Artificial Intelligence and Machine Learning

Project Report

Semester-IV (Batch-2022)

Text Summariser

Supervised By:

Dr. Jatin Arora

Submitted By:

Divya Bhoria 2210990293

Geetansh Sood 2210990323

Ojas Gupta 2210990627

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

Abstract

The exponential growth of digital news content has led to significant challenges in information management, creating a pressing need for efficient summarization tools. This project focuses on developing a machine learning-based text summarizer to effectively condense news articles from major news websites. The primary goal is to address the issue of information overload, which affects both individuals and organizations, by providing a solution that enables quick and easy access to essential information.

Our summarizer utilizes advanced natural language processing (NLP) techniques to automatically generate concise and coherent summaries of lengthy news articles. The project involved collecting and processing a large dataset of archived news articles, which served as the basis for training our machine learning model. The model was designed to identify key points and main ideas within the articles, ensuring that the summaries retain the most important information while discarding extraneous details.

Through rigorous testing and evaluation, the summarizer demonstrated a significant reduction in the time required to comprehend news content, thereby enhancing productivity and decision-making capabilities for users. The results indicate that our approach not only improves information retrieval but also facilitates better retention and understanding of the summarized content.

The implications of this project are far-reaching, with potential applications in various fields such as journalism, academia, finance, and policy-making, where timely and accurate information is crucial. By leveraging the power of machine learning and NLP, this project contributes to the broader field of AI-driven information management solutions, paving the way for future advancements in automated content summarization.

In conclusion, this research underscores the potential of machine learning technologies in transforming how we manage and consume vast amounts of information. The developed text summarizer offers a practical and efficient tool for navigating the digital information landscape, ultimately helping users to stay informed and make better-informed decisions in an increasingly data-saturated world.

CONTENTS

Introduction

In today's digital age, the relentless flow of information from various news sources has become both a boon and a burden. While access to a vast array of news articles enables individuals and organizations to stay informed about current events and trends, the sheer volume of content can be overwhelming. The challenge lies in efficiently sifting through this information to extract relevant and important insights without investing excessive time and effort. This issue of information overload is particularly acute for professionals, researchers, and decision-makers who rely on timely and accurate data to guide their actions. To address this pressing need, our project aims to develop a machine learning-based text summarizer capable of automatically condensing lengthy news articles into concise and coherent summaries. By leveraging advanced natural language processing (NLP) techniques, this tool seeks to enhance productivity, improve information management, and support better decision-making in a world inundated with data.

Background

The digital revolution has dramatically transformed the way news is produced, distributed, and consumed. With the advent of the internet and mobile technologies, news organizations publish a continuous stream of articles, covering an extensive range of topics. Major news websites update their content multiple times a day, contributing to an overwhelming abundance of information. This phenomenon, known as information overload, presents a significant challenge to individuals and organizations alike. For professionals in fields such as finance, marketing, policy-making, and academia, staying informed about the latest developments is crucial. However, the traditional method of manually reading through numerous articles is not only time-consuming but also inefficient.

Moreover, organizations that archive news articles for reference, analysis, or research purposes face difficulties in managing and utilizing this vast repository of data. The valuable insights embedded within these articles often remain untapped due to the impracticality of manually sifting through such large volumes of text. This situation highlights the urgent need for automated solutions that can streamline the process of information extraction and utilization.

Advances in machine learning and natural language processing (NLP) have opened new possibilities for tackling these challenges. Machine learning algorithms can be trained to understand and process human language, enabling the development of tools that can automatically summarize large bodies of text. NLP techniques allow for the extraction of key points and essential information from lengthy articles, creating concise summaries that retain the core message and important details.

This project leverages these technological advancements to create a machine learning-based text summarizer tailored to the needs of users overwhelmed by the influx of news articles. By transforming how news content is consumed, our summarizer aims to make it easier for

individuals and organizations to stay informed, make timely decisions, and efficiently manage their information resources. The development of this tool is not only a response to the current challenges posed by information overload but also a step towards more intelligent and automated information management solutions for the future.

