ISSN No:-2456-2165

Abstrating Wisdom: Text Summarization in the Age of Intelligence

Palaparthi Manichandana[1*]
[1]CSE, Sri Vasavi Engineering College (A), Pedatadepalli, Tadepalligudem – 534101

Nekkanti Durga Sri Jahnavi[2]
[2]CSE, Sri Vasavi Engineering College (A), Pedatadepalli, Tadepalligudem – 534101

Arepalli Shakeena[3]
[3]CSE, Sri Vasavi Engineering College (A), Pedatadepalli, Tadepalligudem –534101

Sontim Lakshmi Meghana[4]
[4]CSE, Sri Vasavi Engineering College (A), Pedatadepalli, Tadepalligudem –534101

Bandaru Neha[5]
[5]CSE, Sri Vasavi Engineering College (A), Pedatadepalli, Tadepalligudem – 534101

Dr. Shirin Bhanu koduri[6] (Associate Professor)
[6]Department of CSE, Sri Vasavi Engineering College (A),Pedatadepalli, Tadepalligudem-534101

Corresponding Authors:- Palaparthi Manichandana[1*]

Abstract:- With the ever-increasing volume of information on the internet, it becomes imperative to develop methods to efficiently curate this data for users. Our overarching goal in this endeavor is to discover an optimal solution for the task of 'Text Summarization'. One of the primary challenges encountered in the realm of text summarization involves the delicate equilibrium between brevity and informativeness. An effective summary must adeptly encapsulate the central concepts and pivotal details of the original text while maintaining conciseness and logical coherence. In essence, text summarization occupies a crucial role in addressing the mounting deluge of textual data witnessed in the digital era. Its potential is noteworthy, offering the capacity to economize time, enhance information retrieval, and amplify knowledge accessibility across diverse domains. Text summarization entails the process of identifying and extracting the most pertinent and meaningful information from a document or a collection of related documents, subsequently condensing this content into a concise version while retaining its overarching significance. In this project, our aim is to assist users in efficiently summarizing various forms of text data, including articles, paragraphs, web pages, and blogs, all within a shorter time frame. In this process, we will utilize Python within the field of 'Natural Language Processing' (NLP) along with the 'Natural Language Toolkit Library' and other Python-based machine learning libraries. NLP is a domain at the intersection of computer science and linguistics, focused on facilitating meaningful interactions between machines and human languages. We currently reside in an age dominated by machines, data abundance, and artificial intelligence. Our aim is to provide output in two formats: text and voice-based, enabling users to save valuable time while accessing the information they require.

Keywords:- *Transformers, Encoder, Decoder, BART, Self-Attention, Pytssx3, Beautiful Soup4.*

I. INTRODUCTION

The Internet has become the go-to resource for people seeking information, and they rely on proficient information retrieval (IR) tools like Bing, Google, Yahoo,Ask.com, AltaVista, and others these tools are primarily crafted to retrieve information from the Internet and furnish users with search outcomes. However, in many instances, users find the process of sifting through search results to uncover the main essence of their query to be tedious and time- consuming. Data and information play a progressively pivotal role in our daily lives. This importance is accentuated by projections from the International Data Corporation (IDC), indicating a remarkable growth in the annual volume of digital data traversing the global Internet. According to IDC, this volume is expected to skyrocket from 33 zettabytes in 2018 to a staggering 175 zettabytes by 2025. Given this exponential surge in online information and the abundance of digital resources, text summarization has emerged as an indispensable tool to navigate our fast-paced existence. In this context, text summarization systems have proven to be a boon for the Academic community and research community. These systems significantly reduce the time and effort required to manually extract key insights from extensive documents. The core objective of Natural Language Processing (NLP) is to empower computers to comprehend, interpret, and produce human language in a manner that adds value, conveys significance, and serves practical purposes. Text Comprehension in NLP strives to equip computers with the ability to grasp the meaning conveyed in human language text. Language Translation in NLP systems possess the capability to translate text from one language to another, as exemplified by online translation services and multilingual chatbots. Text Generation through NLP, is a focus on developing systems that can produce text

resembling human language. This application finds use in chatbots, content creation, and even creative writing. Information Retrieval in NLP aids in constructing systems designed to extract relevant information from vast collections of text data, with search engines serving as a common illustration. Text Summarization NLP techniques are harnessed to create systems capable of automatically generating concise summaries of lengthier texts, such as news articles or research papers. Language Modeling in Language models acquire an understanding of a language's structure and patterns by analyzing extensive text corpora, forming the basis for various NLP tasks. Speech-to-text conversion in NLP encompasses the creation of systems capable of transforming spoken language into written form.

