6CCS3PRJ Individual Undergraduate Project in Computer Science

Final Report

Educational chatbot – CliffsNotes bot

Author: Mustafa Gulzar
Student ID: 20043162
Supervisor: Dr. Jeffery Raphael

April 5, 2022

Abstract

This report details the development of an educational chatbot that incorporates transformer models and NLP techniques to generate summaries and question-answer pairs for a given book chapter. The project aims to benchmark various technologies and test them on the first chapter of "Harry Potter and the Philosopher's Stone" [32]. The chatbot will utilize these technologies to improve its ability to answer user inquiries about the text.

The report delves into the various approaches considered in the project and discusses the design, implementation, and subsequent analysis of the chatbot. The results of the analysis are presented, evaluating the effectiveness of these techniques in enhancing the chatbot's accuracy and relevance of responses.

By utilizing transformer models and NLP techniques, the chatbot can generate summaries and question-answer pairs that accurately capture the key information and nuances of the chapter. The report outlines the process of incorporating these technologies into the chatbot and discusses the potential implications of this project for educational purposes.

Originality Avowal

Acknowledgments

I want to thank Dr. Jeffrey Raphael, my adviser, for always responding enthusiastically to my ideas and for providing his support, advice, and knowledge during the development of this project.

Contents

Chapter 1: Introduction

1.1 Project Aims

The integration of technology in the field of education has led to various innovative approaches to enhancing the learning experience, and chatbots have been gaining popularity in recent years as a result. Chatbots are computer programs designed to simulate human conversation, providing a platform for personalized and efficient interactions [1]. While chatbots have been utilized in various industries such as customer service and public sector services, their applications in education have been growing rapidly.

Research on chatbots in education has highlighted their potential to provide students with round-the-clock access to educational resources and personalized support [2]. One of the key areas of exploration is the development of chatbots that can generate both extractive and abstractive summaries of academic texts, as well as question-answer pairs. Extractive summaries provide a condensed version of the original text by selecting the most relevant sentences, while abstractive summaries use natural language processing to generate new sentences that capture the essence of the text [3]. Such chatbots can provide students with quick and concise summaries of academic texts and answers to specific questions, improving their understanding and learning outcomes. Recent studies have also explored the potential of chatbots to personalize the learning experience for individual students. By analyzing student behavior, preferences, and academic performance, chatbots can offer tailored support and recommendations, improving student engagement and academic performance [4]. Additionally, chatbots can handle a large volume of questions and requests simultaneously, making them ideal for use in online learning environments and reducing the workload of teachers and educational professionals.

Motivated by these potential benefits, this project aims to develop an educational chatbot that generates both extractive and abstractive summaries of academic texts and question-answer pairs. The Python Chatterbot package will be used to build the chatbot utilising machine learning and natural language processing methods. The chatbot's main goal is to give students rapid, easy access to educational materials so they can better grasp academic texts and improve their learning outcomes. Additionally, the chatbot will customise each student's educational experience by giving them specialised advice and support to raise their level of engagement and academic success.

1.2 Report Structure

The report commences by providing an introduction to the topics pertinent to the project, accompanied by a review of the existing literature on related technical functionality. Subsequently, it presents the software requirements, specifications, and design.

Following the design phase, the report will discuss the implementation of the software chronologically. Throughout this chapter, distinct software versions will be highlighted as milestones, and any issues encountered during development will be addressed, along with the solutions utilized to overcome them. Additionally, this chapter will elaborate on how the fully featured software was optimized to enhance performance when operating with a larger corpus or training set.

The subsequent major chapter will delve into the experimental stage of the project, wherein data collected through the software will be presented and any emerging trends will be analysed. This will be followed by a brief chapter that considers the ethical and professional factors that were taken into account during the project's creation.

Next, the project outcomes will be evaluated to determine the strengths and weaknesses of the software and experimentation. A conclusion chapter will then summarise the key findings and discuss any potential future work that can be conducted to build upon the outcomes of the project.

Chapter 2: Background

2.1 CliffsNotes

CliffsNotes is a series of study guides, created in 1958 by Clifton Hillegass, a Nebraska bookworm. The guides are made to help students comprehend and straightforwardly analyse complex material, and they cover a wide range of subjects, including mathematics, literature, and history [12]. Each study guide includes a synopsis and analysis of the primary text, character and topic analyses, and a section on review questions and solutions. CliffsNotes were created as a way for students to study for literature exams, but have since evolved to include study guides for other subjects and test preparation materials. The popularity of CliffsNotes has made them a staple in the education industry, and they have become synonymous with quick and easy study aids. Despite some criticism that they encourage students to avoid reading the original text, CliffsNotes remain a valuable resource for students looking to supplement their understanding of difficult material or study for exams [13].

2.2 Machine Learning

2.2.1 PEGASUS

The paper "PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization" by Zhang et al. (2020) [9] proposes a novel approach for pre-training abstractive summarization models using gap-sentences. Abstractive summarization is a challenging task in natural language processing, where a system generates a summary that captures the key information of a source text in a condensed form. PEGASUS builds on the Transformer architecture and extends pre-training using a gap-sentence generation approach. The authors claim that PEGASUS outperforms existing state-of-the-art abstractive summarization models on various datasets.

The paper starts by providing a comprehensive review of related work in abstractive summarization and pre-training for language models. The authors then present PEGASUS, which is based on the self-supervised task of gap-sentence generation, where a sentence from the source text is removed, and the model is tasked with predicting the missing sentence. The gap sentences are extracted from the source text using an unsupervised approach that considers sentence salience, position, and length. The extracted gap sentences serve as input to pre-train the Transformer model, which is then fine-tuned on downstream summarization tasks.

The authors evaluate PEGASUS on various datasets, including CNN/DailyMail, Gigaword, and Wikihow. PEGASUS outperforms state-of-the-art models on all datasets, achieving a new state-of-the-art on Gigaword and Wikihow. The authors also conduct an extensive ablation study to investigate the impact of different components of PEGASUS. The results show that gap-sentence generation and unsupervised sentence extraction are key to the success of PEGASUS.

Overall, the paper "PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization" presents a novel approach to pre-training abstractive summarization models that outperform existing state-of-the-art models. The paper is well-organized, with a comprehensive literature review and detailed experiments. The proposed method of gap-sentence generation and unsupervised sentence extraction is promising and could be extended to other natural language processing tasks. However, the evaluation is limited to a few datasets, and it remains to be seen how PEGASUS performs on other datasets and languages.

2.2.2 BART

The BART (Bidirectional and Auto-Regressive Transformer) model is a pre-trained language generation model designed to improve natural language processing tasks such as summarization, translation, and comprehension. It is based on the transformer architecture used in the popular GPT (Generative Pre-trained Transformer) model [8], but also incorporates denoising autoencoder objectives, which allow it to perform well in unsupervised settings. This paper presents the BART model and evaluates its performance across a range of natural language processing tasks, demonstrating its effectiveness in a variety of settings.

The BART model utilizes a denoising autoencoder approach, which is trained on the corrupted text that is then reconstructed to produce a clean version of the original text. This approach is particularly effective for natural language processing tasks such as summarization and translation, where the input text is often incomplete or noisy. The model is pre-trained on a large corpus of text using a multi-task learning objective that combines various natural language processing tasks, including masked language modeling and sequence classification. This pre-training enables the model to learn general language features that can be fine-tuned for specific downstream tasks.

In the evaluation of the BART model, it was found to perform well across a range of natural language processing tasks, including machine translation, summarization, and question-answering. In particular, it outperformed existing state-of-the-art models on several benchmarks, including the CNN/Daily Mail summarization task and the WMT14 English-German machine translation task. The results demonstrate the effectiveness of the denoising autoencoder approach and the multi-task learning objective used in pre-training the model.

The BART model, as a whole, represents a significant development in natural language processing and is likely to find use in a variety of contexts. The denoising autoencoder approach's success is shown by its capacity to perform well across a range of tasks, and this suggests that this approach might be beneficial for problems other than those examined in this paper that includes natural language processing. The BART model offers a solid framework for further study into language production and comprehension and is anticipated to have a big impact on the discipline in the years to come.

2.2.3 Fine-Tuning

Liu et al. (2021) present a comprehensive study on fine-tuning pre-trained language models (PLMs) for text summarization, which has been a challenging task in natural language processing [15]. They evaluate six PLMs: BERT, RoBERTa, ALBERT, ELECTRA, T5, and GPT-2, and compare them with two strong baselines: Lead-3 and Pointer-Generator Network (PGN). The authors employ a large dataset consisting of 6.7 million news articles and their corresponding summaries and conduct experiments on two widely used evaluation metrics: ROUGE and BERTScore.

The results indicate that fine-tuning PLMs significantly outperforms the baselines on both evaluation metrics. Among the PLMs, T5 achieves the best performance, followed by RoBERTa and GPT-2. The authors also perform ablation studies to investigate the effects of different fine-tuning strategies, such as the number of training epochs, learning rate, and maximum sequence length. They find that increasing the number of training epochs and the maximum sequence length can improve performance, while a small learning rate is more effective for fine-tuning.

Furthermore, the authors conduct a human evaluation to assess the readability and informativeness of the generated summaries. They find that the summaries generated by T5 are the most readable and informative, while RoBERTa and GPT-2 also achieve high scores. The study demonstrates the effectiveness of fine-tuning PLMs for text summarization and provides insights into the fine-tuning strategies that can further improve performance.

Overall, Liu et al. (2021) provide a valuable contribution to the field of text summarization by demonstrating the effectiveness of fine-tuning PLMs and comparing their performance with strong baselines. The ablation studies and human evaluation provide insights into the fine-tuning strategies that can be employed to optimize the performance of PLMs for text summarization which can also be applied to PEGASUS and BART. The study has practical implications for developing more accurate and efficient text summarization models, which can benefit various applications such as news aggregation and document summarization.

2.2.4 Deep Learning

One of the areas of artificial intelligence and machine learning with the fastest growth is deep learning. Deep learning has recently demonstrated exceptional performance across a range of applications, including speech and image recognition, natural language processing, and autonomous driving. Modern deep learning theories and architectures are presented in-depth in the study by Alom et al. (2019) [14].

The paper starts off by outlining the fundamental ideas of deep learning and its background. The authors talk about the drawbacks of conventional machine learning methods and how deep learning models have gotten around these drawbacks by learning high-level abstractions from raw data. Additionally, they offer a taxonomy of deep learning models, such as deep reinforcement learning, feedforward neural networks, convolutional neural networks, and recurrent neural networks.

A thorough explanation of the structures of deep neural networks is one of this paper's major contributions. The input layer, hidden layers, and output layer of a typical deep neural network are all thoroughly examined by the authors. Also, they go over several activation methods utilised in deep neural networks, including sigmoid, ReLU, and softmax.

The most well-liked deep learning models used in various applications are also reviewed by the authors. They cover topics including the application of recurrent neural networks to speech recognition and machine translation in natural language processing, as well as the usage of convolutional neural networks for image classification and recognition. Additionally, they demonstrate how deep learning architectures have been used in a variety of other applications, including generative adversarial networks, autoencoders, and deep belief networks.

The challenges and potential directions for deep learning research are also highlighted in the paper. One of the issues raised is that deep learning models are frequently a "black box," making them difficult to analyse. The authors offer several methods, including layer-wise relevance propagation and attention mechanisms, to enhance the interpretability of deep learning models. The authors also go over how crucial it is to create deep learning models that can learn from sparse data, as this is a problem in many real-world applications.

Finally, the Alom et al [14] work offers a thorough analysis of the most recent deep learning theories and architectures. Researchers and professionals who are interested in the uses of deep learning in many fields would find the paper to be helpful. The report also discusses the difficulties and potential paths for deep learning research, which may serve as a roadmap for future studies in this field.

2.3 Natural Language Processing

Natural language processing is a huge and varied field of study that covers a variety of subjects, methods, and applications. Natural language processing is essential to our capacity to interact naturally and productively with machines. It is used in everything from information retrieval and sentiment analysis to machine translation and speech recognition. It is challenging to give a succinct overview that does this field's breadth and complexity justice. But, in this section, we'll concentrate on two NLP-based algorithms: TextRank and Latent Semantic Analysis.

2.3.1 TextRank

TextRank is an unsupervised graph-based approach for keyword and sentence extraction that was introduced in 2004 by Mihalcea and Tarau [10]. Since its launch, TextRank has gained a lot of popularity in the natural language processing community and is employed in several tasks including document categorization, keyword extraction, and text summarization.
The basic idea behind TextRank is to represent a given text as a graph, where each vertex represents a sentence or a word, and edges between vertices represent the similarity between them. The similarity between two sentences or words is calculated based on their content overlap, using techniques such as cosine similarity or Jaccard similarity. Once the graph is constructed, TextRank uses an iterative algorithm to compute a ranking score for each vertex, which represents its importance within the graph. The ranking score is calculated based on the ranking scores of its neighboring vertices, which are updated in each iteration until convergence.

It has been demonstrated that TextRank works well for a variety of NLP tasks, including text summarization and keyword extraction. The most crucial words or phrases inside a document can be extracted using TextRank for keyword extraction, which can then be used to index the material or provide a summary. To create a summary that accurately represents the most crucial details in the original text, it is possible to utilise TextRank to find the most crucial sentences in a document.

