SUMMARY GENERATOR WEB APPLICATION

Project report submitted

in partial fulfilment of the requirement for the degree of

Bachelor of Technology

in

Electronics and Telecommunication Engineering

By

Kuladeep Barman (190610026024)

Srinjana Deb (190610026042)

Salik Ahmed (190612826004)

Princi Priya Gogoi (200650026002)

Under the guidance

of

DR. MUKUT SENAPATI

SIDDHANTA BORAH

DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING

ASSAM ENGINEERING COLLEGE

JALUKBARI- 781013, GUWAHATI

June 2023

ASSAM ENGINEERING COLLEGE, GUWAHATI

CERTIFICATE

This is to certify that the thesis entitled "SUMMARY GENERATOR WEB APPLICATION" submitted by KULADEEP BARMAN (190610026024), SRINJANA DEB (190610026042), SALIK AHMED (190612826004), PRINCI PRIYA GOGOI (200650026002) in the partial fulfilment of the requirements for the award of Bachelor of Technology degree in Electronics & Telecommunication at Assam Engineering College, Jalukbari, Guwahati is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Dr. Mukut Senapati

Dept. of Electronics and
Telecomm. Engineering
Assam Engineering College
June, 2023

Siddhanta Borah

Dept. of Electronics and
Telecomm. Engineering
Assam Engineering College
June, 2023

DECLARATION

We, the students pursuing B.Tech at the Department of ETE, Assam Engineering College hereby declare that we have compiled this report reflecting our work during the duration of our final year as a part of our B.Tech programme. We declare that we have included the descriptions etc. of the Project Report and nothing has been replicated from other's work.

We also declare that the same report or any substantial portion of this report has not been submitted anywhere else as part of any requirements for any degree/diploma etc. We understand that any violation of the above will cause disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 26/06/2023

KULADEEP BARMAN	SRINJANA DEB
(190610026024)	(190610026042)
SALIK AHMED	PRINCI PRIYA GOGOI
(190612826004)	(200650026002)

ACKNOWLEDGMENTS

Presentation, inspiration, and motivation have always played a key role in the success of any venture. This project became a reality with the kind support and help of many individuals and we would like to extend our sincere thanks to all of them.

Firstly, we would like to extend our immense and sincere gratitude to the Head of Department of Electronics and Telecommunication Engineering, Dr. Navajit Saikia for giving us the opportunity to work on this project. Secondly, our deepest gratitude goes to our guide Dr. Mukut Senapati, Assistant Professor, Electronics and Telecommunication Engineering, Assam Engineering College, and Siddhanta Borah, Assistant Professor, Electronics and Telecommunication Engineering, Assam Engineering College, for providing us with their valuable insights from time to time that facilitated us to produce a refined work over the period.

Last but not the least, we would like to thank our parents, friends and people who have willingly helped us out in whatever way they can in the completion of this project within the limited time.

KULADEEP BARMAN (190610026024)

SRINJANA DEB (190610026042)

SALIK AHMED (190612826004)

PRINCI PRIYA GOGOI (200650026002)

ABSTRACT

With the dawn of modern civilization, internet has become, more or less, everything in its own right. As the information on the internet is increasing tremendously year after year, text summarization is proving to be a great source to provide relevant information in substantially short spans of time to the readers.

In the last two decades, text summarization has found various applications. In today's world, everything is going digital due to which the amount of information present on the internet has seen an exponential rise, increasing in volume day-by- day. So, users on the internet do not want to spend their time on any information whether it is a document, some paper or any literature without knowing the relevance or importance of that information to their needs. The users want an overview or summarization of those documents before they invest their time in reading the original documents. To overcome this problem, Text Summarization comes into play. Text Summarization is a process of generating a (short) summary of the original document which consists of only important information (i.e. useful and relevant information) in the document.

Natural language Processing gained its attention when humans started to interpret one form of language to machine language. A lot of research works are being carried out in the field of NLP. In this modern era, data retrieval across websites and other informative media are used everywhere irrespective of the languages we speak which made inevitable for having NLP applications like summarization.

Text summarization is a subfield of Natural Language Processing (NLP), and it has been investigated by the NLP community for more than seven decades now. In 1958, the first paper on text summarization was published which gave the first idea regarding extracting the important sentences from a document for summarization purpose, and since then enormous progress has been made in regard of text summarization and its applications. Today, various fields like newspaper publications, game companies, online websites, etc. are implementing automatic text summarization to provide with an overview of the information about the content they offer to their users.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Language is a means of communicating and interacting with others. Language started as abstract symbols to a systematic form of communication. At present, there are more than 7,100 languages spoken across the globe. Language is the institution through which humans communicate and interact with one another using commonly used oral-auditory arbitrary symbols.

As we know that if we need to communicate with a person, we need a specific language. For example, if you need to communicate with two persons, one person knows Hindi and the other person knows Telugu in this case you need to identify the common language between them to communicate with each other. At that time you hear to know that both of them can able to understand and speak in English. In this way, you can communicate with them by using the English language.

Similarly, some specific languages are needed to communicate with computers, and those languages are called programming languages.A programming language is a set of instructions written in a specific language to perform a specific task. It is mainly used to develop desktop applications, operating systems, websites, mobile applications, etc. There are different types of programming languages available which are used for different tasks and one such use case is processing layman language, more specifically called Natural Language[2].

1.1.1 Natural Language Processing

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of Artificial Intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers[6] to process human language in the form of text or voice data and to 'understand' it's full meaning, complete with the speaker or writer's intent and sentiment.

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time. NLP in the form of voice-operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences are present in our day to day life. But NLP also plays a growing role in enterprise solutions that help streamline business operations, increase employee productivity, and simplify mission-critical business processes.

Natural language processing is the driving force behind machine intelligence in many modern real-world applications. Here are a few examples:

- **Spam detection: T**he best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more.

- **Machine translation:** Google Translate is an example of widely available NLP technology at work. Truly useful machine translation involves more than replacing words in one language with words of another. Effective translation has to capture accurately the meaning and tone of the input language and translate it to text with the same meaning and desired impact in the output language.

