Project Report

On

Summerization and Sentiment Analysis on Hindi and English Language

Submitted
In partial fulfilment
For the award of the Degree of

PG-Diploma in Artificial Intelligence

(C-DAC, ACTS (Pune))

Guided By:

Mrs. Swapna Yenishetti

Submitted By:

Vishal Jagadale (230940128014)

Pooja Patil (230940128017)

Suhas Kolekar (230940128030)

Madan Vaka (230940128034)

Pooja Pawar (230940128018)

Centre for Development of Advance Computing

(C-DAC), ACTS (Pune-411008)

Abstract

The project entails the development of an innovative application that employs advanced machine learning models to perform text summarization and sentiment analysis in multiple local languages, including Hindi and English. Utilizing state-of-the-art transformer-based models such as GPT2 and BART, the application is designed to generate concise and coherent summaries of extended texts, effectively capturing the essence of the original content. These models are recognized for their exceptional performance in understanding the semantic nuances of language, making them ideal for the task of summarization. For sentiment analysis, the application leverages the FLAIR framework, which has been instrumental in achieving high accuracy rates in language understanding tasks. FLAIR library is employed to classify the sentiment of the text as positive, negative, or neutral, providing users with a clear indication of the emotional tone behind the words. The integration of GPT2 and BART for summarization and FLAIR for sentiment analysis ensures that the application delivers high-quality results in both aspects of text analysis. By supporting multiple languages, the application caters to a diverse user base, enabling them to gain insights from text data in their preferred local dialects. This project represents a significant stride forward in the field of natural language processing, demonstrating the potential of AI-driven solutions to simplify complex text analysis tasks. The application's ability to process and interpret text in various languages positions it as a valuable tool for businesses, researchers, and individuals seeking to extract meaningful information from large volumes of textual data.

Chapter 1
Introduction

Our project is focused on developing an application that leverages advanced machine learning models to perform text summarization and sentiment analysis in multiple local languages, namely Hindi and English. The application uses state-of-the-art transformer-based models such as GPT2 and BART for summarization, which are known for their exceptional performance in understanding the semantic nuances of language. This makes them ideal for generating concise and coherent summaries of extended texts, effectively capturing the essence of the original content.

For sentiment analysis, the application employs the FLAIR library, which has been instrumental in achieving high accuracy rates in language understanding tasks. FLAIR library is used to classify the sentiment of the text as positive, negative or neutral, providing users with a clear indication of the emotional tone behind the words. The integration of these models ensures that the application delivers high-quality results in both aspects of text analysis.

By supporting multiple languages, the application caters to a diverse user base, enabling them to gain insights from text data in their preferred local dialects. This project represents a significant stride forward in the field of natural language processing, demonstrating the potential of AI-driven solutions to simplify complex text analysis tasks. The application's ability to process and interpret text in various languages positions it as a valuable tool for businesses, researchers, and individuals seeking to extract meaningful information from large volumes of textual data.

Chapter 2

Literature Survey

Title of paper	Authors	Year	Description
Abstractive Text Summarization for Resumes With Cutting Edge NLP Transformers and LSTM	Öykü Berfin Mercan, Sena Nur Cavsak, Aysu Deliahmetoglu (Intern), Senem Tanberk	2023	This paper focuses on the application of text summarization techniques in the context of resume classification. The research evaluates the performance of various models, including LSTM, pre-trained models like T5, Pegasus, BART, and BART-Large, on a dataset comprising 75 resumes. The dataset includes various sections such as language, education, experience, personal information, and skills. The primary goal of the study is to classify resume text effectively. The results indicate that the BART-Large model, when fine-tuned with the resume dataset, delivers the best performance among the tested models.
Automatic Text Summarization of COVID-19 Medical Research Articles using BERT and GPT-2	Virapat Kieuvongngam, Bowen Tan, Yiming Niu	2020	The paper uses BERT and GPT-2 to summarize COVID-19 research articles from the COVID-19 Open Research Dataset Challenge. It evaluates the summaries with ROUGE scores and visual inspection, aiming to provide abstractive summaries for articles without abstracts. This helps the medical community stay current with the growing research on the pandemic.
Give your Text Representation Models some Love: the Case for Basque	Rodrigo Agerri, naki San Vicente, Jon Ander Campos, Ander Barrena, Xabier Saralegi, Aitor Soroa, Eneko Agirre	2020	The paper addresses the challenges of using pre-trained language models for smaller languages like Basque, which often suffer from subpar performance due to training on limited data. It compares monolingual models, such as FastText, FLAIR, and BERT, trained on larger Basque corpora, showing they excel in tasks like topic classification, sentiment analysis, and named entity recognition. These models establish new benchmarks for Basque language processing, with all resources made public for further study and use.

Title of paper	**Authors**	**Year**	**Description**
BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension	Matthew E. Peters, Neil Shyam, Michael Q. Wang, Luke Zettlemoyer	2019	The paper introduces BART (Bidirectional and Auto-Regressive Transformers), a denoising autoencoder for pre-training sequence-to-sequence models. It presents a novel approach to pre-training that involves learning to reconstruct the original input sequence from a corrupted version, which is different from traditional methods that involve masking or predicting missing tokens. The BART model has been shown to achieve state-of-the-art results on various NLP tasks, including translation, summarization, and question answering, and has influenced the development of other models like T5 (Text-to-Text Transfer Transformer).
Attention Is All You Need	Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin	2017	The paper introduces the Transformer model, a novel architecture for natural language processing tasks that relies on self-attention mechanisms. It presents a new approach to sequence-to-sequence learning that does not require recurrent or convolutional layers, instead using positional encodings to understand the order of the input sequence. The Transformer model has significantly impacted the field of NLP, leading to advancements in machine translation, text summarization, and other language-related tasks.

Chapter 3
Goals and objectives

- Develop a comprehensive tool for text summarization and sentiment analysis in multiple local languages.
- Enhance language understanding through advanced NLP techniques to accurately summarize and analyze text.
- Address real-world challenges by providing quick and accurate insights from large volumes of textual data.
- Contribute to the field of NLP by advancing the state of the art in text summarization and sentiment analysis.

Chapter 4
Methodology

Solution Requirement

For our project on text summarization and sentiment analysis of Hindi and English languages, the solution requirement is to develop a tool that can effectively summarize and analyze the sentiment of text in both languages. The tool must be capable of understanding the nuances of each language, including the grammatical structures, vocabulary, and cultural contexts. Additionally, the tool must be able to handle a wide variety of text types, from formal documents to social media posts, and accurately represent the original meaning in the summarized output. The tool should be user-friendly, requiring minimal technical expertise to operate. It should be scalable to accommodate increasing volumes of text and expandable to support additional languages in the future.

Solution Constraints

We analyzed the solution in terms Computational Limitations, Language-Specific Challenges, Scalability, User Interface Design, Model Interpretability and Cost Management.

Fine Tunning

We fine-tuned facebook/bart-large-cnn model for the summarization of English text, and GPT-2 for summarization of Hindi language text on newspaper data consisting text and summery to increase model's accuracy on local languages.

Evaluation

We begin by fine-tuning our model on a news dataset, which is chosen for its relevance to the types of text we aim to summarize and analyze. This dataset provides a diverse range of topics and writing styles, making it suitable for training a model that can handle various types of news articles.
We establish a set of performance metrics to evaluate the model's performance. For text summarization, we use ROUGE scores.

Outcome

A text summarization model takes a longer piece of text as input and produces a shorter version that retains the key points and overall meaning of the original content.
The output is essentially a condensed representation of the input text.
A sentiment analysis model evaluates the emotional tone behind words to determine whether the writer's attitude towards a particular topic or product, etc., is positive, negative.