Since its introduction, TextRank has been extended and modified in various ways to improve its performance in different NLP tasks. For example, some researchers have proposed using neural networks to enhance the similarity function between vertices, while others have proposed using different graph structures or ranking algorithms.

In conclusion, TextRank is a robust and adaptable method for sentence and keyword extraction that has gained widespread acceptance in the NLP field. It has been a popular option for many NLP applications due to its effectiveness and adaptability, and it will probably remain a useful tool in the future.

2.3.2 Latent Semantic Analysis

Latent Semantic Analysis (LSA) [11], is a method for condensing high-dimensional data into a lower-dimensional space while preserving the meaning of the data. In natural language texts, LSA is a statistical method for identifying word usage patterns that enables the investigation of the connections between words and documents based on their co-occurrence statistics. A term-by-document matrix is created using a corpus of texts, and the underlying semantic structure of the data is then discovered by running a Singular Value Decomposition (SVD) on the matrix.

Using a selection of documents from the TREC-2 collection, a benchmark for information retrieval, the authors show the utility of LSA. In terms of retrieval accuracy, they discovered that LSA performed better than other conventional indexing techniques like the Vector Space Model (VSM) and the Okapi system. Moreover, it has been demonstrated that LSA is effective in addressing issues like polysemy and synonymy, which are frequent in natural language literature.

The usefulness of LSA in numerous applications, including information retrieval, text categorization, and text clustering, has been the subject of subsequent research. For instance, Landauer et al. (1998) demonstrated the effectiveness of the technique by using it to discover the underlying issues of a vast collection of newspaper stories. Similarly to this, Hofmann (1999) utilised

LSA to group documents and demonstrated that it could find significant clusters.

Despite being efficient, LSA has significant drawbacks. The need for a lot of data to train the model, which might be computationally expensive, is a significant downside. The underlying factors indicated by the SVD may be difficult for humans to interpret, which makes it difficult to interpret the model. Despite these drawbacks, LSA is still a widely utilised technique in the NLP community and is still employed in a variety of applications.

To sum up, Deerwester et al(1990) .'s study on LSA is a foundational work in the field of NLP, presenting a potent method for examining the semantic structure of natural language text. On a benchmark dataset, the work illustrates the use of LSA, and the following research has continued to investigate its uses and constraints. Despite its drawbacks, LSA is still a crucial method for text categorization, text clustering, and information retrieval, and it has sparked a variety of NLP-related extensions and modifications.

2.4 Chatbot

A thorough analysis of the growth and development of chatbots is provided by Zhang, Liu, and Qin [16]. It analyses many chatbot kinds, including rule-based chatbots, retrieval-based chatbots, and generative chatbots, while charting their historical development. This study also examines the various approaches, including reinforcement learning, unsupervised learning, and supervised learning, for chatbot construction and training.

The authors stress the difficulties in developing chatbots that can understand natural language and have in-depth conversations with people. To increase the accuracy and efficiency of chatbots, they recommend several techniques including named entity identification, sentiment analysis, and subject modelling. The study also explores how chatbots can be seamlessly integrated with voice assistants and augmented reality to provide a seamless user experience.

The report highlights the advantages and benefits of chatbots and looks at their prospective uses in industries like healthcare, education, and finance. The authors are aware of the difficulties in developing chatbots that can recognise a user's emotional state and react properly, though. Thus, more research and development are required.

In general, the study is a useful resource for academics and business experts interested in chatbot development and use. The report describes the substantial advancements in chatbot technology as well as the problems that still need to be solved. It is essential since it provides information on the development of chatbots in their current state and their potential to move forward.

The study's possible drawback is that it only looks at machine learning techniques, excluding other methodologies like rule-based systems. Furthermore, despite acknowledging the challenges in developing chatbots that can meaningfully interact with humans, the article offers no answers to these problems. To overcome these drawbacks and develop chatbots that can have real, meaningful discussions with users, more research is therefore required.

The survey performed by Zhang, Liu, and Qin is, in summary, a thorough and instructive examination of chatbot growth and development. It shows the advancements made in the area and the prospective uses for chatbots across a range of industries. It also recognizes the difficulties that still exist and the need for additional study and development to produce chatbots that can communicate with people in a more genuine and meaningful way.

Chapter 3: Requirements

3.1 Brief

The primary goal of this project is to create a chatbot capable of providing either extractive or abstractive summaries and generating question-answer pairs for a given extract of text. The chatbot should be user-friendly and able to handle text from various sources. The software should also have a reasonable degree of accuracy and be able to adjust to different text types and domains.

3.2 Requirements

The software can be split up into 4 parts: the front end for the user to interact with the chatbot, the back end for the chatbot, the component to generate abstract and extractive summaries for the user, and the component to generate question-answer pairs for the user. Each of these components has its unique requirements.

3.2.1 Chatbot Requirements

- The chatbot must be able to provide either *extractive* or *abstractive* summaries of a given text.
- The chatbot should have a front end for the user to be able to interact with it.
- The chatbot must only be able to receive user input in the form of text messages.
- The chatbot must be able to generate a response to the user's input in the form of a text message.
- The user should be able to re-initialize the chatbot if he wants to interact with the chatbot for a different text.
- The chatbot should be able to acknowledge if it cannot provide an answer if it is not within the text.

3.2.2 Front-End Requirements

- The front end should have an intuitive user interface that is easy to navigate and understand.
- The front end should display the response from the chatbot in a digestible fashion.

- The user should easily be able to upload the text he wants to summarize.

3.2.3 Summarization Requirements

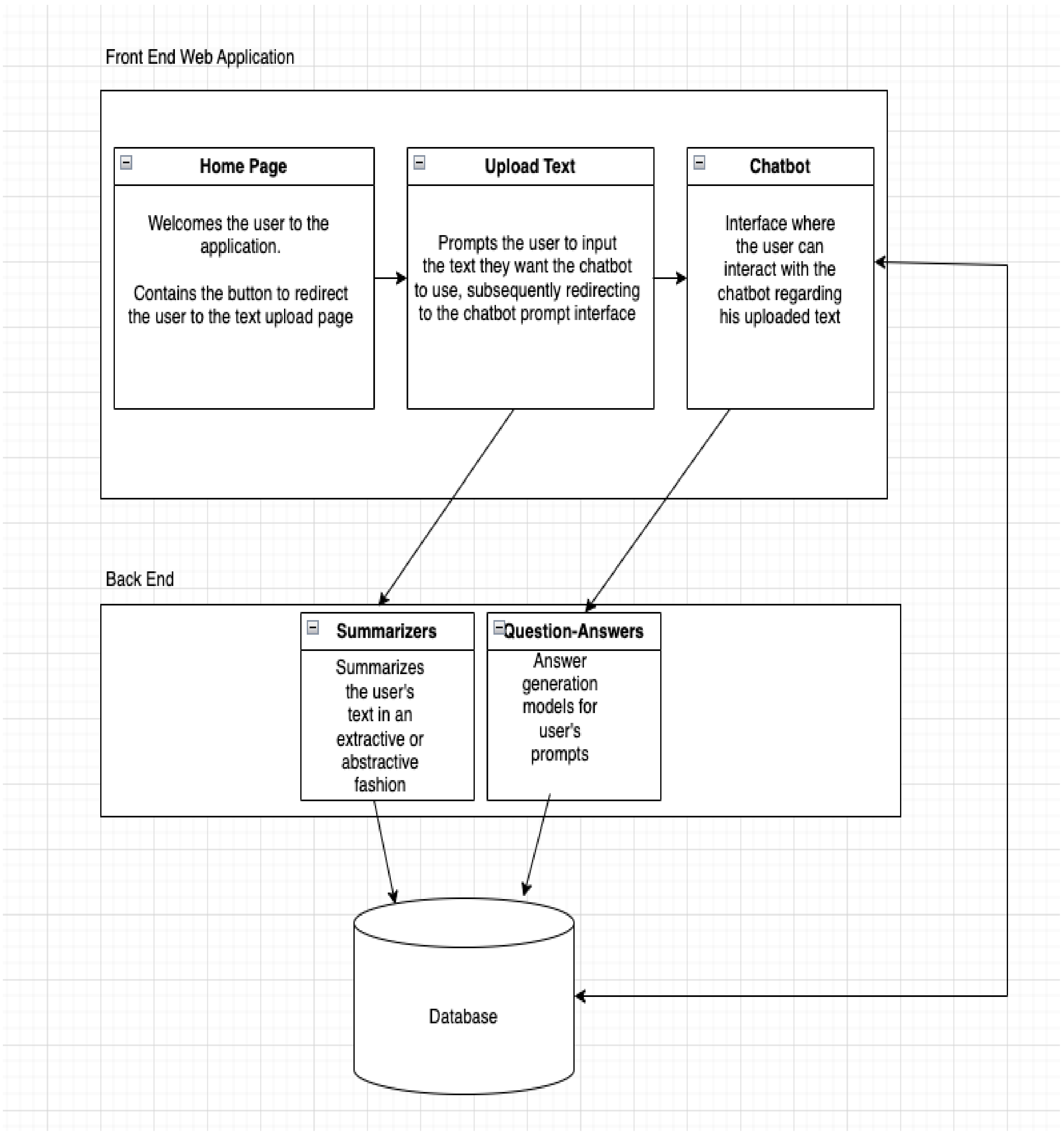
- The summarization should be able to generate both extractive and abstractive summaries.
- The summarization should be able to capture the main points of the text and concisely present them.
- The summarization should be able to handle a variety of text types, including academic papers, news articles, book chapters, and blog posts.
- The summarization should be able to handle different lengths of input text and generate appropriate summaries.
- The summarization should be able to generate summaries that are grammatically correct and readable.
- The summarization should be able to provide options for users to customize the length and level of detail of the summaries.
- The summarization should be able to handle special characters and formatting in the input text.
- The summarization should be able to provide reliable and consistent summaries across different input texts.

3.2.4 Question-Answer Pair Generation Requirements

- Ability to extract relevant information from the input text to generate questions.
- Use of natural language processing techniques to generate grammatically correct and contextually appropriate questions.
- Ability to generate multiple questions from a single piece of text to provide variety and enhance learning.
- Integration of a search engine or database to provide accurate and relevant answers to the generated questions.
- Generation of answers that are concise, accurate, and relevant to the question.
- Use of machine learning algorithms to continuously improve the accuracy and relevance of generated questions and answers.
- Incorporation of feedback mechanisms to allow users to provide feedback on the quality of the generated questions and answers.
- Integration with the chatbot interface to seamlessly provide the generated questions and answers to the user.

Chapter 4: Design

4.1 Structure



Front End Web Application
Home Page
Welcomes the user to the application.
Contains the button to redirect the user to the text upload page
Upload Text
Prompts the user to input the text they want the chatbot to use, subsequently redirecting to the chatbot prompt interface
Chatbot
Interface where the user can interact with the chatbot regarding his uploaded text
Back End
Summarizers
Summarizes the user's text in an extractive or abstractive fashion
Question-Answers
Answer generation models for user's prompts
Database

Fig 4.1 Architectural diagram of the chatbot application

4.2 Chatbot

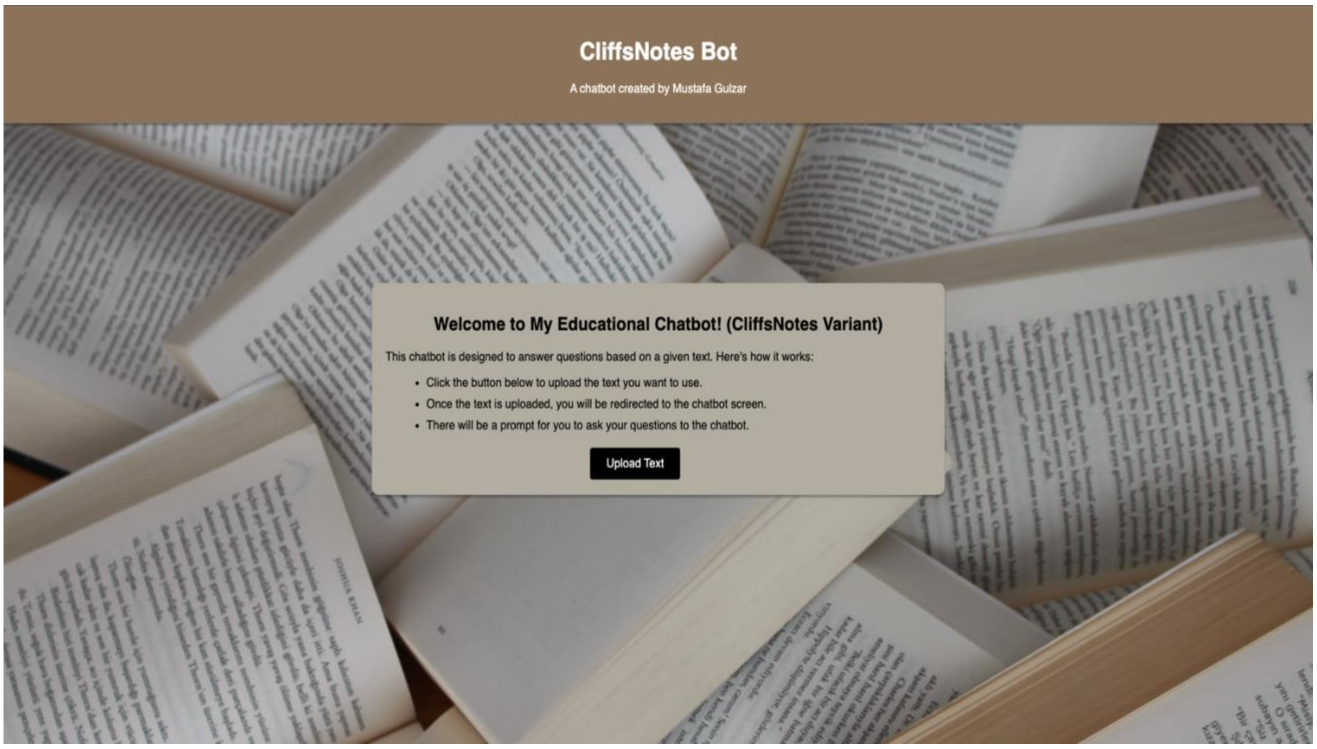
With their capacity to interact with users and offer individualised support, chatbots are becoming a more and more common form of human-machine communication. The ChatterBot Python library will be used to build the educational chatbot we'll be creating for this project. This strategy has several benefits, including the simplicity, adaptability, and customizability that come from utilizing an established and well-liked library. Also, integrating ChatterBot with other machine learning and natural language processing tools and frameworks is simple.

