Sri Lanka Institute of Information Technology

Information Retrieval and Web Analytics (IT3041)

End-to-End Text Summarization and Analysis System

Report

IT22057556	M.A.M.A.Muhsin
IT22628404	M.R.F.Safwa
IT22597342	J.M.H.C.Jayasinghe
IT22003928	F.N.F.Ramla

Abstract

This project introduces an end-to-end text summarization and analysis system that leverages large language models (LLMs) for key tasks such as text summarization and sentiment analysis. Models like Bart and Bert are used for summarization, while tiny llama and Vader handles sentiment analysis, ensuring high-quality results. Additional features such as topic modeling and keyword extraction are implemented using specialized algorithms. The system is built using Streamlit, providing an interactive and user-friendly interface where users can upload text and instantly receive detailed analysis outputs. The backend, developed in Python, integrates LLMs to enhance the accuracy and efficiency of the system. Performance is evaluated using metrics such as ROGUE, precision, recall, and F1-score to ensure reliable outcomes. This project demonstrates the practical use of LLMs in automating text analysis tasks and offers potential applications in fields like news and research analysis. Future improvements may include expanding the system's capabilities and incorporating more advanced features for broader text analysis tasks.

Introduction

With the exponential growth of online news content, readers are often overwhelmed by the sheer volume of information available across multiple platforms. Staying informed about current events and critical issues requires efficient access to key insights from a flood of articles. However, manually reading and summarizing long articles is time-consuming and may result in important details being overlooked. To address this, advancements in Natural Language Processing (NLP) and large language models (LLMs) offer powerful tools to automate the summarization and analysis of news articles.This project focuses on building an end-to-end news article summarization and analysis system that automates the extraction of crucial information from news articles. By utilizing LLMs such as Bart and Bert for summarization and TinyLlama and Vader Model for sentiment analysis, the system enables users to quickly generate summaries of lengthy news articles and evaluate the overall sentiment conveyed in the content. Additionally, the system extracts key topics and keywords, helping users gain a deeper understanding of the article without having to read it in full.

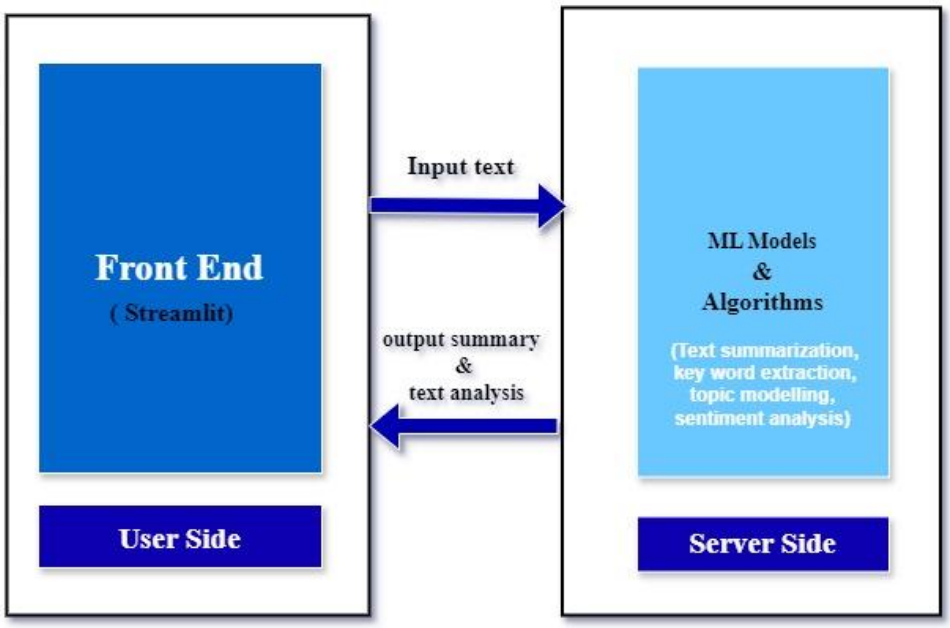
The **objective** of this project is to enhance the accessibility of news content by providing users with concise, accurate summaries and insightful analysis. This system allows users to efficiently process multiple articles, saving time while still obtaining essential information. The tool can be invaluable for journalists, researchers, and everyday readers who need to stay updated on current events but cannot dedicate time to reading every article in detail.

The **scope** of the project is focused on processing individual news articles in real-time through an interactive interface built using Streamlit. The system does not store data or manage long-term records, emphasizing simplicity and ease of use.

To ensure the system delivers high-quality results, the underlying models have been **fine-tuned** specifically for handling news-related content. This fine-tuning process enhances the models' ability to generate more accurate and contextually relevant summaries and analysis, making them better suited for diverse topics and varying styles of news writing. As a result, the system is able to provide concise and meaningful insights, tailored specifically to the nuances of news articles. However, despite the effectiveness of the current system, several potential future improvement could further enhance its capabilities including further multilingual support, Multimedia Integration and Fact-Checking.

System Design and Architecture

Overall High-level Architecture



The high-level architecture diagram illustrates the interaction between the user interface and the backend processing components of the system.

