SumNews: A News Aggregation Website with Summarization

Semester Project

Bahria University Islamabad

Course: Natural Language Processing

Submitted By:

Abdullah Hassan	01-134221-007
Maaz Hussain	01-134221-035
M. Sibtain	01-134221-051

June 2, 2025

Contents

Abstract

In an era of information overload, keeping up with the latest news from various sources can be time-consuming. SumNews is a web-based application developed to address this challenge by aggregating news articles from prominent Pakistani news outlets—Dawn, Tribune, and The News International—and providing concise summaries using a TF-IDF-based text summarization technique. The system leverages web scraping through BeautifulSoup, backend services built with Flask, and a user-friendly frontend interface to present summarized news articles. This report outlines the system's architecture, technologies used, key code implementations, and workflow diagrams. The goal is to enable users to stay informed with minimal time investment by presenting key insights from full articles. Future enhancements aim to expand the range of sources, integrate more sophisticated summarization models, and introduce personalization features for a more tailored user experience.

Chapter 1

Introduction

SumNews is a web application designed to aggregate news from multiple sources and provide concise summaries to users. The primary goal of this project is to help users stay informed about the latest news without having to read through entire articles. By leveraging web scraping and natural language processing techniques, SumNews fetches articles from prominent Pakistani news websites such as Dawn, Tribune, and The News International, and generates summaries using the TF-IDF algorithm.

The website features a user-friendly interface with a header, hero section, main content area displaying news articles, and a footer. Users can navigate to specific news sources or refresh the page to see the latest stories. Each news article is displayed with its title, summary, and a link to the original article.

Chapter 2

Background

In today's digital age, the volume of news content available online is overwhelming. Users often find it challenging to keep up with the latest developments across various topics. SumNews addresses this issue by providing summarized versions of news articles, allowing users to quickly grasp the key points of each story.

The project utilizes several technologies:

- **Flask**: A micro web framework for Python, used to build the backend of the application.
- **BeautifulSoup**: A library for web scraping, which helps in extracting article content from news websites.
- **TF-IDF**: A statistical method for text summarization, which identifies the most important sentences in an article based on word frequency and rarity.

By combining these technologies, SumNews offers a streamlined way for users to consume news. The target audience includes individuals who want to stay informed but have limited time to read full articles.

Chapter 3

System Design

The system architecture of SumNews consists of three main components:

1. **Web Scraping**: Separate Flask applications are developed for each news source (Dawn, Tribune, and The News International). Each application is responsible for fetching the latest article links and scraping the content of those articles.

2. **Summarization**: Once the article content is scraped, it is passed through a TF-IDF-based summarization function that selects the top 5 sentences with the highest TF-IDF scores to create a summary.

3. **Frontend**: The summarized articles are displayed on a user-friendly web interface built with HTML and CSS. The main page features a grid layout of news articles, each showing the title, summary, and a link to the full article.

The backend is implemented using Flask, with three separate applications, each dedicated to one news source:

- **Dawn News**: Runs on port 5002, with the route `/api/dawn/latest` to get the latest articles.

- **Tribune News**: Runs on port 5001, with the route `/api/tribune/latest`.

- **The News International**: Runs on port 5003, with the route `/api/thenews/latest`.

Each application has two main functions:

- **get_[source]_latest_links**(): Fetches the latest article links from the respective news website.

- **scrape_[source]_article(url)**: Scrapes the content of a given article URL and generates a summary using TF-IDF.

The summarization is handled by the `summarize_tfidf` function in the `algorithm/summarization.` module, which takes the full text of the article and returns the top 5 sentences based on TF-IDF scores.

A high-level architecture of SumNews can be visualized as follows:

- **User Interface**: HTML/CSS frontend that displays summarized news articles.
- **Backend Servers**: Three Flask applications, each handling one news source, providing API endpoints for fetching summarized articles.
- **External News Websites**: Dawn, Tribune, and The News International, from which articles are scraped.