- **Virtual agents and chatbots:** Virtual agents such as Apple's Siri and Amazon's Alexa use speech recognition to recognize patterns in voice commands and natural language generation to respond with appropriate action or helpful comments. Chatbots perform the same magic in response to typed text entries.

- **Social media sentiment analysis:** NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyze language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events–information companies can use in product designs, advertising campaigns, and more.

- **Text summarization:** Text summarization uses NLP techniques to digest huge volumes of digital text and create summaries and synopses for indexes, research databases, or busy readers who don't have time to read full text. The best text summarization applications use semantic reasoning and natural language generation (NLG) to add useful context and conclusions to summaries[1].

1.2 TEXT SUMMARIZATION

Text summarization is the process of generating short, fluent, and most importantly accurate summary of a respectively longer text document. The main idea behind automatic text summarization is to be able to find a short subset of the most essential information from the entire set and present it in a human-readable format. As online textual data grows, automatic text summarization methods have the potential to be very helpful because more useful information can be read in a short time.

Possible current uses of summarization are:

i. People need to learn much from texts. But they tend to want to spend less time while doingthis.

ii. It aims to solve this problem by supplying them the summaries of the text from which they want to gain information.

iii. The user will be eligible to select the summary length.

iv. Supplying the user, a smooth and clear interface.

v. Configuring a fast replying server system[1][2].

Advantages of Text Summarization:

- **Instantly effective:** It takes time and effort to read the entire article, deconstruct it, and separate the significant concepts from the raw text. It takes at least 15 minutes to read a 500-word article. In a fraction of a second, automatic summary software summarises texts of 500-5000 words. This enables the user to read less data while still getting the most critical information and drawing sound judgments.

- **Language Independent:** Many summarization software can work in any language, which is a capability that most humans lack. Because summarizers are based on linguistic models, they can automatically summarise texts in a wide range of languages, from English to Russian. As a result, they're great for persons who read and work with multilingual information.

- **Productivity is increased:** Not only do some software summarise documents, but they also summarise web pages. This boosts productivity by accelerating the surfing process. Instead of reading entire news stories that are full of irrelevant information, summaries of such websites can be detailed and accurate while yet being only 20% of the original article's size[4].

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

The availability and ease of access of huge amounts of text data on the web presents both an opportunity as well as a challenge. Increased accessibility of data has led to the information overload problem. Advances in AI and machine learning algorithms have enabled humans to employ the technique of text summarization to simplify many tasks. With the exponential growth in the volume of textual content produced through the internet, it has become imperative to use these techniques to extract content which is most relevant to user's information need.

2.1.1 Brief History

Text summarization in early days was done exclusively using rule-based algorithms. It was called "importance evaluator", which worked based on ranking different parts of a text according to their importance. Two important knowledge bases were used by the evaluator: one being the "importance rule base" that made use of IF-THEN rules and other being the "encyclopedia" which contained domain specific world knowledge represented using a network of frames[7].

Another method called "Production rule system for summarization" was used in 1984, which basically works on three steps: (i) Inferencing (ii) scoring the format rows for their importance and finally (iii) selecting the appropriate ones as summary.

As research in the field of summarization technique progressed, a lot of developments were made to understand or interpret importance of sentences in a textual data corpus. One such method is to calculate the relatedness of a piece of text to other texts in the corpus and then decide on the importance by the degree or quantity by which this text is related to other texts.

Text summarization using neural networks was an important development in Natural language processing area. In this method neural network is trained on a corpus of articles and further modified using feature fusion to create a summary with highly ranked sentences in an article. In the training process, the neural network learns about the type of sentences that should be included in a summary. During feature fusion the neural network is pruned and collapses the hidden layer unit activations into discrete values with frequencies. It then generalizes the important features that must be there in the sentences that are going to be part of summary. Finally, the modified neural network selects the sentences that will be part of summary by ranking the sentences[3].

Many developments were happening in parallel. One such approach is diversity-based approach in extractive summarization, which calculates the diversity of the sentences and tries to remove redundant sentences from final summary. In order to find diversity, K-means clustering algorithm extended with Minimum Description Length Principle was used.

There were other attention-based models introduced in text summarization area, helped in creating a better abstractive summarized output. The base of one such model is standard feed forward Network Neural Language Model (NNLM) which is used for estimating the contextual probability of next word, or known as next word prediction model. In that research, it was first used a Bag-of-words encoder base model, which doesn't retain any order or the relation between neighbouring words[6].

2.2 LITERATURE REVIEW

Some recent papers on "Text Summarization" states that:

- *According to Syed Muqtadir Uddin Hussaini, Faraaz Mohd Khan, Faisal Khan, Dr. Abdul Subhan*

  As with time internet is growing at a very fast rate and with it data and information is also increasing; it will going to be difficult for human to summarize large amount of data. Thus there is a need of this technology because of this huge amount of data.

- *“Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, NazliGoharian “*

  Most summarization works in the literature focus on extractive summarization. Abstractive summarization is an alternative approach where the generated summary may contain novel words and phrases and is more similar tohow humans summarize documents.

- *“Elliott Jobson, Abiel Gutiérrez “*, they proposed that :

  The task of document summarization is an open problem in natural language processing. It can be done by a process that fall under a larger deep learning category called sequence-to-sequence models, which map from an input sequence to a target sequence.

- *“N. Moratanch , Dr. S. Chitrakala”,* they stated :

  Abstractive summarization methods produce highly cohesive, coherent, less redundant summary and information rich.

- *“Sunitha. C ,Dr.A.Jaya , Amal Ganesh”* they claimed*:*

  The enormous amount of digital data available which is increasing day by day make the summarization inevitable in most of the application areas like news summarization , summary abstract of emails , scholarly journals , papers etc.

- *“Mukesh Kumar Rohil, Varun Magotra”*claimed :

  Many of the studies have revealed that the use of automatic text summarization in the biomedical and healthcare domain helps researchers and medical professionals save their time and access more information in considerably short spans of time.