"In the context of text summarization, 'abstractive' and 'extractive' represent distinct approaches or methods employed for generating concise summaries from longer texts. Abstractive summarization entails the creation of a summary that may include sentences or phrases not directly present in the source text. Its objective is to produce a succinct and logically coherent summary by comprehending the text's content and generating human-like sentences that may employ varying vocabulary or sentence structures while preserving the core meaning of the original text. Abstractive summarization poses greater complexity and challenges compared to extractive summarization, as it necessitates a deeper comprehension of the text and the capacity to generate novel content." Conversely, extractive summarization entails the process of choosing and extracting sentences or passages directly from the source text to compose a summary. This approach does not involve the rewording or paraphrasing of content; instead, it identifies the most significant and pertinent sentences within the source text and presents them verbatim. Extractive summarization methods typically rely on algorithms that assess sentences based on various criteria, such as their importance, before selecting the highest-ranked sentences for inclusion in the summary. While extractive summarization is generally considered more straightforward and less complex than abstractive summarization, it may not consistently produce summaries as fluid and coherent as those generated through abstractive techniques.

II. LITERATURE SURVEY

- Lewis, Mike, proposed a paper "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." this paper is all about a denoising autoencoder used for pretraining sequence-to-sequence models in natural language processing. It emphasizes BART's Transformer-based architecture, versatility in tasks, achievement of state-of-the-art results comparable to RoBERTa, and innovative fine-tuning approaches, including improvements in machine translation.
- Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani proposed paper "Self-attention with relative position representations." this paper is all about an improved Transformer attention mechanism that effectively includes relative position representations, leading to enhanced machine translation quality. Unlike traditional absolute position encodings, this method demonstrates notable improvements.
- Kobayashi, Goro,proposed a paper "Attention is not only a weight: Analyzing transformers with vector norms." this paper is all about a norm-based approach to analyzing Transformers, which takes into account the norm of transformed input vectors alongside attention weights. This comprehensive analysis sheds light on the internal mechanisms of Transformers and highlights the inadequacy of relying solely on weight-based analysis. The results reveal a lack of attention towards special tokens in BERT and demonstrate the Transformer attention mechanisms' ability to extract meaningful word alignments.
- Chen, Pu-Chin, proposed a paper "A simple and effective positional encoding for transformers." this paper is all about The significance of positional encoding in Transformer models for Natural Language Processing tasks, which introduces a novel approach named Decoupled Positional Attention for Transformers (DIET). This method not only delivers competitive performance but also boasts faster training and inference times. Additionally, this paper evaluates various positional encoding techniques and their impact on performance.
- Liu, Liyuan, proposed a paper "Understanding the difficulty of training transformers." this paper is all about the present analysis focuses on the challenges encountered when training Transformers in natural language processing (NLP) tasks. To address these difficulties, a solution named Admin (Adaptive model initialization) is proposed in this paper. Admin aims to enhance the stability of initial training stages and optimize overall performance.
- Goyal, Tanya, Junyi Jessy Li, and Greg Durrett proposed a paper "News summarization and evaluation in the era of gpt-3." this paper is all about the influence of GPT-3 on text summarization, particularly in the field of news summarization. The researchers analyze the performance of GPT-3 in comparison to fine-tuned models that were trained on extensive summarization datasets. Moreover, this paper provides sample news articles along with their corresponding GPT-3 summaries.
- Augenstein, Isabelle proposed a paper "Stance detection with bidirectional conditional encoding." this paper is all about the challenging task of stance detection, which involves classifying the attitude expressed in a text towards a target as positive, negative, or neutral. They propose a neural network architecture based on conditional encoding that outperforms previous approaches on the SemEval 2016 Task 6 Twitter Stance Detection corpus.
- Scholak, Torsten, Nathan Schucher, and Dzmitry Bahdanau proposed a paper "PICARD: Parsing incrementally for constrained auto-regressive decoding from language models." this paper is all about a method called PICARD for constraining auto-regressive decoding of language models through incremental parsing, which helps to find valid output sequences by rejecting inadmissible tokens at each decoding step. The

method is evaluated on the Spider and CoSQL text-to-SQL translation tasks, showing state-of-the-art results.