Objectives

Develop an Automated Summarization Tool: Create a machine learning-based text summarizer capable of automatically generating concise summaries of lengthy news articles from major news websites.

Improve Efficiency in Information Consumption: Reduce the time and effort required for individuals and professionals to stay informed by providing quick and accurate summaries of news articles.

Enhance Information Management: Facilitate better organization and utilization of archived news articles by transforming large volumes of text into manageable summaries.

Support Decision-Making: Provide timely and relevant summaries that aid decision-makers in fields such as finance, marketing, policy-making, and academia, ensuring they have access to critical information without delay.

Leverage Advanced NLP Techniques: Utilize state-of-the-art natural language processing methods to accurately identify and extract key points and main ideas from news articles.

Ensure Coherent and Accurate Summaries: Develop algorithms that generate summaries which are not only concise but also maintain the coherence and accuracy of the original content.

Test and Validate the Summarizer: Conduct rigorous testing and evaluation to ensure the effectiveness and reliability of the summarizer in various real-world scenarios and with different types of news content.

Contribute to Technological Advancements: Advance the field of machine learning and NLP by applying these technologies to the practical problem of news summarization, demonstrating their potential and effectiveness.

Provide a User-Friendly Solution: Design the summarizer to be easily accessible and user-friendly, catering to the needs of both individual users and organizations.

Promote Future Research and Development: Lay the groundwork for future innovations in automated content summarization and intelligent information management solutions.

Significance

The development of a machine learning-based text summarizer addresses the pressing issue of information overload, providing a crucial tool for navigating the vast amounts of news content produced daily. By generating concise summaries, the summarizer significantly enhances productivity for professionals, researchers, and decision-makers, allowing them to access critical information quickly and efficiently. This tool supports better decision-making by ensuring that users have timely and accurate information at their fingertips, which is particularly valuable in fields such as finance, marketing, policy-making, and academia. Moreover, the summarizer improves information management by transforming large repositories of archived news articles into manageable and useful summaries, facilitating better organization, retrieval, and utilization of valuable data.

The project also advances the technological frontiers of machine learning and natural language processing, demonstrating their practical applications and encouraging further research and development. By reducing the cognitive load on users and helping them absorb and retain important information more effectively, the summarizer enhances overall comprehension and reduces cognitive fatigue. Additionally, the tool's ability to deliver personalized content aligned with individual preferences improves user engagement and satisfaction.

This project exemplifies the practical applications of artificial intelligence in solving everyday challenges, promoting wider adoption of AI-driven solutions across various sectors. Furthermore, the successful development of this summarizer can inspire new approaches to information management and content consumption, fostering innovation in the field. Ultimately, by addressing current challenges and setting a precedent for effective summarization tools, this project lays a solid foundation for future advancements in automated content summarization and intelligent information systems.

Problem Statement

In the digital age, the constant stream of news articles overwhelms individuals and organizations, leading to information overload. Manually summarizing these articles is time-consuming and inefficient, hindering productivity and decision-making. Despite extensive news archives, the lack of an automated summarization tool results in underutilized data and cognitive fatigue. Thus, there's a critical need for a machine learning-based text summarizer to swiftly condense news articles, addressing information overload and enhancing information management and usability.

Software Requirements

1. **Python (Version 3.6 or higher)**: Python is required as the primary programming language for developing the text summarization tool.
2. **NLTK (Natural Language Toolkit)**: NLTK will be used for natural language processing tasks such as tokenization, stemming, and lemmatization.
3. **TextBlob**: TextBlob, a Python library built on top of NLTK and Pattern libraries, will be utilized for sentiment analysis and other text processing functionalities.
4. **Newspaper3k**: The Newspaper3k library will be used for web scraping news articles from major news websites for training and testing the summarization model.
5. **os module**: The os module will be used for file and directory operations, such as listing files in a directory.
6. **string module**: The string module will be utilized for text processing tasks, such as removing punctuation from text data.
7. **pickle module**: The pickle module will be used for serializing and deserializing Python objects, enabling the saving and loading of trained machine learning models.
8. **Pandas**: Pandas, a powerful data manipulation library, will be used for data preprocessing, analysis, and manipulation tasks.
9. **NumPy**: NumPy, a fundamental package for scientific computing with Python, will be used for numerical computations and data manipulation tasks.
10. **Regular Expressions (re module)**: The re module will be utilized for pattern matching and text manipulation tasks, such as text cleaning and preprocessing.
11. **Matplotlib**: Matplotlib, a comprehensive library for creating static, interactive, and animated visualizations in Python, will be used for data visualization tasks.
12. **Seaborn**: Seaborn, a Python visualization library based on Matplotlib, will be used for creating visually appealing and informative statistical graphics.
13. **Plotly Express**: Plotly Express, a high-level interface for creating expressive and interactive visualizations, will be used for advanced data visualization tasks, such as interactive plots and dashboards.