- *“Minakshi Tomer , Manoj Kumar”* said:

  Therefore, it is necessary to provide the succinct form of the required information without losing its significance. Thus, reducing the reading time is a desirable prospect as it can greatly decrease human effort as well as help in finding the most important parts of any document or corpus of documents .

CHAPTER 3

METHODOLOGY

3.1. INTRODUCTION

Text summarization using AI methodologies refers to the process of condensing a given piece of text into a shorter, more concise summary using artificial intelligence techniques. AI models can analyze and understand the content of the text to extract the most important information and generate a coherent summary. There are two main approaches to text summarization using AI: extractive summarization and abstractive summarization.

- Extractive Summarization

  Extractive summarization involves identifying and selecting the most relevant sentences or phrases from the original text and assembling them to form a summary. The AI model assesses the importance of each sentence based on various features such as word frequency, sentence position, and semantic similarity to the overall context. Sentences with higher importance scores are included in the summary. This method preserves the original wording but may result in a summary that lacks coherence or contains redundant information.

- Abstractive Summarization

  Abstractive summarization aims to generate a summary by understanding the meaning of the text and generating new sentences that capture the key information. AI models for abstractive summarization employ natural language processing techniques, including deep learning and language generation models such as recurrent neural networks (RNNs) or transformer-based models like GPT. These models learn from large amounts of text data and can generate human-like summaries that go beyond the original text. Abstractive summarization can produce more coherent and concise summaries but may introduce some level of interpretation or paraphrasing.

Both extractive and abstractive summarization methods have their advantages and limitations. Extractive methods are often computationally efficient and maintain the original context but may lack the ability to generate novel sentences. Abstractive methods, on the other hand, can generate more fluent and concise summaries but may struggle with generating accurate and coherent summaries for complex texts.

AI methodologies for text summarization require training on large datasets of human-written summaries and their corresponding source documents. These models can then be fine-tuned on specific domains or tasks to improve their performance. By leveraging AI techniques, text summarization can be automated, enabling efficient processing of large amounts of information and assisting users in quickly understanding the main points of a document or article.

3.2. EXTRACTIVE TEXT SUMMARIZATION

Extractive text summarization is a technique that involves selecting and assembling the most relevant sentences or phrases from a given text to create a summary. Instead of generating new sentences, it extracts the most important information directly from the source text. Here's a general outline of the process:

i. Text Preprocessing

   The original text undergoes preprocessing steps such as sentence tokenization, word tokenization, and removing stop words (commonly used words with little semantic value).

ii. Sentence Scoring

   Each sentence in the text is assigned a score based on its importance or relevance to the overall meaning of the document. Various methods can be employed to calculate these scores, including:

- *Term Frequency-Inverse Document Frequency (TF-IDF):* Assessing the significance of each word in a sentence based on its frequency in the document and rarity in the overall corpus.

- *Text Rank Algorithm:* Adapting the PageRank algorithm used in web search to determine the importance of a sentence based on the number and quality of other sentences it is connected to.

iii. Sentence Selection

Sentences with the highest scores are selected to be part of the summary. The number of sentences to include can be predefined or determined dynamically based on the desired summary length.

iv. Summary Generation

The selected sentences are combined to form the final summary. Depending on the requirements, additional post-processing steps such as reordering sentences or ensuring coherence can be applied.

Extractive summarization methods have the advantage of preserving the wording and context of the original text. However, they may result in summaries that contain redundant information or lack coherence since they are composed of disjointed sentences from the source.It's important to note that extractive summarization focuses on sentence selection rather than sentence generation. This approach is computationally efficient and can be used for summarizing large volumes of text quickly. However, it may not capture the entirety of the source text or provide a concise[4] summary that captures the essence of the document as effectively as abstractive summarization methods.

3.2.1. Algorithm for extractive summarization

Step (1): First split the paragraph into n correspondence sentences.

Step (2): Text Processing

Step (3): Tokenization

Step (4): Evaluate the weighted occurrence of frequency of the word

Step (5): Substitute the word with their weighted frequency

Let us take an example paragraph:

Peter and Elizabeth took a taxi to attend the night party in the city. while in the party, Elizabeth collapsed and was rushed to the hospital. Since she was diagnosed with a brain injury, the doctor told Peter to stay beside her until she gets well. Peter stayed with her at the hospital for 3 days without leaving.

Step 1: Break the paragraph into sentences.

1: "Peter and Elizabeth took a taxi to attend the night party in the city."

2: "While in the party, Elizabeth collapsed and was rushed to the hospital."

3: "Since she was diagnosed with a brain injury, the doctor told Peter to stay beside her until she gets well."

4: "Peter stayed with her at the hospital for 3 days without leaving

Step 2: Text processing

Text processing is done by removing extremely common words.

Result of the text processing:

1: Peter Elizabeth took taxi attend night party city.

2: party Elizabeth collapsed rush hospital.

3. diagnose brain injury doctor told peter stay beside get well

4. Peter stayed hospital without leaving.

Step 3: Tokenization

Tokenization is done to get all the words present in the sentences.

["Peter”, " Elizabeth" , "took"..............., "leaving]

Step 4: Evaluating the weighted occurrence

TABLE 3.2.1.1: Tabulating the weighted occurrence

Word	Frequency	Weighted frequency
Peter	3	1
Elizabeth	2	0.67
took	1	0.33
taxi	1	0.33
Attend	1	0.33
Night	1	0.33
Party	2	0.67
City	1	0.33
Collapse	1	0.33
Rush	1	0.33

Step 5:

TABLE 3.2.1.2: Evaluation of the weighted occurrence

1. Peter and Elizabeth took a taxi to attend the night party in the city.	1+0.67+0.33+0.33+0.33+0.67+0.33=3.99
2. while in the party, Elizabeth collapsed and was rushed to the hospital.	0.67+0.67+0.33+0.33+067=2.67
3. since she was diagonised with a brain injury ,the doctor tolds peter to stay besides her while she gets well.	3.97
4. therefore peter stayed with her for three days without leaving.	3.33

So the first sentence carries the most weight in the paragraph. Therefore it can give the best representation of the paragraph. If the first sentence is combined with the third sentence in the paragraph then a better summary can be generated. This is the basic illustration of how to generate an extraction-based text summarization.

