CSI 3016 – Natural Language processing

MULTILINGUAL NEWS SUMMARIZER

SURIYA NARAYANAN – 20MID0059; GURUARUNACHALAM – 20MID0068; PRABHATH SAI – 20MID0137

Guided by: Arivoli

AIM:

- The aim of this project is to develop a multilingual news summarizer that can effectively aggregate English news articles, generate concise summaries, and present these summaries in multiple languages. By leveraging NLP techniques, the system aims to make news content more accessible to users who may not be proficient in English or who prefer to consume news in their native language.

INTRODUCTION:

- In today's interconnected world, access to timely and relevant news is essential for staying informed about global events and developments.
- However, language differences pose a significant barrier to accessing news content, particularly for individuals who are not fluent in English.
- To address this challenge, we propose the development of a multilingual news summarizer, a tool that can aggregate English news articles from various sources, condense them into succinct summaries, and translate these summaries into multiple languages.
- By providing news content in a user's preferred language, regardless of their linguistic proficiency, this system aims to democratize access to information and promote cross-cultural understanding.

OBJECTIVES:

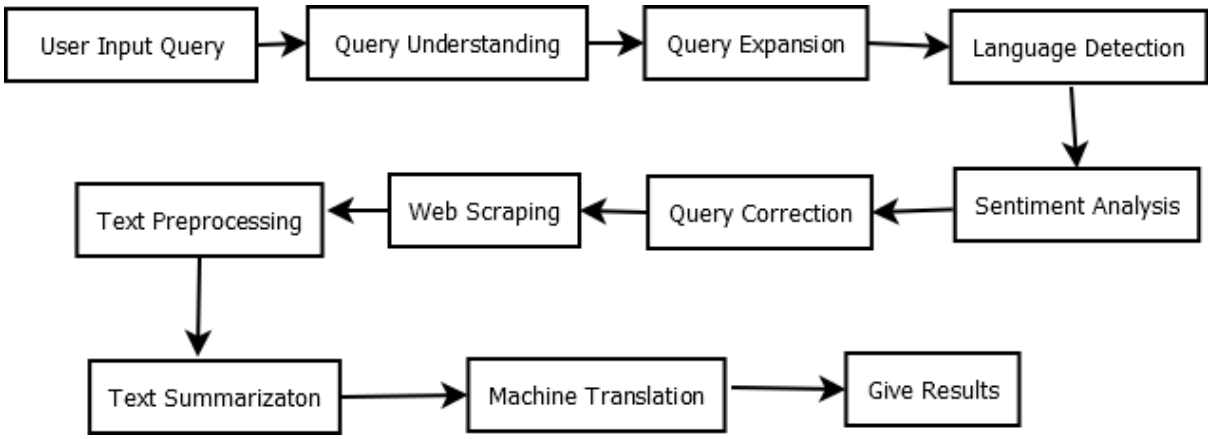
- Develop a web scraping module to collect English news articles from diverse sources.
- Implement natural language processing techniques to summarize the collected articles.

- Integrate machine translation algorithms to translate the summaries into multiple languages.
- Design a user-friendly interface for inputting search queries and displaying multilingual summaries.
- Evaluate the effectiveness and accuracy of the summarization and translation processes.

METHODOLOGY:

- Data Collection: Utilize web scraping techniques to gather English news articles from reputable sources such as news websites and RSS feeds.
- Text Summarization: Implement NLP algorithms such as extractive or abstractive summarization to condense the collected articles into concise summaries.
- Machine Translation: Utilize pre-trained models or APIs for machine translation to translate the summaries into multiple languages.
- User Interface Design: Develop a web-based interface allowing users to input search queries and view multilingual summaries.
- Evaluation: Assess the accuracy and comprehensiveness of the summaries and translations through human evaluation and automated metrics.

PROCESS FLOW:



LITERATURE SURVEY:

S. NO	PAPER - AUTHOR	ALGORITHM	OBSERVATION
1.	Keyword and Keyphrase Extraction Techniques: A Literature Review – Sifatullah Siddiqi	Instead of summarizing the entire content, keyphrase extraction focuses on identifying and extracting the most important phrases or keywords in a document. These keyphrases can provide a quick overview of the main topics covered.	Keyphrase summarization extracts vital phrases or keywords from a document, offering a concise overview of its main topics. It simplifies information retrieval and aids in understanding core themes quickly. Challenges include relevance determination and potential ambiguity in extraction.
2.	Automatic Text Summarization Using Gensim Word2Vec and K-Means Clustering Algorithm – Mofiz Mojib Haider; Md. Arman Hossin; Hasibur Rashid Mahi; Hossain Arif	TextRank: Unsupervised extractive summarization algorithm using sentence tokenization, word embeddings, and graph-based ranking. Efficiently extracts key information without needing labeled training data.	Gensim: Python library for TextRank extractive summarization algorithm. Quickly identify key sentences to summarize text. Easy to use and flexible. Popular for basic summarization tasks and broader NLP functionalities.