14. **tkinter**: Tkinter, the standard GUI (Graphical User Interface) toolkit for Python, will be used for creating a user-friendly interface for interacting with the text summarization tool.

These software requirements will provide the necessary tools and libraries for developing a robust and effective text summarization application.

Hardware Requirements

The hardware requirements for running the GitHub profile viewer are relatively modest, as the primary processing will be handled on the client-side and the backend server:

1. Computer or Server:

2. Internet Connection

Proposed Design / Methodology

1. Data Collection:

   - Scrape news articles using the Newspaper3k library.

   - Clean the data by removing HTML tags and non-textual content.

2. Text Preprocessing:

   - Tokenize text into words or sentences with NLTK or spaCy.

   - Remove stop words and punctuation, and apply stemming or lemmatization.

3. Feature Extraction:

   - Extract features such as word frequency, TF-IDF, or word embeddings.

4. Model Development:

   - Implement and train extractive (e.g., TextRank) or abstractive (e.g., seq2seq with attention) summarization models.

5. Evaluation:

   - Use ROUGE or BLEU scores to evaluate model performance.

   - Refine models based on evaluation results and user feedback.

6. Deployment:

   - Package the tool into an executable or web service for deployment.

7. Testing and Validation:

   - Perform thorough testing to ensure functionality and reliability.

   - Validate with real users to gather feedback for improvements.

8. Documentation and Maintenance:

   - Document the tool's design, implementation, and usage.

   - Provide updates and maintenance for bug fixes and enhancements.

9. Scalability and Extensibility:

    - Ensure the tool is scalable and extensible for future enhancements and larger datasets.

Schematic structure:

1. Data Loading and Preprocessing:

- The process begins by loading the data from a CSV file containing URLs of news articles using the `Load_data` class.

- Unnecessary columns such as "Id", "Author", and "Date" are dropped from the DataFrame to clean the data and prepare it for analysis.

2. Sentiment Analysis:

- The `sentiment_EDA()` method of the `Load_data` class performs sentiment analysis on the news articles.

- For each article, the sentiment polarity and subjectivity scores are calculated using the TextBlob library.

- The results are stored in a DataFrame, which is then used for exploratory data analysis (EDA).

3. Exploratory Data Analysis (EDA):

- EDA is performed to gain insights into the sentiment distribution of the news articles.

- Two types of plots are generated:

- Scatter plot: Polarity vs Subjectivity to visualize the relationship between sentiment polarity and subjectivity.

- Histograms: Distribution of polarity and subjectivity scores to understand their frequency distribution.

4. Visualization:

- Matplotlib is used to create visualizations for EDA, including scatter plots and histograms.

- These visualizations provide a graphical representation of the sentiment analysis results, enabling easy interpretation and analysis.

5. Individual Article Analysis:

- The `analyze_data()` method of the `Load_data` class analyzes individual news articles.

- For each article, the title, summary, and sentiment analysis results (polarity and subjectivity) are printed.

- This analysis provides insights into the sentiment of each article and helps understand the overall sentiment distribution across the dataset.

6. Integration and Execution:

- The code snippets for data loading, preprocessing, sentiment analysis, EDA, and individual article analysis are integrated into a single script.