3.3. ABSTRACTIVE TEXT SUMMARISATION

In our project we use the transformer based model.Abstractive text summarization using Transformer models has been a significant advancement in the field. Transformers, such as the widely used model known as GPT (Generative Pre-trained Transformer), have demonstrated impressive performance in generating coherent and human-like summaries. Here's an overview of the process:

i. Pre-training the Transformer Model:

Transformer models are pretrained on large amounts of text data to learn the patterns and structures of language. This pretraining phase involves predicting missing words in a sentence or generating the next word in a sequence, enabling the model to capture the semantics and grammar of the text.

ii. Fine-tuning for Summarization

After pre-training, the Transformer model is further fine-tuned specifically for abstractive summarization. This involves training the model on a dataset consisting of source documents and their corresponding human-written summaries. During fine-tuning, the model learns to generate concise and coherent summaries by understanding the relationship between the source text and its summary.

iii. Encoding the Source Text

When given a source document for summarization, the input text is tokenized and encoded into numerical representations that the Transformer model can process. The encoding captures the contextual information of the text, allowing the model to understand the dependencies between different words and sentences[5].

iv. Decoding the Summary

The Transformer model uses the encoded source text to generate a summary. The decoding process involves generating each word or phrase in the summary one by one, considering the context and previously generated words. This step includes both language generation and understanding the semantic meaning of the text to produce a concise and coherent summary.

v. Post-processing

The generated summary may undergo post-processing steps to improve its coherence, remove redundancies, or adjust the length to match desired constraints.

Abstractive summarization using Transformer models offers the advantage of generating summaries that are not limited to sentences present in the source text. The models can generate new sentences that convey the key information from the original text while maintaining coherence. However, they may sometimes introduce slight paraphrasing or reordering of information compared to the source text.

It's worth noting that training and fine-tuning Transformer models for abstractive summarization typically require large amounts of annotated data and significant computational resources. Additionally, careful evaluation and refinement are necessary to ensure the generated summaries are accurate, informative, and free of biases or factual errors.

3.3.1. Algorithm for abstractive summarization

i. **Preprocessing**: This step involves cleaning the input text by removing unnecessary characters, punctuation, and extra white spaces. The purpose is to standardize the text and remove any noise that might hinder the summarization process. Additionally, converting the text to lowercase ensures consistency in the subsequent steps.

ii. **Sentence Tokenization**: Sentence tokenization is the process of splitting the text into individual sentences. It allows the algorithm to analyze and process the text at a more granular level. Tokenization can be done using pre-trained models or libraries like NLTK or SpaCy, which provide reliable sentence tokenization capabilities.

iii. **Text Representation**: In this step, each sentence is transformed into a numerical representation that captures its semantic meaning. This representation is necessary for the subsequent steps of the algorithm. Commonly used techniques include word embeddings such as Word2Vec or GloVe. These embeddings convert each word in the sentence into a dense vector representation. The word vectors are then combined to obtain a sentence vector, which represents the semantic meaning of the sentence.

iv. **Sequence-to-Sequence Model**: A sequence-to-sequence model is a neural network architecture consisting of an encoder and a decoder. This model is trained to learn the mapping between the input text and the desired summary. The encoder processes the input text, capturing its context and semantic representation. The decoder generates the summary by predicting the next word or phrase based on the previously generated words.

   a. **Encoder**: The encoder network takes in the encoded representations of the input sentences and processes them. The encoder's task is to capture the context and semantic information of the text, condensing it into a fixed-size encoded representation. This representation serves as the input for the decoder.

   b. **Decoder**: The decoder network takes the encoded representation from the encoder and generates the summary. It is trained to predict the next word or phrase based on the encoded representation and the previously generated words in the summary. The decoder generates the summary sequentially, word by word, until a stopping criterion is met.

v. **Training**: To train the sequence-to-sequence model, a dataset is needed that contains pairs of input texts and their corresponding summaries. The dataset should be labeled or generated using human-written summaries. During training, the model learns to map the input text to the summary by optimizing its parameters using techniques like back propagation and gradient descent. This process involves minimizing a loss function that measures the discrepancy between the predicted summary and the human-written summary.

vi. **Inference**: Once the model is trained, it can be used to generate summaries for new, unseen texts. The inference process involves the following steps:

   a. **Encoding**: The input text is encoded using the trained encoder network. The encoder processes the input text and produces an encoded representation that captures its context and semantic information.

   b. **Decoding**: The decoder is initialized with the encoded representation from the encoder. It generates words or phrases sequentially to form the summary. During decoding, the model predicts the next word or phrase based on the encoded representation and the previously generated words. This process can be done using techniques like beam search or sampling to explore different possible sequences of words.

   c. **Stopping Criterion**: A stopping criterion is necessary to determine when to end the generation process. This criterion can be a maximum length limit for the summary or a predefined end-of-sentence token. It ensures that the generated summary is of a reasonable length.

vii. **Post-processing**: After generating the summary, post-processing steps are applied to refine it. This involves removing any redundant or irrelevant information from the generated summary. Redundant information may occur due to the repetition of words or phrases. Additionally, proper punctuation and formatting are added to creates well-formed summary that is readable and coherent.

It's important to note that abstractive text summarization is a challenging task, and achieving high-quality summaries can be complex. The quality of the generated summaries depends on various factors such as the size and quality of the training dataset, the architecture of the sequence-to-sequence model, and the techniques used for decoding and post-processing. Fine-tuning the model, incorporating attention mechanisms to focus on important parts of the input text, or using more advanced architectures like transformer-based models (e.g., BART or T5) can help improve the quality of the generated summaries.