- Awangga, Rolly Maulana, Syafrial Fachri Pane, and Restiyana Dwi Astuti proposed a paper "Implementation of web scraping on GitHub task monitoring system." this paper is all about the implementation of web scraping technology on GitHub for task management in e-learning. It also covers the use of BeautifulSoup4 for XML parsing and provides an algorithm for scraping content.
- Chandra, Rakesh Vidya, and Bala Subrahmanyam Varanasi proposed a paper "Python requests essentials." this paper is all about an introduction and detailed usage of HTTPretty, a Python library for mocking HTTP requests and responses, and includes an example of building a Flask web application.
- Yadav, Avanish Vijaybahadur, Sanket Saheb Verma, and Deepak Dinesh Singh proposed a paper "Virtual Assistant for blind people." this paper is all about the development of a basic abettor app for dark individuals application bogus intelligence, apparatus learning, angel and argument recognition. It additionally includes a abstract analysis on accustomed human- computer alternation for basic claimed abettor systems and the abutting bearing of basic claimed assistants.
- Ifeanyi, Nwakanma, Oluigbo Ikenna, and Okpala Izunna prposed a paper "Text–To– Speech Synthesis (TTS)." this paper is all about the Text-to-Speech Synthesis technology and its applications, as well as the methodology and analysis of a new system that aims to improve upon existing systems by incorporating Object Oriented Analysis and Development Methodology and Expert System.

➢ *Model:*

Transformers are a class of deep learning models that have revolutionized natural language processing (NLP) tasks, such as text classification, text summarization.

➢ *Data Set:*

The CNN/DailyMail dataset is a widely used summarization dataset that associates important news as summaries. It is mostly used for text summarization tasks and it consists of news articles and their corresponding human-understandable summaries. The CNN/Daily Mail datasets consist of bullet points that describe the articles, whereas multiple sentence summaries are created by combining the bullet points of the article.

The current CNN/DailyMail version of the dataset supports both abstractive summarization and extractive summarization.

By predefined dataset has been tokenized and the dataset contains pairs of articles and articles can have multiple summaries. The DailyMail dataset contains a wide range of articles and feature stories.

Researchers and developers use the CNN/DailyMail dataset for tasks such as abstractive text summarization, extractive text summarization, etc. It performs duties as a standard for evaluating the performance of summarization models.

III. METHODOLOGY OF TRANSFORMERS

➢ *Encoder-Decoder Structure:*

Transformers consist of an encoder and a decoder. The input text is encoded by the encoder and the summary will be generated by the decoder with the help of multiple self-attention and feed-forward layers.

➢ *Self-Attention Mechanism:*

The self-attention mechanism is a key component in many modern deep learning models, particularly in the field of natural language processing (NLP). It is used to weigh and combine different elements within a sequence of data, such as words in a sentence, in a way that gives more importance to relevant elements while downplaying the impact of less relevant ones. The self-attention mechanism has been a crucial innovation, especially in the context of transformer-based models like BERT, GPT-3, and many others.

➢ *Here's a High-Level Explanation of how Self-Attention Works:*

- Input Sequence: Suppose you have a sequence of elements, such as a sentence. Each element is represented as a vector, often in the context of NLP, these vectors are word embeddings.
- Key, Query, and Value: For each element in the sequence, the self-attention mechanism creates three vectors: the key, query, and value vectors. These vectors are learned from the input data.

✓ The query vector represents the element you're currently interested in understanding.
✓ The key vector represents the elements in the sequence that are used to compare against the query to determine their relevance.
✓ The value vector contains information about the element, which is typically used when combining the information from other elements.