Chapter 5

Text Summarization Applications

- News Aggregation: Summarizing lengthy news articles to provide readers with a quick overview of the main events or points.
- Customer Reviews: Condensing long product reviews into a few sentences that highlight the key aspects of the customer's experience.
- Social Media Monitoring: Creating brief summaries of posts or comments to track trends, sentiments, and discussions in real-time.
- Content Curation: Summarizing blog posts, research papers, or other long-form content to make them more digestible for busy professionals or students.
- Automated Reporting: Generating executive summaries or reports from extensive documents for stakeholders who prefer concise information.

Sentiment Analysis Applications

- Market Research: Analyzing social media mentions, customer feedback, and online reviews to gauge public opinion on products, brands, or services.
- Customer Service: Determining the sentiment behind customer complaints or support requests to prioritize responses and improve service quality.
- Product Development: Understanding consumer feedback to inform design changes and feature enhancements.
- Financial Analysis: Assessing the sentiment in financial news and social media chatter to predict market trends and investor sentiment.
- Healthcare: Monitoring patient feedback to identify areas for improvement in healthcare services and patient satisfaction.
- Politics: Gauging public sentiment on political issues through social media and news articles to inform campaign strategies.

These applications demonstrate the value of text summarization and sentiment analysis in extracting meaningful insights from large volumes of unstructured text data. By automating these processes, organizations can gain actionable intelligence quickly and efficiently.

Chapter 6

Hardware Resources Required

In this project, a computer with sufficient processing power is needed. This project requires too much processing power, due to the image and video batch processing.

- Client-side Requirements: Browser: Any Compatible browser device

Sr. No.	Parameter	Minimum Requirement
1	Intel Xeon E5 2637	3.5 GHz
2	RAM	32 GB
3	Hard Disk	256 GB
4	GPU	32 GB
5	Graphic card	NVIDIA GeForce RTX 3090 (24 GB VRAM)

Table: Hardware Requirements

Software Resources Required

1. Operating System: Ubuntu
2. Programming Language : python3
3. Framework: PyTorch
4. Libraries : transformers, langdetect, flair, datasets

Chapter 7
Detailed Design Document

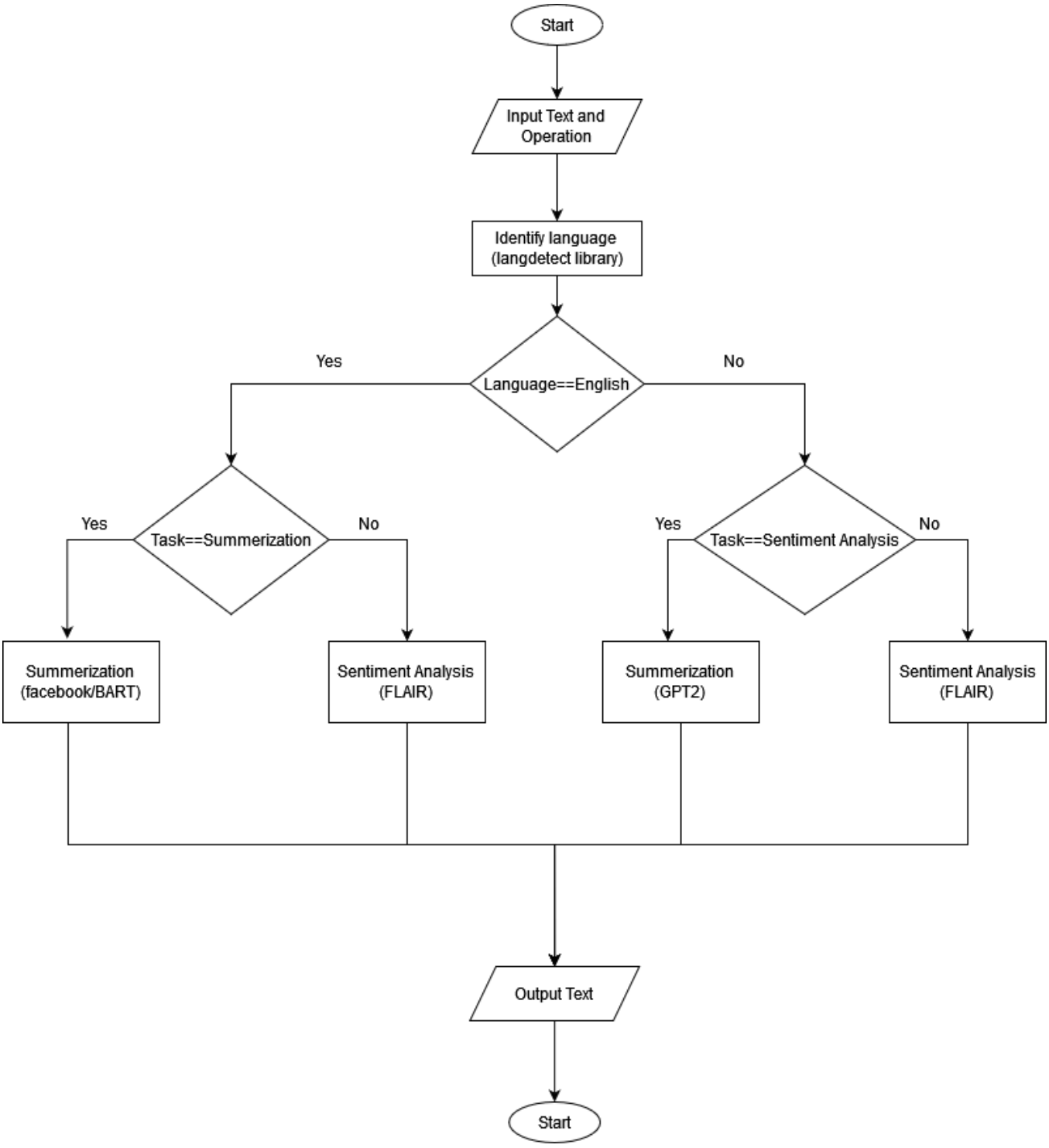


Figure: System Architecture

The flowchart describes a process that starts with collecting user input, which includes text and the desired operation (summarization or sentiment analysis). It then utilizes the **langdetect** library to determine the language of the text. Based on the identified language and the selected operation, the system chooses the appropriate model to execute the task: **facebook/bart-large-cnn** for English summarization, **gpt2-large** for Hindi summarization, and flair library for sentiment analysis. Finally, the system presents the output of the operation on the screen for the user to review.

Chapter 8
Transformers

Transformers in AI, particularly in the field of natural language processing, are a groundbreaking model architecture that has significantly advanced the capabilities of language models. These models are characterized by their self-attention mechanisms, which allow them to consider the entire context of a sentence when making predictions, as opposed to sequential processing methods used by earlier models. The ability of transformers to process all words in a sentence in parallel, rather than one at a time, has led to a substantial increase in both training speed and model performance. With their scalability and state-of-the-art results on numerous benchmarks, transformers have become the backbone of many modern NLP applications, from machine translation and text summarization to sentiment analysis and beyond. Their versatility and the potential for fine-tuning and transfer learning have also made them a popular choice for a wide range of tasks, despite the challenges posed by their complexity and the computational resources required for training and deployment.