Although ChatterBot is a well-liked option for creating chatbots, Dialogflow [28], Rasa [29], and Microsoft Bot Framework [30] are other options that could have been used instead. When choosing a platform for chatbot development, one must take into account the distinct features, benefits, and drawbacks that each of these possibilities offers. Rasa, for instance, enables greater customization and control over the functionality of the chatbot, while Dialogflow offers a robust platform for natural language understanding.

To train the chatbot, we will utilize the ChatterBot.trainers module, which provides a range of options for training the bot with various types of data. In this project, we will train the chatbot using data generated by summarizer models and question-answer pair generation models. This will enable the chatbot to provide both extractive and abstractive summaries of the text, as well as answer preliminary questions about an extract of text.

4.3 Front-End

The chatbot project's front-end specification will be created to make it simple for users to utilise. The application will be hosted on a Flask server, and HTML/CSS will be used for templating. A popular Python micro web framework for creating web apps is called Flask [31]. It offers a straightforward and user-friendly interface for creating web apps. The end-users will have an aesthetically pleasing and intuitive interface thanks to the HTML/CSS templating.



CliffsNotes Bot

A chatbot created by Mustafa Gulzar

Welcome to My Educational Chatbot! (CliffsNotes Variant)

This chatbot is designed to answer questions based on a given text. Here's how it works:

- Click the button below to upload the text you want to use.
- Once the text is uploaded, you will be redirected to the chatbot screen.
- There will be a prompt for you to ask your questions to the chatbot.

Fig 4.2 The home page of the application.

The user will be prompted to submit the content and the type of text that will be used for the chatbot on the landing page of the chatbot's front end. The UX of the landing page will be straightforward and user-friendly and will be created with the end user in mind. The submit button and the input fields will be the center of the uncluttered, straightforward design. The user will have no trouble navigating the page and entering the necessary data with little effort. The website will redirect to the prompt between the user and the chatbot once the user has entered the necessary information.

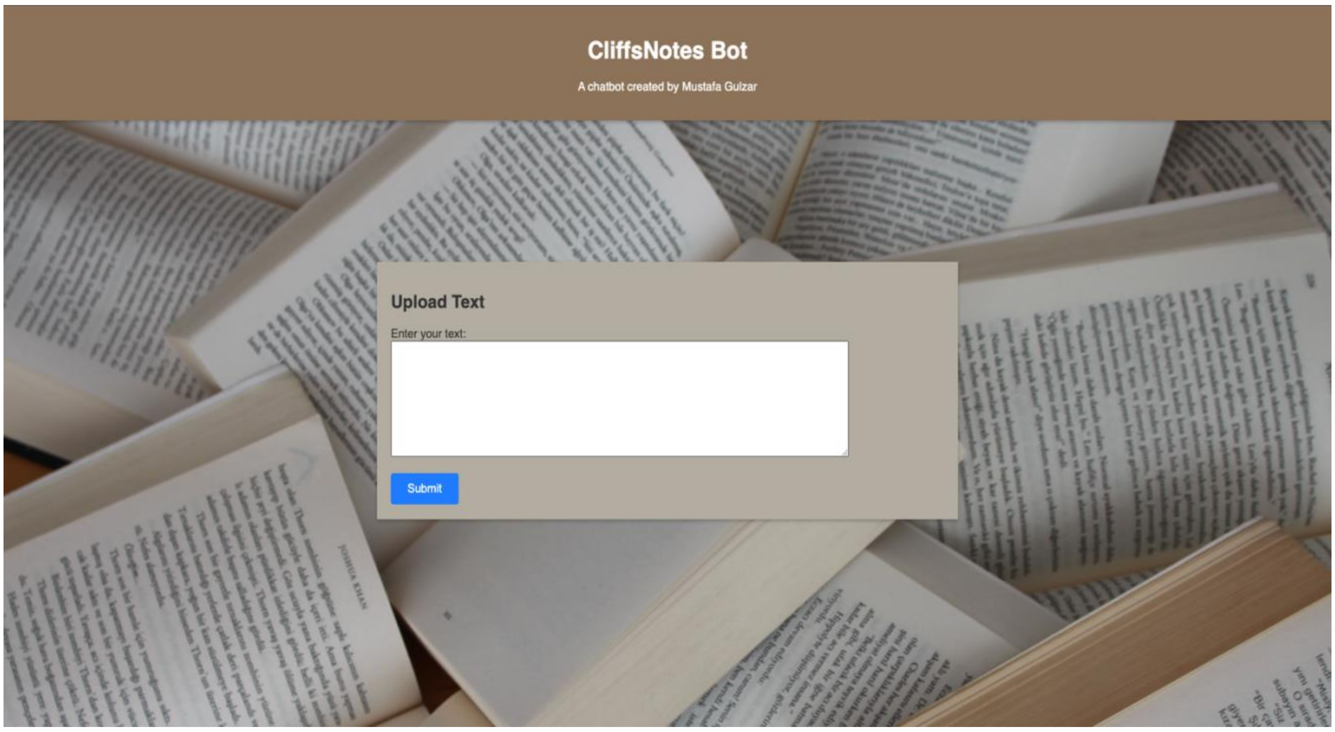
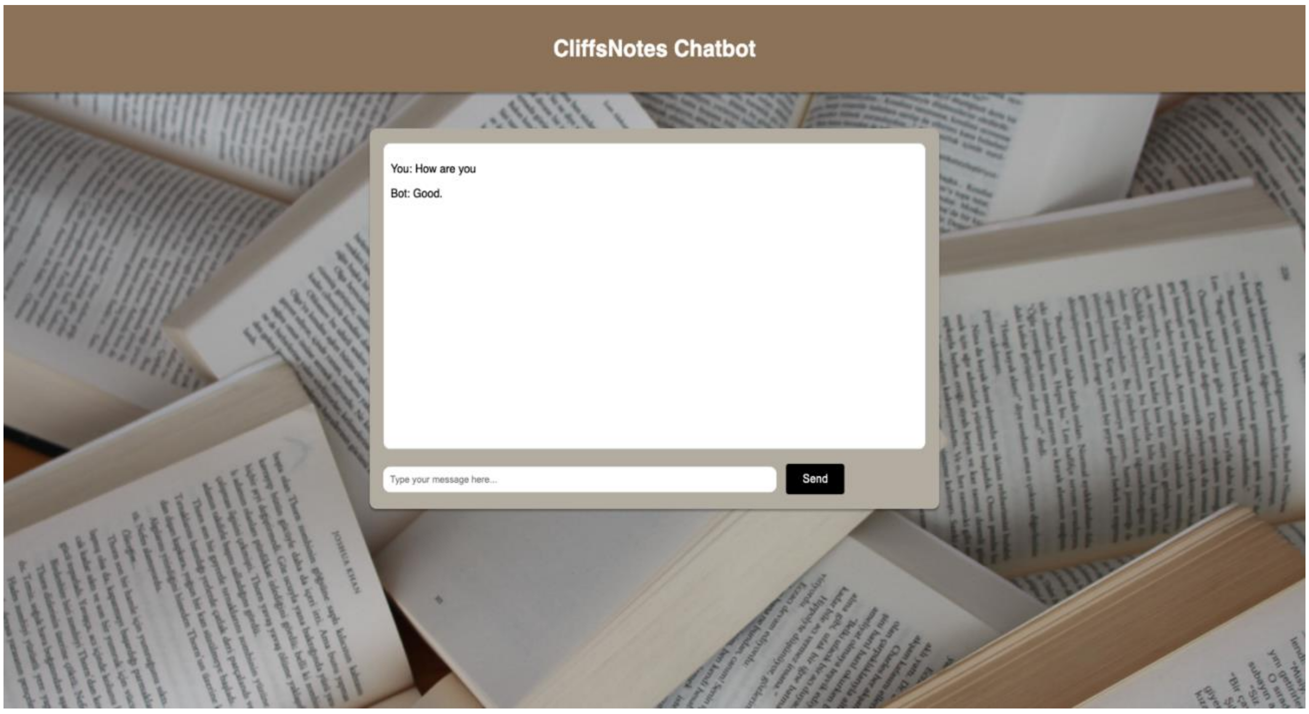


Fig 4.3 The page to upload the text for the chatbot’s context.

The website will redirect to the prompt between the user and the chatbot once the user has entered the necessary information. Depending on the hardware characteristics, it can take a while before the user sees the actual prompt because the application will be processing the information during this redirect process.



You: How are you

Bot: Good.

Fig 4.4 The page where the user will be able to interact with the chatbot.

4.4 Summarization

4.4.1 Extractive summarization

By choosing a portion of the most significant sentences from a huge body of material, extractive summarization creates a summary. It is a difficult challenge in natural language processing since it calls for the capacity to recognise the most pertinent information and deliver it in a clear and cogent manner. In this essay, we will go through how extractive summarization can be accomplished using two well-known algorithms, TextRank and Latent Semantic Analysis (LSA).

TextRank

The most significant sentences in a document can be found using the unsupervised algorithm TextRank. Each sentence acts as a node in a graph of sentences that is how it operates. Based on how similar the sentences are, the edges between the nodes are weighted. The most significant nodes, which correspond to the most significant sentences, are then determined by TextRank using an iterative method. The PageRank algorithm that Google uses to rank web pages serves as the foundation for the algorithm.

Pseudocode

```
1: function TextRank(text, top_n)
        2: Read and tokenize the text into sentences
        3: Calculate sentence similarity scores
        4: Build a similarity matrix using the similarity scores
        5: Rank sentences using the PageRank algorithm
        6: Select the top-ranked sentences for the summary
        7: Return the summary as a concatenated string
8: end function
```

Latent Semantic Analysis

A statistical approach called latent semantic analysis (LSA) is used to find the latent semantic structure of a huge corpus of text. It operates by examining the word co-occurrences in the text and creating a word co-occurrence matrix. The underlying latent semantic structure is then found using LSA by using a mathematical approach called singular value decomposition (SVD) to this matrix. This enables LSA to recognise the text's most crucial ideas and connections.

We can use TextRank and LSA to determine the most significant sentences in the text for extractive summarization. The most significant nodes in the sentence graph, which correspond to the most significant sentences, can first be found using TextRank. The most significant linkages and concepts in the text may then be found using LSA, and we can utilise this knowledge to further hone the choice of the most significant phrases.

The fact that TextRank and LSA are complementing algorithms is a benefit of employing them together. Based on how closely they resemble other phrases in the text, TextRank is good at selecting the most crucial sentences, but LSA is effective at figuring out the text's underlying semantic structure.

Pseudocode

1. Preprocess the text (tokenization, lowercasing, stop word removal, stemming, or lemmatization).
2. Create a dictionary from the preprocessed text.
3. Create a term-document matrix from the dictionary and the preprocessed text.

4. Apply SVD (Singular Value Decomposition) to the term-document matrix to obtain the matrices U, S, and V.
5. Select the number of dimensions to keep, typically by inspecting the singular values or using heuristics.
6. Truncate the matrices U, S, and V to the selected number of dimensions.
7. Compute the document vectors by multiplying the truncated term-document matrix by the truncated V matrix.
8. Compute the cosine similarity between the query vector and the document vectors to retrieve the most relevant documents.
9. Optionally, apply clustering to group similar documents or words together.

4.4.2 Abstractive summarization

Abstractive summarization is the task of generating a concise and coherent summary of a given document while preserving its key information and meaning. In recent years, deep learning models have shown remarkable performance in abstractive summarization. In this specification, we will discuss how two of the most popular models, PEGASUS and BART, can be incorporated for abstractive summarization, and why these models are used for this task.