- Attention Scores: For each element in the sequence, the self-attention mechanism computes a score that represents how relevant that element is to the current query. This score is calculated by taking the dot product of the query vector of the current element and the key vectors of all other elements. These scores are then scaled and passed through a softmax function to obtain a distribution of attention weights that sum to 1.
- Weighted Sum: The attention scores obtained in the previous step are used to compute a weighted sum of the value vectors of all elements in the sequence. This weighted sum represents the information from the entire sequence that is most relevant to the current query.
- Output: The weighted sum computed in the previous step is the output of the self-attention mechanism for the current element. It is typically combined with the input embedding for that element and passed through a feedforward neural network to obtain the final

representation of that element.

By applying this self-attention mechanism across all elements in the sequence, you can capture complex relationships and dependencies between elements, which is particularly useful for tasks like language understanding and translation.

Transformers, which are models built on top of self-attention mechanisms, have been highly successful in various NLP tasks and have also found applications in computer vision, speech recognition, and other domains due to their ability to model long-range dependencies and capture contextual information effectively.

- *Training:*

Transformers are trained on large text corpora using objectives like masked language modeling (predicting masked words) and next-sentence prediction. Fine-tuning is performed on specific

Where Q, K andV denotes query vector, key vectors, and value vectors, respectively. dk denotes the dimension of one vector of K. Softmax function outputs the attention weights distributed over V . Attention(Q, K, V) is a vector of the weighted sum of elements of V, and represents current context information as such as summarization, by minimizing loss functions tailored to those tasks.

