **93.73%**.

5. **Long Short-Term Memory (LSTM):** LSTM is a type of recurrent neural network known for capturing long-term dependencies. The LSTM model in our evaluation showcased a test accuracy of **98%**, indicating strong performance in fake news detection tasks.

The accuracies indicate that Logistic Regression performed the best among the models tested, followed closely by Naive Bayes and Decision Tree, while KNN showed relatively lower performance in this context.

Model Comparison

- **Logistic Regression:**
  - **Accuracy:** 0.965
  - **Strengths:** High precision and recall, making it reliable for binary classification.
  - **Weaknesses:** Slightly lower accuracy compared to SVM and Decision Tree.

- **Decision Tree:**
  - **Accuracy:** 0.936

  - **Strengths:** Perfect recall for class 0 and high precision.
  - **Weaknesses:** Tends to overfit the data (indicated by the confusion matrix).

- **Support Vector Machine (SVM):**
  - **Accuracy:** 0.975
  - **Strengths:** Highest accuracy among the models, indicating robust generalization.
  - **Weaknesses:** Computationally expensive, especially on large datasets.

- **Naive Bayes:**
  - **Accuracy:** 0.937
  - **Strengths:** Fast and performs well given its simplicity.
  - **Weaknesses:** Assumes feature independence, leading to slightly lower precision and recall compared to SVM and Logistic Regression.

- Based on the evaluation metrics, **SVM** is the best-performing model with the highest accuracy and a well-balanced precision-recall tradeoff.
- **Logistic Regression** and **Naive Bayes** are strong contenders, especially for computational efficiency and ease of interpretation.
- **Decision Tree** shows signs of overfitting and may require additional regularization or pruning.

4 Implementation

4.1 Tools and Frameworks Used

The implementation of fake news detection involves various tools and frameworks. Some of the popular tools and libraries include:

1. **Python:** Python is a widely used programming language in the field of data science and machine learning due to its versatility and extensive libraries.
2. **Scikit-learn:** Scikit-learn is a popular machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It offers various algorithms for classification, regression, clustering, and more, making it useful for fake news detection tasks.
3. **TensorFlow:** TensorFlow is an open-source machine learning framework developed by Google. It is commonly used for building and training deep learning models, which can be beneficial for tasks requiring complex neural network architectures.
4. **NLTK (Natural Language Toolkit):** NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources, such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.
5. **Pandas:** Pandas is a powerful data manipulation library in Python that offers data structures and tools for data analysis. It provides easy-to-use data structures like DataFrames, which are beneficial for handling and preprocessing datasets in fake news detection tasks.
6. **Gensim:** Gensim is a robust open-source vector space modeling and topic modeling toolkit. It is useful for tasks like semantic analysis, text summarization, and similarity detection, which can enhance the capabilities of fake news detection systems.

7. **Word2Vec:** Word2Vec is a technique for natural language processing that is used to learn word embeddings. These embeddings can capture semantic relationships between words, which is valuable for tasks like text classification and similarity analysis in fake news detection.

4.1.1 Step by Step Implementation

A high-level overview of the step-by-step implementation process for building a fake news detection system:

1. **Data Collection and Loading:** Gather a dataset of labeled news articles, where each article is classified as either fake or real. Load the dataset into a suitable data structure using tools like Pandas in Python.

2. **Data Preprocessing:** Perform data cleaning tasks such as removing HTML tags, punctuation, stopwords, and special characters. Tokenize the text data and convert it into a numerical format that machine learning models can process. Handle missing values and standardize the text data for consistency.

3. **Feature Engineering:** Extract relevant features from the text data that can help differentiate between fake and real news articles. Common features include word frequency, n-grams, sentiment analysis scores, and readability metrics.

4. **Splitting the Data:** Divide the preprocessed data into training and testing sets to evaluate the performance of the machine learning model accurately.