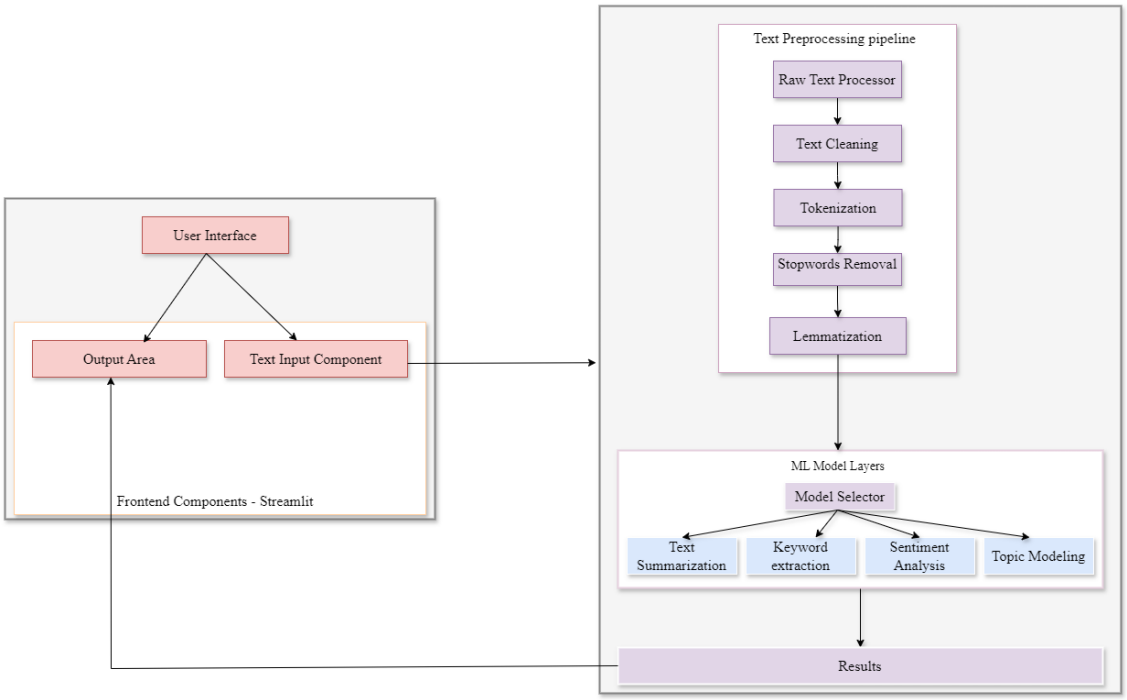
- **Front End (Streamlit)**: This component serves as the user interface, where users can input their text. The Streamlit framework provides an interactive platform for users to engage with the application, facilitating easy input of text for processing.
- **Input/Output Flow**: The diagram shows a clear flow of information:
- **Input Text**: Users enter their text into the front end, which is then sent to the server-side for processing.
- **Output Summary & Text Analysis**: After processing, the backend returns the generated summary and analysis results to the front end for display to the user.
- **Server Side (ML Models & Algorithms)**: This component encompasses the machine learning models and algorithms that perform the core tasks of the application, including:
- Text summarization
- Keyword extraction
- Topic modeling
- Sentiment analysis

Overall, this architecture highlights the seamless integration between user input and backend processing, allowing for efficient text analysis and summarization through an easy-to-use interface.

Flow of Data

- User Input:
- Input Text Field: The user enters a news article or any text they want to summarize.
- Summary Length Input: The user specifies the desired length of the summary (short, medium, long).
- Type of Summary: The user selects the type of summary either extractive, abstractive.
- Overall Text Preprocessing:
- The text is preprocessed based on an overall preprocessing required by all the tasks.
- Key Word Extraction:
- The preprocessed text is used to initially generate the keywords in the input text.
- Summarization:
- Summary is generated based on user text and other user inputs of length and type of summary.
- Sentiment Analysis:
- Sentiment Analysis on the summarized text is generated giving results as positive, negative or neutral
- Topic Modeling:
- Topics are modeled based on the summarized text.
- Output Display:
- Generated Summary: The final summarized text is presented to the user next.
- Key Words: The extracted key words are shown for the user's reference.
- Sentiment Result: The identified sentiment is displayed alongside the summary.
- Topics Modeled: The identified topics are displayed below sentiment analysis

The diagram below shows the flow of work


User Interface
Output Area
Text Input Component
Frontend Components - Streamlit
Text Preprocessing pipeline
Raw Text Processor
Text Cleaning
Tokenization
Stopwords Removal
Lemmatization
ML Model Layers
Model Selector
Text Summarization
Keyword extraction
Sentiment Analysis
Topic Modeling
Results

Technology Stack

Frontend Technolgy

- Streamlit

Backend Technolgy:

- Python
- Libraries: Transformers, Pandas

Machine Learning Frameworks:

- PEFT (Parameter-Efficient Fine-Tuning)

Natural Language Processing Libraries:

- NLTK (Natural Language Toolkit)
- SpaCy

Deployment and Hosting

- Streamlit sharing

Model Training Resources:

- Google Colab
- Local CPU/GPU (Using Jupyter Notebook)

Version Control

- Git

Implementation

Backend Development

The backend of our news article summarization and analysis system is designed to process text inputs (news articles) in real-time, generating summaries, performing sentiment analysis, keyword extraction and topic modelling. It was implemented in Python, using pre-trained and fine-tuned models from Hugging Face Transformers for Natural Language Processing (NLP) tasks and also through the use of algorithms built from scratch. The backend interacts seamlessly with the Streamlit frontend to deliver results in a user-friendly interface.

Backend Architecture:

The backend architecture of the news article summarization and analysis system is designed to provide a robust and efficient environment for processing news articles in real time. It integrates various components that work together seamlessly to deliver accurate summaries, sentiment analysis, keyword extraction and topic modeling. Below is a detailed breakdown of the architecture:

- Framework:
- Streamlit: The entire application is built using Streamlit, a Python-based framework that enables rapid development of interactive web applications. Streamlit provides a user-friendly interface, allowing users to input text and receive results in real time.
- Component Initialization:
- Upon launching the application, all necessary components, including machine learning models and algorithms, are loaded into memory. This includes:
- Fine-tuned Bart Model: A transformer model fine-tuned for text summarization. It processes the input text to generate concise summaries that encapsulate the main ideas of the news articles.
- TinyLlama and Vader Model: A transformer model for sentiment analysis. It evaluates the text to determine the emotional tone, classifying it as positive, negative, or neutral.
- LDA Algorithm: An algorithm used for modeling topic based on text
- Keyword extraction Algorithm: An algorithm used for keyword extraction. It identifies important keywords within the input text, aiding users in understanding the main ideas discussed.

- Input Preprocessing:
- Before any analysis, the input text undergoes an initial preprocessing step that is common to all tasks. This step involves:
- Cleaning: Removing extraneous characters, punctuation, html tags, white spaces and formatting issues to ensure the text is in a consistent state.
- Keyword Extraction:
- Once preprocessing is complete, the cleaned text is passed to the keyword extraction component. The keyword extraction algorithm analyzes the text to extract significant keywords, providing users with highlights and essential terms that reflect the content of the article.
- Processing Workflow:
- The processing of the text is initiated when the user clicks the process button in the Streamlit interface. The workflow is structured as follows:
- Summarization: The first step is to summarize the cleaned input text using the finetuned model. The model generates a summary that captures the key points and essence of the news article.
- Sentiment Analysis: Following summarization, a model analyzes the summarized text to assess its sentiment. This analysis categorizes the sentiment into positive, negative, or neutral, providing insights into the emotional context of the article.
- Topic Modeling: Simultaneously, the system performs topic modeling on the summarized text to identify prevalent themes and topics. This enhances the overall understanding of the article's main ideas.
- Real-Time Processing:
- The architecture allows for real-time processing, meaning users receive immediate feedback after submitting their articles. This responsiveness enhances the user experience and ensures that the application meets the needs of users seeking quick analysis.
- Stateless Design:
- The backend operates on a stateless design principle. Each user request is handled independently, without retaining any session data. This approach simplifies the architecture, making it easier to scale and maintain.

