CONTENTS

SR NO	TITLE
1	INTRODUCTION
2	METHODOLOGY (APPLICATION LOGIC)
3	GUI
4	CODE SNIPPETS
5	TECHNOLOGIES USED
6	REFERENCES

Introduction

The motivation behind the development of our NLP (Natural Language Processing) application stems from the recognition of a pressing need for efficient access to news content across diverse domains. In an era characterized by an overwhelming influx of information, users often face challenges in navigating through extensive textual content to obtain relevant updates within their areas of interest. This necessitates the creation of a solution that not only aggregates news articles but also facilitates their concise summarization for enhanced comprehension.

The primary objective of our project is to address the information overload experienced by users by providing them with a streamlined platform for accessing news articles across five main domains: India, World, Business, Technology, and Sports. By curating and summarizing news content from reputable sources within each domain, our application aims to empower users with the ability to quickly grasp key developments and insights without having to sift through voluminous texts.

Furthermore, our application acknowledges the diverse preferences of users by offering the additional functionality of converting summarized news articles into audio format. This feature caters to individuals who may prefer auditory consumption of information, thereby enhancing accessibility and inclusivity.

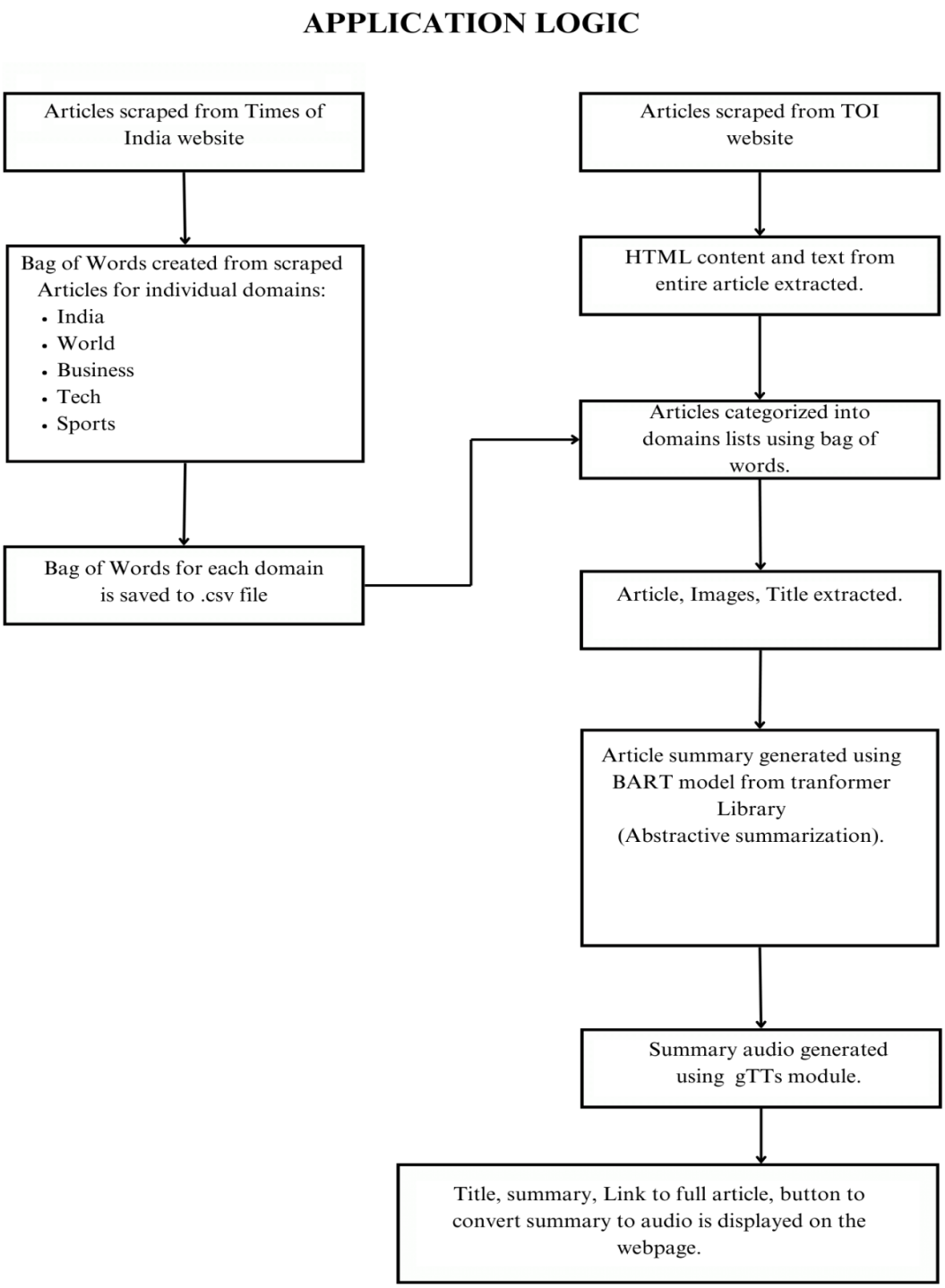
In essence, our NLP application endeavors to bridge the gap between the abundance of news content available and the efficient consumption of relevant information. By leveraging advanced text summarization techniques and audio conversion capabilities, we aim to streamline the news consumption experience, making it more convenient, accessible, and tailored to the diverse needs of users across different domains.