The data flow is as follows: When a user visits the SumNews website, the frontend makes requests to the respective Flask APIs to get the latest summarized articles. Each API scrapes the news website, summarizes the articles using TF-IDF, and returns the data to the frontend, which then displays it to the user.

3.1 Key Code Snippets

Some key parts of the code include:

- The `get_dawn_news_links()` function, which uses `requests` and `BeautifulSoup` to fetch and parse the Dawn website to extract article links.
- The `scrape_dawn_article(url)` function, which scrapes the content of a given Dawn article URL and generates a summary.
- Similarly, for Tribune and The News International, there are corresponding functions.

The summarization function `summarize_tfidf(text, number_of_sentences=5)` uses TF-IDF to score sentences and select the top ones for the summary. Here is an example of how the summarization is implemented:

```
def calculate_sentence_tfidf(sentences):
    vectorizer = TfidfVectorizer(stop_words='english')
    tfidf_matrix = vectorizer.fit_transform(sentences)
    sentence_scores = np.array(tfidf_matrix.sum(axis=1)).flatten()
    return sentence_scores

def summarize_tfidf(text, number_of_sentences=5):
    original_sentences = nltk.sent_tokenize(text)
    if len(original_sentences) == 0:
        return []
    sentence_scores = calculate_sentence_tfidf(original_sentences)
    ranked_sentences = sorted(((score, sentence) for score, sentence in zip(sentence_scores, original_sentences)), reverse=True)
    summary = [ranked_sentences[i][1] for i in range(min(number_of_sentences, len(ranked_sentences)))]
    return summary
```

3.2 Workflow Diagrams

To better understand how each component operates, the following UML-style workflow diagrams illustrate the internal structure and flow of the code for each major module in the project.

3.2.1 Dawn News Workflow

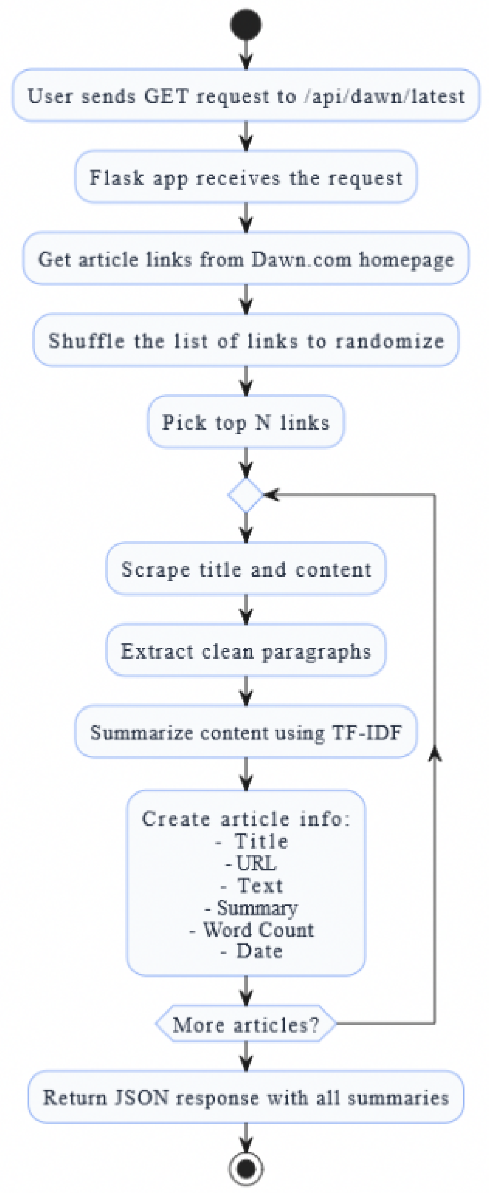
User sends GET request to /api/dawn/latest
Flask app receives the request
Get article links from Dawn.com homepage
Shuffle the list of links to randomize
Pick top N links
Scrape title and content
Extract clean paragraphs
Summarize content using TF-IDF
Create article info:
- Title
- URL
- Text
- Summary
- Word Count
- Date
More articles?
Return JSON response with all summaries