Key Features:

- **Text Summarization**: The system uses fine-tuned models like BART and T5 for abstractive summarization and BERT for extractive summarization. Users can define the desired summary length, which is dynamically adjusted based on the input text.
- **Keyword Extraction**: A TF-IDF-based algorithm identifies and ranks important keywords from the article, providing insights into key ideas.
- **Topic Modeling**: The LDA algorithm is used to generate relevant topics from the text, allowing for deeper understanding and categorization of the news content.
- **Sentiment Analysis**: VADER and Tiny LLaMA models analyze the sentiment of the article, providing an overall view of its emotional tone (positive, negative, neutral).

Frontend Development

Design and Functionality of the User Interface

The front-end is designed with a user-centric approach, focusing on simplicity and ease of use. The layout is clean and intuitive, allowing users to use the application seamlessly. The frontend is built using Streamlit, which enables rapid development of interactive web applications. This framework allows for straightforward integration of Python code, making it easy to deploy machine learning models and present results dynamically.

The design of the UI consists of an input section that has a text input box, users can paste or upload news articles into a designated text input box. This box is clearly labeled to ensure users know where to enter their content. In addition to text input, the interface supports file uploads (.txt format), enhancing flexibility for users who may have articles in different formats. Further a slider is also included to determine the length of the summary desired.

A prominent "Process" button is included, which users can click to initiate the summarization and analysis of the entered text. This button is designed to be easily accessible, encouraging users to interact with the application.

The results of the summarization, sentiment analysis, keyword extraction and topic modeling are displayed on the same page, immediately after processing. The keywords extracted from the input text are displayed in a section below the input text. The summarized text is presented in a distinct section, visually separated from the input area to enhance clarity. The sentiment analysis results are displayed with clear indicators (e.g., positive, negative, neutral) below the summarized text followed by the topics modeled.

Key UI Features and User Experience Considerations

- Intuitive Layout: The user interface is structured logically, with a clear division between the input section, processing area, and results display.
- Responsiveness: The interface is designed to be responsive, adapting to different screen sizes and devices to ensure a consistent user experience across platforms.
- Visual Appeal: A visually appealing color scheme and typography based on the user's browser theme are used to create an engaging user experience.

- Error Handling: Clear error messages and guidance are provided if users input invalid data
- Session Management: Although the system follows a stateless design, users can still have a seamless experience with session management features like preserving inputs temporarily during processing.
- The key UI components and features in detail have been included above in the design section

Database Management

Data Storage Solutions and Schema Design

The system is designed to operate without a persistent database, focusing on real-time processing of input data. As a result, no long-term data storage or schema design is required.

All data processing occurs in memory during the session. Input text, cleaned text, and processed outputs (summaries, sentiments, topics, keywords) are temporarily held in memory for the duration of the session.

The system follows a stateless architecture, meaning each user interaction is processed independently, with no data stored across sessions. This ensures that the application remains lightweight and simple, reducing complexity and overhead.

How Data is Handled and Accessed

Users input news articles or text directly into the interface. Once the text is submitted, it is immediately preprocessed, and then the keyword extraction is performed. On click of the process text button summarization, sentiment analysis, and topic modeling tasks are performed.

Key Features and Algorithms

Text Summarization

Text summarization is an essential task in the field of Natural Language Processing (NLP), aiming to condense large amounts of text into shorter, meaningful summaries. In our project, Summarizing was mainly focused on news articles, leveraging both abstractive and extractive summarization techniques. The goal is to allow users to quickly grasp the key points of a lengthy news article through concise and accurate summaries.

The system was built to provide real-time summarization through an interactive interface, primarily targeting two summarization strategies:

- Abstractive Summarization: Where the system generates new sentences that may not exist in the original text. It interprets and understands the important aspects of a text and generates a more “human” friendly summary.
- Extractive Summarization: Where the system extracts key sentences directly from the article to form a summary.

Additionally, preprocessing techniques, chunking mechanisms for handling long articles, and post-processing steps were integrated into the workflow to ensure high-quality summaries.

Below is a detailed breakdown of the models used, the preprocessing involved, and other key components that form the summarization pipeline.

Models Used for Text Summarization

1. **Abstractive Summarization**

In abstractive summarization, the models generate new sentences and rephrase the content in the text. For this task, we fine-tuned two state-of-the-art transformer models: **BART** and **T5**.

BART (Bidirectional and Auto-Regressive Transformers):

BART is a versatile transformer-based model that combines bidirectional encoding (where the context of a word is understood from both its left and right sides) with autoregressive decoding (where the summary is generated one token at a time). This dual approach allows BART to excel at tasks like abstractive summarization by producing grammatically coherent and contextually relevant summaries.

BART was fine-tuned using the **MultiNews dataset**, which contains news articles paired with human-written abstractive summaries. This dataset provides a comprehensive base for learning how to generate news summaries in a structured format.

The input news article is first tokenized and passed through the encoder, where it captures the relationships between words and phrases. The decoder then takes the encoded representation and generates the summary one token at a time, ensuring each token fits grammatically and contextually with the previous tokens.

BART excels at handling longer sequences and generating human-like text, it can handle up to 1024 tokens. It was particularly effective at producing coherent and concise summaries of news articles, even for those with complex sentence structures.

T5 (Text-to-Text Transfer Transformer):

T5 is a model designed with a text-to-text framework in mind, meaning that it treats every NLP task, including summarization, as a text input-output problem. This unified approach allows T5 to be applied to a variety of NLP tasks with minimal structural changes.

Like BART, T5 was also fine-tuned on the **MultiNews dataset**. During fine-tuning, the model learns the mapping from a long news article to its corresponding abstractive summary.