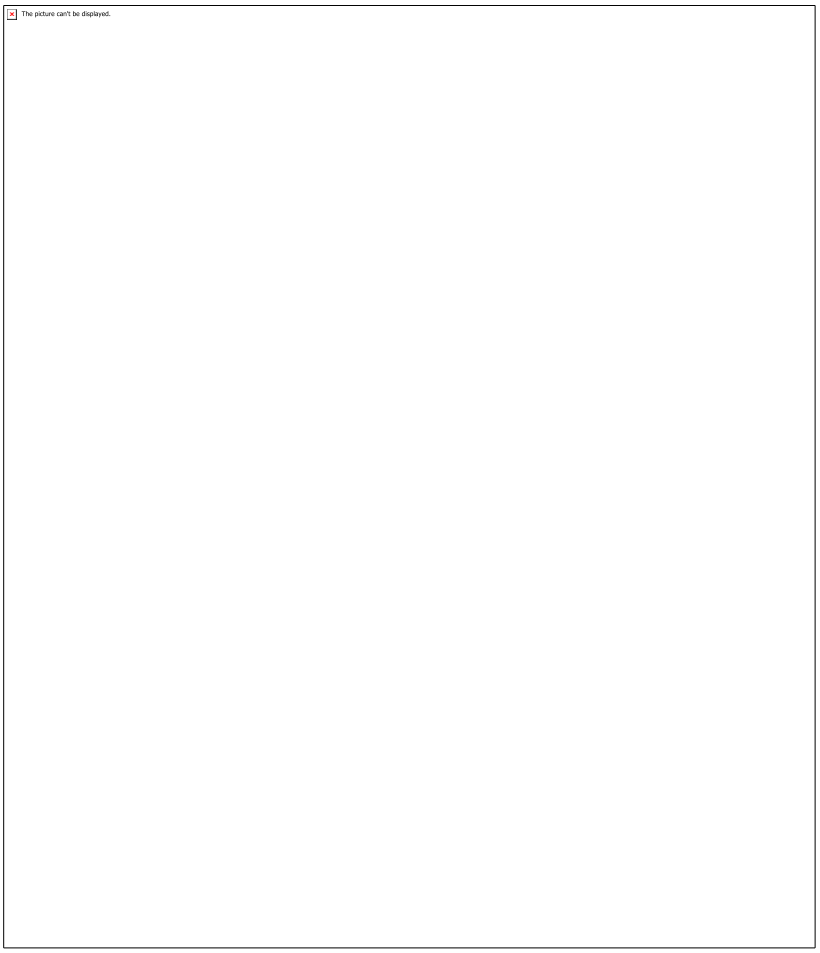


Figure: Transformer Architecture

Large Language Model

A Large Language Model (LLM) is a type of artificial intelligence model that is designed to understand and generate human-like text. These models are trained on vast amounts of text data, which allows them to learn the structure, grammar, and context of the language. They can be used to answer questions, write essays, translate languages, and even compose music or artwork. LLMs are a key component of many natural language processing (NLP) applications and are continually improving as researchers and developers push the boundaries of what is possible with machine learning.

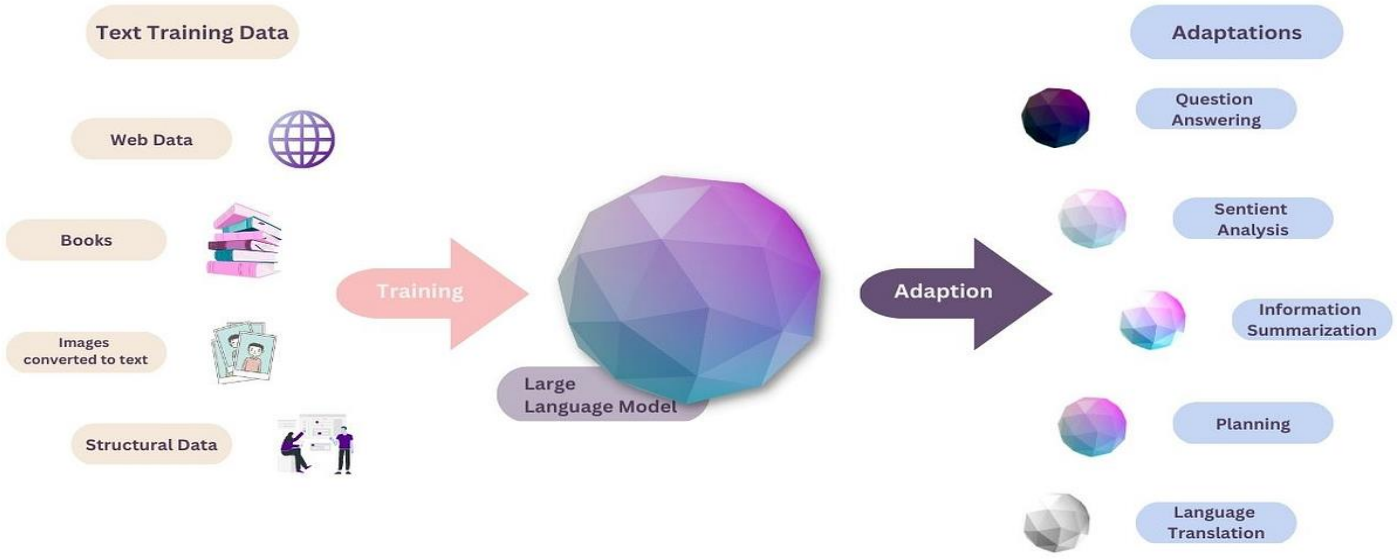


Figure: Large Language Model

Abstractive summarization

Abstractive summarization is a natural language processing task that involves generating a concise summary of a given text, often by rephrasing and condensing the original content. Unlike extractive summarization, which simply extracts key sentences or phrases from the source material, abstractive summarization creates new sentences that may not appear verbatim in the original text. This process typically employs advanced machine learning models, such as sequence-to-sequence models with attention mechanisms, which can understand the context and semantics of the text to produce coherent and relevant summaries. Abstractive summarization is particularly useful in scenarios where the goal is to provide a high-level overview or a summary that captures the essence of the information in a more creative and flexible manner than traditional extractive methods.

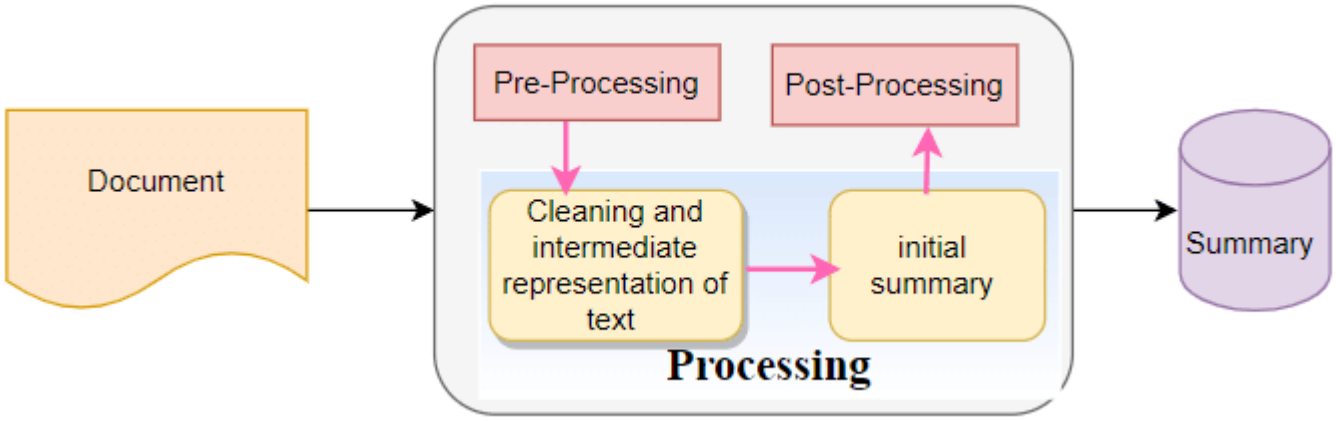


Figure: Abstractive Summerization System Architecture

Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a subfield of natural language processing and computational linguistics that aims to identify and extract subjective information from source materials. The objective is to determine the sentiment expressed in a piece of text, which can be positive, negative, or neutral. Sentiment analysis is widely used in various applications, such as social media monitoring, customer feedback analysis, and market research, to gauge public opinion, customer satisfaction, and brand reputation. It involves the use of machine learning algorithms and natural language processing techniques to classify text into sentiment categories. These algorithms can be trained on large datasets and can recognize subtle nuances in language that may indicate sentiment, such as the use of certain words or phrases, tone, and context. The insights gained from sentiment analysis can be valuable for businesses and organizations looking to understand their audience's feelings and perceptions, which can inform decision-making and strategy.