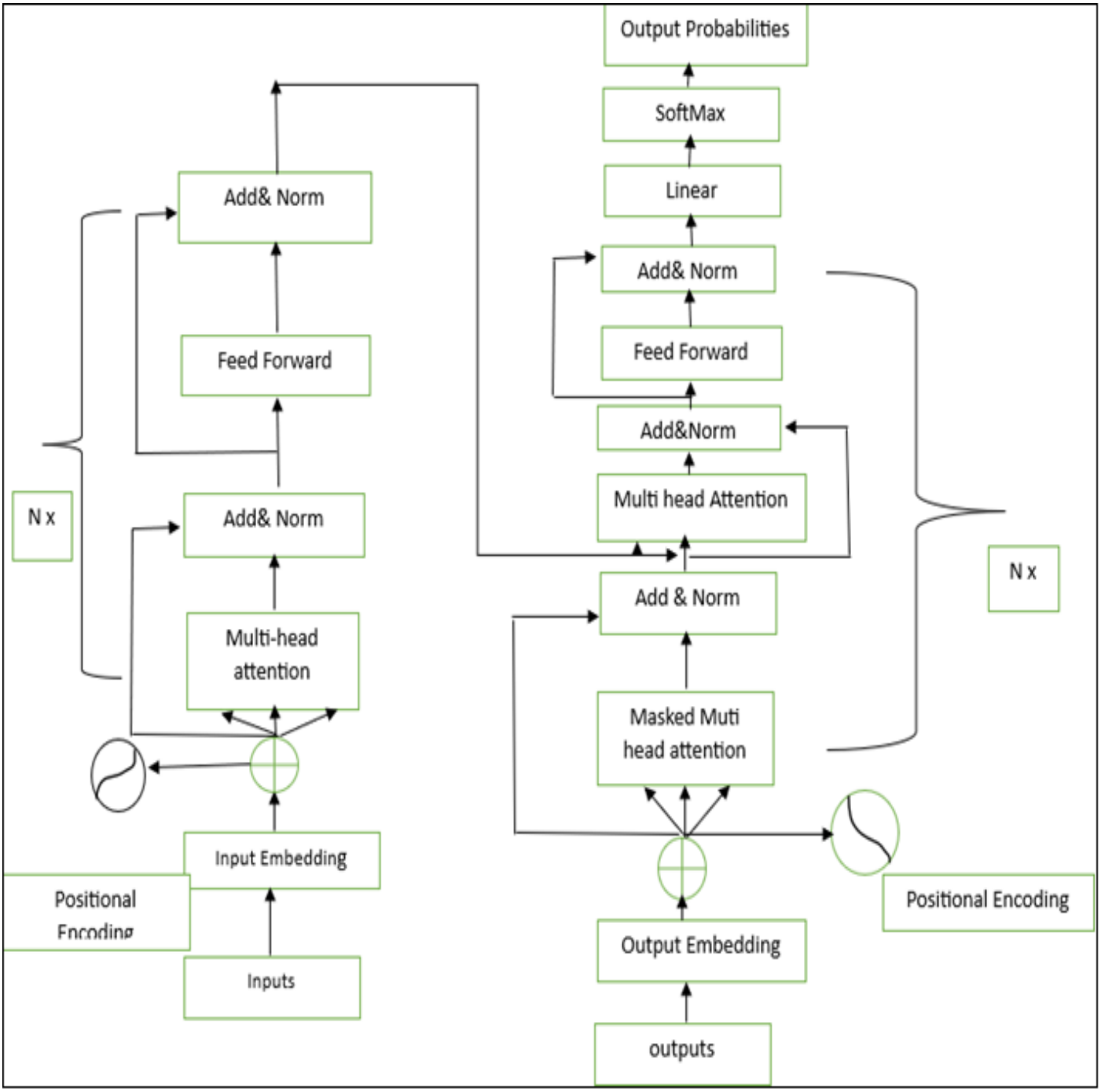

Fig 1 Transform Architecture

➢ *BART:*

BART architecture comprises both an encoder and a decoder, and it combines ideas from autoencoders and sequence-to-sequence models. Let's delve into the architecture of both the encoder and decoder in depth.

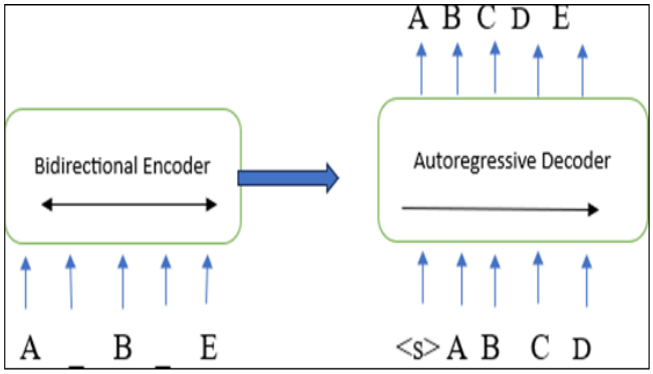

Fig 2 BART Architecture

➢ *BART Encoder:*

The encoder in BART is responsible for processing the input text and transforming it into a fixed-dimensional representation. It typically takes a sequence of tokens as input, where tokens can be subword units (e.g., Byte-Pair Encoding or BPE tokens) or words.

➢ *Token Embeddings:*

The input tokens are first converted into dense vector representations called token embeddings. These embeddings are often pre-trained on large text corpora using methods like Word2Vec, GloVe, or more commonly, contextual embeddings like BERT embeddings.

➢ *Positional Encodings:*

To provide positional information to the model, positional encodings are added to the token embeddings. These encodings help the model distinguish between different positions in the input sequence.

➢ *Multi-Head Self-Attention Layers:*

The core of the transformer architecture is the multi-head self-attention mechanism. In the encoder, multiple self-attention heads are applied to the token embeddings in parallel. This allows the model to capture relationships between different tokens in the input sequence.

➢ *Feedforward Neural Networks (FNNs):*

After the self-attention layers, each position's representation is passed through a feedforward neural network. This network applies linear transformations and activation functions to the representations, adding non-linearity to the model.

➢ *Layer Normalization and Residual Connections:*

Layer normalization and residual connections are applied after each sub-layer (self-attention and FFN). These techniques help stabilize training and improve the flow of gradients during training.

➢ *Stacked Layers:*

BART typically has multiple layers of the above components stacked on top of each other (e.g., 12 or 24 layers), allowing the model to capture increasingly complex patterns in the input data.

➢ *BART Decoder:*

The decoder in BART is responsible for generating the output sequence based on the encoded information. It operates autoregressively, meaning it generates one token at a time while conditioning on previously generated tokens.

➢ *Token Embeddings:*

Similar to the encoder, the decoder starts with token embeddings for the previously generated tokens. During training, this includes both the ground truth tokens and the tokens generated by the model itself during the autoregressive process.