T5 operates using a similar encoder-decoder architecture as BART. After encoding the input text, the decoder generates the summary in a sequence-to-sequence manner. However, T5 uses a slightly different approach in its token handling, which often results in more diverse and creative summaries compared to BART.

T5's architecture is highly adaptable and effective in generating abstractive summaries that are concise and clear. Its performance is particularly strong on news data, where both grammatical accuracy and coverage of important content are essential. However, it could handle only a maximum of 512 tokens.

2. **Extractive Summarization**

In extractive summarization, rather than generating new text, the model selects key sentences from the original article that encapsulate the most important information. For this task, we used **BERT**.

BERT (Bidirectional Encoder Representations from Transformers)

BERT is a bidirectional transformer that excels at capturing sentence-level relationships within a text. For extractive summarization, BERT is fine-tuned to identify and rank the most important sentences in a document based on their context and relevance.

BERT was fine-tuned on an extractive summarization dataset consisting of news articles paired with their extractive summaries. The model learned to classify sentences based on their importance to the overall meaning of the article.

The input article is tokenized and passed through BERT, which assigns an importance score to each sentence. The model then selects the top-ranked sentences to create the extractive summary. This approach ensures that the key information is retained, while irrelevant or redundant information is excluded.

BERT provides a highly accurate extractive summary by directly selecting relevant sentences from the original text. This ensures that the summary remains factual and aligned with the content of the article.

Implementation

Text Preprocessing

Text preprocessing involved several key steps to clean the text data effectively. Excess whitespace was removed to standardize the input and eliminate any unnecessary gaps, enhancing readability. HTML tags present in the articles were stripped away to ensure that the text was free from formatting issues. Additionally, punctuation marks were eliminated to focus on the content of the text without distractions. These preprocessing steps contributed to creating a clean dataset, which is crucial for accurate and efficient summarization.

Tokenization

The text was divided into individual tokens (words or phrases) to facilitate processing by the models. This step is essential for both abstractive and extractive summarization techniques. During training of the model, the entire dataset was tokenized after which the tokenized dataset was divided into training and test splits.

Chunking Mechanism

As the models have a limited token length it can process during a given instance, that is 512 tokens for the T5 model and 1024 for Bart model. The chunking mechanism was designed to process long texts while adhering to the token limits of language models. It begins by tokenizing the input text and creates chunks of a specified maximum length, allowing for a defined number of overlapping tokens between consecutive chunks. This overlap helps retain contextual information across the chunks. The mechanism ensures that each chunk is manageable for processing, thus preventing errors related to token limits. Additionally, the chunks are paired with their corresponding summaries, making it suitable for tasks like supervised learning. Overall, this approach allows for effective handling of lengthy articles while maintaining the integrity of the content.

Summary Generation

After chunking the text, each chunk was processed individually to generate its corresponding summary. This chunk-by-chunk approach allows for the relevant information in each section to be distilled effectively. Once all the chunks have been summarized, these individual summaries are combined to form a comprehensive final summary of the entire text. This method ensures

that the overall context and key points from the original content are preserved, resulting in a coherent and informative summary that captures the essence of the lengthy article.

Post Processing

Post-processing involved refining the generated summaries to enhance their quality and coherence. This step included removing any iterative sentences that may have appeared in the summaries due to the overlapping tokens used during the chunking process. While the overlap was essential for maintaining contextual integrity, it sometimes resulted in repetitive content in the final summary. By eliminating these redundant sentences, the post-processing phase aimed to produce a more concise and coherent summary, ensuring that the output is clear, engaging, and free from unnecessary repetition.

Fine-Tuning Process

In this project, we employed **Parameter-Efficient Fine-Tuning (PEFT)** using **LoRA (Low-Rank Adaptation)** to adapt our models for the summarization tasks. LoRA allows for fine-tuning large language models efficiently by introducing low-rank matrices into the architecture. This method significantly reduces the number of trainable parameters, leading to faster training times and lower memory requirements while maintaining or even enhancing model performance.

1. Dataset Preparation

The first step involved the careful preparation of datasets to train the models. The datasets used to finetune the model was cleaned by removing duplicates, irrelevant content, and any inconsistencies. Articles with an excessive number of tokens were removed from the dataset. This decision was made to reduce computational time and improve processing efficiency. These steps are crucial to ensure that the models are trained on high-quality, unique data, which improves the performance of the summarization algorithms. The **MultiNews dataset** for the abstractive summarization task, which comprises long news articles paired with their corresponding abstractive summaries. This dataset is particularly valuable for training as it contains diverse topics and well-structured summaries. For the **extractive summarization task**, I utilized an external dataset containing articles and their extractive summaries, ensuring a robust training process for the BERT model.

2. Model Preparation

The previously mentioned models and their respective tokenizers were intialized to process text as mentioned in the section previously.

3. LoRA Implementation

Low-Rank Adaptation (LoRA) was integrated into the model architecture by introducing low-rank matrices into the weight matrices of the transformer layers. This mechanism allows for fine-tuning the models with significantly fewer parameters. For example: the Bart model had a total of over 407 million parameters but only 786k parameters were trained. By keeping the original model weights frozen and only training the LoRA parameters, we leverage the pre-trained knowledge while adapting the model to our specific task. This technique effectively balances computational efficiency and model performance. The introduction of LoRA layers helps mitigate overfitting risks, especially with datasets that may not have extensive samples for every possible scenario.

4. Hyperparameter Tuning

The training process involved rigorous hyperparameter tuning, focusing on key parameters such as:

- **Learning Rate:** Adjusted to ensure effective convergence without overshooting the optimal parameters
- **Batch Size:** Set to balance between memory constraints and training stability, affecting the model's ability to generalize from the training data. This is represented by per_batch_train_size.
- **Number of Training Epochs:** Determined based on the model's performance on a validation set to avoid overfitting while ensuring the model adequately learns from the data .
    - Output_dir: output directory saves the checkpoints based on the number of save steps allowing to train the model effectively by loading it gradually as the computational capacities were limited.

5. Training Process

The models underwent a structured training process using the prepared datasets, where each model was fine-tuned iteratively. During training, various checkpoints were created to save model states and facilitate easy restoration in case of issues.