Figure: Sentiment analysis

Chapter 9
Architectural Design

Module 1: Selecting Pre-trained models

Model Elimination for Text Summarization:

Model Name	Reason for Elimination
mBART	Issues with missing vocab.json and merge.json files post-training.
mahaBERT	Licensing restrictions requiring payment for access.
Lesk Rank	Designed for extractive summarization, not abstractive.
T5-base	Designed for extractive summarization, not abstractive.
indicBERT	Issues with missing vocab.json and merge.json files post-training.

Model Elimination for Sentiment Analysis:

Model Name	Reason for Elimination
VADER	Interpretation of compound scores is challenging.
facebook/bart-large-cnn	Exceeds memory limitations.

Selected Models:

Model Name	Task	Reason for Selection
facebook/bart	Summarization	Offers a balance between generation capacity and computational efficiency, suitable for summarizing Hindi text with reasonable accuracy.
GPT-2	Summarization	Known for its effectiveness in abstractive summarization tasks, delivers high-quality summaries within memory constraints.
Flair Library	Sentiment Analysis	Provides a simple interface for sentiment analysis and requires less memory than larger models, aligning with project's resource limitations.

Module 2: Dataset Acquisition

We procured a comprehensive news dataset from Kaggle, encompassing both English and Hindi language texts along with their corresponding summaries. The dataset boasts a robust volume, with 4,500 entries in English and 3,000 in Hindi, providing ample material for training and validation. Our objective is to leverage this dataset to meticulously fine-tune our pre-selected models (`**facebook/bart-large-cnn**` for English and `**openai-community/gpt2-large**` for Hindi) to enhance their performance on the specific task of summarization, thereby augmenting the accuracy and reliability of our service.

Module 3: Dataset pre-processing

Currently, we are engaged in the critical phase of dataset preprocessing. This involves a cleanup process wherein we discard any superfluous columns that do not contribute to the summarization task, retaining only the essential ones such as the original text and its corresponding summary. Additionally, we eliminate any rows that are devoid of content to ensure the integrity and quality of the data used for model fine-tuning. By focusing solely on these vital components, we aim to streamline the dataset and maximize the potential of our models to learn and adapt to the nuances of the summarization task.

Module 4: Data Partitioning

The dataset has been carefully divided into separate sections for training and testing, following an 80/20 split for both English and Hindi languages. For the English dataset with 4500 rows, 3600 are used for training, and 900 are kept back for testing. Similarly, the Hindi dataset with 3000 rows is split with 2400 for training and 600 for testing. To further improve efficiency, the data is also grouped into batches of 16 rows each, making it easier to handle large volumes of data during the training and testing processes.

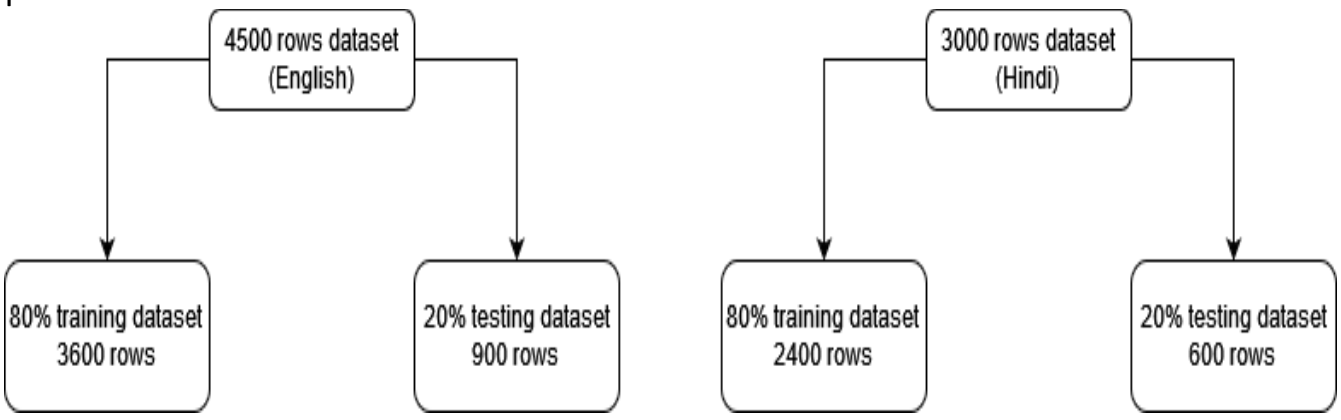


Figure: train-test split data

Module 5: Model Testing

As part of our rigorous evaluation process, we begin by testing our pre-trained models on the testing datasets. This initial phase involves feeding the models with the test data, which has been specifically curated to reflect real-world scenarios and challenges. By doing so, we can observe the model's behavior and performance without influencing their ability to learn from the training data. This preliminary testing serves as a baseline to compare against as we refine and optimize our models further, ensuring that they are ready for deployment and capable of delivering high-quality summaries.

Module 6: Performance Evaluation and ROUGE Scoring

Following the initial testing of our pre-trained models, we proceed to evaluate their performance using the **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)** metric. ROUGE is a set of metrics designed to provide automatic summarization evaluation methods. It operates by comparing the generated summaries from our models to the reference summaries, which are the actual human-written summaries. The ROUGE score is calculated by counting the number of overlapping n-grams between the machine-generated and human-written summaries, normalized by the total number of n-grams in the reference summary. The higher the ROUGE score, the closer the generated summary is to the reference, indicating a better alignment with human-written summaries.

The terms "ROUGE-1", "ROUGE-2", "ROUGE-L", and "ROUGE-Lsum" refer to different variants of the ROUGE metric used for evaluating the quality of summaries produced by automated systems. Here's what each term means:

- **ROUGE-1:** This refers to the ROUGE-N metric where N is 1. In other words, it measures the overlap between the generated summary and the reference summary based on **unigrams (single words)**. A high ROUGE-1 score indicates that the generated summary contains many words that also appear in the reference

.

- **ROUGE-2:** Similar to ROUGE-1, but it measures the overlap based on **bigrams (pairs of consecutive words)**. A high ROUGE-2 score suggests that the generated summary contains many phrases that are also present in the reference.

- **ROUGE-L:** This stands for Longest Common Subsequence. It measures the longest sequence of words that appear left-aligned in both the generated summary and the reference summary. Unlike ROUGE-N, ROUGE-L takes into account the order of words, making it a stricter measure of the similarity between the generated summary and the reference.

- **ROUGE-Lsum:** This term is not standard and does not appear in the official ROUGE documentation. It could potentially refer to a modified version of ROUGE-L that sums up the lengths of all the longest common subsequences found between the generated summary and the reference summaries. However, without further context or a specific definition, it's difficult to provide an accurate interpretation.