➢ *Positional Encodings:*

Like the encoder, the decoder also includes positional encodings to provide positional information.

➢ *Multi-Head Self-Attention and Cross-Attention:*

The decoder uses multi-head self- attention layers to capture dependencies among the tokens in the generated sequence. Additionally, it uses multi-head cross-attention layers to attend to the encoder's output, allowing it to focus on different parts of the input during decoding.

➢ *Feedforward Neural Networks (FNNs):*

After the attention layers, the decoder passes the representations through feedforward neural networks with residual connections and layer normalization.

➢ *Softmax Layer:*

The final layer of the decoder is a softmax layer that produces a probability distribution over the vocabulary for the next token in the sequence. During training, this distribution is used to compute the loss, and during inference, it is sampled to generate the next token.

➢ *Stacked Layers:*

Similar to the encoder, the decoder consists of multiple stacked layers of the components mentioned above.

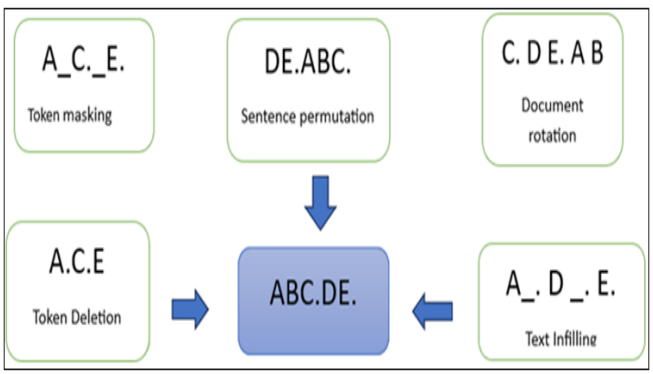

Fig 3 Summarization Process

➢ *Tokenization:*

Tokenization is a fundamental and necessary step in NLP tasks, including text summarization. It is used to convert raw text into tokens that can be processed by the model. Tokenization is a preprocessing step that occurs before training and inference with BART.

➢ *Token Deletion:*

Token deletion may be used in the context of text summarization, but it's not always a primary technique. Token deletion can be employed when generating a summary from a longer piece of text. You may selectively delete tokens from the original text to create gaps for the model to fill in and generate a concise summary.

➢ *Text Infilling:*

Text infilling, which involves generating missing portions of text, can be useful in text summarization, especially when you want to complete or condense a text while maintaining context. You can use text infilling to generate missing sentences or phrases in the summary.

➢ *Sentence Permutation:*

Sentence permutation is typically not used in the core process of text summarization. Summarization aims to produce a coherent and concise summary from the input text, and shuffling the order of sentences within the input text may not align with that goal. However, sentence permutation could be used for data augmentation or text diversity in training data.

➢ *Document Permutation:*

Document permutation, involving the rearrangement of entire documents, is generally not part of the standard text summarization process. Summarization focuses on extracting key information from a single document or text, and altering the order of entire documents may not be relevant to this task.

➢ *BeautifulSoup4:*

Beautiful Soup (BeautifulSoup4): A Python Library for Web Scraping and Parsing HTML

Beautiful Soup, often referred to as BeautifulSoup4 or simply BS4, is a widely-used Python library that plays a crucial role in web scraping and parsing HTML and XML documents. It serves as a powerful tool for extracting structured data from web pages and navigating the complex HTML structure of websites. Beautiful Soup simplifies the process of web scraping by providing a Pythonic and intuitive interface.

➢ *Working :*

HTML Parsing: Beautiful Soup is capable of parsing HTML and XML documents, making it suitable for handling web content.

➢ *Navigational Capabilities:*

It provides methods and attributes for navigating the HTML structure, allowing users to locate specific elements based on tag names, attributes, class names, or IDs.

➢ *Element Extraction:*

Users can extract data from located HTML elements, including text content, attributes, and more.

➢ *Traversal:*

Beautiful Soup enables traversal of the HTML tree structure, supporting operations like finding parent, sibling, and child elements.