S. NO	PAPER - AUTHOR	ALGORITHM	OBSERVATION
3.	QBSUM: A large-scale query-based document summarization dataset from real-world applications - Mingjun Zhao 1 a, Shengli Yan 1 b, Bang Liu c, Xinwang Zhong b, Qian Hao b, Haolan Chen b, Di Niu a, Bowei Long b, Weidong Guo b	Query-Based Summarization: This technique involves generating a summary based on specific queries or keywords provided by the user. The system extracts relevant information related to the query and forms a concise summary.	Query-based summarization tailors summaries to user queries or keywords, offering personalized and focused content. It adapts dynamically to changing queries but faces challenges in accurately interpreting user input and ensuring comprehensive coverage in summaries.
4.	Automatic Text Summarization in Natural Language Processing - M. R. Desai; Bhagyashree Gachhinakatti; Pooja Balaganur; Rajeshwari Y; Laxmi Rathod	Algorithms for sentence ranking in extractive summarization include TF-IDF (Term Frequency-Inverse Document Frequency) and Graph-Based Methods (e.g., TextRank, PageRank).	Prioritize relevant information. Avoid redundancy. Identify and order important sentences. Include diverse content. Maintain sensitivity to context. Demonstrate consistency and scalability.

S. NO	PAPER - AUTHOR	ALGORITHM	OBSERVATION
5.	Natural Language Processing (NLP) based Text Summarization - A Survey - Ishitva Awasthi; Kuntal Gupta; Prabjot Singh Bhogal; Sahejpreet Singh Anand; Piyush Kumar Soni	Algorithms used in extractive summarization include: TF-IDF (Term Frequency-Inverse Document Frequency) TextRank PageRank For abstractive summarization, algorithms include: Sequence-to-Sequence models (e.g., using LSTM or Transformer architectures) Pointer-Generator Networks	Extractive Summarization: Pros: Preserves the original wording and context of important information. Cons: May lack coherence as it relies on existing sentences and might not offer a concise summary. Abstractive Summarization: Pros: Provides a more concise and coherent summary by generating new language. Cons: Requires a deeper understanding of the content, and the quality of output may vary based on the language model used.
6.	Approaches to Fake News Detection and Their Applicability to Romanian-Language News Analysis - Costin BUSIOC, Stefan RUSETI, Mihai DASCALU	SVM, Multinomial Naive Bayes, Support Vector Machine, Random Forest Classifier, CNN, Bi-LSTMs, Affinity Propagation, K-Means, Bert, XLNet, RoBERTa. **Datasets:** The Fake News Challenge (FNC-1), LIAR dataset, advocate.com, naturalnews.com, greenvillegazette.com, politicot.com.	Constructing a large and diversified dataset for Romanian fake news detection, leveraging pretrained NLP models like RoBERTa for improved accuracy. **Limitations:** Limited size of datasets for training complex neural networks, potential bias in highly credible source selection for dataset construction.

S. NO	PAPER - AUTHOR	ALGORITHM	OBSERVATION
7.	Classification of “Hot News” for Financial Forecast Using NLP Techniques - Savas¸ Yıldırım, Dhanya Jothimani, Can Kavaklıo˘glu, Ays¸e Bas¸ar	Logistic Regression, Support Vector Machine, k-Nearest Neighbors, multinomial Naive Bayes were utilized for classifying financial news articles as "hot" or "non-hot.“ **Dataset:** Dow Jones Newswires text feed spanning from 2013 to 2017 was employed for analysis, with a focus on balanced and imbalanced datasets.	**Proposed Work:** Future research could explore deep learning approaches and extend analysis to predict stock prices in both developed and emerging markets. **Limitations:** Performance degradation observed with imbalanced datasets despite attempted remedies such as SMOTE technique.
8.	Text Similarity Measures in News Articles by Vector Space Model Using NLP - Ritika Singh Satwinder Singh 1	Cosine similarity, Jaccard similarity, Euclidean distance. **Dataset:** Google News dataset containing news articles in Hindi and English.	**Proposed Work:** Expanding analysis to other languages and exploring enhancement with Doc2Vec model for improved accuracy. **Limitations:** Potential improvement with Doc2Vec model, limited to Hindi and English languages.
9.	Fake News Detection in Social Media using Graph Neural Networks and NLP Techniques: A COVID-19 Use-case - Abdullah Hamid, Nasrullah Sheikh, Naina Said,Kashif Ahmad, Asma Gul, Laiq Hassan, Ala Al-Fuqaha.	**Algorithms:** Bag of Words (BoW), BERT embedding, Graph Neural Networks (GNNs). **Dataset:** COVID-19 related tweets from MediaEval 2020 task.	**Proposed work:** Integration of textual and structural information for enhanced fake news detection. **Limitations:** Imbalanced dataset, reliance on text and structure independently.