5. **Model Selection and Training:** Choose a suitable machine learning model for fake news detection tasks, such as Support Vector Machines (SVM), Random Forest, or Neural Networks. Train the selected model on the training data and tune hyperparameters to improve performance.

6. **Model Evaluation:** Evaluate the trained model using the testing dataset to assess its accuracy, precision, recall, and other relevant metrics. Fine-tune the model if necessary based on the evaluation results.

4.1.2 Challenges Faced

The implementation of a fake news detection system may encounter several technical and practical challenges. Here are some common challenges faced and potential ways to address them:

1. **Data Quality and Quantity:**

   - *Challenge:* Insufficient or noisy data can hinder the effectiveness of the model.
   - *Addressing:* Use data augmentation techniques to increase the dataset size, perform extensive data cleaning to reduce noise, and consider leveraging transfer learning or pre-trained language models to improve performance with limited data.

2. **Feature Selection:**

   - *Challenge:* Identifying the most relevant features for fake news detection can be challenging.
   - *Addressing:* Conduct feature importance analysis, experiment with different feature sets, and leverage techniques like TF-IDF, word embeddings, and topic modeling to extract meaningful features from the text data.

3. **Model Overfitting:**

   - *Challenge:* Overfitting occurs when the model performs well on the training data but poorly on unseen data.

- *Addressing:* Regularize the model by tuning hyperparameters, using techniques like dropout in neural networks, and employing cross-validation to ensure the model's generalizability.

4. **Class Imbalance:**
   - *Challenge:* Class imbalances in the dataset, where fake news samples are significantly fewer than real news samples, can lead to biased model predictions.
   - *Addressing:* Implement techniques like oversampling, undersampling, or using algorithms that handle class imbalances well, such as SMOTE (Synthetic Minority Oversampling Technique).
5. **Interpretable Models:**
   - *Challenge:* Ensuring the model's decisions are interpretable and explainable is crucial for building trust in the system.
   - *Addressing:* Use techniques like SHAP (SHapley Additive exPlanations) values, LIME (Local Interpretable Model-agnostic Explanations), or attention mechanisms in neural networks to provide insights into how the model makes predictions.
6. **Computational Resources:**
   - *Challenge:* Training complex models like deep neural networks may require significant computational resources.
   - *Addressing:* Utilize cloud computing services or distributed computing frameworks to scale up computational resources, optimize code for efficiency, and consider model compression techniques to reduce computational overhead.
7. **Model Maintenance and Updates:**
   - *Challenge:* Ensuring the model stays relevant and effective over time as fake news tactics evolve.
   - *Addressing:* Set up a system for continuous monitoring, retraining, and updating the model with new data periodically. Stay informed about emerging trends in fake news and adapt the model accordingly.

5 Results and Analysis

5.1 Model Performance

We analyze the performance of different machine learning models in detecting fake news. The models evaluated include Logistic Regression, Decision Trees, Support Vector Machines (SVM), Naive Bayes, and Long Short-Term Memory (LSTM) networks. The evaluation is based on accuracy scores obtained from the testing phase of the models. The evaluation of various machine learning models for fake news detection highlights the superior performance of LSTM and SVM models. These models exhibit high accuracy rates and robust predictive capabilities, making them suitable choices for combating fake news. Regular evaluation, optimization, and model refinement are essential for ensuring continued effectiveness in fake news detection efforts.

6 Conclusion and Future Work

Summary

In this project, we developed an innovative machine learning model aimed at detecting fake news by analyzing news events and corresponding Twitter reviews. Our results indicated a promising accuracy rate in differentiating between real and fake news articles, demonstrating the

effectiveness of classification algorithms in this context. The model leverages sentiment analysis and natural language processing techniques to interpret the nuances of social media discussions, providing a robust framework for fake news detection. Overall, the project contributes to the ongoing efforts to mitigate the spread of misinformation, particularly on social media platforms.