➢ *Modifying HTML:*

While its primary purpose is parsing and extraction, Beautiful Soup can also modify HTML documents by adding, modifying, or deleting elements and attributes.

➢ *HTML Cleanup:*

It can clean up poorly formatted HTML, making it easier to work with inconsistent web data.

➢ *Compatibility with various parsing tools:*

Beautiful Soup supports various HTML and XML parsers, including html.parser, lxml, and others, allowing users to choose the most suitable one for their needs.

➢ *Pytssx3:*

Text-to-speech (TTS) engines, including those used by libraries like pyttsx3, convert text into voice using a combination of linguistic and acoustic modeling techniques. Here's a simplified overview of how a typical TTS engine converts text into voice.

➢ *Text Analysis:*

The input text is first analyzed linguistically to identify linguistic features such as words, phonemes (the smallest units of sound), punctuation, and sentence structure.
This linguistic analysis is crucial for generating natural and intelligible speech.

➢ *Text-to-Phoneme Conversion:*

The text is then converted into phonemes, which are the individual sound units of a language. Each language is composed of their own phonemes.

This step involves mapping words or sequences of characters to their corresponding phonemes.

➢ *Language rules and concept:*

Language rules and concept are applied to determine pronunciation, stress patterns, and intonation. These rules help ensure that the speech sounds natural and follows the language's prosody (rhythmic and intonational patterns).

➢ *Acoustic Modeling:*

Acoustic models are used to generate the actual speech waveform. These models capture how phonemes are pronounced and how they transition from one to another.

Acoustic models can be complex machine learning models, such as Hidden Markov Models (HMMs) or neural networks, trained on large datasets of recorded human speech.

ISSN No:-2456-2165

➢ *Voice Synthesis:*

The TTS engine selects a voice from a collection of available voices. Each voice has its own unique characteristics, including pitch, tone, and accent.

The acoustic model generates the speech waveform for each phoneme based on the selected voice.

➢ *Speech rhythm and pitch variation:*

Speech rhythm and pitch variation patterns, such as rising and falling pitch, are applied to the speech waveform to convey emotions, emphasis, and natural speech patterns.

This step helps make the synthesized speech sound more expressive and human-like.

➢ *Audio Rendering:*

The synthesized speech waveform is converted into an audio format (e.g., WAV or MP3) that can be played back through speakers or headphones.

➢ *Playback:*

Finally, the generated audio is played back to the user, making the synthesized text audible.

IV. RESULT AND ANALYSIS

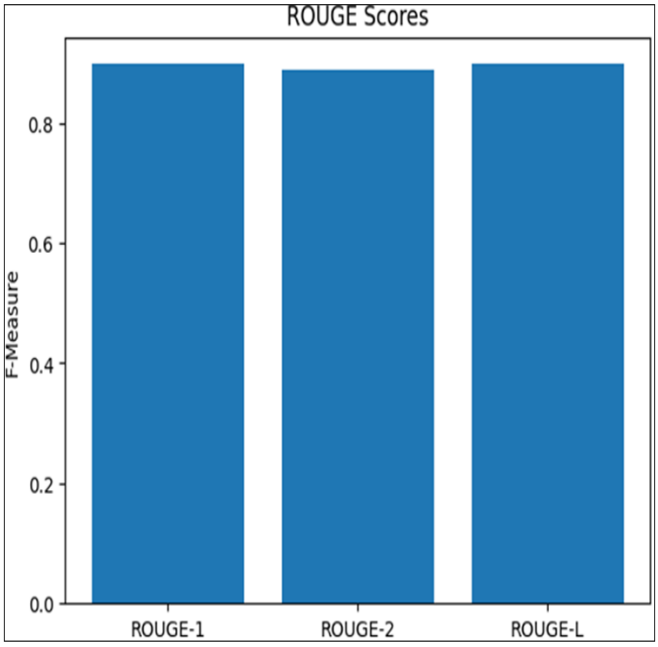

Fig 4 ROUGE Scores

➢ *ROUGE Scores:*

- Rouge1: Score(precision=0.8706896551724138, recall=0.926605504587156, fmeasure=0.8977777777777779)
- Rouge2: Score(precision=0.8608695652173913, recall=0.9166666666666666, fmeasure=0.8878923766816144)
- RougeL: Score(precision=0.8706896551724138, recall=0.926605504587156, fmeasure=0.8977777777777779)

Table 1 ROUGE Scores

Model	Rouge-1 F1 Measure	Rouge-2 F1 Measure	Rouge -3 F1 Measure
BERT	0.507	0.390	0.506
BART	0.897	0.887	0.898

- BART is better than BERT as the F1 scores of BART is higher than BERT.