- The script is executed to perform the entire analysis pipeline, from loading the data to visualizing the sentiment distribution and analyzing individual articles.

Overall, the schematic structure outlines the sequential flow of operations, starting from data loading and preprocessing to sentiment analysis, EDA, and individual article analysis. It demonstrates a systematic approach to analyzing sentiment in news articles, facilitating data-driven insights and decision-making.

Algorithms Used:

Algorithm Type: Lexicon-based sentiment analysis algorithm.

Library Used: TextBlob library in Python.

Sentiment Analysis Process:

1. Text Preprocessing: The algorithm preprocesses the text data, including tokenization and removal of stopwords and punctuation.
2. Lexicon-based Scoring: Each word in the text is assigned a pre-defined sentiment score based on a lexicon or dictionary of sentiment words.
3. Calculation of Sentiment Polarity: The sentiment polarity score of the text is computed by aggregating the sentiment scores of individual words and

normalizing the result. The polarity score ranges from -1 (negative sentiment) to 1 (positive sentiment), with 0 representing neutral sentiment.

4. Calculation of Subjectivity: The subjectivity score of the text is calculated by averaging the subjectivity scores of individual words. The subjectivity score ranges from 0 (objective/factual) to 1 (subjective/opinionated).

Output: The algorithm generates sentiment analysis results, including the sentiment polarity and subjectivity scores for each news article.

Purpose: The algorithm is utilized for exploratory data analysis (EDA) to analyze the sentiment distribution across the dataset, visualize sentiment trends, and gain insights into the overall sentiment of the news articles.

Summary:

The sentiment analysis algorithm utilized for the analysis of news articles employs a lexicon-based approach facilitated by the TextBlob library in Python. Through this method, the algorithm systematically preprocesses the text data, assigning sentiment scores to individual words based on pre-defined lexicons or dictionaries. By aggregating these scores, the algorithm computes sentiment polarity and subjectivity scores for each article. This process enables the algorithm to generate comprehensive sentiment analysis results, offering insights into the emotional tone and subjective nature of the news content. Despite its simplicity and efficiency, the algorithm's reliance on pre-defined lexicons may limit its ability to capture nuanced sentiments accurately. However, it remains a valuable tool for conducting exploratory data analysis, visualizing sentiment trends, and gaining a deeper understanding of the overall sentiment landscape within the dataset.

Result

In [5]:

```
import tkinter as tk
import nltk
from textblob import TextBlob
from newspaper import Article
from os import listdir
import string
from pickle import dump, load
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

1. import tkinter as tk:
    - Tkinter is a standard GUI (Graphical User Interface) toolkit in Python. It is used to create graphical user interfaces for applications. The alias tk is commonly used to simplify the reference to the Tkinter module.
2. import nltk:
    - NLTK (Natural Language Toolkit) 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 along with a suite of text processing libraries.
3. from textblob import TextBlob:
    - TextBlob is a library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.
4. from newspaper import Article:
    - Newspaper is a Python library used for web scraping articles from online newspapers. The Article class is used to extract and parse article data from URLs.
5. from os import listdir:
    - The listdir function from the os module is used to list the files and directories in a specified path. It is useful for file manipulation and directory operations.
6. import string:
    - The string module provides a collection of string operations and constants, such as string manipulation methods, predefined string constants (like string.ascii_letters, string.digits, etc.).
7. from pickle import dump, load:

- The pickle module implements binary protocols for serializing and deserializing a Python object structure. The dump function is used to serialize an object and write it to a file, and the load function is used to read a serialized object from a file.

8. import pandas as pd:
    - Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like Series (1-dimensional) and DataFrame (2-dimensional) for handling and analyzing data. The alias pd is commonly used to simplify the reference to the Pandas module.
9. import numpy as np:
    - NumPy is the fundamental package for scientific computing with Python. It provides support for arrays, matrices, and many mathematical functions to operate on these data structures. The alias np is commonly used to simplify the reference to the NumPy module.
10. import re:
    - The re module provides support for regular expressions in Python. Regular expressions are used for string matching and manipulation based on patterns.
11. import matplotlib.pyplot as plt:
    - Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. The pyplot module is used for creating plots and charts. The alias plt is commonly used to simplify the reference to the pyplot module.
12. import seaborn as sns:
    - Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. The alias sns is commonly used to simplify the reference to the Seaborn module.
13. import plotly.express as px:
    - Plotly Express is a high-level data visualization library built on Plotly. It provides a concise and easy-to-use API for creating a wide range of interactive plots. The alias px is commonly used to simplify the reference to the Plotly Express module.

In [1]:
```
DATA_PATH= 'CNNLinks.csv'
```

- DATA_PATH: This is the name of the variable. In this context, it's used to store the path to a file.
- 'CNNLinks.csv': This is a string representing the filename or the path to the CSV file. The file presumably contains links to CNN news articles.