3.4. BLEU METRIC

BLEU (Bilingual Evaluation Understudy) is a metric commonly used to evaluate the quality of machine-generated translations or summaries by comparing them to one or more reference translations or summaries. BLEU measures the similarity between the machine-generated output and the human-generated reference texts. The closer the BLEU score is to 1, the better the machine-generated output matches the reference texts[1].

BLEU calculates precision by counting the n-gram matches (contiguous sequences of n words) between the machine-generated output and the reference texts. It also considers the brevity penalty to avoid favouring overly short translations or summaries. The formula for BLEU is as follows:

$$\text{BLEU} = \text{BP} * \exp\,(\textstyle\sum(n=1 \text{ to } N)\ w\_n * \log(p\_n))$$

where

- BP (Brevity Penalty) penalizes the generated output if it is shorter than the reference texts to discourage excessively short outputs.
- N is the maximum n-gram size considered.
- p_n is the modified precision for n-grams, which is the ratio of matching n-grams between the generated output and the reference texts to the total number of n-grams in the generated output.

- w_n is the weight assigned to each n-gram precision. Typically, equal weights (1/N) are used for n-grams of different lengths.

The BLEU score ranges from 0 to 1, with 1 being a perfect match to the reference texts. However, it's important to note that BLEU is just one of many evaluation metrics and should be used in conjunction with other metrics and human evaluation to get a comprehensive understanding of the quality of the generated summerization.

3.5 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of evaluation metrics commonly used for assessing the quality of automatic text summarization systems. ROUGE measures the overlap between the generated summaries and the reference summaries created by humans. It calculates various metrics based on the number of overlapping n-grams (contiguous sequences of words) and the longest common subsequence (LCS) between the generated and reference summaries.

There are several variants of ROUGE metrics, including ROUGE-N, ROUGE-L, and ROUGE-S. Here's a brief explanation of each:

- ROUGE-N: This metric measures the overlap of n-grams between the generated and reference summaries. It calculates precision, recall, and F1-score based on the number of shared n-grams.
- ROUGE-L: This metric computes the LCS (longest common subsequence) between the generated and reference summaries. It considers the length of the LCS as the matching criterion and calculates precision, recall, and F1-score based on this measure.
- ROUGE-S: This metric focuses on skip-bigram (2-gram with a maximum skip of 4) matches between the generated and reference summaries. It calculates precision, recall, and F1-score based on the number of skip-bigram matches.

Each ROUGE metric provides an evaluation score that indicates the quality of the summarization output. The scores typically range from 0 to 1, with higher values indicating better performance[2][4].

ROUGE scores provide quantitative measures to assess the quality of summarization systems and compare different approaches or models. They can be useful in evaluating and benchmarking the performance of automatic text summarization algorithms.

CHAPTER 4

RESULT ANALYSIS

4.1. INTRODUCTION

Abstractive summarization has applications in various domains, including news summarization, document summarization, social media summarization, and automatic summarization of research articles. It enables users to quickly grasp the key points of a lengthy text, facilitates information retrieval, and supports tasks like document clustering and recommendation systems.

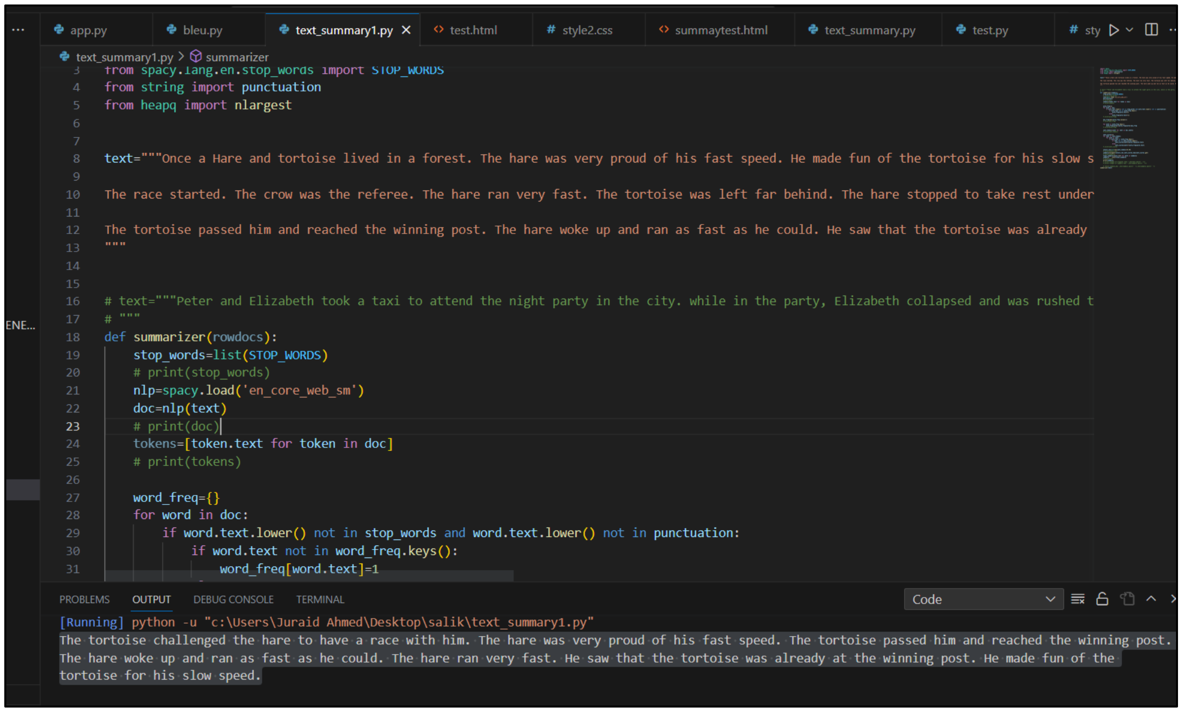
As research and advancements in NLP continue, abstractive summarization techniques are expected to improve, leading to more accurate, coherent, and informative summaries that assist users in efficiently digesting and understanding large volumes of text.