Regular evaluations on a separate validation set were conducted to monitor model performance, employing metrics like ROUGE scores to assess the quality of generated summaries. This iterative feedback loop enabled timely adjustments to training strategies as necessary

6.Performance evaluation

After fine-tuning, comprehensive performance evaluations were conducted. Standard metrics were employed such as:

- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): This metric assesses the overlap between the generated summaries and the reference summaries, providing scores for recall, precision, and F1 metrics.
- Qualitative Assessments: Manual evaluations were also performed to examine the coherence and relevance of the summaries in relation to the original articles.

By leveraging LoRA within the PEFT framework, we achieved efficient fine-tuning of our summarization models, resulting in high-quality outputs that are suitable for real-time processing of news articles. The combination of these methodologies allowed us to enhance both the speed and accuracy of summarization, addressing the challenges of large-scale text processing effectively.

User Inference

The user inputs the text they wish to summarize through an interactive interface. Additionally, users can specify the desired length of the summary and the type of summary they would like extractive or abstractive, allowing for tailored outputs based on their preferences. The system processes the input according to the implemented backend logic as mentioned previously in the implementation. Once the processing is complete, the generated summaries and analysis results are presented back to the user, ensuring a seamless and customizable experience.

Topic Modeling

The End-to-End Text Summarization and Analysis System includes several components such as sentiment analysis, keyword extraction, and topic modeling. The focus of this section is on the topic modeling component, which utilizes the Latent Dirichlet Allocation (LDA) algorithm to uncover hidden themes in the dataset. Topic modeling enables the extraction of meaningful topics from a large volume of text, making it easier to categorize and summarize the dataset.

LDA is a widely used probabilistic model that identifies clusters of words in documents to represent topics. These topics help provide a high-level summary of the dataset, allowing for an efficient understanding of the data.

Implementation

- Data Preprocessing

- Preprocessing is a crucial step in preparing the dataset for topic modeling. Given that the dataset consists of over 1 million rows of text data, the following preprocessing steps were applied:
- Tokenization: Breaking the text into individual words.
- Stopword Removal: Removing common words (e.g., “the,” “is,” “in”) that do not contribute meaningfully to the topics.
- Lemmatization: Reducing words to their base forms to capture the root meaning.
- Bigrams and Trigrams: Constructing multi-word expressions (e.g., "machine learning") to preserve the context of words.
- Filtering: Removing words that were either too frequent or too rare, ensuring the model focuses on significant terms.

- LDA Model Building

The Latent Dirichlet Allocation (LDA) algorithm, implemented using the Gensim library, was employed to perform topic modeling. The following steps outline the model development process:

- Corpus Creation: The preprocessed text was converted into a Bag of Words (BoW) representation, where each word in the dataset was mapped to a unique ID using a dictionary.
- Hyperparameter Tuning:
- Number of Topics: After experimenting with different values, 10 topics were found to be optimal for this dataset.
- Alpha and Beta Parameters: These were adjusted to balance the diversity and coherence of the topics.
- Iterations and Passes: Multiple passes (set to 20) were used to ensure a thorough analysis of the data and better topic representation.
- Model Training:
- The LDA model analyzed patterns in word co-occurrences and assigned each word a probability distribution across topics.
- Each topic was then represented by a distribution of words, where the most probable words define the theme of the topic.
- Topic Extraction:
- The top words from each topic were extracted. Topics ranged from finance to technology and health, based on the most frequent words associated with each theme.

5. Topic Visualization

- To aid in interpreting the topics, word clouds were generated to visualize the most significant words in each topic. In a word cloud, larger words represent higher frequency within the topic, providing a quick visual summary of the key terms.
- The word clouds were generated for the top 10 words in each topic, providing an effective way to compare topics and understand their differences.

Keyword Extraction

In the ever-growing world of text data, extracting relevant keywords is crucial for efficient information retrieval and data analysis. Keyword extraction helps in identifying the most important terms in a document that represent the core idea of the text. This project focuses on using the Term Frequency-Inverse Document Frequency (TF-IDF) method, a widely used approach for measuring the importance of words in a document relative to a corpus. This project utilizes natural language processing (NLP) techniques to preprocess text and extract top keywords from paragraphs.

Implementation

1. Preprocessing

Before applying the TF-IDF method, the text undergoes several preprocessing steps to clean and normalize the data. This ensures that the extracted keywords are meaningful and relevant to the content. The following steps were performed,

- Lowercasing: All text is converted to lowercase to ensure uniformity.
- HTML Tag Removal: HTML tags are stripped from the text using BeautifulSoup.
- Special Character and Number Removal: Only alphabetic characters are retained, and any numbers or special characters are removed.
- Stopword Removal: Common English stopwords are removed to focus on more meaningful words.
- Tokenization: The text is tokenized into individual words to break it down for further analysis.
- Verb Filtering: Verbs are filtered out using POS tagging to prioritize nouns and adjectives, making the keywords more meaningful.

2. TF-IDF Model

- The TF-IDF method was implemented using the following steps:
- Text Vectorization: Once the text is cleaned, a CountVectorizer is used to generate word frequency counts.
- TF-IDF Calculation: The word count vector is then transformed into a TF-IDF matrix using TfidfTransformer, which computes the importance of each word relative to the document.

- Top Keywords Extraction: A helper function, extract_topn_from_vector(), retrieves the top N keywords based on these scores.
- Sorting and Ranking: The words are ranked based on their TF-IDF scores, and the top keywords are extracted from the paragraph.

Sentiment Analysis

Sentiment analysis is a popular task in Natural Language Processing (NLP) that involves determining the sentiment expressed in a piece of text by understanding the mood or the mentality that is expressed in the text.

Sentiment analysis tasks range from the humble binary classification (positive/negative) to more nuanced multi-class classification (e.g., very positive, positive, neutral, negative, very negative). Sentiment analysis uses a range of NLP algorithms such as Rule-based, Automatic and Hybrid.

Several categories and types of sentiment analysis are available for any given sentiment analytics,

- Fine Grained Sentiment Analysis
- Emotion Detection
- Aspect based
- Multilingual

In our project, Since the primary focus is in fact news articles in the English language, the primary focus of our research was on leveraging Fine grained and emotion detection methods.various techniques and models to analyse sentiments effectively.

Several steps of preprocessing and stripping down of tokens were used to provide accurate and effective sentiment predictions and context window challenges encountered throughout the project.