V. CONCLUSION

In this project, we explored the dynamic field of Natural Language Processing (NLP) with a specific focus on text summarization, a crucial component in handling the vast volume of textual data available on the internet. As the digital world continues to expand, the need for efficient data curation and information retrieval becomes more pronounced. Our project sought to address this need by investigating and implementing advanced techniques for text summarization.

We delved into two prominent models, BART and transformers, which have revolutionized NLP tasks. BART, with its encoder-decoder architecture and token-based processing, offers a versatile solution for summarizing text. On the other hand, transformers, built upon the self-attention mechanism, have demonstrated their prowess in capturing complex relationships and dependencies within text data.

Throughout our exploration, we highlighted the importance of tokenization as a foundational step in NLP, enabling the conversion of raw text into machine-processable units. Token deletion, text infilling, and sentence permutation were discussed as techniques to enhance text summarization.

Furthermore, we introduced supporting technologies such as BeautifulSoup, a Python library for web scraping and parsing HTML, and pyttsx3, a tool for text-to-speech conversion. These tools, when integrated with NLP models, contribute to a comprehensive solution for information extraction and consumption.

In a world inundated with data, our project has underscored the significance of text summarization as a time-saving and information-enhancing tool. It bridges the gap between data overload and efficient knowledge acquisition, benefiting various domains, from academia to industry. As we continue to witness the exponential growth of digital data, the role of NLP in managing this abundance becomes even more crucial.

With this project, we've taken a step towards achieving the goal of effective text summarization, and we anticipate further advancements and applications in this ever-evolving field.

REFERENCES

[1]. Lewis, Mike, et al. "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension." *arXiv preprint arXiv:1910.13461* (2019).
[2]. Shaw, Peter, Jakob Uszkoreit, and Ashish Vaswani. "Self-attention with relative position representations." *arXiv preprint arXiv:1803.02155* (2018).
[3]. Kobayashi, Goro, et al. "Attention is not only a weight: Analyzing transformers with vector norms." *arXiv preprint arXiv:2004.10102* (2020).
[4]. Chen, Pu-Chin, et al. "A simple and effective positional encoding for transformers." *arXiv preprint arXiv:2104.08698* (2021).
[5]. Liu, Liyuan, et al. "Understanding the difficulty of training transformers." *arXiv preprint arXiv:2004.08249* (2020).
[6]. Goyal, Tanya, Junyi Jessy Li, and Greg Durrett. "News summarization and evaluation in the era of gpt-3." *arXiv preprint arXiv:2209.12356* (2022).
[7]. Augenstein, Isabelle, et al. "Stance detection with bidirectional conditional encoding." *arXiv preprint arXiv:1606.05464* (2016).
[8]. Scholak, Torsten, Nathan Schucher, and Dzmitry Bahdanau. "PICARD: Parsing incrementally for constrained auto-regressive decoding from language models." *arXiv preprint arXiv:2109.05093* (2021).
[9]. Awangga, Rolly Maulana, Syafrial Fachri Pane, and Restiyana Dwi Astuti. "Implementation of web scraping on GitHub task monitoring system." *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 17.1 (2019): 275-281.
[10]. Chandra, Rakesh Vidya, and Bala Subrahmanyam Varanasi. *Python requests essentials*. Birmingham, UK: Packt Publishing, 2015.
[11]. Yadav, Avanish Vijaybahadur, Sanket Saheb Verma, and Deepak Dinesh Singh. "Virtual Assistant for blind people." *International Journal* 6.5 (2021).
[12]. Ifeanyi, Nwakanma, Oluigbo Ikenna, and Okpala Izunna. "Text–To–Speech Synthesis (TTS)." *International Journal of Research in Information Technology* 2.5 (2014): 154-163.