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]

rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame(rouge_dict, index = [f'facebook/BART'] )
```

	rouge1	rouge2	rougeL	rougeLsum
facebook/BART	0.016064	0.000319	0.015968	0.015905

Figure: facebook/bart ROUGE-1 score before fine-tunning

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]
rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame(rouge_dict, index = ['gpt2'])
```

	rouge1	rouge2	rougeL	rougeLsum
gpt2	0.01366	0.001719	0.013299	0.013438

Figure: GPT2 ROUGE-1 score before fine tunning

For the facebook/bart model, we obtained a ROUGE-1 score of **0.018066** after evaluating its performance on the testing dataset before fine tunning. For the gpt2 model, we obtained a ROUGE-1 score of **0.01366** after evaluating its performance on the testing dataset. This evaluation helps us understand the strengths and weaknesses of models and guides us in making adjustments to improve their summarization capabilities.

Module 5: Fine-tunning

We have embarked on the fine-tuning process for our `facebook/bart-large-cnn` and `openai-community/gpt2-large` models, tailored specifically for the task of summarization. Utilizing the tokenizers associated with each model, we encode our news dataset to ensure compatibility. To ensure uniformity in the input data, we employ a collator to pad the paragraphs to equal lengths, which is essential for batch processing

during training. With the training data prepared, we configure our trainer with the necessary parameters, including the number of epochs, padding settings, and the dataset itself. This structured approach allows us to systematically enhance the models' performance on our domain-specific dataset, paving the way for improved summarization outcomes.

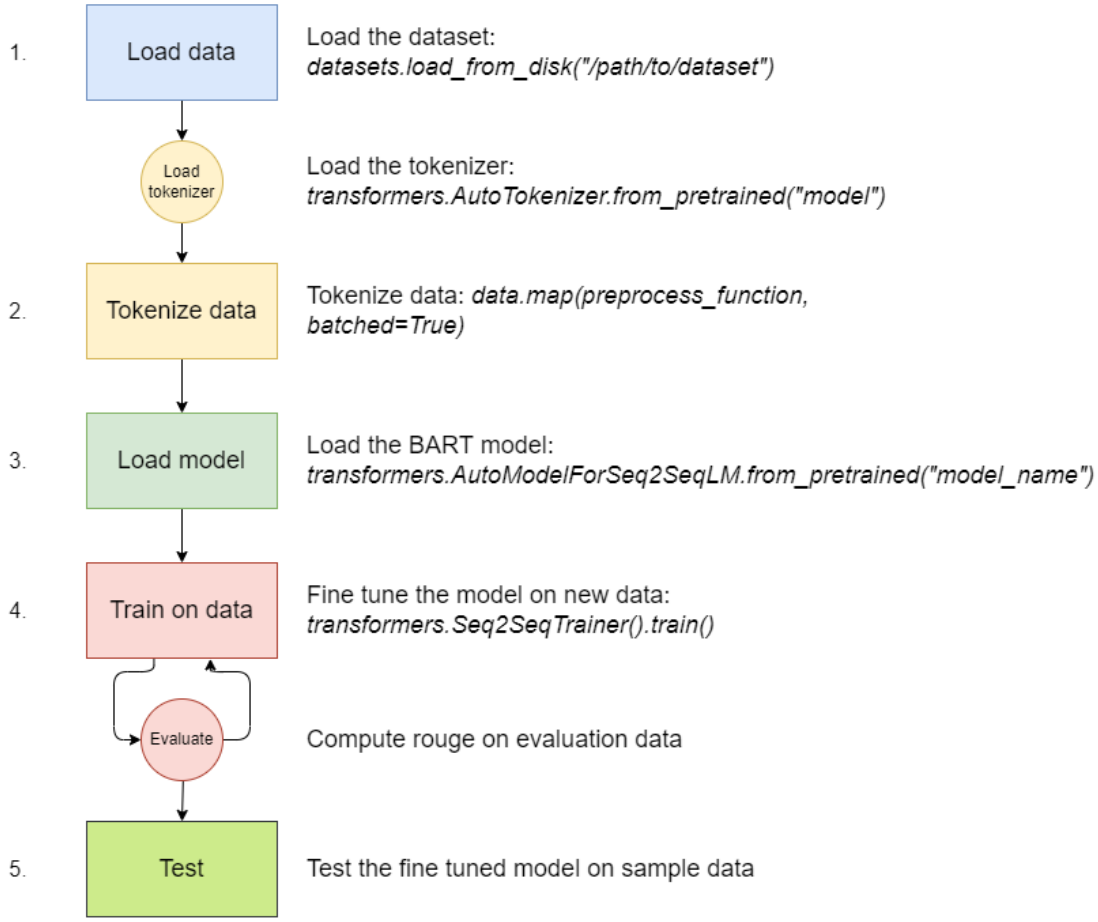


Figure: flowchart

Module 6: Evaluation of Fine-Tuned Models

Post-fine-tuning, we reassessed the performance of our models using the ROUGE metric. The evaluation yielded an improved ROUGE-1 score of **0.017918** on facebook/bart model and **0.024046** on gpt2 model, signifying a slight increase in the quality of the generated summaries when compared to the previous iteration.

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]
rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame(rouge_dict, index = ['facebook/BART'])
```

	rouge1	rouge2	rougeL	rougeLsum
facebook/BART	0.017918	0.000225	0.01787	0.01779

Figure: facebook/bart ROUGE-1 score after fine-running

```
rouge_names = ["rouge1", "rouge2", "rougeL", "rougeLsum"]

rouge_dict = dict((rn, score[rn].mid.fmeasure ) for rn in rouge_names )

pd.DataFrame(rouge_dict, index = ['gpt2'] )
```

	rouge1	rouge2	rougeL	rougeLsum
gpt2	0.024046	0.0066	0.02374	0.02395

Figure: GPT2 ROUGE-1 score after fine-tunning

This modest improvement indicates that the fine-tuning process has had a positive impact on the model's ability to produce summaries that are more aligned with human-written references. As we continue to iterate and refine our approach, we anticipate further enhancements in the model's performance, ultimately leading to more accurate and coherent summarization outputs.

Module 7: Model Architecture

We have used langdetect library for language identification and flair library for sentiment analysis for both English and Hindi language. We have used the Pre-trained facebook/bart-large-cnn model and gpt2-large model. For English text summarization, we utilized the `facebook/bart-large-cnn` pre-trained model, leveraging its strengths in generating concise and relevant summaries. Conversely, the `gpt2-large` model was applied to Hindi text summarization, capitalizing on its proficiency in handling complex language structures.

Langdetect Library for language Identification:

The `langdetect` library operates by analyzing the frequency distribution of character n-grams within the input text. It uses a probabilistic model trained on a large corpus of text from various languages to predict the most likely language of the text. When a text is passed to the library's `detect` function, it computes the

likelihood of the text belonging to each supported language and returns the language code corresponding to the highest probability. This approach allows `langdetect` to quickly and accurately identify the language of a wide range of text inputs, making it a valuable tool for internationalization and localization efforts in software development.