Limitations

Despite the encouraging results, our approach has several limitations. One significant concern is the potential bias in the dataset used for training the model, which may not represent the full spectrum of news topics or social media sentiments. Additionally, the scalability of the model remains a challenge, particularly when processing large volumes of real-time data. The reliance on Twitter as a primary source for news validation may also limit the model's applicability to other social media platforms where user behavior and content dynamics differ.

Future Scope

Looking ahead, there are several avenues for improvement and expansion of this work. Implementing real-time detection capabilities could enhance the model's responsiveness to emerging fake news stories. Furthermore, adding multilingual support would broaden the model's applicability across diverse linguistic contexts, making the tool more accessible globally. Integrating the detection system with established fact-checking databases could also improve the accuracy of predictions and provide users with verified information. Exploring these enhancements will be crucial in developing a comprehensive solution to combat the pervasive issue of fake news.

7 References

INCET2021Paper0529
https://www.sciencedirect.com/science/article/pii/S1877050918318210?via
https://www.researchgate.net/publication/373532276$_{F}ake_{N}ews_{D}etection_{U}sing_{M}achine_{L}earning$

```
true_news = pd.read_csv('True.csv')
fake_news = pd.read_csv('Fake.csv')

true_news['output'] = 1
fake_news['output'] = 0

fake_news['news']= fake_news['title']+fake_news['text']
fake_news= fake_news.drop(['title', 'text'], axis=1)

#Concatenating and dropping for true news
true_news['news']= true_news['title']+true_news['text']
true_news= true_news.drop(['title', 'text'], axis=1)

#Rearranging the columns
fake_news = fake_news[['subject', 'date', 'news','output']]
true_news = true_news[['subject', 'date', 'news','output']]

frames = [fake_news, true_news]
news_dataset = pd.concat(frames)
news_dataset

news_dataset.drop('date', axis=1, inplace=True)

def review_cleaning(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

news_dataset['news']= news_dataset['news'].apply(lambda x:review_cleaning(x))
news_dataset.head()

stop = stopwords.words('english')
news_dataset['news'] = news_dataset['news'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
news_dataset.head()
```

```
text = fake_news["news"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
text = true_news["news"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
# Download required NLTK data
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('stopwords')
stop_words = set(stopwords.words("english"))
lemmatizer = WordNetLemmatizer()
corpus = []
for i in range(0, len(news_features)):
    news = re.sub('[^a-zA-Z]', ' ', news_features['news'][i])
    news = news.lower()
    news = news.split()  # Using split() instead of word_tokenize()
    news = [lemmatizer.lemmatize(word) for word in news if word not in stop_words]
    news = ' '.join(news)
    corpus.append(news)
```

```
tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
X= tfidf_vectorizer.fit_transform(news_features['news'])
X.shape

y= news_dataset['output']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)

def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j, i, cm[i, j],
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
logreg_cv = LogisticRegression(random_state=0)
dt_cv = DecisionTreeClassifier()
svm_cv = SVC()
nb_cv = MultinomialNB(alpha=0.1)
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree', 2: 'SVM', 3: 'Naive Bayes'}
cv_models = [logreg_cv, dt_cv, svm_cv, nb_cv]
for i, model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i], cross_val_score(model, X, y, cv=10, scoring='accuracy').mean()))
```

```
#LSTM
voc_size=10000
onehot_repr=[one_hot(words,voc_size)for words in corpus]

sent_length=5000
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
embedded_docs[1]
#Creating the lstm model
embedding_vector_features=40
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Dropout(0.3))
model.add(LSTM(100))
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))

#Compiling the model
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
# Converting the X and y as array
X_final=np.array(embedded_docs)
y_final=np.array(y)
X_final.shape,y_final.shape

X_train, X_test, y_train, y_test = train_test_split(X_final, y_final, test_size=0.33, random_state=42)
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=2,batch_size=64)
y_pred_prob = model.predict(X_test)
y_pred = (y_pred_prob > 0.5).astype(int)
y_pred
```