Multiple approaches of sentiment analysis were experimented to assess their effectiveness,

- Rules based – VADER
- Machine Learning - Random Forest
- Neural Network Transformer based approaches (BERT, small Llama)

VADER

VADER is a rule-based sentiment analysis tool, it’s a lexicon based model where it has a lexicon of words , phrases and their associated sentiment scores which are commonly used and it's specifically used for sentiments expressed in social media. It is known for its simplicity and effectiveness in handling both polarity (positive/negative/neutral) and intensity (strength - sentiment scores) of sentiments.

VADER is computationally efficient and well suited for real time applications because of it’s rule based and Lexicon based approach.

Machine Learning

Random operates by building multiple decision trees during training and outputting the mode of the classes (classification) of the individual trees.Random Forest is an ensemble method, meaning it combines the predictions of multiple decision trees to improve robustness and accuracy.

By averaging multiple decision trees, Random Forest reduces the risk of overfitting, which is a common issue in sentiment analysis due to the complexity and variability of natural language.

Text data, when converted into numerical features (TF-IDF), often results in high-dimensional spaces. Random Forest can effectively handle such high-dimensional data and a proper and complete understanding of features importance can be evaluated as well to identify which word or phrase are most influential for a sentiment prediction.

BERT and Small LLaMA Neural Network-Based Sentiment Analysis Methods

BERT is a transformer-based model developed by Google. That has significantly advanced the field of NLP by enabling deep contextual understanding of text. BERT is pre-trained on a large corpus of text and can be fine-tuned for specific tasks, including sentiment analysis.

An existing fine-tuned model of BERT was used to access the sentiment of the text for this application. Due to computational restriction and the training requirements of BERT an alternative was needed.

Small Llama was the perfect alternative in thatinstance, being a model that is much smaller than BERT it was much easier to fine tune using the “News_Sentiment_Analysis” dataset.

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model developed by Google that has significantly advanced the field of Natural Language Processing (NLP). BERT enables deep understanding of text by processing words in both directions (left-to-right and right-to-left) simultaneously. This approach allows BERT to capture the full context of a word based on its surrounding words, making it highly effective for various NLP tasks.

BERT is pre-trained on a large corpus of text, including the entirety of Wikipedia and a vast collection of books. This extensive pre-training allows BERT to understand a wide range of language patterns and nuances. After pre-training, BERT can be fine-tuned on specific datasets for particular tasks, such as sentiment analysis, named entity recognition, and question answering.

For this application, an existing fine-tuned model of BERT was used to assess the sentiment of the text. The fine-tuning process involves training the pre-trained BERT model on a labelled sentiment analysis dataset, enabling it to learn the specific patterns associated with positive, negative, and neutral sentiments.

However, due to the computational resources required for training and deploying BERT, an alternative solution needed to be fine-tuned to our application's custom needs.

Small LLaMA (Large Language Model) emerged as the perfect alternative in this instance. Small LLaMA is a more compact and efficient model compared to BERT, making it easier to fine-tune and deploy, especially with limited computational resources.

Even with its smaller size, Small LLaMA has the ability to comprehend complex language patterns and nuances, making it highly desirable for sentiment analysis tasks.

The Small LLaMA model was fine-tuned using the “News_Sentiment_Analysis” dataset, which contains a rich collection of news articles labelled with sentiment scores.

This fine-tuning process involved training the Small LLaMA model on this dataset to adapt it to the specific language and sentiment patterns found in news articles. As a result, the fine-tuned Small LLaMA model was able to provide accurate and efficient sentiment analysis for the application, overcoming the computational limitations associated with BERT.

The sentiments Anaysis was implemetated using the same technique of the text summarization, but the only difference was that the model was tiny llama and it was able to accept 2500 tokens.

Evaluation and Testing

Text Summarization

Performance Evaluation

Evaluating a text summarization model is a crucial part of the system development process to ensure that the generated summaries are accurate, concise, and informative. The evaluation is typically done through both automatic metrics and, where possible, human evaluation. Below is a detailed explanation of the methods used to evaluate and test the performance of the text summarization component of your system.

- Evaluation Metrics

ROUGE (Recall-Oriented Understudy for Gisting Evaluation): ROUGE is the most common evaluation metric used for summarization tasks. It works by comparing the overlap of n-grams (words or sequences of words) between the generated summary and the reference (human-written) summary. Several variants of ROUGE are used:

- ROUGE-1: Measures the overlap of unigrams (single words) between the generated summary and the reference summary.
- ROUGE-2: Measures the overlap of bigrams (two consecutive words) between the generated and reference summaries.
- ROUGE-L: Measures the longest common subsequence (LCS) between the generated and reference summaries. This is particularly useful as it evaluates how well the order of information in the summary matches the reference.

These metrics provide a measure of recall (how much of the reference summary is captured by the generated summary), precision (how much of the generated summary is relevant to the reference), and F1-score (the harmonic mean of recall and precision).

- Human Evaluation

  While automatic metrics are fast and useful, they may not always capture the full quality of a summary. For instance, a summary may have a low ROUGE score but still conveys the main points accurately. Human evaluation is often necessary for understanding the actual quality of the summary, specifically in terms of:

- **Fluency**: Is the summary grammatically correct and easy to read?
- **Coherence**: Does the summary make logical sense? Is it easy to follow?
- **Relevance**: Does the summary include all the important information from the source text without including unnecessary details?
- **Informativeness**: How well does the summary capture the essential content of the original text?

Both the models for abstractive summarization were evaluated based on their ROUGUE score mainly based Rougue1 scores, the T5 model had an overall rougue score of about 0.365 while Bart model had a score of over 0.6 this maybe be due to the fact that T5 model has more chunks than Bart since it has 512 tokens less than Bart and so context maybe slighty deviating as compared to when more tokens are available, Therefore the best model chosen for the task of abstractive summarization was Bart.
The same evaluations metrics were also used for the extractive summarization.

Furthermore, apart from the overall evaluation, during the model training cross validation also was used by splitting the dataset to train and test to facilitate this.

- Model Testing and Validation

After fine-tuning the summarization models, various techniques can be employed to test and validate the model:

- **Cross-Validation**: Perform k-fold cross-validation to split the dataset into training and validation sets, ensuring that the model is not overfitting and performs well on unseen data. This helps generalize the model's performance across different types of news articles.
- Validation Set: The model should be evaluated on a separate validation set during training. Monitoring the validation loss and ROUGE scores during fine-tuning helps in selecting the best model.