4.2 . RESULTS

A) For Extractive Summarization

Input: "Once a Hare and tortoise lived in a forest. The hare was very proud of his fast speed. He made fun of the tortoise for his slow speed. The tortoise challenged the hare to have a race with him. The hare accepted the challenge.The race started. The crow was the referee. The hare ran very fast. The tortoise was left far behind. The hare stopped to take rest under a tree. He fell asleep. The tortoise passed him and reached the winning post. The hare woke up and ran as fast as he could. He saw that the tortoise was already at the winning post. He won the race"

Summary generated: "The tortoise challenged the hare to have a race with him. The hare was very proud of his fast speed. The tortoise passed him and reached the winning post. The hare woke up and ran as fast as he could. The hare ran very fast. He saw that the tortoise was already at the winning post. He made fun of the tortoise for his slow speed."



```python
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
from heapq import nlargest


text="""Once a Hare and tortoise lived in a forest. The hare was very proud of his fast speed. He made fun of the tortoise for his slow s

The race started. The crow was the referee. The hare ran very fast. The tortoise was left far behind. The hare stopped to take rest under

The tortoise passed him and reached the winning post. The hare woke up and ran as fast as he could. He saw that the tortoise was already
"""


# text="""Peter and Elizabeth took a taxi to attend the night party in the city. while in the party, Elizabeth collapsed and was rushed t
# """
def summarizer(rowdocs):
    stop_words=list(STOP_WORDS)
    # print(stop_words)
    nlp=spacy.load('en_core_web_sm')
    doc=nlp(text)
    # print(doc)
    tokens=[token.text for token in doc]
    # print(tokens)

    word_freq={}
    for word in doc:
        if word.text.lower() not in stop_words and word.text.lower() not in punctuation:
            if word.text not in word_freq.keys():
                word_freq[word.text]=1
```

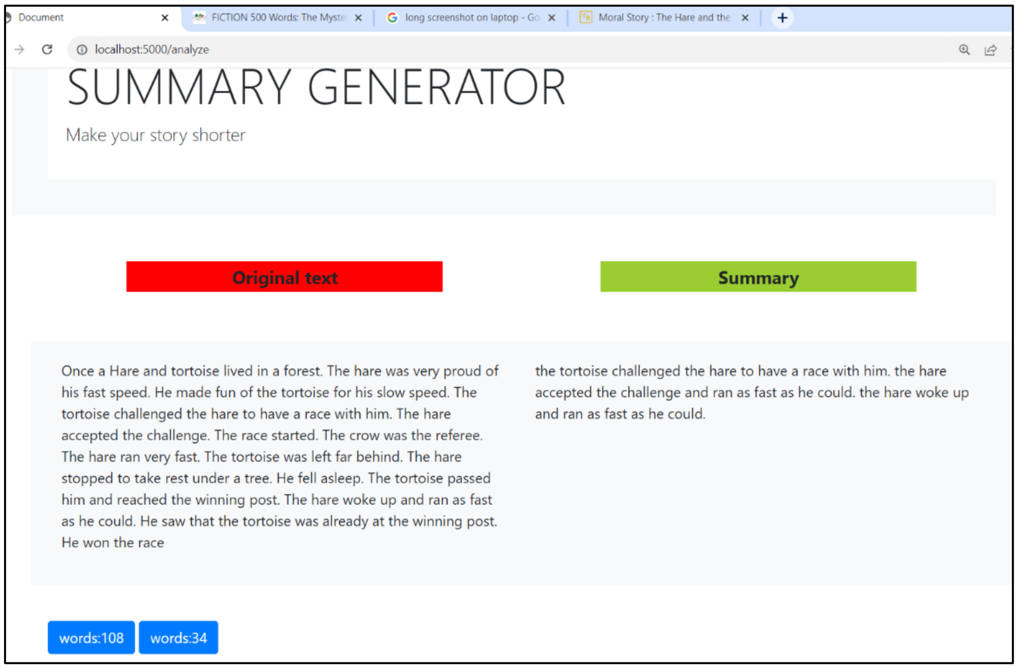
```
[Running] python -u "c:\Users\Juraid Ahmed\Desktop\salik\text_summary1.py"
The tortoise challenged the hare to have a race with him. The hare was very proud of his fast speed. The tortoise passed him and reached the winning post.
The hare woke up and ran as fast as he could. The hare ran very fast. He saw that the tortoise was already at the winning post. He made fun of the
tortoise for his slow speed.
```

Fig 4.2.1: Screenshot of extractive summarization result

B) For Abstractive summarization

Input: “Once a Hare and tortoise lived in a forest. The hare was very proud of his fast speed. He made fun of the tortoise for his slow speed. The tortoise challenged the hare to have a race with him. The hare accepted the challenge.The race started. The crow was the referee. The hare ran very fast. The tortoise was left far behind. The hare stopped to take rest under a tree. He fell asleep. The tortoise passed him and reached the winning post. The hare woke up and ran as fast as he could. He saw that the tortoise was already at the winning post. He won the race”

Generated summary: “the tortoise challenged the hare to have a race with him. the hare accepted the challenge and ran as fast as he could. the hare woke up and ran as fast as he could.”



Once a Hare and tortoise lived in a forest. The hare was very proud of his fast speed. He made fun of the tortoise for his slow speed. The tortoise challenged the hare to have a race with him. The hare accepted the challenge. The race started. The crow was the referee. The hare ran very fast. The tortoise was left far behind. The hare stopped to take rest under a tree. He fell asleep. The tortoise passed him and reached the winning post. The hare woke up and ran as fast as he could. He saw that the tortoise was already at the winning post. He won the race

the tortoise challenged the hare to have a race with him. the hare accepted the challenge and ran as fast as he could. the hare woke up and ran as fast as he could.