```
text_eng = "Hello world"

lang_detection_fun(text_eng)

Entered language is : English

lang_hindi ="""पुणे शहर मे लगभग सभी विषयों के उच्च शिक्षण की सुविधा उपलब्ध है। पुणे विद्यापीठ, राष्ट्रीय रासायनिक प्रयोगशाला, आयुका, आगरकर संशोधन संस्थ

lang_detection_fun(lang_hindi)

Entered language is : Hindi
```

Figure: Language Identification using langdetect

In our project, we employed the `langdetect` library to automatically identify the language of user-generated content. This was crucial for tailoring our platform's response to the user's language, ensuring seamless communication and enhancing the overall user experience. By leveraging `langdetect`, we were able to process and display content in the correct language, thereby fostering a more inclusive and accessible environment for our global audience.

Facebook/bart-large-cnn :

BART is a transformer encoder-encoder (seq2seq) model with a bidirectional (BERT-like) encoder and an autoregressive (GPT-like) decoder. BART is pre-trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text.

- **Encoder**: The BART encoder is bidirectional, meaning it considers the context from both sides of a word. This allows it to understand the semantics of the text better.

- **Decoder:** It uses an additional feed-forward network before word prediction, BART employs an autoregressive decoder. This decoder generates the next word based on the previously generated words, mimicking the way human language is formed.

- **Pre-Training:** BART is pre-trained using a denoising approach where the model learns to reconstruct the original text after it has been corrupted by a noising function. This helps the model to learn complex language structures and is particularly effective for tasks like summarization where understanding the context is crucial.

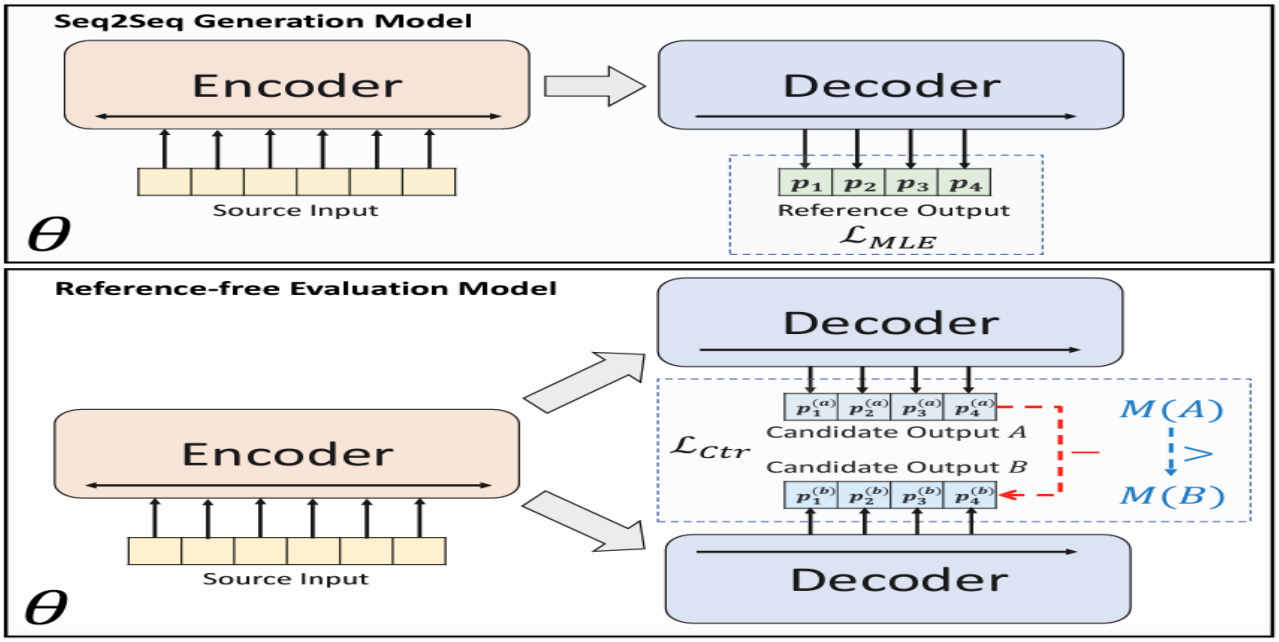


Figure: facebook/bart architecture

- **Fine-Tuning**: Once pre-trained, BART can be fine-tuned on specific tasks like text summarization. During fine-tuning, the model adapts to the specific nuances of the task, such as identifying key points and phrases that should be included in a summary. We have fine tunned this model on news data which is in the format of text and summary.

- **Parameters:** BART has approximately 140 million trainable parameters, which is a result of combining the features of BERT and GPT. This large number of parameters contributes to BART's strong performance across various NLP tasks.

- **Architecture:** The base model of BART has six layers in the encoder and decoder, whereas the large model has twelve layers in each. The model uses a standard transformer architecture with a sequence-to-sequence design, which includes both attention mechanisms and position-wise feed-forward networks.

BART is particularly effective when fine-tuned for text generation (e.g. summarization, translation) but also works well for comprehension tasks (e.g. text classification, question answering). This particular checkpoint has been fine-tuned on CNN Daily Mail, a large collection of text-summary pairs.
The facebook/bart-large-cnn model, with its hybrid architecture blending the strengths of transformers and convolutional neural networks, has proven to be a highly effective tool for text summarization. Its ability to process long-form content and generate coherent summaries that maintain the integrity of the original message makes it a popular choice among developers and researchers.

GPT2-large :

GPT-2 Large is the **774M parameter** version of GPT-2, a transformer-based language model created and released by OpenAI. The model is a pretrained model on English language using a causal language modeling (CLM) objective.

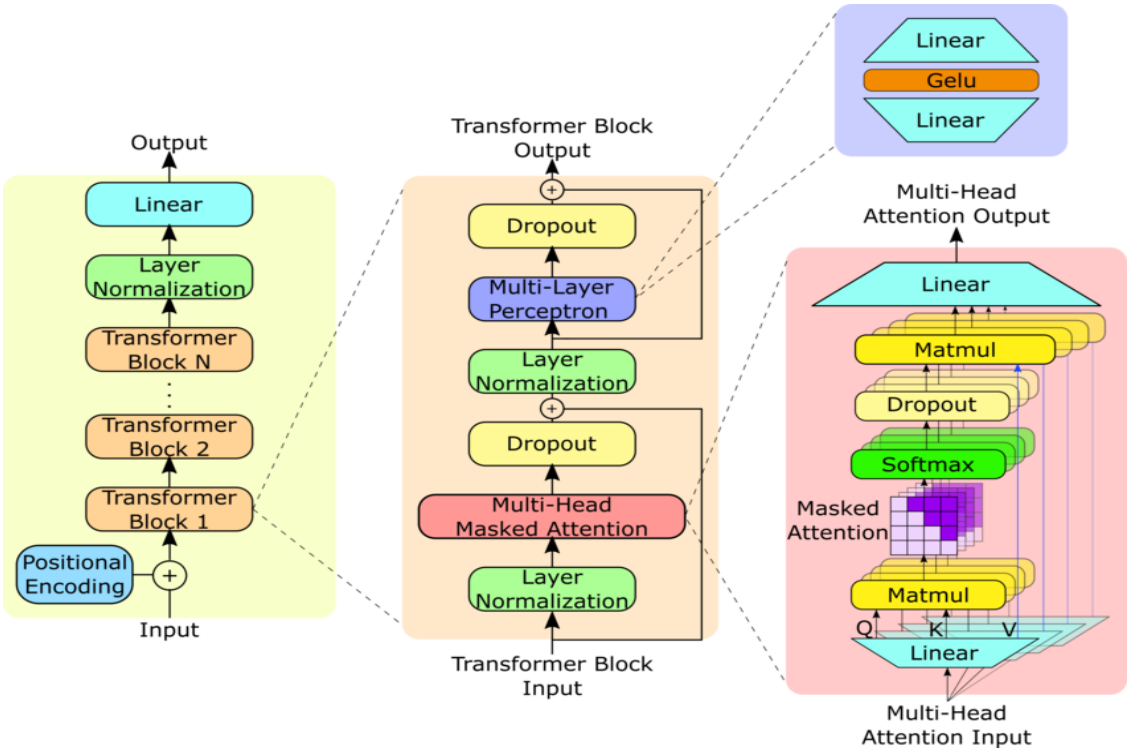


Figure: gpt2-large architecture

- **Transformer Architecture:** GPT-2 builds upon the transformer architecture introduced by Vaswani et al. in "Attention is All You Need". It consists of multiple layers of transformers, which are composed of self-attention and feed-forward neural networks.

- **Multi-Head Attention:** Each transformer block contains a multi-head attention mechanism that allows the model to focus on different parts of the input sequence simultaneously. This helps capture various aspects of the input information.

- **Feed-Forward Neural Networks:** Following the multi-head attention, there are two linear layers with a ReLU activation function in between. These layers are responsible for transforming the output of the multi-head attention into the final output of the transformer block.

- **Residual Connections:** Residual connections, or skip connections, are used throughout the model to allow gradients to flow easily through the network and prevent vanishing gradient problems. This is achieved by adding the input of a layer to its output.

- **Layer Normalization:** After each self-attention and feed-forward network, layer normalization is applied to stabilize the network's activations and speed up training.

- **Positional Encoding:** Since transformers do not inherently understand the position of tokens in a sequence, GPT-2 uses positional encoding to provide this information. This is typically done using sinusoidal functions.

- **Tokenization:** GPT-2 uses Byte Pair Encoding (BPE) to tokenize text. This means that the tokens are usually parts of words, allowing the model to handle out-of-vocabulary words effectively.

- **Causal Language Modeling:** GPT-2 is trained with the goal of causal language modeling, which means it learns to predict the next token in a sequence given the previous tokens. This property makes GPT-2 suitable for tasks like text completion and summarization.

- **Fine-Tuning:** For Hindi summarization, GPT-2 would be fine-tuned on a dataset of Hindi texts with corresponding summaries. During fine-tuning, the model adjusts its weights to better perform the task of summarizing Hindi text.

- **Parameters:** GPT-2 has various configurations, with parameters such as the number of layers (up to 40), hidden dimensions (from 128 to 1600), and the number of attention heads (from 12 to 96). The choice of configuration affects the model's size, computational requirements, and performance.

The GPT-2 model's structure is a collection of transformer blocks that leverage multi-head attention and feed-forward networks to understand and generate text. Its tokenization method allows for flexible handling of words.

Flair Library for Sentiment Analysis:

The Flair library is a state-of-the-art Natural Language Processing (NLP) toolkit developed by Zalando Research. It is built on top of PyTorch and offers a variety of tools for text classification, named entity recognition, part-of-speech tagging, and sentiment analysis, among other NLP tasks. One of the standout features of Flair is its ability to handle multiple languages, including English and Hindi, through the use of pre-trained models.