```
In [ ]: class Load_data:
            def __init__(self, dir_path):
                self.dir_path = dir_path

            def load(self):
                data = pd.read_csv(self.dir_path)
                return data

            def drop(self):
                data = self.load()
                data = data.drop(columns=["Id","Author", "Date"])

                return data

            def analyze_data(self):
                data = self.drop()

                for sr_no, url in enumerate(data['URL']):
                    article = Article(url)
                    article.download()
                    article.parse()
                    article.nlp()
                    analysis = TextBlob(article.text)
                    print(f'Title: {article.title}')
                    print(f'Summary: {article.summary}')
                    print('-----------------------------------------------')

            def sentiment(self):
                data = self.drop()
```

```
              for sr_no, url in enumerate(data['URL']):
                  article = Article(url)
                  article.download()
                  article.parse()
                  article.nlp()
                  analysis=TextBlob(article.text)
                  print(analysis.sentiment)
                  print(analysis.polarity)
                  print(f'Polarity: {"positive" if analysis.polarity>0 else "negative"
                  if analysis.polarity<0 else "neutral"}')
                  print(f'Subjectivity: {"Biased And Less Factual"
                  if 1 >= analysis.subjectivity > 0.4 else "Unbiased And More Factual"
                  if 0.4 >= analysis.subjectivity >= 0 else "Invalid subjectivity score"}')
                  print('-----------------------------------------------')
                  def sentiment_EDA(self):
              data = self.drop()
              sentiment_data = []
              for sr_no, url in enumerate(data['URL']):
                  article = Article(url)
                  article.download()
                  article.parse()
                  article.nlp()
                  analysis = TextBlob(article.text)
                  sentiment = [
                      analysis.polarity,
                      analysis.subjectivity,
                  ]
                  sentiment_data.append(sentiment)
              self.sentiment_data = pd.DataFrame(sentiment_data, columns=['Polarity', 'Subjectivity'])
              return self.sentiment_data
```

Class: Load_data

This class handles loading, processing, and analyzing news article data from a CSV file.

__init__(self, dir_path)

- Purpose: Initializes the class with the path to the CSV file.
- Parameters: dir_path - Path to the CSV file.

load(self)

- Purpose: Loads the CSV file into a pandas DataFrame.
- Returns: A pandas DataFrame containing the data from the CSV file.

drop(self)

- Purpose: Loads the data and drops unnecessary columns ("Id", "Author", and "Date").
- Returns: A pandas DataFrame with the specified columns removed.

analyze_data(self)

- Purpose: Analyzes the data by:
    - Loading and dropping columns.
    - Iterating through each URL in the data.
    - Downloading, parsing, and performing NLP on each article.
    - Printing the article's title and summary.

sentiment(self)

- Purpose: Performs sentiment analysis on the articles by:
    - Loading and dropping columns.
    - Iterating through each URL in the data.