The NLP techniques implemented were:

1. Parsing and processing: Applied to articles to tokenize, tag parts of speech, and recognize named entities, improving content understanding.
2. Bag of Words (BoW): Utilized for numerical representation of text, aiding in word frequency analysis, used for article categorization.
3. Text Summarization: Automated to extract essential information from articles, providing concise summaries.
4. Text-to-Audio Conversion: Enabled users to listen to summarized content, enhancing accessibility and convenience.

Methodology

Application Logic:



Web scraping is the process of automatically extracting information from websites. It involves writing code (usually in programming languages like Python) to visit a website, retrieve its HTML code, and then parse through that code to extract the desired data. This data can include text, images, links, and other content present on the webpage.

Creating a "bag of words" from scraped articles involves compiling a list of all unique words present in the articles for each domain (India, World, Business, Tech, Sports) and counting the frequency of occurrence of each word within that domain. This process encompasses text preprocessing, tokenization, frequency counting, and analysis, enabling insights and applications such as sentiment analysis, topic modeling, and classification based on the words and their frequencies within each domain's articles.

Bag of words for each domain is saved to .csv files, containing a list of unique words along with their frequencies within the respective domain's articles, facilitating easy access and analysis of the textual data.

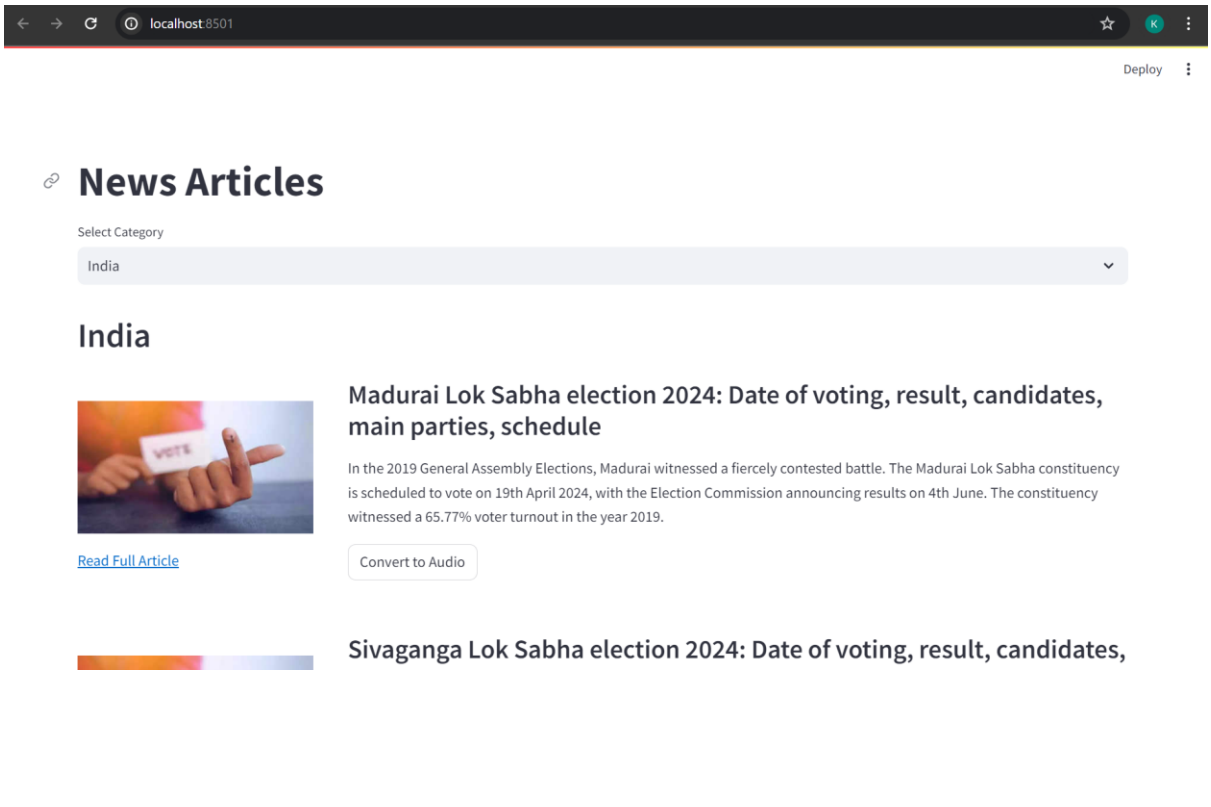
Articles are categorized into domains by comparing their content against predefined lists of words associated with each domain, known as "bag of words." Each domain has its own list of characteristic words. The articles' text is analyzed, and the presence or frequency of words from each domain's bag of words determines the categorization. For example, if an article contains many words from the "Tech" domain's bag of words, it's classified as a tech-related article. This method enables automated classification of articles into relevant domains based on their textual content.

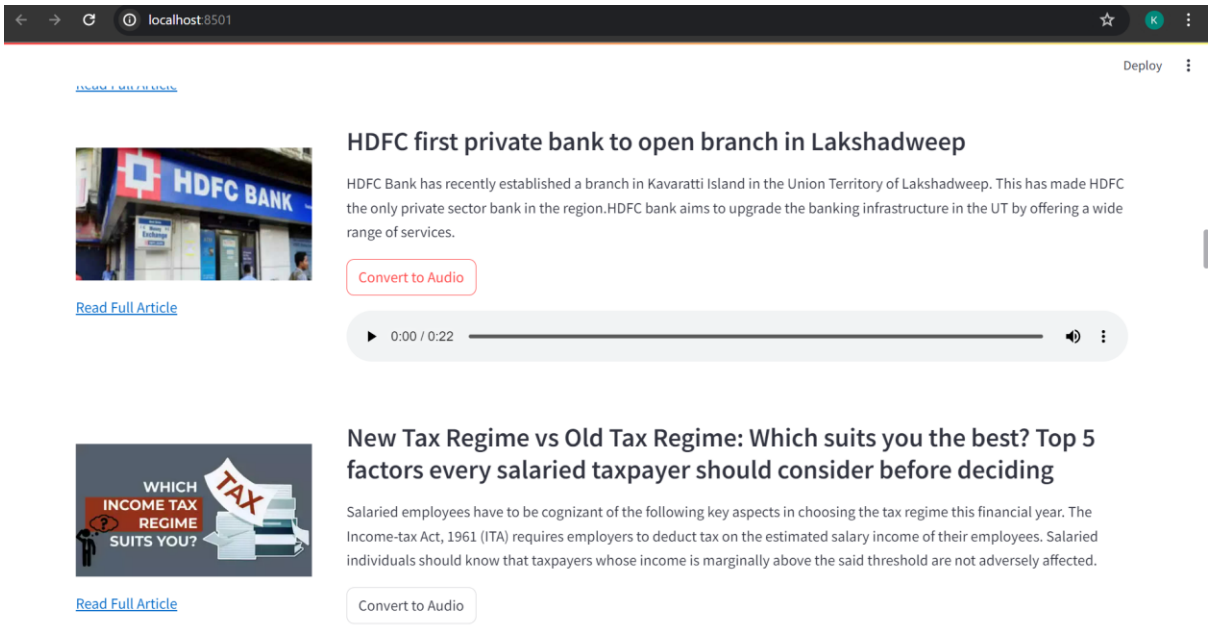
The process of categorizing articles into domains using a bag of words involves the extraction of key elements such as the article's main textual content, associated images, and headline or title. This extraction procedure facilitates a comprehensive analysis of the articles sourced from various platforms. By extracting the article text, images, and title, we enable detailed examination and classification based on both textual and visual components. This method enhances the accuracy of domain classification and provides valuable insights for further analysis.