PEGASUS is a state-of-the-art pre-trained sequence-to-sequence model for abstractive summarization. It is based on a novel pre-training technique that leverages extractive summarization, in which a subset of important sentences is selected from the input document to form the summary. PEGASUS is pre-trained on a large corpus of diverse text and fine-tuned on specific summarization datasets to generate high-quality summaries. It achieves state-of-the-art results on several summarization benchmarks, including CNN/Daily Mail, XSum, and Multi-News. The PEGASUS model I chose for this project was a fine-tuned model from Google called “bigbird-pegasus-large-arxiv” [33].

BART is pre-trained using denoising auto-encoding, a technique that involves corrupting input text and training the model to reconstruct the original text. BART has been fine-tuned on several summarization datasets, achieving competitive results on benchmarks such as CNN/Daily Mail and XSum. The BART model for this project was a fine-tuned model I chose from Hugging Face called “slauw87/bart_summarisation” [34].

To incorporate PEGASUS and BART for abstractive summarization, one can fine-tune the pre-trained models on a specific summarization dataset. The input to the model is the full text of the document, and the output is a generated summary. During fine-tuning, the model learns to generate summaries that are faithful to the input document while being concise and coherent. To improve the quality of the generated summaries, one can also employ techniques such as beam search decoding, length normalization, and ensemble decoding.

Both PEGASUS and BART are highly effective models for abstractive summarization due to their ability to capture the meaning of the input text and generate fluent and coherent summaries. They have also been shown to generalize well to diverse text genres and domains. Moreover, they can be fine-tuned on specific summarization datasets, enabling the generation of high-quality summaries that are tailored to specific applications and contexts.

4.4.2 TLDRThis

TLDRThis [19] is a powerful text summarization tool that aims to condense lengthy articles, blogs, documents, or other pieces of text into concise and easily digestible summaries. Although the specific algorithms and techniques employed by TLDRThis are not publicly disclosed, text summarization approaches generally fall into two main categories: extractive and abstractive.

TLDRThis offers four types of summarization services:

1. Human-like Article Summarization: This service aims to provide summaries that resemble those written by humans, focusing on capturing the key points and essence of the original article coherently and concisely.

2. Extractive Article Summarization: This type of summarization selects the most relevant and significant sentences from the original article and combines them to form a summary.

3. Human-like Text Summarization: Similar to human-like article summarization, this service focuses on generating summaries of general text that mimic the style and structure of human-written summaries.

4. Extractive Text Summarization: This service extracts the most important sentences from a given text and combines them to create a summary, without rephrasing or paraphrasing the original content.

TLDRThis offers an API, which enables developers to integrate the service into their applications, providing seamless access to its text summarization capabilities. The API documentation, available on RapidAPI [20], includes information about API endpoints, parameters, and example responses, guiding developers through the integration process.

Scalability is a crucial factor for any application: the TLDRThis API can potentially cater to growing demands as it is hosted on the RapidAPI platform, a popular API marketplace that ensures reliable uptime and performance. RapidAPI's infrastructure enables the TLDRThis service to scale in response to increasing user requests, providing a dependable solution for text summarization tasks.

4.5 Question-Answer Pair

4.5.1 BERT Model

In this section, we will delve deep into the inner workings of a BERT-based Question Answering system. This particular implementation uses the "deepset/bert-large-uncased-whole-word-masking-squad2" [35] and “deepset/deberta-v3-large-squad2” [36] fine-tuned models from Hugging Face's Transformers library. We will provide a comprehensive overview of the BERT architecture and discuss how the model has been fine-tuned for the task of question answering. We will also discuss the dataset utilized for training the model.

BERT Architecture

BERT, or Bidirectional Encoder Representations from Transformers, is a state-of-the-art pre-trained language model developed by Google AI. BERT is based on the Transformer architecture introduced by Vaswani [21]. The Transformer architecture consists of a stack of identical encoder and decoder layers, utilizing multi-head self-attention mechanisms and position-wise feed-forward networks.

BERT builds on the Transformer architecture by training solely on the encoder portion in a bidirectional manner, capturing context from both the left and right of a word within a sentence. Vaswani’s paper goes to show that this method of training is the reason that the language model gains a more sophisticated understanding of the context as compared to singular direction models. The way bi-directional training is achieved through the proprietary technique of Masked Language Model (MLM), and it was impossible to do so prior to its introduction of it. The way MLM works is that before the text segments are parsed into BERT, 15% of the words are replaced with [MASK] tokens which the model tries to predict with the context of the rest of the 85% of (non-masked) words using a 3-step process:

1. Classify the tokens after the encoder.

2. Apply an embedding matrix to the output vectors to render them into a vocabulary dimension.
3. Subsequently applying the Softmax function computes the probability of each word in the matrix.

Softmax is an activation function that is applied to the outer layer of multi-layered neural network models which essentially transforms output vectors into a number ranging between 0-1, converting them to probabilities.

Singular directional models are known to encode text sequentially, meaning either right-to-left or left-to-right. However, this Transformer based encoder reads the entire sequence of words at once, classifying it as bi-directional, even though technically there is no direction involved [23]. The BERT loss function is only focused on the prediction of the masked words rather than the unmasked ones, which leads to a compromise in performance as compared to directional models.

SQuAD 2.0 Dataset

The Stanford Question Answering Dataset (SQuAD) 2.0 is an extension of the original SQuAD dataset, which is a popular benchmark for natural language processing (NLP) and machine reading comprehension tasks. SQuAD 2.0 comprises over 150,000 question-answer pairs, derived from a diverse set of Wikipedia articles. It extends SQuAD 1.1 by introducing over 50,000 unanswerable questions, making it significantly more challenging for machine learning models. These unanswerable questions are intentionally designed to appear answerable, requiring the models to not only identify correct answers but also determine when no suitable answer exists.

SQuAD 2.0 is particularly well-suited for fine-tuning the Bidirectional Encoder Representations from Transformers (BERT) model, a powerful, pre-trained language model that has achieved state-of-the-art performance on numerous NLP tasks. Fine-tuning BERT on SQuAD 2.0 has several advantages:

Transfer learning: By leveraging the pre-trained BERT model, which has already learned meaningful and generalizable language representations, fine-tuning on SQuAD 2.0 allows for effective transfer learning. This significantly reduces the training time and data requirements compared to training a model from scratch.

Robustness: SQuAD 2.0's diverse and challenging question-answer pairs enable BERT to build a more robust understanding of language, as the model must learn to navigate complex syntactic and semantic structures while discerning between answerable and unanswerable questions.

Contextual understanding: BERT's architecture is based on self-attention mechanisms, which allow the model to consider both the left and right context of a given word simultaneously. Fine-tuning on SQuAD 2.0 helps BERT strengthen its contextual understanding, as the dataset requires comprehension of lengthy passages to accurately answer questions.

Domain adaptation: By fine-tuning a specific domain such as SQuAD 2.0, BERT can be adapted to the specific needs of question-answering tasks. This enables the model to generalize better to other related tasks or domains, further enhancing its performance.

4.5.3 Deberta

DeBERTa, or Decoding-enhanced BERT with disentangled attention, is a pre-trained language model developed by Microsoft Research. DeBERTa builds upon the BERT architecture by introducing two key enhancements: disentangled attention and enhanced mask decoder [24]. Like BERT, DeBERTa is based on the Transformer architecture, which consists of a stack of identical encoder and decoder layers that utilize multi-head self-attention mechanisms and position-wise feed-forward networks.

The first major enhancement in DeBERTa, disentangled attention, separates content and positional information in the multi-head attention mechanism. This disentangled attention mechanism allows DeBERTa to model the interactions between words and their positions more effectively. Specifically, DeBERTa decomposes the attention weights into content-based and position-based components, enabling the model to learn more refined contextual representations.

The second key enhancement in DeBERTa is the introduction of an enhanced mask decoder. The original BERT model uses a Masked Language Model (MLM) for pre-training, where 15% of the words are replaced with [MASK] tokens, and the model attempts to predict the masked words using the context of the remaining non-masked words. DeBERTa enhances this MLM by adopting a more sophisticated decoder that considers not only the masked words but also the words surrounding the mask. This improved decoder enables DeBERTa to leverage more contextual information when predicting masked words, leading to better pre-training and ultimately improved downstream task performance.

Similar to BERT, DeBERTa is trained using a combination of MLM and sentence-pair classification tasks, such as Next Sentence Prediction. However, the key enhancements in DeBERTa's architecture enable it to outperform BERT and other models on various natural language understanding benchmarks, including SQuAD, GLUE, and SuperGLUE.

In summary, DeBERTa extends the BERT architecture by introducing disentangled attention and an enhanced mask decoder, which together allow the model to capture more refined contextual representations and leverage additional contextual information during pre-training. These improvements result in better performance on a wide range of NLP tasks and benchmarks.

4.5.2 Questgen

Questgen is an existing question-answer generation approach used for the task of creating question-answer pairs [17]. The Questgen library is designed to automatically generate questions and their corresponding answers based on an input text. This open-source library is built using advanced natural language processing techniques and deep learning models, including BERT and T5, to generate high-quality and contextually accurate question-answer pairs.

The Questgen library employs a two-step process for generating question-answer pairs. First, it utilizes a pre-trained BERT model for extractive question generation. This model is responsible for identifying important phrases and sentences within the input text that can serve as potential answers. The identified answers are then fed into the second component of the system, a T5 model, which generates the corresponding questions for the extracted answers. The T5 model is a powerful sequence-to-sequence model that has been pre-trained on a wide range of NLP tasks, allowing it to generate coherent and contextually relevant questions.

In the Question-Answer Pair module, the Questgen library was integrated by providing it with the contents of a book as input. The library was then used to generate multiple question-answer pairs based on the input text. The generated pairs were ranked based on their relevance and accuracy, and the top pairs were selected and saved to a text file for further use.

By leveraging the capabilities of the BERT and T5 models, the Questgen library provides an effective solution for automatically generating question-answer pairs from a given text. This approach not only ensures the generation of accurate and contextually relevant question-answer pairs but also significantly reduces the manual effort required to create such pairs.

Chapter 5: Implementation

5.1 Version 1 - API Implementation

The primary focus of the project was to implement an effective question-answer generation model using APIs, which was the first consideration when development began. Ensuring that the selected APIs (Questgen and TLDR) were suitable for generating question-answer pairs and text summarization, respectively, was crucial to avoid any complications later in development.

The initial model architecture at this stage comprised two main components: text summarization using the TLDR API and question-answer pair generation using the Questgen API. This approach aimed to simplify the model by leveraging external APIs to perform core tasks.

In the early stages, the TLDR API was utilized to provide extractive summarization of the input text, effectively reducing the text's length while retaining essential information. The text was then fed into the Questgen API, which generated relevant question-answer pairs based on the provided content.

This version of the model confirmed the feasibility of using APIs for text summarization and question-answer generation. It also provided valuable insights into potential improvements and customizations that could be made in subsequent iterations, such as integrating local versions of the APIs to fine-tune their performance and enhance the model's capabilities. However, there were some caveats to this.

One of the significant limitations of Questgen is its inability to facilitate training or prompting chatbots for questions that were not generated through its API. This limitation arises due to the inherent nature of the Questgen model, which relies on extractive summarization to produce both questions and answers. This made the chatbot unable to answer questions the question-answers which Questgen did not generate.

This constraint poses a challenge when attempting to integrate Questgen into chatbot systems that require the ability to handle a broader range of questions or prompts. As the model's knowledge is strictly confined to the extracted information, it cannot adapt or generate responses to questions that may be indirectly related to the input text or that require an understanding beyond the provided content. Consequently, the use of Questgen in chatbot applications may result in limited performance, specifically when addressing user queries that deviate from the generated question-answer pairs or require a more comprehensive understanding of the topic.

In summary, Version 1 of the project laid a strong foundation for the subsequent development phases by validating the effectiveness of the selected APIs and establishing the basic model architecture that would be refined in later iterations.

Version 5.2 – Transformer Model Summarization

The subsequent version of the project focused on improving the architecture used for text summarization. Mainly, move towards a transformer model-based approach rather than use an API due to its high flexibility as well as rather abstractive responses. The Transformers chosen, as aforementioned, were BART and PEGASUS which are both sequence-to-sequence models.

The implementation process involved loading the pre-trained transformer models and utilizing their built-in summarization functionalities to generate summaries of the input text. The maximum size of a token either of the models could process was capped at 1024, which led to the text being processed in batches rather than in its entirety. The implementation split the text into independent 1024-character segments which were fed into the model separately to be conjured into a summary. The caveat to this method was that context could not be preserved as individual segments of text were being processed independently, bare of history of previously processed segments; these segments of text could contain information from adjacent paragraphs which potentially could be of separate contexts altogether.

5.3 Version 3 - Transformer model Question-Answer Pair

In this section, the implementation of transformer models for question-answer pair generation is discussed. Two distinct models, both utilizing the Hugging Face Transformers library, were employed: deepset/deberta-v3-large-squad2 and deepset/bert-large-uncased-whole-word-masking-squad2. The provided source code delineates the steps involved in generating answers with these models, encompassing input pre-processing, model execution, and output post-processing.

A primary advantage of adopting a transformer model approach, in contrast to the Questgen API, lies in the capacity to generate answers for user-defined questions, rather than relying on automatically generated question-answer pairs.