    - Downloading, parsing, and performing NLP on each article.
    - Analyzing sentiment using TextBlob and printing the sentiment, polarity, and subjectivity.

sentiment_EDA(self)

- Purpose: Performs sentiment analysis and prepares data for exploratory data analysis (EDA) by:
    - Loading and dropping columns.
    - Iterating through each URL in the data.
    - Downloading, parsing, and performing NLP on each article.
    - Collecting polarity and subjectivity scores.
    - Storing the collected data in a pandas DataFrame and returning it.

In [3]:
```
obj= Load_data(DATA_PATH)
```

1. DATA_PATH = 'CNNLinks.csv':
    - This line sets the variable DATA_PATH to the string 'CNNLinks.csv', which is the path to the CSV file containing the data.
2. obj = Load_data(DATA_PATH):
    - This line creates an instance of the Load_data class. Here's what happens step-by-step:
        - The Load_data class's __init__ method is called with DATA_PATH as the argument.
        - Inside the __init__ method, the dir_path attribute of the instance (self.dir_path) is set to the value of DATA_PATH, which is 'CNNLinks.csv'.
        - After the __init__ method executes, the new instance of Load_data is assigned to the variable obj.

Purpose - By creating an instance of Load_data and assigning it to obj, you can now use this object to call the methods defined in the Load_data class. This allows you to load, process, and analyze the data in the CSV file.

In [6]:
```
data= obj.load()
data.head()
```

Out[6]:

	Id	Author	Date	URL
0	1	Stephen Collinson	April 29, 2024	https://edition.cnn.com/2024/04/28/politics/bi...
1	2	John Towfighi	April 28, 2024	https://edition.cnn.com/2023/10/11/business/ha...
2	3	Andrew Carey and Olga Voitovych	April 28, 2024	https://edition.cnn.com/2024/04/28/europe/russ...
3	4	Andrew Carey and Olga Voitovych	April 28, 2024	https://edition.cnn.com/2024/03/13/travel/aust...
4	5	Silvia Marchetti	April 28, 2024	https://edition.cnn.com/travel/italy-house-bou...

1. data = obj.load():
    - This line calls the load method of the obj instance, which is an instance of the Load_data class.
    - The load method reads the CSV file specified by the dir_path attribute ('CNNLinks.csv') into a pandas DataFrame and returns it.
    - The returned DataFrame is assigned to the variable data.
2. data.head():
    - This line calls the head method on the data DataFrame.
    - The head method returns the first five rows of the DataFrame, providing a quick look at the top of the dataset.

In [46]:
```
data= obj.drop()
data.head()
```

Out[46]:

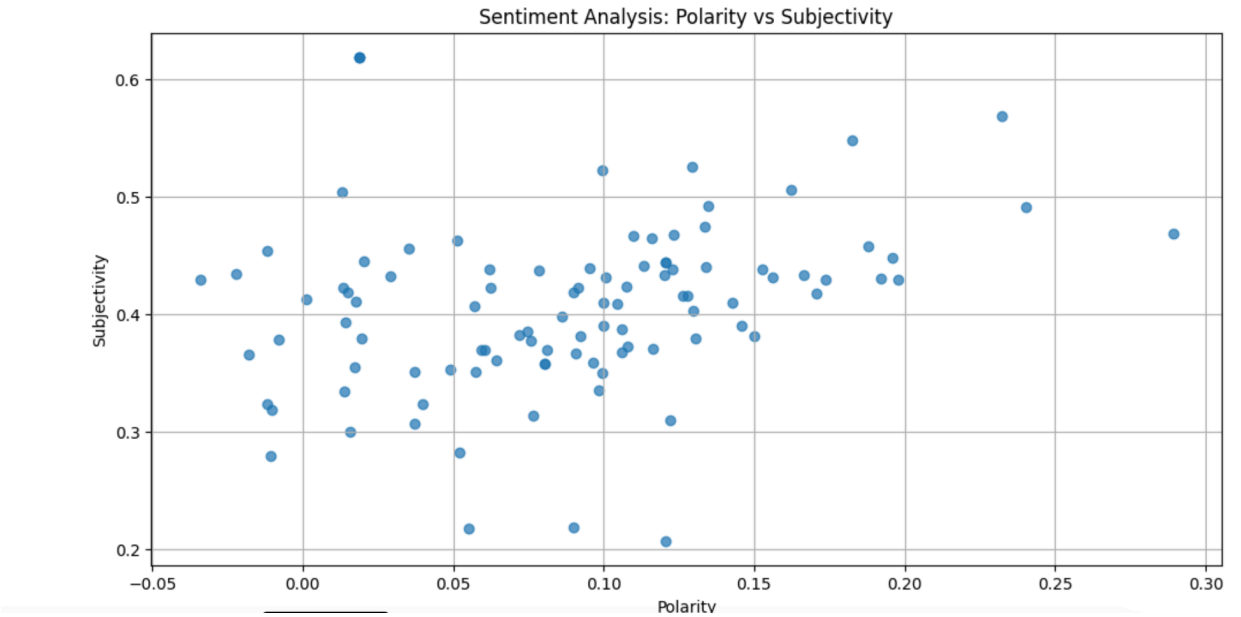
	URL
0	https://edition.cnn.com/2024/04/28/politics/bi...
1	https://edition.cnn.com/2023/10/11/business/ha...
2	https://edition.cnn.com/2024/04/28/europe/russ...
3	https://edition.cnn.com/2024/03/13/travel/aust...
4	https://edition.cnn.com/travel/italy-house-bou...

1. data = obj.drop():
    - This line calls the drop method of the obj instance, which is an instance of the Load_data class.
    - The drop method first calls the load method to read the CSV file specified by the dir_path attribute ('CNNLinks.csv') into a pandas DataFrame.
    - It then drops the columns "Id", "Author", and "Date" from the DataFrame.
    - The modified DataFrame is returned and assigned to the variable data.
2. data.head():
    - This line calls the head method on the data DataFrame.
    - The head method returns the first five rows of the DataFrame, providing a quick look at the top of the dataset.