Fig 4.2.2: Screenshot of abstractive summarization result

4.3 ROUGE Metrics

A) For extractive summarization

The output provided is a dictionary containing ROUGE scores for different metrics. Each metric has its own sub-dictionary with precision (p), recall (r), and F1-score (f) values. Here's a breakdown of the information:

rouge-1: This metric represents ROUGE-1, which measures the overlap of unigrams (single words) between the generated and reference summaries.

r: Recall value for ROUGE-1. In this case, it is 1.0, indicating that all the unigrams in the reference summary are present in the generated summary.

p: Precision value for ROUGE-1. It is 0.2, suggesting that only 20% of the unigrams in the generated summary are present in the reference summary.

f: F1-score for ROUGE-1. It is approximately 0.3333, calculated based on precision and recall.

rouge-2: This metric represents ROUGE-2, which measures the overlap of bigrams (pairs of words) between the generated and reference summaries.

r: Recall value for ROUGE-2. It is 0.0, indicating that none of the bigrams in the reference summary are present in the generated summary.

p: Precision value for ROUGE-2. It is also 0.0, suggesting that none of the bigrams in the generated summary are present in the reference summary.

f: F1-score for ROUGE-2. Since both recall and precision are 0.0, the F1-score is also 0.0.

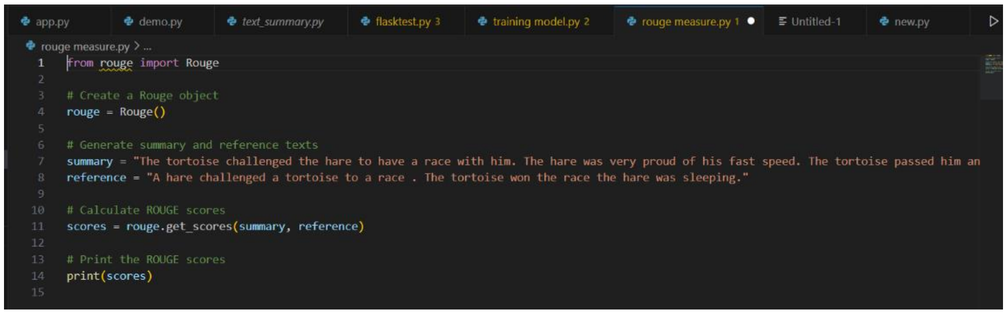
rouge-l: This metric corresponds to ROUGE-L, which calculates the longest common subsequence (LCS) between the generated and reference summaries.

r: Recall value for ROUGE-L. It is 1.0, indicating that the generated summary contains the entire reference summary.

p: Precision value for ROUGE-L. Here, it is 0.2, suggesting that only 20% of the generated summary is part of the reference summary.

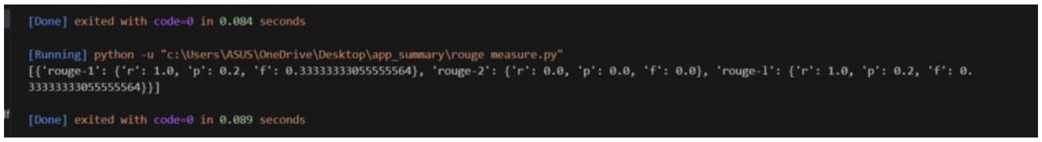
f: F1-score for ROUGE-L. Similar to ROUGE-1, it is approximately 0.3333.

These scores provide an evaluation of the generated summary based on the overlap and similarity with the reference summary. Keep in mind that the values provided are specific to the example or evaluation performed and may vary depending on the specific text and the evaluation criteria used[3].



```python
from rouge import Rouge

# Create a Rouge object
rouge = Rouge()

# Generate summary and reference texts
summary = "The tortoise challenged the hare to have a race with him. The hare was very proud of his fast speed. The tortoise passed him an
reference = "A hare challenged a tortoise to a race . The tortoise won the race the hare was sleeping."

# Calculate ROUGE scores
scores = rouge.get_scores(summary, reference)

# Print the ROUGE scores
print(scores)

```

Fig 4.3.1: Calculation of ROUGE value for extractive model



```
[Done] exited with code=0 in 0.084 seconds

[Running] python -u "c:\Users\ASUS\OneDrive\Desktop\app_summary\rouge measure.py"
[{'rouge-1': {'r': 1.0, 'p': 0.2, 'f': 0.33333333055555564}, 'rouge-2': {'r': 0.0, 'p': 0.0, 'f': 0.0}, 'rouge-l': {'r': 1.0, 'p': 0.2, 'f': 0.
33333333055555564}}]

[Done] exited with code=0 in 0.089 seconds
```

Fig 4.3.2: Output of Extractive Rouge calculation

B) For abstractive summarization

The output provided is a dictionary containing ROUGE scores for different metrics. Each metric has its own sub-dictionary with precision (p), recall (r), and F1-score (f) values. Here's a breakdown of the information:

rouge-1: This metric represents ROUGE-1, which measures the overlap of unigrams (single words) between the generated and reference summaries.

r: Recall value for ROUGE-1. In this case, it is approximately 0.5833, indicating that about 58.33% of the unigrams in the reference summary are present in the generated summary.

p: Precision value for ROUGE-1. It is approximately 0.35, suggesting that about 35% of the unigrams in the generated summary are present in the reference summary.

f: F1-score for ROUGE-1. It is approximately 0.4375, calculated based on precision and recall.

rouge-2: This metric represents ROUGE-2, which measures the overlap of bigrams (pairs of words) between the generated and reference summaries.

r: Recall value for ROUGE-2. It is approximately 0.125, indicating that about 12.5% of the bigrams in the reference summary are present in the generated summary.

p: Precision value for ROUGE-2. It is approximately 0.08, suggesting that about 8% of the bigrams in the generated summary are present in the reference summary.

f: F1-score for ROUGE-2. It is approximately 0.0976, calculated based on precision and recall.