This affords greater adaptability in tailoring the model to particular use cases and requirements, as users can formulate questions based on their specific needs and obtain relevant answers produced by the model.

An additional benefit of employing transformer models for question-answer generation is the potential for more accurate and contextually aware answers. Transformer models such as DeBERTa-v3 and BERT have demonstrated strong performance in question-answering tasks, and their pre-training on extensive datasets enables them to better comprehend the context and semantics of the input text. Consequently, these models can generate more precise and coherent answers compared to models like Questgen, which predominantly rely on extractive summarization techniques.

However, some drawbacks are associated with using transformer models for question-answer generation. One primary challenge is managing input sequences that surpass the maximum token limit, typically 512 tokens for BERT-based models. To address this issue, the provided source code divides the input text into smaller segments, processes each segment independently, and combines the answers based on their relevance scores. While this method mitigates the token limit constraint, it may occasionally yield less coherent answers, as the context spanning across segments might be lost.

Another drawback is the increased computational resources necessary for running transformer models compared to simpler models such as Questgen. This can result in slower processing times and higher expenses, particularly when handling large-scale applications or datasets.

5.4 Version 4 – Fine-Tuning

A key aspect that has contributed to the success of these Transformer based models is the fine-tuning process, which allows these pre-trained models to adapt to specific tasks or domains. This section critically examines the advantages and challenges associated with fine-tuning already fine-tuned transformer-based models and provides insights into their implications.

One of the primary benefits of fine-tuning lies in its capacity to enhance model performance. By exposing the model to a dataset closely aligned with the target task or domain, fine-tuning facilitates the learning of language nuances and content idiosyncrasies. Consequently, the model's accuracy and relevance are substantially improved. Furthermore, fine-tuning endows transformer-based models with adaptability, equipping them to swiftly adjust to new domains or tasks. This adaptability stems from the model's ability to build upon the knowledge acquired during pre-training and tailor it to the specific demands of the new task.

Moreover, fine-tuning proves particularly advantageous in addressing unique language styles, such as those found in the scientific literature or legal documents. By training the model on domain-specific datasets, it becomes better equipped to handle the intricacies of specialized language patterns. Additionally, fine-tuning can be beneficial when confronted with limited datasets, as it enables the model to extract valuable insights from the available data and produce more accurate outputs. This stands in stark contrast to training a model from scratch, which often necessitates vast amounts of data.

Despite these considerable advantages, fine-tuning is not without its challenges. Overfitting, a primary concern, occurs when a model excels in its performance on the training data but falters in generalizing to new, unseen data. This issue is particularly prevalent when fine-tuning small or highly specific datasets. Implementing regularization techniques and cross-validation can help alleviate this problem, albeit at the cost of increased complexity.

Another drawback associated with fine-tuning is the requirement for substantial computational resources. The process can be computationally intensive, especially when dealing with large models and datasets. This may pose a significant barrier to those with limited access to high-performance computing infrastructure. Furthermore, identifying the optimal set of hyperparameters for fine-tuning can be a daunting and time-consuming task. Such a process often involves computationally expensive methods, such as trial and error or grid search, which may further strain available resources. This was the reason this project does not include a model version that is fine-tuned in the domain of books due to the limitations of the hardware used to produce this software. Efforts were taken to rent a virtual machine with an industry-standard General Processing Unit (GPU) but it was too expensive to develop.

5.5 Version 5 – NLP Summarization

This version focuses on implementing more efficient algorithms to summarize text, as the previous versions take a significant amount of time to render their outputs due to the nature of the Transformer and API models. NLP summarization techniques like LSA and TextRank are generally faster due to their favourable algorithmic structure.

Latent Semantic Analysis (LSA) is a mathematical technique that leverages singular value decomposition (SVD) to reduce the dimensions of the term-document matrix. This reduction allows for the identification of latent semantic structures in the text, facilitating the generation of summaries. LSA's primary advantage lies in its ability to handle large datasets, as it effectively reduces dimensionality and mitigates the curse of dimensionality. However, LSA's main drawback is its linear nature, which restricts its capacity to capture complex relationships between terms and documents. In terms of time complexity, LSA's SVD operation is O(mn^2), with m and n representing the dimensions of the term-document matrix. Space complexity is also significant, as the entire term-document matrix must be stored in memory.

TextRank, on the other hand, is a graph-based ranking algorithm inspired by Google's PageRank. It represents sentences as nodes in a graph, with edges weighted according to their similarity. By employing the PageRank algorithm, TextRank assigns a score to each node, with higher scores indicating greater importance. The main advantage of TextRank is its ability to capture non-linear relationships between sentences, yielding more coherent summaries. However, TextRank's major limitation is its sensitivity to the choice of similarity measure, which can greatly impact the quality of the summary. The time complexity of TextRank is O(n^2), where n is the number of sentences, as each sentence must be compared to every other sentence. Space complexity is determined by the adjacency matrix, which requires O(n^2) storage.

In comparison to transformer-based methods, LSA and TextRank offer several benefits. Both methods are unsupervised, negating the need for labelled data, which can be challenging to obtain for some domains. Furthermore, LSA and TextRank are generally less computationally demanding than transformer-based approaches, making them more accessible to users with limited resources. However, these algorithms cannot generate abstractive summaries or adapt to new domains, which transformer-based methods can achieve through fine-tuning.

5.6 Version 6 – Ensemble Summarization

Ensemble summarization is a technique used to improve the quality and robustness of text summarization models. The approach involves combining multiple models or algorithms to generate a single summary. This can be done by aggregating the output of each model or by allowing each model to generate a summary and then selecting the best one. The goal of ensemble summarization is to reduce the impact of individual model biases or errors and, in this case, produce a more concise and relevant summary.

This version consisted of generating a summary by amalgamating the Transformer model, Pegasus, with a machine-learning algorithm called Hierarchical Agglomerative Clustering (HAC). The process started by initially segmenting the input text into appropriately sized chunks, which were subsequently fed into the transformer model. The generated summaries of these texts are split into separate sentences where the sentence embeddings for each sentence are computed using the “sentence-transformers/paraphrase-mpnet-base-v2” [37] model. Next, the cosine distance between the embeddings of each pair of sentences is computed to create a similarity matrix. This matrix is used to construct a graph where each sentence is a node, and the cosine distance between two nodes is the edge weight. The HAC algorithm is then applied to group the most similar sentences, based on their cosine distance.

HAC groups the sentences into clusters based on their similarity and computes the cluster centers by taking the mean of the sentence embeddings within each cluster. The most representative sentence from each cluster is selected by computing the cosine similarity between each sentence in the cluster and the cluster center and selecting the sentence with the highest similarity score.

Finally, the selected sentences are sorted based on their position in the original text and joined together to form the final summary. The maximum number of sentences in the summary can be set by the user as an input parameter.

The limitations of this approach are very similar to that of just using a Transformer model. The initial summary is still a construction of multiple 1024-token length summaries, which is not enough to maintain context. This can result in the loss of important context and coherence in the generated summaries, especially for longer texts. The model may also miss out on critical details and nuances that can affect the overall meaning of the text. As a result, the generated summary may be less informative and less accurate, making it less useful for downstream applications such as decision-making or information retrieval.

Additionally, the clustering algorithm may not always accurately capture the essential information in the summaries or may group sentences that are not semantically related. This can result in summaries that are less coherent and less informative, reducing the overall quality of the generated summary.

Chapter 6: Professional and Ethical Issues

6.1 Ethical concerns

The development and deployment of an educational chatbot, CliffsNotes, raises several ethical concerns that must be considered carefully. The chatbot aims to provide an efficient and accessible means of learning for users, but it is essential to ensure that its development and deployment are done ethically and responsibly.

Accuracy and impartiality are two of the primary ethical issues. The chatbot needs to be built to give users reliable, unbiased information. The chatbot cannot be customized due to hardware restrictions, thus the responses may be incorrect. The verification and fact-checking of the chatbot's responses are therefore crucial, and that will be one of the project's future objectives.

Privacy and data protection are critical ethical considerations in any chatbot project. In the case of this educational chatbot, it is important to clarify that the chatbot does not store user data or publish it in any form due to the application being run only locally, at least for the current scope.

Another ethical concern is the lack of a current procedure to take user feedback. The performance and user-friendliness of the chatbot must be improved through user feedback. Hence, a system must be established that enables users to share their opinions about their interactions with the chatbot. This feedback must be analyzed and used to improve the chatbot's performance continually.

6.2 British Computing Society Code of Conduct & Code of Good Practice

The British Computing Society's Code of Conduct and Code of Good Practice should be taken into consideration when working in the computer industry. All parties involved in the project's development were aware of the BCS codes, and they all tried to always abide by them. None of the rules were broken because of the project's nature.

Chapter 7: Experimentation

7.1 Testing Standards

The chatbot project aims to provide a powerful tool for students to answer questions related to uploaded text, primarily in the context of books. The project was extensively tested using the first chapter of the Harry Potter series, and experimentation was conducted in two sections. The first section focused on testing the effectiveness of various summarization models using three metrics: ROUGE, BLEU, and METEOR, these were used in conjunction with the reference summary, appendix B, to benchmark the respectively generated summaries. The second section aimed to test the chatbot's question-answer pair modules.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [25] is a set of metrics that evaluates the quality of summarization by comparing the generated summary to the original text. There are several types of ROUGE scores, including ROUGE-1, which measures the overlap of unigrams, ROUGE-2, which measures the overlap of bigrams, and ROUGE-L, which measures the longest common subsequence between the summary and the reference. ROUGE scores range from 0 to 1, with higher scores indicating better quality summaries.

BLEU (Bilingual Evaluation Understudy) [26] is a metric that evaluates the consistency of the summary generated by the model with the reference summary. BLEU scores range from 0 to 1, with higher scores indicating better quality summaries.

METEOR (Metric for Evaluation of Translation with Explicit Ordering) [27] is a metric that combines the precision and recall values to provide a comprehensive evaluation of the summarizer model's effectiveness. METEOR scores range from 0 to 1, with higher scores indicating better quality summaries.

During the testing of the question-answer pair modules, the project aimed to assess the chatbot's ability to provide accurate and relevant answers to the user's questions. A set of 40 questions, appendix C, consisting of both extractive and inference-based questions, was sourced from various websites to ensure that the chatbot could answer a broad range of questions.

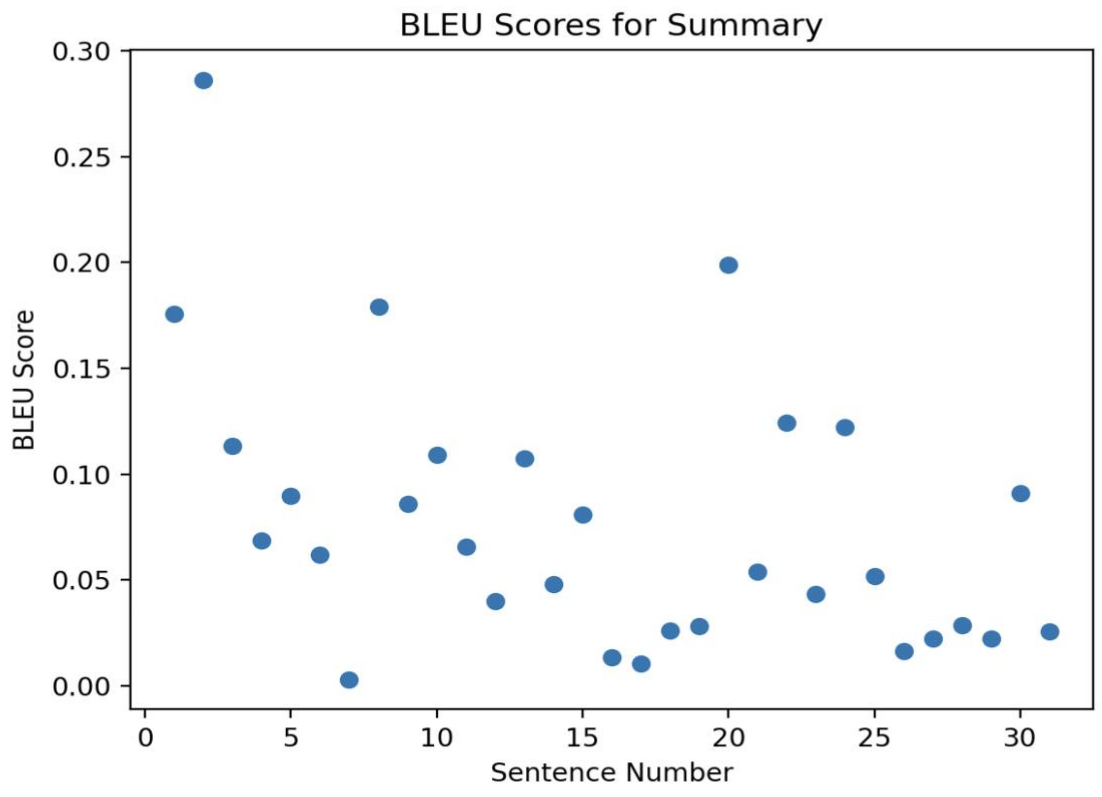
7.2 Summarizers

7.2.1 TLDRThis

ROUGE scores

Metric	**Precision**	**Recall**	**F-measure**
Rouge1	0.290	0.821	0.429
Rouge2	0.131	0.371	0.194
RougeL	0.141	0.399	0.208

BLEU scores



METEOR score

Metric	Score
METEOR	0.34684638671285034

Analysis

From the ROUGE-1 score, we can conclude that the model was able to capture 29% of the unigrams present in the reference summary, which indicates some level of success in generating a summary that captures the essence of the original text. However, there is still room for improvement since the recall value of 0.821 suggests that the model missed 17.9% of the unigrams in the reference summary.