```
sentiment_df = obj.sentiment_EDA()

# Plot the EDA
plt.figure(figsize=(12, 6))
plt.scatter(sentiment_df['Polarity'], sentiment_df['Subjectivity'], alpha=0.7)
plt.title('Sentiment Analysis: Polarity vs Subjectivity')
plt.xlabel('Polarity')
plt.ylabel('Subjectivity')
plt.grid(True)

plt.show()
```

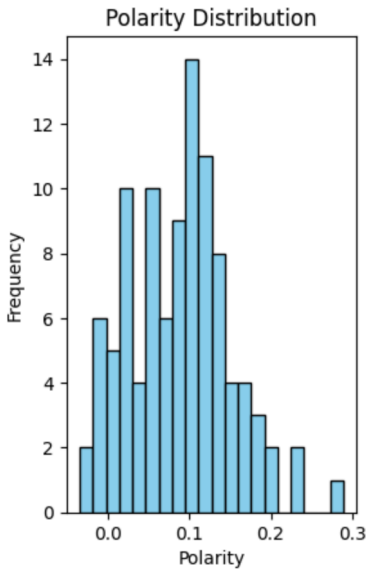


1. sentiment_df = obj.sentiment_EDA():
    - Calls the sentiment_EDA method of the obj instance, which is an instance of the Load_data class.
    - This method performs sentiment analysis on the articles, calculates polarity and subjectivity scores, and stores the data in a pandas DataFrame.
    - The resulting DataFrame containing sentiment scores is assigned to the variable sentiment_df.
2. Plotting with Matplotlib:
    - plt.figure(figsize=(12, 6)): Sets the figure size for the plot.
    - plt.scatter(sentiment_df['Polarity'], sentiment_df['Subjectivity'], alpha=0.7): Creates a scatter plot with polarity on the x-axis and subjectivity on the y-axis. The alpha parameter sets transparency.
    - plt.title('Sentiment Analysis: Polarity vs Subjectivity'): Sets the title of the plot.
    - plt.xlabel('Polarity'): Sets the x-axis label.
    - plt.ylabel('Subjectivity'): Sets the y-axis label.
    - plt.grid(True): Adds a grid to the plot.

- plt.show(): Displays the plot.

In [48]:
```
plt.subplot(1, 2, 1)
plt.hist(sentiment_df['Polarity'], bins=20, color='skyblue', edgecolor='black')
plt.title('Polarity Distribution')
plt.xlabel('Polarity')
plt.ylabel('Frequency')
```

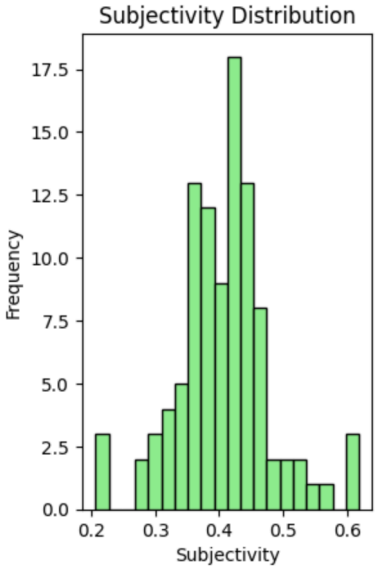
Out[48]: `Text(0, 0.5, 'Frequency')`



1. plt.subplot(1, 2, 1):
   - Creates a subplot grid with 1 row and 2 columns, and selects the first subplot for the following plot.
2. plt.hist(sentiment_df['Polarity'], bins=20, color='skyblue', edgecolor='black'):
   - Plots a histogram of the polarity values from the DataFrame sentiment_df.
   - bins=20 specifies the number of bins (intervals) for the histogram.
   - color='skyblue' sets the color of the bars to sky blue.
   - edgecolor='black' sets the color of the edges of the bars to black.
3. plt.title('Polarity Distribution'):
   - Sets the title of the subplot to "Polarity Distribution".
4. plt.xlabel('Polarity'):
   - Sets the label for the x-axis to "Polarity".
5. plt.ylabel('Frequency'):
   - Sets the label for the y-axis to "Frequency".

In [49]:
```
plt.subplot(1, 2, 2)
