DOCUMENT SUMMARISER & QUESTION GENERATOR

D M V S V Karthik Jayanthi *†,*Burugupalli Sai Chaitanya* †, Kodati Jaideep †,Shanmukha Rahul *

Email: dmvsvkarthik.jayanthi.21cse@bmu.edu.in*
burugupallisai.chaitanya.21cse@bmu.edu.in *
kodati.jaideep.21ces@bmu.edu.in*
sivapuramshanmukha.rahul.21cse@bmu.edu.in*

***Abstract*—In today's digital age, extracting meaningful information from PDF documents poses a significant challenge due to their extensive content and complex nature. PDF documents are widely used to maintain information structure and integrity in a variety of contexts, such as corporate reports, legal documents, and academic research.**

Nonetheless, users may find it difficult to efficiently extract and understand important insights due to their dense content and occasionally complicated structure. Acknowledging this challenge, the Document Summarizer & Question Generator tool appears as a way to assist users in more efficiently navigating the vast geography of PDF documents.

This study describes an approach to design a comprehensive Document summarizer & Question Generator tool. This tool leverages start-of-art natural language processing models, including finetuning of FLAN T5 for document summarization and GPT-3.5-turbo for question generation. This report details the development process, methodologies, and evaluation results of this application.

Notably, FLAN T5 variants exhibit robust summarization capabilities when finetuned on DialogSum dataset demonstrating exceptional performance across rouge metrics like rouge 1 rouge 2, rougeL, rougeLsum. The GPT-3.5-turbo excels in generating the contextually relevant questions which can be seen by its rouge scores. *Keywords*—PDF documents, Document summarization, Question generation Natural language processing, FLAN T5, GPT-3.5-turbo, DialogSum dataset, Rouge metrics, Information extraction, Digital age, Complexity, Navigation efficiency

I. Introduction

Understanding this challenge, the Document Summarizer & Question Generating tool emerges as a solution to help the users explore the vast geography of the PDF documents more effectively. This programme provides concise summaries of PDFs and produces relevant questions based on their content. The programme uses advanced language processing algorithms to help users to understand the required content properly.

Understanding this challenge, the Document Summarizer & Question Generating tool emerges as a solution to help the users explore the vast geography of the PDF documents more effectively. This programme provides concise summaries of PDFs and produces relevant questions based on their content. The programme uses advanced language processing algorithms to help users to understand the required content properly. This study shows how to create a comprehensive Document Summarizer and Question Generator tool. This programme uses advanced natural language processing models, including FLAN T5 for document summarization and GPT-3.5-turbo for question formulation. In essence, this application will allow the users to overcome the problems of extracting insights from PDF documents, which enhances their capacity to access and absorb the information effectively.

Whether for academic, professional, or personal use, the Document Summarizer & Question Generator tool is a valuable tool for anybody looking to explore the depth of information contained within the PDF documents.

II. Related Works

In [1], the authors have collected 40 different transcripts from Spotify podcast summarization dataset and generated the seed summaries due to low reference quality and also used Shakespeare corpus collected the 40 scenes where each 700 words long. The authors of [8] have generated 40 fictional sales transcripts and emails from ChatGPT. [4] extracted the dataset from Journalist-written health-related claims, known as PUBHEALTH, were used to evaluate their accuracy. Each claim was categorised as True, False, Mixed, or Unproven. The training set contains 9466 claims, while the validation and test sets have 1183 claims each.[4] has also used the FEVER Dataset which consist of 185k claims the dataset for FEVER shared tasks has been released in 2018. During the fact-checking task, claim assertions were classified as Supports, Refutes, or Not enough Data. [4] The e-FEVER dataset is a subset of the original FEVER dataset and consists of 67687 claims with evidence documents retrieved using the DOMLIN system. [7] has used the CodeXGLUE. CodeXGLUE was derived from the CodeSearchNet dataset. It is multilingual, having data in six languages (Ruby, JavaScript, Java, Go, PHP, and Python). The bulk of foundation models [1, 2, 7] are assessed on this dataset for code summarising. In [9], the paper describes LaMini, an instructional dataset, for fine-tuning models across tasks like text summarization, providing guidelines for model fine-tuning using this dataset. [10] Creating different datasets from multiple domains to test the resilience of current QA methods is challenging due to the high human resource requirements. The primary emphasis is on the automated production of long-tail QA datasets. However, they have encountered various problems during this procedure, including noise filtering, question granularity, difficulty control, and prompt engineering. These difficulties require further investigation to develop basic solutions have generated the dataset from Coarse-tail QA, Fine-tail QA,