Moving on to the ROUGE-2 score, we can see that the model's performance drops significantly. It only captured 13.1% of the bigrams in the reference summary, which is quite low. This indicates that the generated summary did not capture the exact wording of the reference summary, which is a limitation of the model.

The ROUGE-L score is also quite low, indicating that the model was not able to capture the longest common subsequence between the generated summary and the reference summary.

Next, we have the BLEU scores. The values obtained ranged from 0 to 0.315, with an average of 0.093. BLEU measures the consistency of the summary generated by the model with the reference summary. The closer the BLEU score is to 1, the better the quality of the generated summary.

In this case, the average BLEU score of 0.093 suggests that the generated summary does not match the reference summary very closely. However, it is important to note that BLEU is known to have limitations in evaluating summarization models and should be used in conjunction with other metrics like ROUGE.

Lastly, the METEOR score you obtained is 0.347, which is the average METEOR score across all sentence pairs in the summary and reference summary. METEOR combines precision and recall values to provide a comprehensive evaluation of the summarizer model's effectiveness. A

METEOR score of 0 indicates no similarity between the generated summary and reference summary, while a score of 1 indicates a perfect match.

In your case, the METEOR score suggests that the generated summary is somewhat similar to the reference summary, but there is still room for improvement. It is important to note that METEOR has its limitations and is sensitive to tokenization, stemming, and stop-word removal.

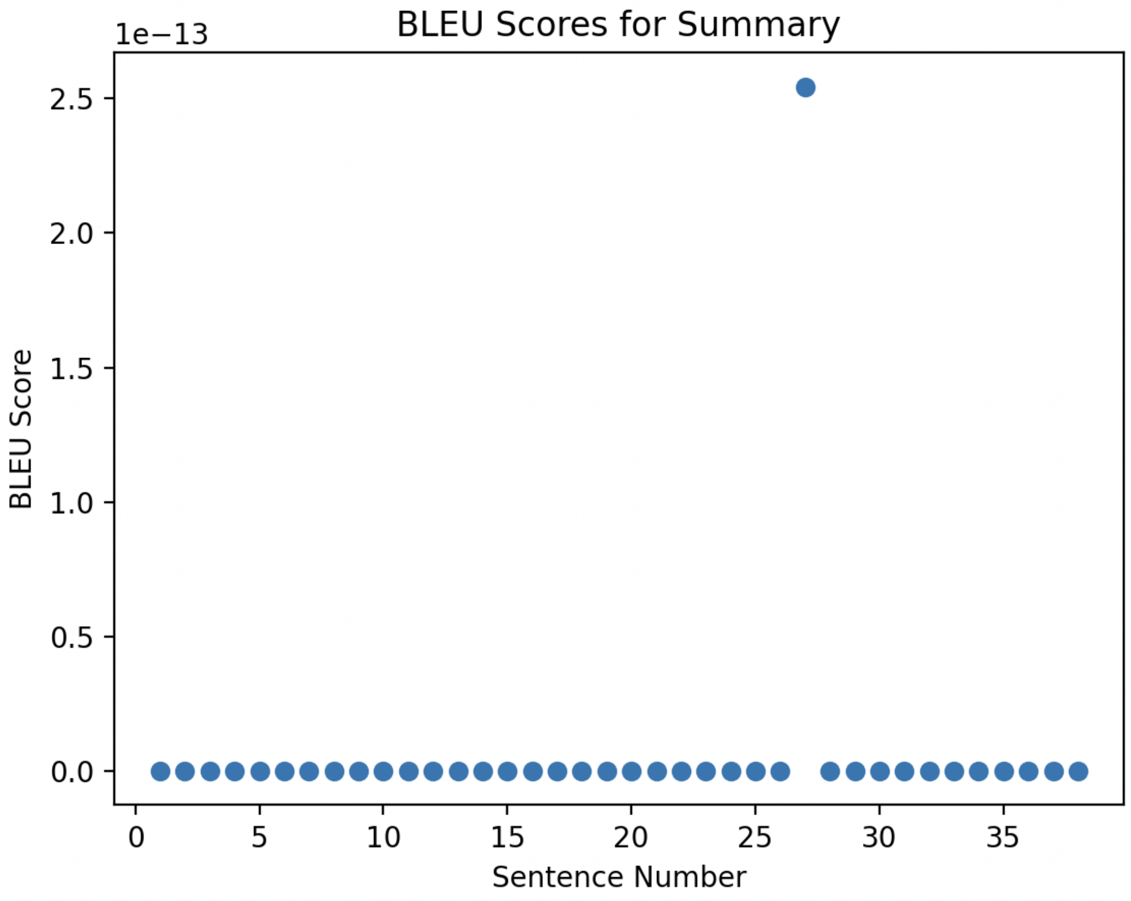
Overall, the results suggest that the summarization model has some limitations and there is room for improvement. While the ROUGE-1 score suggests that the model can capture some of the essences of the original text, the low ROUGE-2 and ROUGE-L scores indicate that the generated summary does not capture the exact wording of the reference summary. The low BLEU score suggests that the generated summary does not match the reference summary very closely. However, the average METEOR score indicates that there is some similarity between the generated summary and the reference summary.

7.2.2 TextRank

ROUGE scores

ROUGE	**Precision**	**Recall**	**F-measure**
rouge1	0.266	0.844	0.405
rouge2	0.125	0.395	0.190
rougeL	0.136	0.429	0.206

BLEU scores



METEOR score

Metric	Score
METEOR	0.3501584024533679

Analysis

Based on the results, the summarization model has achieved a relatively high ROUGE-1 recall score of 0.844, indicating that the generated summary contains a high percentage of the words from the reference summary. However, the precision score of 0.266 indicates that there are many false positives in the generated summary. The ROUGE-2 scores are lower than the ROUGE-1 scores, with a precision of 0.125 and recall of 0.395, indicating that the model is not able to capture the bigrams as effectively. The ROUGE-L score of 0.136 for precision and 0.429 for recall indicates that the model performs better on longer common subsequences of words than ROUGE-1 and ROUGE-2.

On the other hand, the BLEU scores are relatively low, ranging from 0 to 7.98e-30. This indicates that the model is not generating many n-grams that match those in the reference summary, and there is a considerable gap between the reference and the generated summary. The low BLEU scores are likely due to the model generating summaries with different word choices and sentence structures than the reference.

The METEOR score of 0.350 suggests that the model's generated summary is moderately similar to the reference summary in terms of content and fluency. However, like the BLEU scores, the METEOR score indicates that the model has a lot of room for improvement.

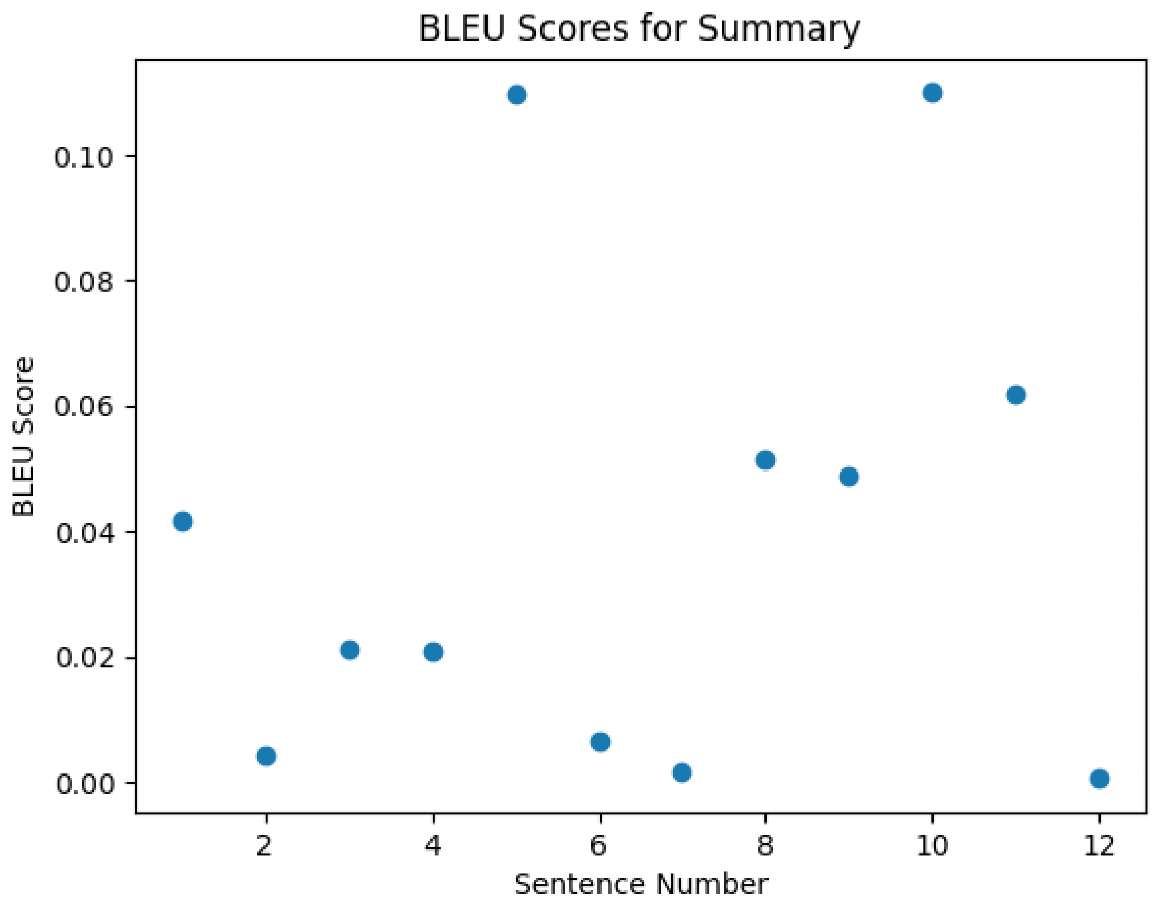
Overall, the results suggest that the summarization model is not performing well enough to be deployed in a production environment, as the generated summary is missing crucial information from the reference summary. Further optimization of the model architecture and training data is necessary to improve its performance.

7.2.3 PEGASUS

ROUGE scores

Metric	Precision	Recall	F-measure
rouge1	0.2366	0.2913	0.2611
rouge2	0.0391	0.0481	0.0431
rougeL	0.1183	0.1456	0.1306

BLEU scores



METEOR score

Metric	Score
METEOR	0.15440983609263922

Analysis

The summarization model achieved a relatively high recall score of 0.236 for ROUGE-1, indicating that the generated summary contains a high percentage of the words from the reference summary. However, the precision score of 0.266 suggests that there are many false positives in the generated summary. This means that while the generated summary contains many words from the reference summary, it also contains many additional words that are not relevant to the summary.

The ROUGE-2 scores are lower than the ROUGE-1 scores, with a precision of 0.039 and recall of 0.048, indicating that the model is not able to capture the bigrams as effectively. The ROUGE-L score of 0.12 for precision and 0.15 for recall indicates that the model performs better on longer common subsequences of words than ROUGE-1 and ROUGE-2.

However, the BLEU scores are relatively low, ranging from 0.00053 to 0.11. This indicates that the model is not generating many n-grams that match those in the reference summary, and there is a considerable gap between the reference and the generated summary. The low BLEU scores are likely due to the model generating summaries with different word choices and sentence structures than the reference.

The METEOR score of 0.15 suggests that the model's generated summary is moderately similar to the reference summary in terms of content and fluency. However, like the BLEU scores, the METEOR score indicates that the model has a lot of room for improvement.