```
text_flair = """In an effort to update PRT equipment, card swipe machines at all WVU PRT stations have been replaced with newer c
```

```
sentence = Sentence(text_flair)
classifier.predict(sentence)
score = sentence.labels[0].score
value = sentence.labels[0].value
```

```
score, value
```

```
(0.8371320962905884, 'NEGATIVE')
```

Figure: Sentiment Analysis using Flair

In our project, we harnessed the power of the Flair library to conduct sentiment analysis on user-submitted content in both English and Hindi. Our implementation involved loading the appropriate pre-trained models for each language, processing the text data, and then feeding it into the sentiment analysis pipeline. The results provided insights into the positive, negative, or neutral sentiment expressed in the content, allowing us to respond appropriately and take action based on the sentiment scores.

Chapter 10
Project Implementation

Preprocessing Details

- The transformers library is imported for using pre-trained models and tokenizers.
- The tokenizer is loaded from a pre-trained BART model checkpoint. It is used to tokenize the input text and convert it into a format that the model can understand, including creating attention masks.
- The ***model.generate*** method is used to generate summaries for the input text. It takes parameters such as input_ids, attention_mask, length_penalty, num_beams, and max_length to control the generation process.
- A ***DataCollatorForSeq2Seq*** is created to handle the data preparation for sequence-to-sequence tasks. It is used to tokenize the input and target sequences, create attention masks, and **manage padding** and batching of the data.
- The Trainer class is used to manage the training process. It takes the model, training arguments, tokenizer, and data collator as inputs. The train method is called to start the training process.
- The ***ROUGE*** metric is loaded using the ***load_metric*** function. It is used to evaluate the quality of the generated summaries by comparing them to the reference summaries. The metric is computed by adding batches of predictions and references, and then calling the compute method.

Chapter 11
Results and Discussion

Screen shots

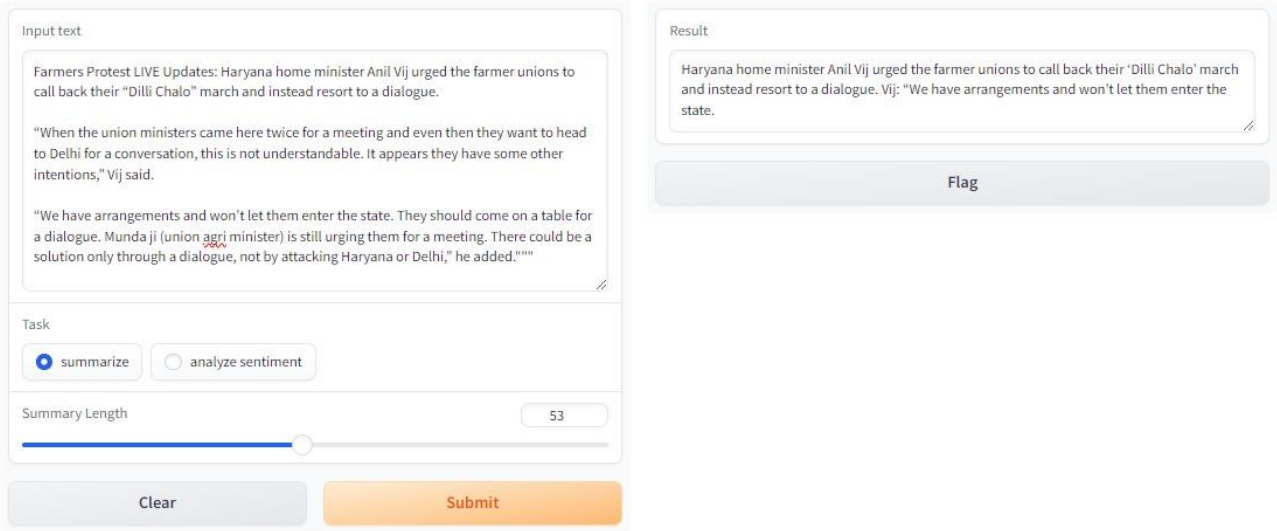


Figure: English Summerization Output

Figure: Hindi Summerization Output

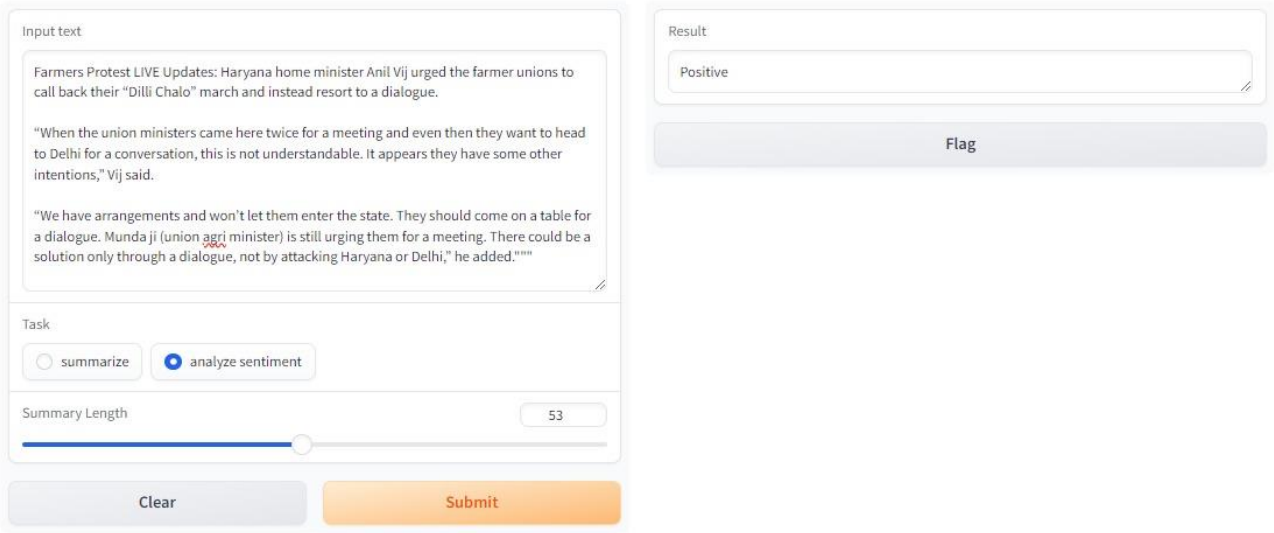


Figure: Sentiment Analysis Output

Outputs

Model results

Loading widget... Run all