Figure 3.1: Workflow diagram for Dawn news scraping and summarization

3.2.2 Tribune News Workflow

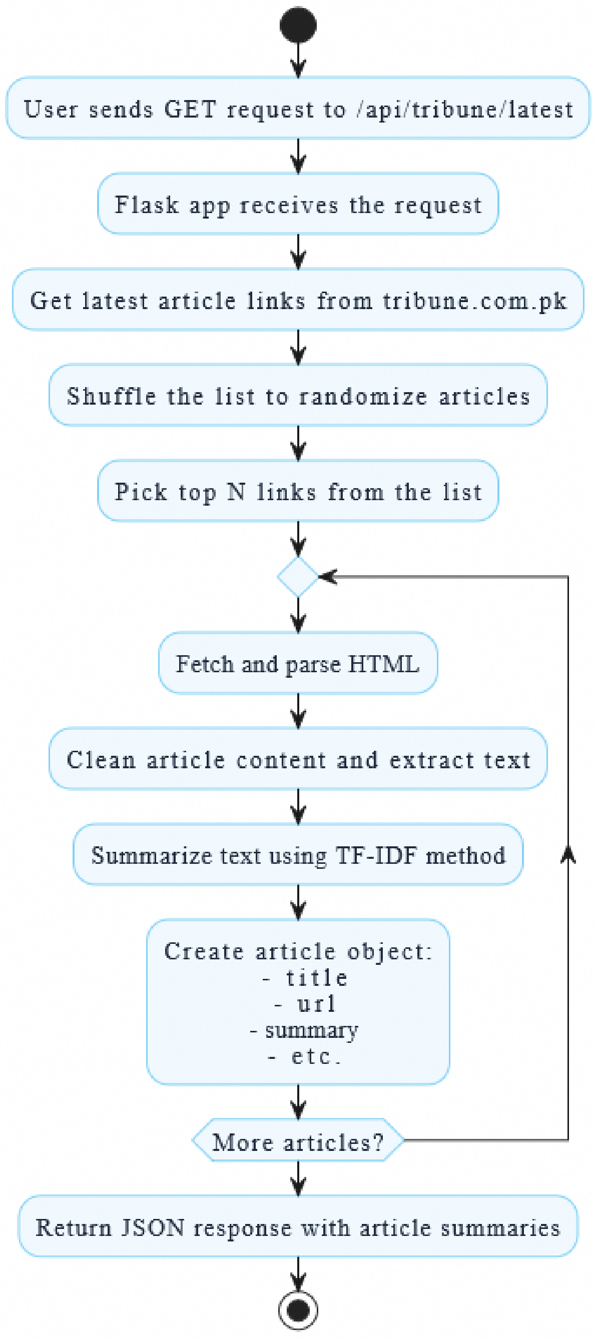
User sends GET request to /api/tribune/latest
Flask app receives the request
Get latest article links from tribune.com.pk
Shuffle the list to randomize articles
Pick top N links from the list
Fetch and parse HTML
Clean article content and extract text
Summarize text using TF-IDF method
Create article object:
- title
- url
- summary
- etc.
More articles?
Return JSON response with article summaries