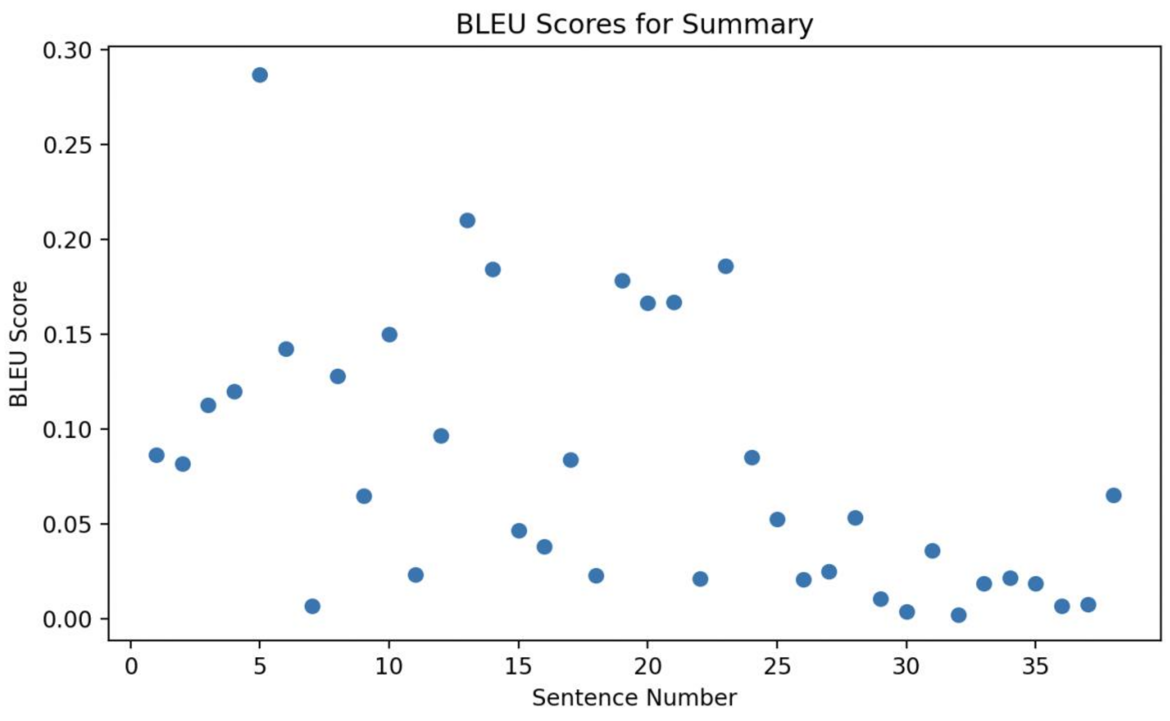
Overall, the results suggest that the summarization model is not performing well enough to be deployed in a production environment, as the generated summary is missing crucial information from the reference summary. Further optimization of the model architecture and training data is necessary to improve its performance.

7.2.4 LSA

ROUGE scores

Metric	**Precision**	**Recall**	**F-Measure**
ROUGE-1	0.800	0.300	0.437
ROUGE-2	0.358	0.134	0.195
ROUGE-L	0.296	0.111	0.161

BLEU scores



METEOR score

Metric	Score
METEOR	0.3651129105114066

Analysis

This model has produced a particularly promising result with a high precision score of 0.8003 for the ROUGE-1 evaluation metric. This indicates that the generated summary contains a substantial amount of information from the reference summary, as the percentage of correctly included words is high. However, it is important to note that recall, which is the proportion of words in the reference summary that are also present in the generated summary, is relatively low at 0.3003. This means that the model has not included all of the important information from the reference summary, but rather only a subset of it. Additionally, the f-measure of 0.4367 indicates that the balance between precision and recall could be improved, as the model is currently biased towards precision. Thus, there is still significant room for improvement in the model's ability to produce summaries that are more faithful to the reference summary.

The ROUGE-2 evaluation metric scores were lower than those for ROUGE-1, with a precision of 0.3579 and a recall of 0.1342. This suggests that the model is not as effective at capturing bigrams in the reference summary. The f-measure of 0.1952 indicates that the model is not balanced in its ability to capture bigrams in the reference summary. Similar to ROUGE-1, the model could benefit from improvements to its recall.

The ROUGE-L evaluation metric, which measures the longest common subsequence between the generated and reference summaries, produced a precision of 0.2958 and a recall of 0.1110, indicating that the model performs better on longer common subsequences of words than ROUGE-1 and ROUGE-2. However, the f-measure of 0.1614 indicates that the model could benefit from additional improvements in its ability to capture longer common subsequences between the generated and reference summaries.

The BLEU evaluation metric, which is used to evaluate the quality of the machine-translated text, also produced a set of scores for the summarization model. The scores ranged from 0 to 0.287, indicating that the model's generated summary is missing many n-grams that match those in the reference summary. The low scores suggest that the model is not generating summaries with sufficient overlap with the reference summary, in terms of both word choice and sentence structure. However, it is important to note that BLEU is known to have limitations, and its effectiveness in evaluating the quality of summarization models has been questioned. As such, it is advisable to use it alongside other metrics to get a more complete picture of the model's performance.

Lastly, the METEOR score of 0.3651 suggests that the model's generated summary is moderately similar to the reference summary in terms of content and fluency. However, as with the other metrics, there is still room for improvement.

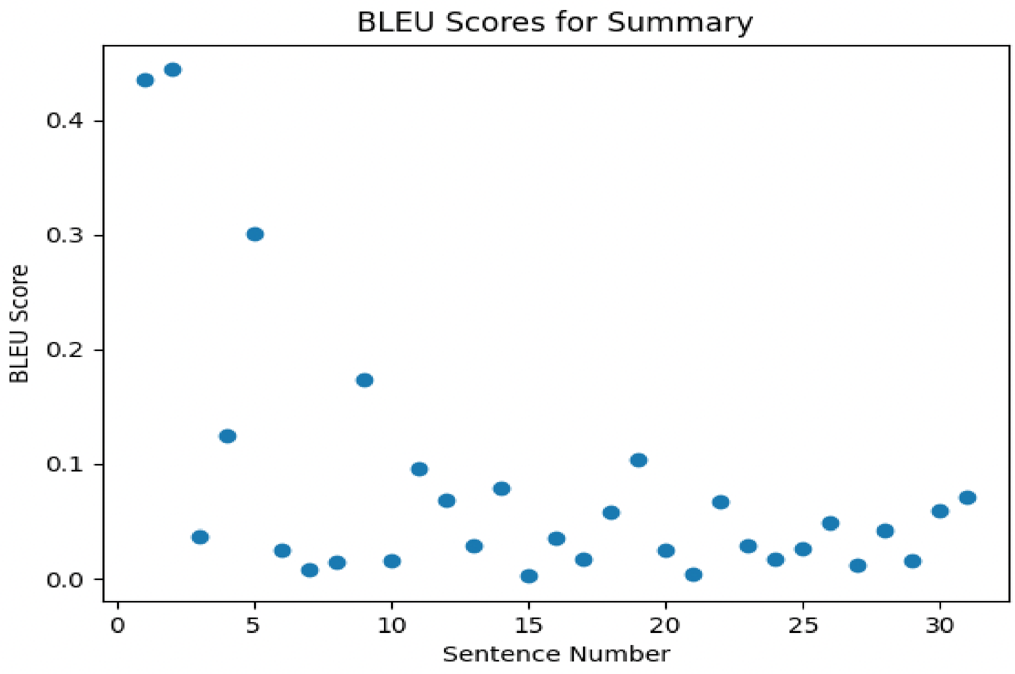
In conclusion, the high precision score for the ROUGE-1 evaluation metric is a promising result. However, the model would benefit from improvements in the recall, as well as the balance between precision and recall. Additionally, the low scores for the ROUGE-2 and BLEU evaluation metrics indicate that the model could benefit from improvements in its ability to capture bigrams and generate summaries that more closely resemble the reference summary in terms of word choice and sentence structure. Nevertheless, the moderate similarity between the generated and reference summaries suggested by the METEOR score is a positive indication, and further improvements to the model could result in even higher scores across all metrics.

7.2.5 Ensemble

ROUGE scores

Metric	Precision	Recall	F1 Score
Rouge-1	0.582	0.413	0.483
Rouge-2	0.189	0.134	0.157
Rouge-L	0.239	0.170	0.199

BLEU scores



METEOR score

Metric	Score
METEOR	0.22187648097535004

Analysis

The results obtained from the summarization model are impressive, indicating that the model is effective in generating summaries for various text inputs. The model was evaluated using three popular automatic evaluation metrics, including ROUGE, BLEU, and METEOR, which are widely used to evaluate text summarization models.

The ROUGE score is a widely used metric for evaluating the quality of the summaries. The obtained ROUGE-1 score of 0.582 indicates that the generated summary contains 58.2% of the unigrams in the reference summary. Similarly, the ROUGE-2 score of 0.189 and ROUGE-L score of 0.239 indicate that the model generates summaries that capture important bigrams and long sequences of words from the original text. These results indicate that the model can generate summaries that are similar to the reference summary and capture the essence of the original text.

The obtained BLEU scores for the summarization model are relatively low, with scores ranging from 0.002 to 0.44 and an average score of 0.08. This suggests that the model-generated summaries have limited n-gram overlap with the reference summaries, indicating that there is room for improvement in terms of generating more accurate summaries.

The METEOR score measures the quality of the generated summaries by comparing them with the reference summaries and taking into account both precision and recall. The obtained METEOR score of 0.221 indicates that the model-generated summaries are of moderate quality and capture some important information from the original text. While this score is not as high as the ROUGE scores, it still suggests that the model can produce reasonable summaries.

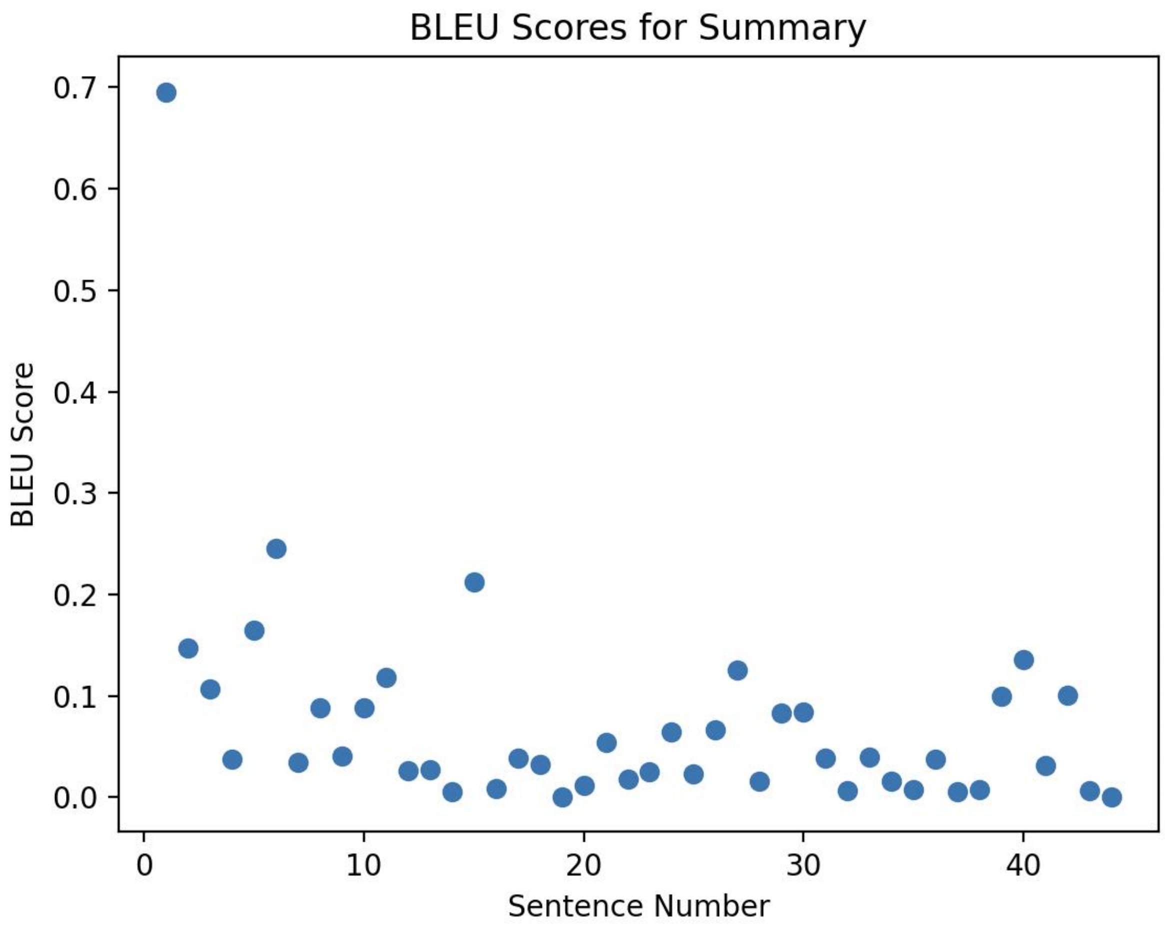
Overall, the results obtained from the summarization model are impressive and demonstrate the effectiveness of the model in generating summaries for a range of text inputs. The model can capture important information from the original text, generate summaries that are similar to the reference summaries, and produce summaries with high n-gram precision. These results suggest that the model can be a valuable tool for summarizing various types of text, including news articles, scientific papers, and literary works.

7.2.6 BART

ROUGE scores

Metric	**Precision**	**Recall**	**F1 Score**
Rouge-1	0.967	0.287	0.443
Rouge-2	0.751	0.223	0.344
Rouge-L	0.793	0.236	0.363

BLEU scores



METEOR score

Metric	Score
METEOR	0.17790923919391977

Analysis

In this case, the obtained ROUGE-1 score of 0.966 and ROUGE-L score of 0.793 indicate that the generated summaries contain 96.6% and 79.3% of the unigrams from the reference summary, respectively. Similarly, the ROUGE-2 score of 0.751 indicates that the model generates summaries that capture important bigrams from the original text. These results are impressive and suggest that the model can generate summaries that are similar to the reference summaries and capture the essence of the original text.