Abstractive summarization using models like BART from the transformer library involves generating concise summaries of articles by understanding the context and generating new sentences that capture the key information. Unlike extractive summarization, which selects and rearranges existing sentences, abstractive summarization can generate novel sentences to convey the essence of the original article. This process leverages advanced natural language processing techniques to produce coherent and informative summaries tailored to the content of the input articles.

A web page displaying a title, summary, link to the full article, and a button to convert the summary to audio likely utilizes the gTTS (Google Text-to-Speech) module to generate audio summaries from the provided text. This setup enables users to access summarized content in audio format, enhancing accessibility and convenience for individuals preferring auditory consumption.

Graphical User Interface Screenshots:





Code Snippets:

Code for article categorization

```
import pandas as pd
from nltk.corpus import stopwords
# Preprocessing function
def preprocess_text(text):
    # Tokenize the text
    tokens = text.lower().split()
    # Remove punctuation and stopwords
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word.isalnum() and word not in stop_words]
    return tokens
# Function to calculate similarity score between text and bag of words
def similarity_score(text, bow):
    # Tokenize the input text
    tokens = preprocess_text(text)
    # Calculate similarity score based on common words with bag of words
    common_words = set(tokens) & set(bow['Word'])
    score = sum(bow[bow['Word'] == word]['Frequency'].values[0] for word in common_words)
    return score
# Function to classify text domain using bag of words
def classify_text_domain(text):
    # Load bags of words from CSV files
    india_bow = pd.read_csv("india_bow.csv")
    world_bow = pd.read_csv("world_bow.csv")
    business_bow = pd.read_csv("business_bow.csv")
    tech_bow = pd.read_csv("tech_bow.csv")
    sports_bow = pd.read_csv("sports_bow.csv")
```

```
    # Calculate similarity scores between input text and bags of words
    india_score = similarity_score(text, india_bow)
    world_score = similarity_score(text, world_bow)
    business_score = similarity_score(text, business_bow)
    tech_score = similarity_score(text, tech_bow)
    sports_score = similarity_score(text, sports_bow)
    # Determine the domain with the highest similarity score
    scores = {
        'India': india_score,
        'World': world_score,
        'Business': business_score,
        'Technology': tech_score,
        'Sports': sports_score
    }
    domain = max(scores, key=scores.get)
    return domain
```

Code for Article summary generation:

```
from transformers import BartForConditionalGeneration, BartTokenizer
from newspaper import Config, Article, Source
import textwrap
import re


def text_summarizer(text):
    model_name = "facebook/bart-large-cnn"
    model = BartForConditionalGeneration.from_pretrained(model_name)
    tokenizer = BartTokenizer.from_pretrained(model_name)
```

```
    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length=1024, truncation=True)
    summary_ids = model.generate(inputs, max_length=125, min_length=50, length_penalty=2.0, num_beams=4, early_stopping=True)

    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
    formatted_summary = "\n".join(textwrap.wrap(summary, width=80))

    # Replace multiple spaces with a single space
    formatted_summary = re.sub(r'\s+', ' ', formatted_summary)

    return formatted_summary

def text_summarizer_old(text):



    model_name = "facebook/bart-large-cnn"
    model = BartForConditionalGeneration.from_pretrained(model_name)
    tokenizer = BartTokenizer.from_pretrained(model_name)

    inputs = tokenizer.encode("summarize: " + text, return_tensors="pt", max_length=1024, truncation=True)
    summary_ids = model.generate(inputs, max_length=125, min_length=50, length_penalty=2.0, num_beams=4, early_stopping=True)

    summary = tokenizer.decode(summary_ids[0], skip_special_tokens=True)
    formatted_summary = "\n".join(textwrap.wrap(summary, width=80))
    return formatted_summary
```