TriviaQA, WebQA, NaturalQA. [6] have used WIKI-ENG, WIKI-SIMP, Wall Street Journal from Penn Treebank with diversified questions and themes which help the model training. Detecting Factual Errors using automated methods for detecting inconsistencies include question-based and entailmentbased approaches. Question-based techniques assume that linked materials and summaries will yield consistent answers [1]. observed that GPT-4 underperforms human precision by 8% in factual reasoning. In their experimental setups they have used three non-LLM approaches and have used ten LLM approaches to reduce computational costs, deployed a single ZeroShot prompt across all the LLM models. [2] Initial studies in the last decade focused on extractive 12 summarization, with some incorporating sentence-level operations like compression and reordering as postprocessing. most of the frameworks can be described as:

- Sentence scoring: Determines the importance of each sentence. Summarization preserves valuable information by extracting key sentences.
- Sentence selection: Choose the most important sentences to create a paragraph-length summary. This section should consider global aspects like content coherence and repetition in descriptions.
- Sentence reformulation: Modifying or paraphrasing extracted sentences from original materials might result in clearer, more cohesive, and concise summaries.

Large Language Models Supervised fine-tuning with natural language instructions empowers the large language models (LLMs) to achieve remarkable zeroshot performance on a diverse set of applications [9]. The Pegasus model architecture defines the encoderdecoder architecture for summarization work which have proposed some unique techniques to improve the summarization work. [11] proposes a new selfsupervised pre-training objective for abstractive summarization with gap sentence generation and a study for selecting those sentences. They have evaluated this across so many different summarizations task for that where they have used 568M parameters for training the Pegasus model for all the datasets and there aim to show that how good and affectively summarization works by Pegasus with little finetuning like with 1000 examples. [5] they have fine-tuned the Llama-2 model and gpt-3.5 turbo which has more parameters which are trainable for Socratic question generation like for the given context they have split the data into two sets, and they will give one as an input to the Llama and another to model to generate the question based on context. So, to measure the similarity between generated Socratic questions and ground truth questions they have used BERT score and ROUGE-L which measures n-gram overlap based over long-common sequence. The fine-tuning procedure improves Pegasus' capacity to write short and informative summaries by adjusting its parameters for the summarization task. This adaptation guarantees that the model learns to prioritise important information while maintaining coherence in the output summaries. Furthermore, fine-tuning enables Pegasus to generalise more effectively to previously unknown data by learning domain-specific properties from training data. Overall, finetuning of the SamSum dataset allows Pegasus to excel at producing highquality abstractive summaries.

III. Methodology

A. Introduction to Languages

The document summarizer and question generator application is developed using a combination of different front-end and back-end technologies to ensure a robust and user-friendly experience for the users.

Front End and Back End Languages:

- **Python:** Python is a high-level, general-purpose programming language. The design philosophy of python emphasises the code readability through the use of extensive indentation. Python is dynamically typed, and garbage collected. It supports a variety of programming paradigms, including structured, object-oriented, and functional programming.
- **Streamlit:** Streamlit is an open-source Python toolkit used for developing the interactive web apps with minimum code. It streamlines the process of designing the user interfaces by letting developers to create Python scripts that generate interactive components like sliders, buttons, graphs, etc.... Streamlit sets these pieces in a web browser, allowing the users to easily design and deploy data driven applications without the need for the extra frontend languages or frameworks like the HTML, CSS, and JavaScript.

B. Supporting Libraries

The supporting libraries used in this project are:

- **Streamlit:** Streamlit is a Python library for developing the interactive web applications with minimum code. It streamlines the process of designing user interfaces by letting developers to create Python scripts that generate interactive components like sliders, buttons, graphs, etc....
- **Hugging Face Transformers:** The Hugging Face Transformers library will contain the cutting edge natural language processing (NLP) models and tools for interacting with the pre-trained models such as BERT, GPT, and T5. It includes features for text production, text classification, summarization, and question answering.
- **PyTorch:** PyTorch is an open-source machine learning framework that includes many tools and frameworks for developing and training neural networks. It is commonly used for the deep learning tasks including the natural language processing, computer vision, and the reinforcement learning.
- **PyPDF2:** PyPDF2 is a Python library which is used for handling the PDF files. It contains utilities that can be used for reading, writing, and altering the PDF files.
- **Langchain:** Langchain is a very unique Python package used in this project that is used to provide a variety of NLP related functions such as text splitting, document loading, summarization, question generating, and conversation modelling. It is mostly contains bespoke implementations or wrappers for the NLP models and utilities that are suited to this project's unique requirements.
- **Base64:** The base64 module implements methods for the encoding and decoding of binary data into base64-encoded texts.