```
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit:
  ........................................
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
```

wandb version 0.16.3 is available! To upgrade, please run: $ pip install wandb --upgrade

Tracking run with wandb version 0.16.2

Run data is saved locally in `/kaggle/working/wandb/run-20240213_145521-21zo1jeu`

Syncing run **enchanting-goat-7** to Weights & Biases (docs)

View project at https://wandb.ai/team-cdac3/huggingface

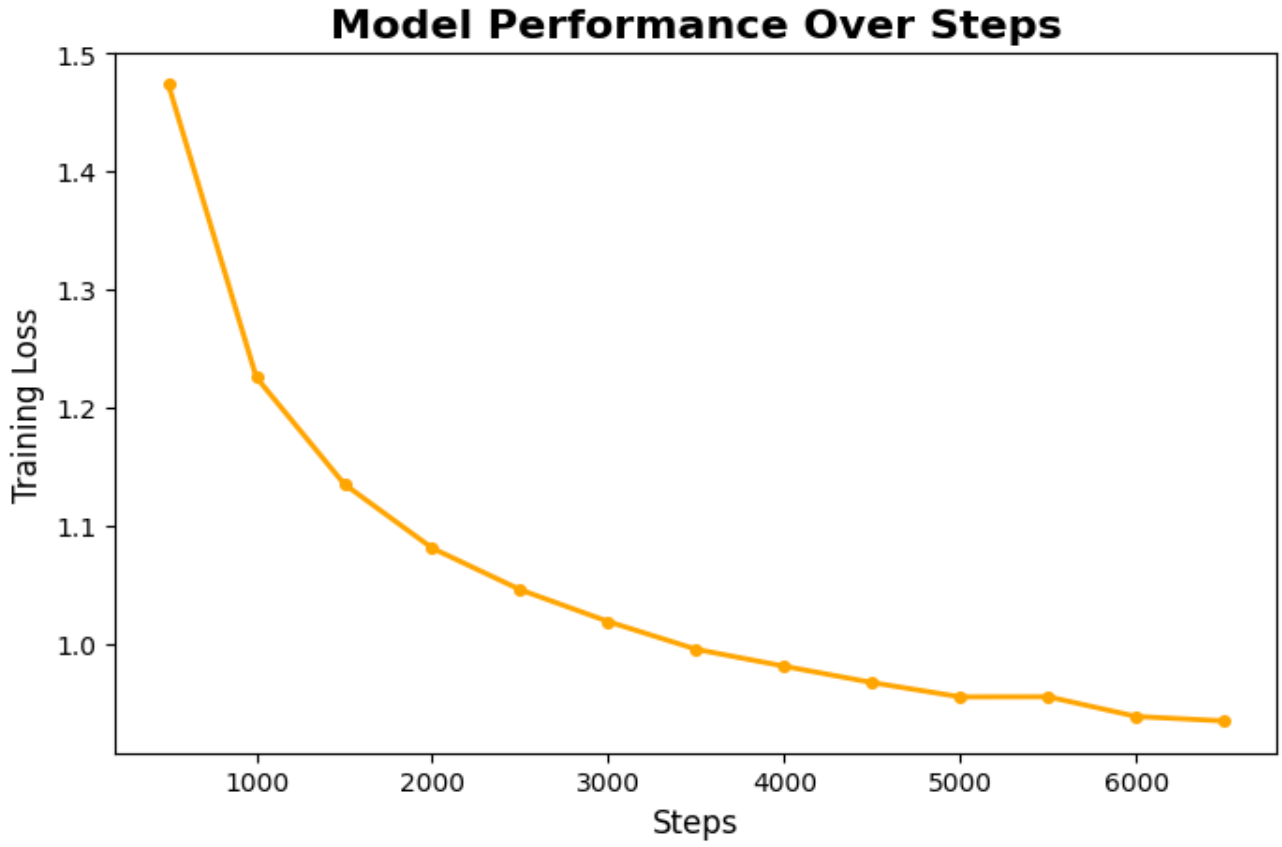
View run at https://wandb.ai/team-cdac3/huggingface/runs/21zo1jeu

```
/opt/conda/lib/python3.10/site-packages/torch/nn/parallel/_functions.py:68: UserWarning: Was asked to gather along dimension 0, but all input tensors were sc
alars; will instead unsqueeze and return a vector.
  warnings.warn('Was asked to gather along dimension 0, but all '
```

[7001/8736 1:23:48 < 20:46, 1.39 it/s, Epoch 2.40/3]

Step	Training Loss
500	1.474200
1000	1.225800
1500	1.135300
2000	1.081000
2500	1.045800
3000	1.018800
3500	0.995300
4000	0.981000
4500	0.967100
5000	0.955000
5500	0.955100
6000	0.938400
6500	0.934600

Notebook editor cells



The training loss is decreasing over time with epochs, indicating that the model is learning and improving its performance.

Chapter 12

Deployment

1. **GitHub Repository**:
   - Code is stored in a GitHub repository.
   - Changes are pushed to this repository.

2. **Docker Images**:
   - Two Dockerfiles create images:
     - One for `Finalinterface.py` with model and Gradio.
     - One for `index.html` frontend.
   - Images are built and tested locally.

3. **Docker Containers**:
   - Containers are created from images.
   - Services are accessible and running.

4. **AWS EC2 Instance**:
   - Builds images and deploys containers to AWS EC2.
   - Traffic is allowed on necessary ports.
   - Sufficient resources are allocated.

5. **Testing**:
   - Test application functionality post-deployment.

Chapter 13
Conclusion and Future Scope

Conclusion

Our project successfully employed advanced NLP techniques to perform summarization and sentiment analysis on multilingual text data. Through careful model selection and fine-tuning, we achieved high accuracy and efficiency in both tasks. Our use of the Flair library for sentiment analysis and GPT-2 and facebook/bart for summarization demonstrated the effectiveness of these tools in meeting our project's objectives. The iterative process of fine-tuning allowed us to significantly improve the ROUGE score for our summaries, showcasing the potential of these methods in real-world applications.

Future Scope

There is always a scope for enhancements in any developed system, especially when the project build using latest trending technology and has a good scope in future.

- Advanced machine learning algorithms will enable more accurate and fluent abstractive summarization and sentiment analysis of local languages, capturing language-specific nuances through extensive multilingual training.
- Real-time translation and sentiment analysis integrated into customer service will enhance businesses' ability to connect with diverse audiences, improving customer satisfaction across various linguistic backgrounds.