Challenges Faced and Solutions

- **Handling Long Documents**: Many news articles exceed the token limit of the models. This was addressed by implementing the chunking mechanism with overlapping windows to ensure content coverage while summarizing long articles
- **Computational Efficiency**: Fine-tuning large models such as BART and T5 can be resource-intensive. This was mitigated by using PEFT (Parameter-Efficient Fine-Tuning) and LoRA (Low-Rank Adaptation) techniques to reduce the number of trainable parameters and decrease memory usage. Other methods such as saving checkpoints and and traning models through halting the training until the resources are restored and continuing training from the point itwas stopped.
- **Maintaining Summary Coherence**: Overlapping chunks sometimes introduced repeated information, which was handled using post-processing techniques to remove redundancy and improve the overall flow of the summary

Topic Modeling

Performance Evaluation

The evaluation of the LDA model was performed using two key metrics: Perplexity and Coherence Score.

- Perplexity: Perplexity measures the model's predictive power. A lower perplexity score indicates better performance. In this case, the LDA model achieved a Perplexity score of -8.579, indicating a strong model fit to the data.
- Coherence Score: The coherence score evaluates the semantic similarity between words in a topic. A higher coherence score signifies more meaningful and interpretable topics. The final model achieved a Coherence Score of 0.644, reflecting the strong semantic structure of the generated topics.

1. Coherence Score

- The coherence score of 0.644 indicates that the words within each topic are semantically related, making the topics interpretable and meaningful. This score demonstrates the model's ability to capture coherent topics that accurately represent the data's underlying themes.

2. Perplexity Score

- With a Perplexity score of -8.579, the LDA model demonstrated strong predictive performance. While perplexity alone does not guarantee topic interpretability, it complements the coherence score, confirming that the model effectively captured the relationships between words and topics.

3. Qualitative Evaluation

- Qualitative analysis was performed by examining the top words for each topic. For example, a topic labeled as finance included words such as “growth,” “investment,” and “economy,” while a technology topic featured terms like “AI,” “innovation,” and “data.” The word clouds further enhanced the interpretability of the topics, providing a clear visual representation of the dominant words.

Challenges Faced and Solutions

- Computational Complexity: Processing a dataset of this size required significant computational resources, making it challenging to train the LDA model efficiently.
- Topic Overlap: Initially, there was some overlap between topics, which was reduced through hyperparameter tuning and refinement of the preprocessing steps.

- Interpretability: While LDA is effective at identifying distinct topics, some topics require manual interpretation to fully understand their meaning.

Keyword Extraction

Performance Evaluation

The system was tested on multiple paragraphs to ensure the accuracy of keyword extraction. Here are the steps used to evaluate the method

- Preprocessing Evaluation
  The preprocessing step was checked manually to ensure proper tokenization, removal of unwanted words, and successful elimination of verbs.

- Keyword Quality
  The extracted keywords were compared with manually selected keywords to check the relevance. Keywords such as nouns, important adjectives, and entities were expected to be part of the result.
- TF-IDF Effectiveness
  The ranking of words by TF-IDF score was reviewed to ensure that high-frequency but less important words did not dominate the top keywords. The use of TF-IDF successfully balanced the word frequency with inverse document importance.

Challenges Faced and Solutions

Several challenges were encountered during the implementation

- Verb Removal Accuracy
- Identifying and eliminating verbs without affecting meaningful words was challenging. Fine-tuning the POS tagging process was necessary to strike the right balance between removing irrelevant words and retaining important ones.
- Handling Short Words
- Many meaningful three-letter words (such as "car," "man," "law") were initially removed during preprocessing. To address this, a predefined list of useful short words was incorporated.
- Scalability

- The current implementation handles single documents at a time. Scaling it for larger datasets or corpora requires extending the vectorization approach to a collection of documents, which could demand higher computational resources.

Sentiment Analysis

Performance Evaluation

After implementing and fine-tuning various sentiment analysis models, comprehensive and detailed performance evaluations were conducted. Standard metrics were used to assess the effectiveness of VADER, Random Forest classification, pre-tuned BERT, and fine-tuned small LLaMA models.

VADER (Valence Aware Dictionary and Sentiment Reasoner)

Performance Metrics:

Accuracy: VADER provides quick and reasonably accurate sentiment analysis for short, informal texts.

Precision, Recall, F1-Score: These metrics were calculated to evaluate the balance between precision and recall.

Assessment: VADER's simplicity and efficiency make it suitable for real-time applications, VADER also provides highly confident scores on assessment but it struggles heavily with more complex or nuanced texts.

Random Forest Classification

Performance Metrics:

Accuracy: Random Forest achieved average accuracy by leveraging multiple decision trees.

Precision, Recall, F1-Score: These metrics were used to assess the model's performance in classifying sentiments.

Qualitative Assessment: Random Forest handles high-dimensional data well and provides insights into feature importance, however the accuracy remains average and also struggles with nuanced language and proper sentiment scoring to evaluate against other models.

Pre-Tuned BERT (RoBERTa)

Performance Metrics:

Accuracy: BERT achieved best performance with high accuracy however it was much less confident with the sentiment scoring compared to VADER showing an understanding for nuanced language .

Precision, Recall, F1-Score, Sentiment Scoring: These metrics were used to evaluate BERT's ability to capture nuanced sentiments.

Qualitative Assessment: RoBERTa has deep contextual understanding that allows it to handle complex texts effectively, but it requires significant computational resources for both training and inference.

Fine-Tuned Small LLaMA Model

The small LLaMA model is a compact and efficient neural network fine-tuned on the "News_Sentiment_Analysis" dataset.

Performance Metrics:

Accuracy: The fine-tuned small LLaMA model has decent competitive accuracy lower than RoBERTa but is much better at evaluating natural language than other models using lower computational resources.

Qualitative Assessment: The small LLaMA model provided a good balance between performance and computational efficiency, making it the most suitable for limited resource systems.

Comparative Analysis

RoBERTa and the fine-tuned small LLaMA model achieved the highest accuracy, followed by Random Forest and VADER.

Computational Efficiency- VADER and the small LLaMA model were the most computationally efficient, while BERT required the most resources.