Figure 3.2: Workflow diagram for Tribune news scraping and summarization

3.2.3 The News International Workflow

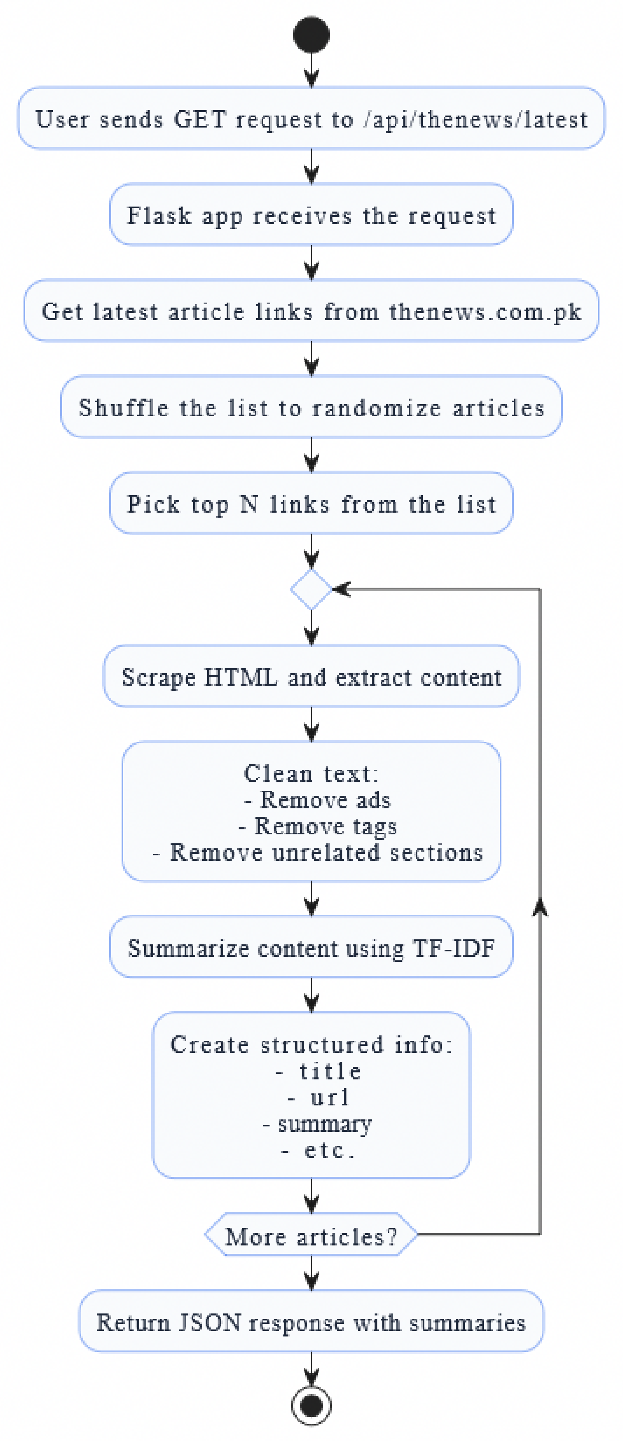
User sends GET request to /api/thenews/latest
Flask app receives the request
Get latest article links from thenews.com.pk
Shuffle the list to randomize articles
Pick top N links from the list
Scrape HTML and extract content
Clean text:
- Remove ads
- Remove tags
- Remove unrelated sections
Summarize content using TF-IDF
Create structured info:
- title
- url
- summary
- etc.
More articles?
Return JSON response with summaries

Figure 3.3: Workflow diagram for The News International scraping and summarization