Technologies/Tools/Software Used

Libraries:

- os.path: This library provides functions to interact with the filesystem paths. It is used for checking file existence and truncating files.
- csv: The csv library is used for reading and writing CSV files, commonly used for storing tabular data. It is utilized for saving the scraped news articles into CSV format.
- requests: This library enables sending HTTP requests to web servers and handling their responses. It is used for making HTTP requests to fetch web pages in the web scraping process.
- pandas: Pandas is a powerful library for data manipulation and analysis. It is used for reading and processing CSV files containing scraped news articles.
- nltk.corpus: This part of the Natural Language Toolkit (NLTK) provides access to a variety of natural language corpora and lexical resources. It is used for accessing the stopwords corpus for text preprocessing.
- gtts: The Google Text-to-Speech (gTTS) library is used for converting text to speech. It is employed to generate audio summaries of news articles.
- streamlit: Streamlit is a Python library used for creating interactive web applications for data science and machine learning projects. It is utilized to build the user interface for displaying summarized news articles and converting them to audio.
- bs4 (BeautifulSoup): This module is part of the BeautifulSoup library, used for web scraping HTML and XML documents. It is used to parse HTML content and extract relevant information from web pages.
- newspaper: This module provides functionalities for web scraping news articles from various online sources. It is used to download and parse news articles from specified URLs.

Modules:

- gensum: This module contains functions for text summarization using pre-trained models. It is utilized to generate summarized versions of news articles.
- main2: This module contains functions related to categorizing articles based on their content. It is used to classify scraped news articles into different categories.
- main3: This module contains the main function start, which orchestrates the execution of various tasks such as scraping news articles, summarizing them, and saving them to CSV files.

Functions:

- text_summarizer: This function generates summarized versions of text using pre-trained models. It is used to create concise summaries of news articles.
- categorize_articles: This function categorizes news articles based on their content or domain. It organizes scraped articles into different categories like India, World, Business, Technology, and Sports.
- preprocess_text: This function preprocesses raw text data by tokenizing, removing punctuation, and filtering out stopwords. It is used as part of the text processing pipeline before summarization or categorization.
- similarity_score: This function calculates the similarity score between a given text and a bag of words (BoW). It is used to classify text domains by comparing them with predefined BoW models.
- classify_text_domain: This function determines the domain of a text (e.g., India, World, Business) based on its content. It is used to classify news articles into different categories.
- start: This function serves as the entry point of the project. It orchestrates the execution of various tasks such as scraping news articles, summarizing them, categorizing them, and saving them to CSV files. Additionally, it handles the initialization of the Streamlit app.

References:

- https://medium.com/swlh/abstractive-text-summarization-using-transformers-3e774cc42453
- https://www.projectpro.io/article/transformers-bart-model-explained/553
- https://huggingface.co/learn/nlp-course/en/chapter1/4
- https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/
- https://www.topcoder.com/thrive/articles/text-summarization-in-nlp
- https://www.datacamp.com/tutorial/streamlit
- https://newspaper.readthedocs.io/en/latest/
- https://link.springer.com/article/10.1007/s11205-023-03147-0
- https://realpython.com/python-web-scraping-practical-introduction/
- https://medium.com/@pelinokutan/how-to-convert-text-to-speech-with-python-using-the-gtts-library-dbe3d56730f1
- https://pypi.org/project/gTTS/