Contextual Understanding - BERT excelled in understanding complex and nuanced texts followed by the fine tuned small Llama model, whereas VADER was more suited for simpler

Texts and language.

Challenges Faced and Solutions

- Computational Inefficiencies – Use of

Conclusion and Future work

Conclusion

The project successfully integrated multiple natural language processing (NLP) components, including text summarization, keyword extraction, topic modeling, and sentiment analysis, to create a robust text analysis system. The text summarization models, leveraging both abstractive (BART and T5) and extractive (BERT) approaches, provided concise yet informative summaries, with options for users to customize the length of the summaries. The fine-tuning of these models, utilizing PEFT techniques such as LoRA, ensured computational efficiency while maintaining high-quality results. Additionally, the chunking mechanism and post-processing steps, including redundancy removal, enhanced the coherence of the summaries, particularly for long-form text.

The keyword extraction method, implemented using TF-IDF along with preprocessing and POS tagging, effectively identified key terms that represented the core themes of the input documents. This approach provided a scalable solution for efficiently extracting high-quality keywords across diverse text types.

Topic modeling, using the LDA algorithm, was instrumental in summarizing and categorizing large text datasets into meaningful themes. With a Coherence Score of 0.644 and a Perplexity Score of -8.579, the model demonstrated strong interpretability and coherence in topic discovery. The use of word clouds provided an additional layer of insight, allowing users to visually understand the key terms within each topic.

Sentiment analysis, implemented using models like Tiny Llama and VADER, added another layer of understanding by analyzing the emotional tone of the summarized content. This component, combined with the other features, created a holistic system for analyzing and gaining insights from large textual data.

In conclusion, the system offers a powerful and scalable solution for text summarization, keyword extraction, topic modeling, and sentiment analysis, making it suitable for applications that require processing and understanding of vast amounts of textual data. While the results were satisfactory, future improvements could involve further fine-tuning of the models, optimization of the topic modeling process, and extension of the system's capabilities to handle a wider variety of datasets and NLP tasks.

Limitations of current system

- **Lack of Multilingual Support**: The system is currently limited to processing text in English only. This restricts its usability in non-English-speaking regions or for datasets in different languages. Expanding to multilingual capabilities requires additional model fine-tuning and testing on diverse datasets.
- **Limited Scalability**: Since the system is built using Streamlit and does not utilize cloud-based solutions or distributed processing, it may struggle to handle large datasets or real-time, high-traffic environments. This restricts its usage in scenarios that demand heavy computation or processing large volumes of data simultaneously.
- **Basic User Interface**: Although the system provides interactive features, the user interface is relatively simple, lacking more advanced customization options, such as detailed input configurations for model parameters, fine-tuning, or advanced summary options beyond length control.
- **Summarization Quality for Complex Texts**: While the models are fine-tuned for news articles, the summarization quality may degrade for more complex or highly domain-specific texts. Abstractive models, in particular, may generate less accurate summaries when the input text contains intricate technical details.
- **Performance on Long Texts**: Although chunking mechanisms have been implemented to process long articles, this process can still be inefficient, especially for texts that exceed token length limits. The overlapping chunks may also lead to repetitive or disjointed summaries, impacting the coherence of the final output.
- **Sentiment Analysis Granularity**: The current sentiment analysis implementation provides a basic positive, negative, and neutral sentiment output. This limits its ability to capture nuanced emotions or mixed sentiments within longer or more complex texts.
- **Keyword Extraction and Topic Modeling for Specialized Domains**: The current keyword extraction and topic modeling algorithms (TF-IDF and LDA) work well for general-purpose tasks but may not perform optimally for specialized fields like law, medicine, or finance, where the language and keyword importance can be significantly different.
- **Lack of Persistent Data Storage**: The system does not store user inputs, results, or processed data. While this enhances simplicity and privacy, it also limits its ability to analyze historical trends or provide long-term insights based on previously processed data.

Future Work

For the future scope of the overall system, several improvements can be implemented to enhance its functionality. Multilingual support is a key area of expansion, allowing the system to handle and process text in multiple languages, which would greatly widen its application. Incorporating real-time processing capabilities would enable faster and more dynamic text analysis, making the system more responsive to user inputs. Additionally, deploying the system on a cloud-based platform would improve scalability, allowing it to manage larger datasets and handle more complex tasks efficiently.

The text summarization component can benefit from integrating more advanced models like **transformer-based models** such as GPT-4. These models can generate more coherent and contextually accurate summaries. Additionally, implementing **multilingual summarization** will allow the system to handle non-English articles effectively. Exploring **summary customization options** like abstraction levels, or focus points (e.g., focusing on specific sections of the text) will offer users more control over the summarization process.

In sentiment analysis, using more **sophisticated models** and expanding the sentiment scale (beyond just positive, negative, or neutral) can give deeper emotional insights. Real-time sentiment analysis would allow the system to dynamically analyze content as it's being updated. Further fine-tuning models on **domain-specific datasets** could also enhance the sentiment detection accuracy in specialized fields such as finance, healthcare, or entertainment.

For topic modeling, the LDA model can be further enhanced by experimenting with **neural topic models** like BERTopic, which provide better context understanding and improved topic coherence. The introduction of **dynamic topic modeling** could allow the system to track how topics evolve over time in real-world applications. Expanding the model to handle **cross-domain or more complex datasets** will improve its applicability to a wider range of industries.

The keyword extraction component can be enhanced by incorporating **contextual keyword extraction techniques** that go beyond simple statistical models like TF-IDF. Algorithms that leverage the context within the document, such as **BERT-based keyword extraction models**, can significantly improve the relevance of extracted keywords. Furthermore, allowing users to specify **custom keyword preferences** based on domain-specific needs can make the system more flexible.

References

1.https://www.kaggle.com/datasets/clovisdalmolinvieira/news-sentiment-analysis/data

2.Python Sentiment Analysis Project with NLTK and 🤗 Transformers, https://www.youtube.com/watch?v=QpzMWQvxXWk

3.https://www.geeksforgeeks.org/what-is-sentiment-analysis/

4. https://mlarchive.com/machine-learning/sentiment-analysis-with-random-forest/

5.Fine-tuning Tiny LLM on Your Data | Sentiment Analysis with TinyLlama, https://www.youtube.com/watch?v=_KPEoCSKHcU