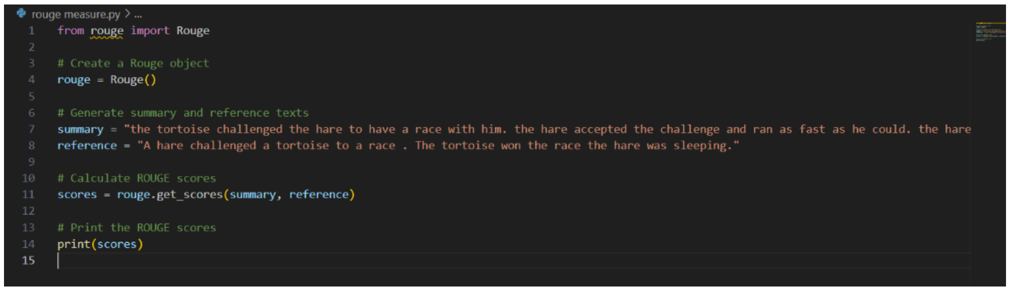
rouge-l: This metric corresponds to ROUGE-L, which calculates the longest common subsequence (LCS) between the generated and reference summaries.

r: Recall value for ROUGE-L. It is approximately 0.5, indicating that the generated summary covers about 50% of the reference summary.

p: Precision value for ROUGE-L. Here, it is approximately 0.3, suggesting that about 30% of the generated summary is part of the reference summary.

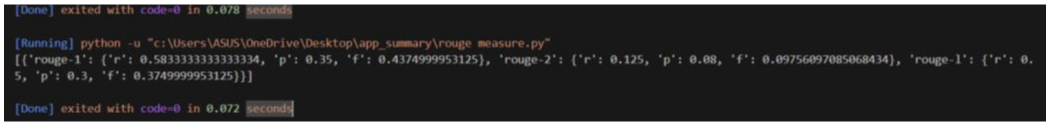
f: F1-score for ROUGE-L. Similar to ROUGE-1, it is approximately 0.375, calculated based on precision and recall[3].

These scores provide an evaluation of the generated summary based on the overlap and similarity with the reference summary. Keep in mind that the values provided are specific to the example or evaluation performed and may vary depending on the specific text and the evaluation criteria used.



```
rouge measure.py > ...
from rouge import Rouge

# Create a Rouge object
rouge = Rouge()

# Generate summary and reference texts
summary = "the tortoise challenged the hare to have a race with him. the hare accepted the challenge and ran as fast as he could. the hare
reference = "A hare challenged a tortoise to a race . The tortoise won the race the hare was sleeping."

# Calculate ROUGE scores
scores = rouge.get_scores(summary, reference)

# Print the ROUGE scores
print(scores)

```

Fig 4.3.3: Calculation of ROUGE value for abstractive model



```
[Done] exited with code=0 in 0.078 seconds

[Running] python -u "c:\Users\ASUS\OneDrive\Desktop\app_summary\rouge measure.py"
[{'rouge-1': {'r': 0.5833333333333334, 'p': 0.35, 'f': 0.4374999953125}, 'rouge-2': {'r': 0.125, 'p': 0.08, 'f': 0.09756097085068434}, 'rouge-l': {'r': 0.
5, 'p': 0.3, 'f': 0.3749999953125}}]

[Done] exited with code=0 in 0.072 seconds
```

Fig 4.3.4: Output of Abstractive Rouge calculation

CHAPTER 5

CONCLUSION AND FUTURE SCOPE OF WORK

5.1. CONCLUSION

The evaluation of automatic summarization is still an open issue. Recent automatic methods have proven more consistent that human based evaluation. However, the lack of consensus between humans when evaluating summaries is a real problem. The development of more focused summaries may lead to a more consistent evaluation and to a better convergence between human and automatic evaluation methods, which are highly desirable. However, automatic summarization evaluation is still a very promising research area with many challenges ahead.

While our results are encouraging, there is still much room for improvement for this challenging task; our new datasets can help the community to further explore this problem.

5.2. FUTURE SCOPE OF WORK

The most effective and versatile methods used so far in automatic summarization rely one extractive methods: they aim at selecting the most relevant sentences from the collection of original documents in order to produce a condensed text rendering important pieces of information. As we have seen, these kinds of methods are far from being ideal: in multi-document summarization, the selection of sentences from different documents leads to redundancy, which in turn must be eliminated. Moreover, most of the time only a part of a sentence is relevant, but extracting only sub-sentences is still far from being operational. Lastly, extracting sentences from different documents may produce an inconsistent and/or hard-to-read summary.

These limitations suggest a number of desirable improvements. We detail here three very active research trends.

i. In order to overcome the redundancy problem, researchers are actively working on a better representation of the text content and, more interestingly, are now trying to provide summaries tailored towards specific user needs.

ii. One of the main drawback of extractive methods is often the lack of readability of the text produced. In most systems, sentence ordering is based on simple heuristics (e.g. location of the sentence in the original documents) that are not enough to produce a coherent text. Recent research aims at finding new methods for producing more coherent texts.

iii. Can cause problems if the text is multi-lingual.

References

[1] A. T. Al-Taani, "Automatic text summarization approaches," 2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS), Dubai, United Arab Emirates, 2017, pp. 93-94, doi: 10.1109/ICTUS.2017.8285983.

[2] Neelima Bhatia and Arunima Jaiswal. Article: Literature Review on Automatic Text Summarization: Single and Multiple Summarizations. *International Journal of Computer Applications* 117(6):25-29, May 2015.

[3] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez and Krys Kochut, "Text Summarization Techniques: A Brief Survey" *International Journal of Advanced Computer Science and Applications(IJACSA),* 8(10), 2017. Available: http://dx.doi.org/10.14569/IJACSA.2017.081052

[4] Pankaj Gupta, Ritu Tiwari and NirmalRobert,"Sentiment Analysis and Text Summarization of Online Reviews: A Survey "International Conzatiference on Communication and Signal Processing, August 2013

[5] Vishalgupta,Gurpreet Singh Lehal, "A Survey of Text Summarization Extractive Techniques." *JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE*, VOL. 2, NO. 3, AUGUST 2010

[6] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive Document Summarization with a Graph-Based Attentional Neural Model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181, Vancouver, Canada. Association for Computational Linguistics.

[7] Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural c Language Learning, pages 256–265, Jeju Island, Korea, 12–14 July 2012. 2012 Association for Computational Linguistics