plt.hist(sentiment_df['Subjectivity'], bins=20, color='lightgreen', edgecolor='black')
plt.title('Subjectivity Distribution')
plt.xlabel('Subjectivity')
plt.ylabel('Frequency')
```

Out[49]: Text(0, 0.5, 'Frequency')



1. plt.subplot(1, 2, 2):
    - Creates a subplot grid with 1 row and 2 columns, and selects the second subplot for the following plot.
2. plt.hist(sentiment_df['Subjectivity'], bins=20, color='lightgreen', edgecolor='black'):
    - Plots a histogram of the subjectivity values from the DataFrame sentiment_df.
    - bins=20 specifies the number of bins (intervals) for the histogram.
    - color='lightgreen' sets the color of the bars to light green.
    - edgecolor='black' sets the color of the edges of the bars to black.
3. plt.title('Subjectivity Distribution'):
    - Sets the title of the subplot to "Subjectivity Distribution".
4. plt.xlabel('Subjectivity'):
    - Sets the label for the x-axis to "Subjectivity".
5. plt.ylabel('Frequency'):
    - Sets the label for the y-axis to "Frequency".

```
In [50]: obj.analyze_data()

Title: Biden is up against nostalgia for Trump's first term
Summary: More than half, 55%, of all Americans say they see Trump's presidency as a success, while 44% see it as a
failure.
Four years ago, he was able to assail Trump's time in office from his position as a challenger.
Biden won all three in 2020 after Trump won them in 2016 in his victory over Democratic nominee Hillary Clinton.
In the poll, 92% of Republicans view Trump's time in office a success, while just 73% of Democrats say Biden's has
been a success.
And while 85% of Democrats polled say they back Biden, 91% of Republicans say they support Trump.
--------------------------------------------------
```

1. obj.analyze_data():
    - This line invokes the analyze_data() method, which is defined within the Load_data class.
    - The analyze_data() method is responsible for analyzing the data, specifically the news articles, stored in the CSV file.
    - Inside the analyze_data() method:
        - It loads the data from the CSV file.
        - It drops unnecessary columns such as "Id", "Author", and "Date".
        - It iterates through each URL in the DataFrame.
        - For each URL, it downloads the article, parses it, performs natural language processing (NLP), and prints the title and summary of the article.

Reference

Summarize News Articles with Machine Learning in Python