Work Progress:

- **Request Package**: The request package is utilized for HTTP communication, primarily for fetching data from online sources. In this project, it facilitates the retrieval of content from various websites, such as news platforms, to enable web scraping.
- **Googletrans 4.0.0**: Googletrans 4.0.0 is a Python wrapper for the Google Translate API. It is employed to perform machine translation tasks, allowing the generated summaries to be translated into multiple languages.
- **Beautiful Soup**: Beautiful Soup stands as a widely-used Python library designed for web scraping. It aids in parsing HTML and XML documents, thereby facilitating the extraction of pertinent information from web pages. In this context, it serves the purpose of extracting article content from online sources.
- **Plaintext Parser**: While not a specific library, a plaintext parser likely refers to a tool or method employed for handling unformatted text content. This may involve preprocessing the text extracted from articles during the web scraping process.
- **Text Summarization Libraries**:

- **Edmundson Summarizer**: This library specializes in text summarization, offering capabilities for generating extractive summaries from textual data.
- **Gensim**: Gensim is a renowned Python library used for topic modeling and various natural language processing tasks, including text summarization.
- **Sumy**: Sumy is a dedicated library tailored for text summarization purposes. It supports multiple algorithms, including LexRank and Latent Semantic Analysis (LSA), for summarizing textual content.

- **SentimentIntensityAnalyzer**: The SentimentIntensityAnalyzer, a component of the NLTK (Natural Language Toolkit) library, is applied for sentiment analysis. This tool assigns sentiment scores to text, indicating the degree of positivity or negativity expressed within the content.

Algorithm used:

Edmund Summarizer

The Edmundson summarizer is a method for extractive text summarization that identifies and scores sentences based on the presence of specific cue words or phrases.

- Cue Phrase Identification: Cue phrases are identified based on domain knowledge or through analysis of the text. These phrases typically represent key concepts, significant events, or important topics within the text.
- Sentence Scoring: Each sentence in the text is scored based on the presence of cue phrases. Sentences containing more cue phrases or specific combinations of cue phrases receive higher scores, indicating greater importance or relevance.
- Content Overlap: Redundancy is minimized by penalizing sentences that contain similar or redundant information. This ensures that the summary contains diverse and representative content from the original text.

- Top Sentence Selection: The top-ranked sentences, based on their scores, are selected to form the summary. The number of sentences selected depends on the desired length of the summary, typically specified by the user.

CODE:

```
import requests
from bs4 import BeautifulSoup
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.edmundson import EdmundsonSummarizer
from googletrans import Translator
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4


def get_news_urls_bing(topic, limit=10):
    base_url = 'https://www.bing.com/news/search'
    search_params = {'q': topic}

    response = requests.get(base_url, params=search_params)
    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')

        no_results_message = soup.find('div', {'class': 'empty'})
```

```
        if no_results_message and "No results" in no_results_message.text:
            return "No articles found for the given topic."

        article_links = soup.find_all('a', {'class': 'title'})

        if article_links:
            return [link['href'] for link in article_links[:limit]]
        else:
            return "No news articles found for the given topic."
    else:
        return "Failed to retrieve data from Bing News."

def get_article_content(url):
    response = requests.get(url)

    if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        text_content = soup.get_text(separator='\n')
        return text_content
    else:
        return f"Failed to retrieve data from the URL. Status Code: {response.status_code}"

def summarize_article(url):
```