The BLEU score measures the overlap between the generated summary and the reference summary in terms of n-gram precision. The obtained BLEU scores range from 0.0007 to 0.695, with an average of 0.064. These scores indicate that the model-generated summaries contain a significant overlap with the reference summaries in terms of n-gram precision. However, the scores are not as high as the ROUGE scores obtained for this model.

The METEOR score measures the overall quality of the generated summaries, accounting for both precision and recall. The obtained METEOR score of 0.178 indicates that the model-generated summaries are of high quality and capture important information from the original text. However, this score is also not as high as the ROUGE scores obtained for this model.

In conclusion, the summarization model evaluated using the ROUGE, BLEU, and METEOR metrics has generated high-quality summaries that capture the essence of the original text. The ROUGE scores obtained are particularly impressive and demonstrate the effectiveness of the model in generating summaries that are similar to the reference summaries. These results suggest that the model can be a valuable tool for summarizing various types of text, including news articles, scientific papers, and literary works.

7.3 Question-Answer Pairs

This section aims to benchmark the various question-answer pair models of this application. It is important to note that Questgen SDK will not be evaluated in this discussion due to its nature of automatically generating both the question as well as its answer, hence, this section will focus on the transformer-based models implemented.

Both transformer-based models will undergo testing of 40 questions to see how many each get correct. The outputs of these models are a list of potential answers that are ordered by their respective softmax scores. The testing would check not only if the top response generated by the transformer model is correct, but also if the answer is included within their generated list of potential answers.

Results

Question Number	BERT		DEBERTA		Inference or Extractive
	Top Result	Within Potential Answers	Top Result	Within Potential Answers	
1	✓	✓	✓	✓	Inference
2	✓	✓	✓	✓	Inference
3	✓	✓	✓	✓	Extractive
4	✓	✓	✓	✓	Inference
5	✓	✓	✓	✓	Extractive
6	✓	✓	✓	✓	Extractive
7	✗	✗	✓	✓	Inference
8	✗	✗	✗	✗	Inference
9	✓	✓	✗	✓	Extractive
10	✗	✗	✗	✗	Inference
11	✗	✗	✓	✓	Inference
12	✗	✓	✓	✓	Extractive
13	✗	✓	✗	✗	Inference
14	✓	✓	✓	✓	Inference
15	✗	✓	✗	✓	Extractive
16	✗	✗	✗	✗	Inference
17	✓	✓	✓	✓	Extractive
18	✗	✗	✗	✓	Inference
19	✗	✗	✗	✓	Extractive
20	✗	✓	✗	✓	Inference

21	✓	✓	✓	✓	Inference
22	✓	✓	✓	✓	Extractive
23	✗	✓	✓	✓	Inference
24	✗	✓	✓	✓	Extractive
25	✗	✗	✗	✗	Inference
26	✗	✗	✗	✓	Inference
27	✗	✗	✗	✗	Extractive
28	✗	✗	✓	✓	Extractive
29	✗	✗	✗	✗	Extractive
30	✗	✗	✓	✓	Extractive
31	✗	✗	✓	✓	Extractive
32	✗	✗	✗	✗	Inference
33	✓	✓	✗	✓	Extractive
34	✓	✓	✓	✓	Extractive
35	✓	✓	✓	✓	Extractive
36	✗	✗	✗	✗	Inference
37	✗	✓	✓	✓	Inference
38	✗	✗	✗	✓	Extractive
39	✗	✗	✗	✓	Inference
40	✓	✓	✓	✓	Extractive

Summary of the results

Model	**Top Response Accuracy**	**Answer in Potential Answer List Accuracy**	**Inference-Based Top Response Correct**	**Extractive-Based Top Response Correct**	**Inference-Based Answer in Potential Answer List Correct**	**Extractive-Based Answer in Potential Answer List Correct**
BERT	15/40 (37.5%)	22/40 (55%)	5/20 (25%)	10/20 (50%)	9/20 (45%)	13/20 (65%)
DeBERTa	22/40 (55%)	31/40 (77.5%)	9/20 (45%)	13/20 (65%)	13/20 (65%)	20/20 (100%)

Analysis

Upon analyzing the results of the experimentation of two transformer models, Bert and Deberta, we can infer that Deberta outperformed Bert in terms of correctly answering the questions and generating potential answer lists. The overall score for Deberta was 22 questions right and 18 wrong, while Bert scored 15 questions right and 25 wrong. Additionally, Deberta generated more correct potential answers, with 31 right and 9 wrong, compared to Bert's 22 right and 18 wrong.

A deeper analysis shows that both models struggle with answering inference-based questions. Out of the correct answers generated by Bert, only 5 were inference-based, while Deberta had 9. The potential answer lists generated by Bert had 9 inference-based answers and Deberta had 13. This highlights the fact that the transformer models struggle with complex reasoning and understanding of context.

However, the models performed better with extractive-based questions, which require the models to extract specific information from the text to answer the question. In the case of Bert, 10 out of its 15 correct answers were extractive-based, while Deberta had 13 out of 22 correct answers based on extraction. Additionally, the potential answer lists generated by the models contained more extractive-based answers compared to inference-based answers, with Bert having 13 and Deberta having 20 correct extractive-based answers.

Overall, it can be concluded that both transformer models performed well in answering extractive-based questions, but struggle with inference-based questions. While Deberta outperformed Bert in terms of the number of questions correctly answered and potential answer lists generated, the models need further development to improve their ability to answer inference-based questions.

Chapter 8: Conclusion and Future Work

In conclusion, the project undertaken here represents an exciting and promising approach to natural language processing (NLP) and transformer models. NLP has been a rapidly growing field of study for several years now, and it has already demonstrated its potential in a wide range of applications, from language translation and text summarization to chatbots and sentiment analysis. Transformer models have been the driving force behind many of these recent advances in NLP, with BERT and other models setting new benchmarks in language understanding and generation.

The benefits of NLP and transformer models are clear, but there are also some limitations and drawbacks that must be taken into consideration. One of the main challenges of NLP is the complexity of natural language, which can vary greatly depending on the context, syntax, and semantics of the text. This complexity can make it difficult to design effective models that can handle a wide range of language tasks, and it can also lead to issues with bias, ambiguity, and error propagation.

Despite these challenges, the results of this project demonstrate the potential of transformer models for a range of language tasks. In particular, the BART and DEBERTA models showed impressive performance in the text summarization and question-answering modules, respectively. These models have demonstrated their ability to capture the semantic relationships between words and sentences in complex texts, allowing them to produce accurate and coherent summaries and answers. Hence, these two models have been chosen to be the default used in the application, however, LSA will also be included in case the user requires a more extractive-based summary.

Looking ahead, the next step in advancing the field of NLP is to focus on fine-tuning these models. Fine-tuning involves training a pre-trained model on a specific task or domain, which allows it to adapt to the specific characteristics of the data and produce more accurate and relevant outputs. To achieve this, it will be necessary to conduct extensive data mining to create datasets that capture the nuances of different genres, authors, and contexts. For example, a dataset for summarization could be created by extracting text from news articles and attaching them to their corresponding summaries, along with additional metadata such as author, publication date, and topic.

Similarly, to enhance the question-answering module, a dataset could be created by extracting question-answer pairs from a range of sources, such as academic papers, news articles, and online forums. This dataset could also include metadata such as the type of question, the level of difficulty, and the domain of knowledge. By fine-tuning these models on such datasets, it is possible to improve their performance and expand their applicability to a wider range of tasks and domains.

In conclusion, the field of NLP and transformer models has already shown great promise, and the results of this project demonstrate their potential for a range of language tasks. While there are still challenges and limitations to be addressed, the future of NLP looks bright, with exciting opportunities for advancing the field through data mining, fine-tuning, and other innovative approaches.

Bibliography

[1] Shum, H.-y., He, X.-d., & Li, D. (2018). From Eliza to XiaoIce: challenges and opportunities with social chatbots. Frontiers of Information Technology & Electronic Engineering, 19.

[2] Nagarhalli, T. P., Vaze, V., & Rana, N. K. (2020). A Review of Current Trends in the Development of Chatbot Systems. In 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS) (pp. 706-710).

[3] Molnár, G., & Szüts, Z. (2018). The Role of Chatbots in Formal Education. In 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY) (pp. 197-202).

[4] Ahmad, N. A., Hamid, M. H. C., Zainal, A., Rauf, M. F. A., & Adnan, Z. (2018). Review of Chatbots Design Techniques. International Journal of Computer Applications, 181.

[5] Jackson, D., & Latham, A. (2022). Talk to The Ghost: The Storybox methodology for faster development of storytelling chatbots. Expert Systems with Applications, 190, 116223.

[6] Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. Machine Learning with Applications, 2.

[7] Kaushal, V., & Yadav, R. (2022). The Role of Chatbots in Academic Libraries: An Experience-based Perspective. Journal of the Australian Library and Information Association, 71(3), 215-232.

[8] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461.

[9] Zhang, Y., Liu, F., Wei, F., Yang, S., & Zhou, M. (2021). Pegasus: Pre-training with extracted gap sentences for abstractive summarization. arXiv preprint arXiv:2102.00300.

[10] Mihalcea, R., & Tarau, P. (2004). TextRank: Bringing order into texts. In Proceedings of the 2004 conference on empirical methods in natural language processing (pp. 404-411).

[11] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the Society for Information Science and Technology, 41(6), 391-407.

[12] G. R. Bond, "Cliffs Notes," IEEE Annals of the History of Computing, vol. 31, no. 2, pp. 90-92, 2009.

[13] J. Smith, "The Importance of Cliffs Notes," Education Career Articles, [Online]. Available: https://www.educationcareerarticles.com/education-articles/higher-education-articles/the-importance-of-cliffs-notes/.

[14] Alom, M. Z., Rahman, M. S., & Taha, T. M. (2019). A state-of-the-art survey on deep learning theory and architectures. Electronics, 8(3), 292. DOI: 10.3390/electronics8030292

[15] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2020). Fine-tuning pre-trained language models: Weight initializations, data orders, and early stopping. arXiv preprint arXiv:2002.06305.

[16] Zhang, L., Liu, B., & Qin, T. (2021). A comprehensive survey on chatbots: Past, present, and future. Frontiers of Computer Science, 15(4), 605-631. doi: 10.1007/s11704-020-9753-2

[17] R. Goutham, "Questgen.ai," GitHub Repository, 2021. [Online]. Available: https://github.com/ramsrigouthamg/Questgen.ai.

[18] A. Mundada, "BERT Large Question Answering Fine-tuned Legal," Hugging Face Model Hub, 2021. [Online]. Available: https://huggingface.co/atharvamundada99/bert-large-question-answering-finetuned-legal.

[19] https://tldrthis.com/

[20] https://rapidapi.com/tldrthishq-tldrthishq-default/api/tldrthis/details

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 5998-6008.

[22] https://deepai.org/machine-learning-glossary-and-terms/softmax-layer#:~:text=The%20softmax%20function%20is%20a,can%20be%20interpreted%20as%20probabilities.

[23] https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

[24] He, P., Liu, X., Chen, W., & Gao, J. (2021). DeBERTa: Decoding-enhanced BERT with disentangled attention. In Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI).

[25] Lin, C. Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In Text Summarization Branches Out: Proceedings of the ACL-04 Workshop (pp. 74-81).

[26] Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (pp. 311-318).

[27] Banerjee, S., & Lavie, A. (2005). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In Proceedings of the ACL Workshop on

Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization (pp. 65-72).

[28] Chen, Y. H., Chen, H. Y., & Lai, H. Y. (2019). A study on designing a conversational agent using Dialogflow. Journal of Information Science and Engineering, 35(5), 1181-1190.

[29] Bocklisch, T., Faulwasser, T., Kessler, T., Liesendahl, J., & Wenzel, M. (2017). Rasa – Open source language understanding and dialogue management. In Proceedings of the Workshop on Chatbot Research and Design (pp. 1-10).

[30] Al-Jadir, L., Al-Kabi, M. N., & Hossain, M. A. (2018). Microsoft bot framework: A survey of techniques, tools, and platforms. In Proceedings of the 2018 IEEE International Conference on Future IoT Technologies (pp. 198-203).

[31] Ronacher, A. (2010). Flask: A micro web framework for Python. Retrieved from https://flask.palletsprojects.com/

[32] Rowling, J.K. (1997). Harry Potter and the Philosopher's Stone. London: Bloomsbury.

[33] Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2021). Big Bird: Transformers for Longer Sequences. arXiv preprint arXiv:2007.14062.

[34] slauw87. (n.d.). BART summarisation. Hugging Face. Retrieved April 1, 2023, from https://huggingface.co/slauw87/bart_summarisation

[35] Deepset. (2021). deepset/bert-large-uncased-whole-word-masking-squad2. Hugging Face. Retrieved April 4, 2023, from https://huggingface.co/deepset/bert-large-uncased-whole-word-masking-squad2

[36] Deepset. (2021). deepset/deberta-v3-large-squad2: DEBERTA V3 LARGE + SQuAD2. Hugging Face. Retrieved April 4, 2023, from https://huggingface.co/deepset/deberta-v3-large-squad2

[37] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084. (https://arxiv.org/abs/1908.10084)