3.2.4 Summarization Module Workflow

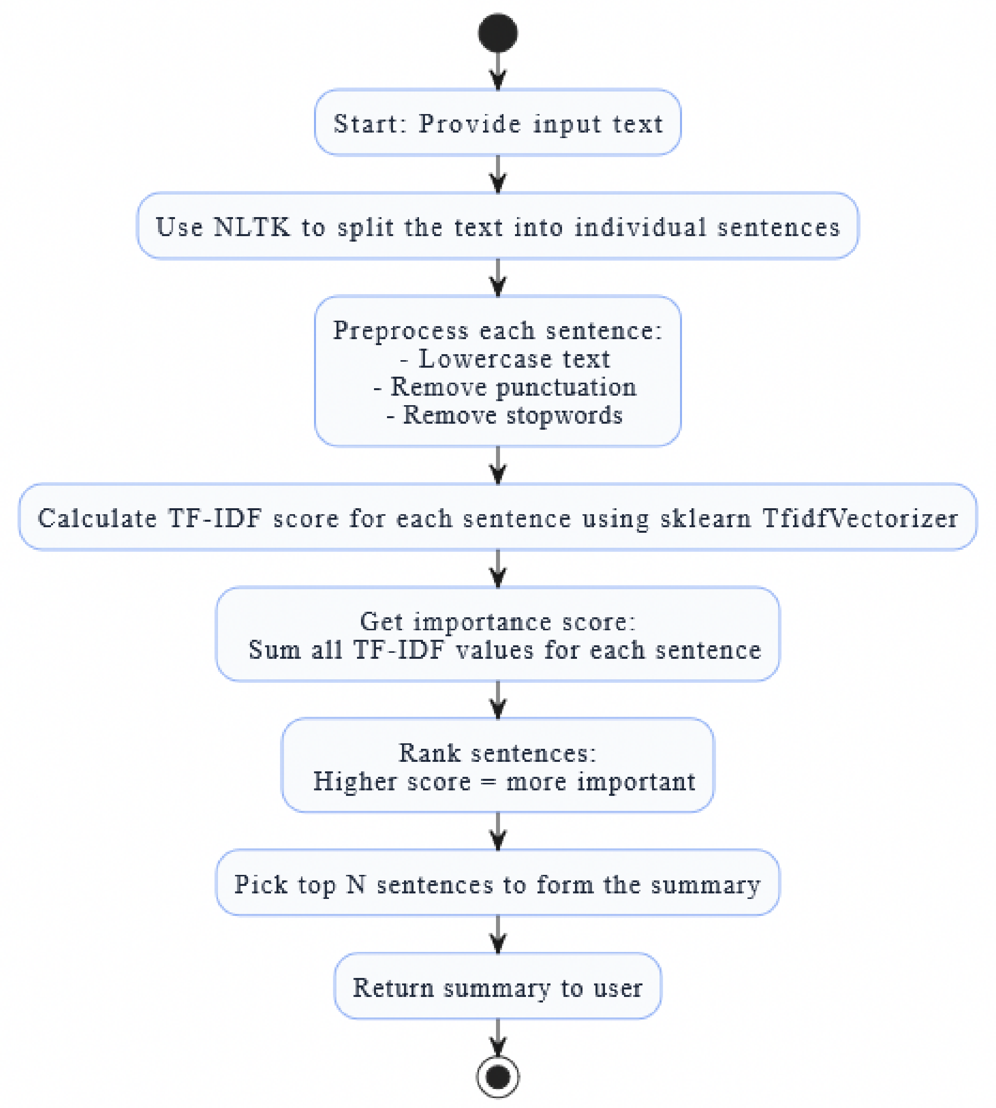
Start: Provide input text
Use NLTK to split the text into individual sentences
Preprocess each sentence:
- Lowercase text
- Remove punctuation
- Remove stopwords
Calculate TF-IDF score for each sentence using sklearn TfidfVectorizer
Get importance score:
Sum all TF-IDF values for each sentence
Rank sentences:
Higher score = more important
Pick top N sentences to form the summary
Return summary to user

Figure 3.4: Workflow diagram for TF-IDF-based summarization process

Chapter 4

Conclusions and Future Work

SumNews demonstrates the feasibility of creating a news aggregation platform with automatic summarization. The project successfully integrates web scraping, natural language processing, and web development to provide a valuable service to users.

However, there are several areas for improvement:

- **Expand News Sources**: Add more news websites to provide a broader range of news.
- **Improve Summarization**: Explore more advanced summarization techniques, such as abstractive summarization using machine learning models, to generate more coherent and informative summaries.
- **Real-Time Updates**: Implement a mechanism to fetch and update news articles in real-time, ensuring users always have access to the latest news.
- **User Personalization**: Allow users to customize their news feed based on their interests and preferences.

In conclusion, SumNews is a step towards making news consumption more efficient and accessible. With further development, it can become a comprehensive platform for staying informed in today's fast-paced world.