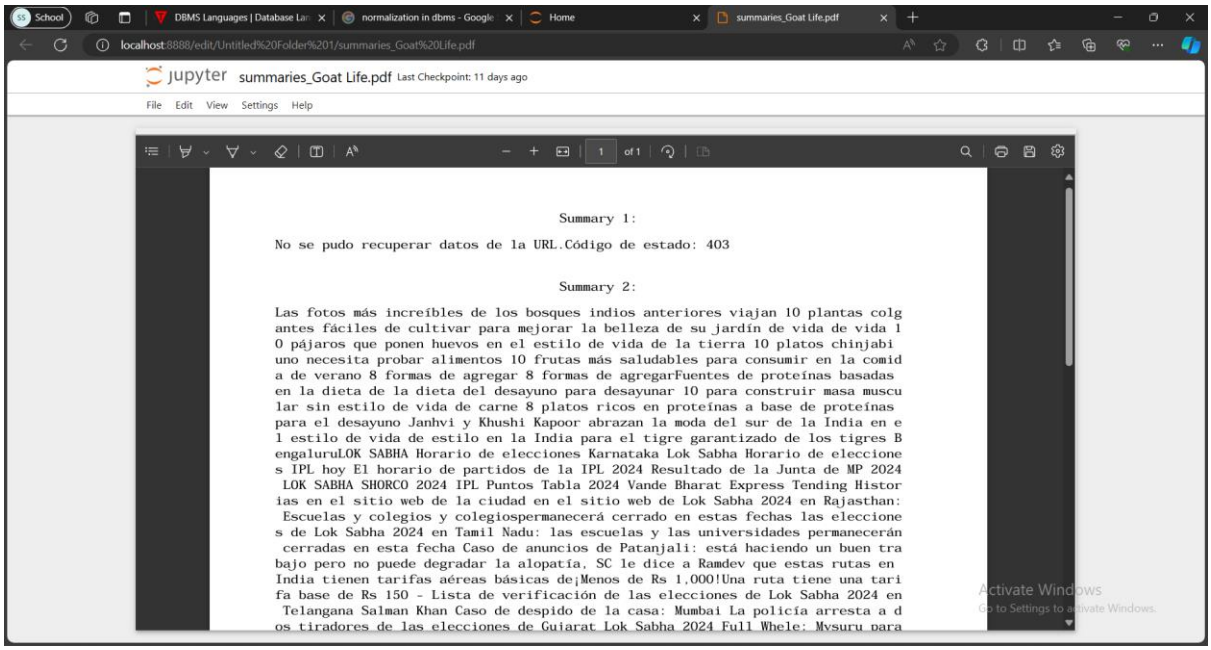
```
    article_content = get_article_content(url)

    if "Failed to retrieve data" in article_content:
        return article_content

    parser = PlaintextParser.from_string(article_content, Tokenizer("english"))

    summarizer = EdmundsonSummarizer()
    summarizer.bonus_words = ('example', 'keywords', 'for', 'your', 'domain')
    summarizer.stigma_words = ('avoid', 'these', 'words', 'if', 'possible')
    summarizer.null_words = ('irrelevant', 'words', 'that', 'don', 't', 'contribute')

    summary_sentences = summarizer(parser.document, sentences_count=5)

    summary = " ".join(str(sentence) for sentence in summary_sentences)
    return summary

def translate_text(text, target_language='es'):
    try:
        translator = Translator()
        translated_text = translator.translate(text, dest=target_language)
        return translated_text.text
    except:
```

```
        print("Sorry, can't translate")
        return text

def save_to_pdf(file_path, summaries):
    pdf_canvas = canvas.Canvas(file_path, pagesize=A4)
    pdf_canvas.setFont("Courier-Bold", 10)

    y_position = 800
    max_line_length = 80

    for idx, summary in enumerate(summaries, start=1):
        pdf_canvas.drawCentredString(A4[0] / 2, y_position, f"Summary {idx}:")
        y_position -= 20

        for line in summary.split('\n'):
            pdf_line = line.encode('latin-1', 'replace').decode('latin-1')


            segments = [pdf_line[i:i+max_line_length] for i in range(0, len(pdf_line),
max_line_length)]

            for segment in segments:
                pdf_canvas.drawString(50, y_position, segment)
                y_position -= 12
```

```
        y_position -= 20

    pdf_canvas.save()

topic = input("Enter the topic you want to search for : ")
target_language = input("Enter the target language : ")

summaries = []


urls = get_news_urls_bing(topic, limit=20)

translated_summaries_count = 0
for idx, url in enumerate(urls, start=1):
    print(f"\nProcessing Article {idx} ({url}):\n")
    summary = summarize_article(url)

    if target_language != 'en':
        translated_summary = translate_text(summary, target_language)
        if translated_summary.strip():
            print(translated_summary)
            summaries.append(translated_summary)
```

```
            translated_summaries_count += 1
        else:
            print(summary)
            summaries.append(summary)

        if translated_summaries_count == 5:
            break

pdf_file_path = f"summaries_{topic}.pdf"
save_to_pdf(pdf_file_path, summaries)
print(f"PDF saved: {pdf_file_path}")
```

Output:



CHALLENGES:

- ❖ Language Complexity: Dealing with nuances, idioms, and cultural references in different languages poses a challenge for accurate summarization and translation.

- Data Quality: Ensuring the reliability and relevance of the collected news articles is crucial for generating informative summaries.
- Cross-Language Ambiguity: Ambiguities and polysemy present in one language may not have direct equivalents in other languages, leading to potential loss of meaning during translation.
- Performance Optimization: Balancing the computational resources required for web scraping, summarization, and translation to ensure efficient system performance.
- Evaluation Metrics: Developing appropriate metrics to evaluate the quality of summaries and translations across different languages.