It is often used to represent the binary data in ASCII letter format, which is beneficial for delivering the binary data through the text-based protocols like HTTP.

- **OpenAI:** the OpenAI library is used in this project to access the OpenAI's language models or APIs. The OpenAI offers several powerful AI technology and models for different kinds of applications, including NLP, text generation, and the conversational AI.

C. User Flow Diagram

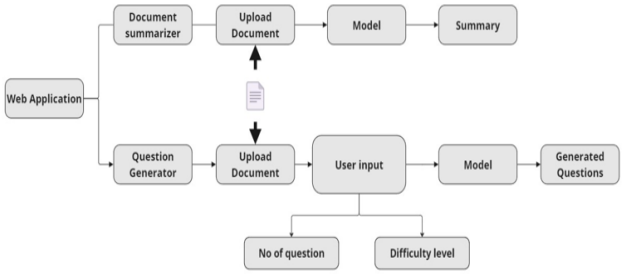

Figure 1: User Flow Diagram showing how a user interact with the platform.

- **User Input:** The process starts with the user uploading a document to the web application.
- **Document Summarizer**: The document is then uploaded to the document summarizer.
- **Summary**: The document summarizer analyses the document and generates a summary.
- **Question Generation:** The user then specifies the number of questions and the difficulty level for the questions they want to be generated.
- **Generated Questions:** The model then generates questions based on the document and the userspecified criteria.

D. Architecture and Implementation

The methodology suggested in this paper contains three phases to develop a document summarizer and question generator pipeline. The initial phase comprises of developing a model for document summarization. The development of question generation is done in the second phase, and the final phase consists of integrating these two models into a single pipeline. The description of each of these phases is discussed further in this report.

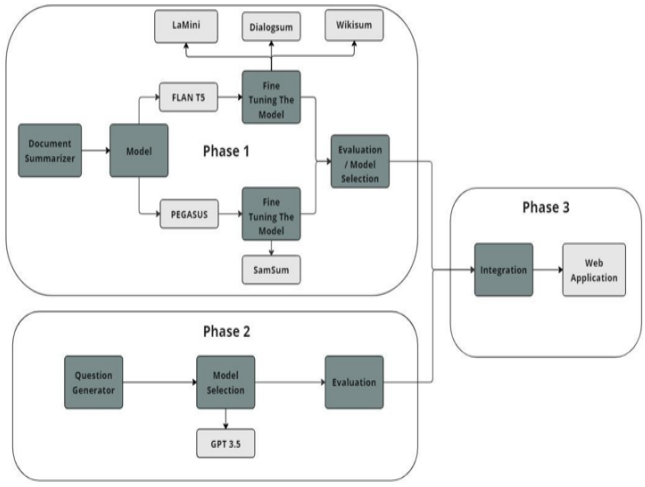

Figure 2: Diagram showing flow how the 3 phases are linked.

E. Development of Document Summarizer

FLAN T5, or Finetuned LAnguage Net is a variant of the T5 (Text-to-Text Transfer transformer) model that has been finetuned for several tasks such as text summarization, translation, question answering, and more. FLAN T5 is a versatile model that was developed by the Google. It is capable of performing various natural language processing tasks when provided with the text-to-text input and output pairs. The Document summarization involves reducing the large documents into short and more concise summaries while preserving the key information. The FLAN T5 is good at this task due to its ability to understand and generate the text across different domains.

FLAN T5 is benefited from the pros of the T5 base architecture, including its transformer based design, which enables it to capture the complex linguistic patterns in the text [3]. The encoderdecoder structure of the FLAN T5 architecture contains the feedforward networks and the self attention mechanisms. For tasks involving natural language processing, it tokenizes the both input and output sequences using the text-to-text approach [3]. This also allows the FLAN T5 to produce the summaries that are not only just concise but also very coherent and contextually accurate, which makes it powerful model for the various document summarization tasks.

Pegasus is a large Language model (LLM) developed by the Google. This model represents a significant advancement in the natural language processing domain. Just like GPT, Pegasus also has a transformerbased architecture, but it is used for abstractive text summarization applications only. It encodes input text and decodes it into coherent summaries using an encoder-decoder architecture with self-attention techniques. The transformer architecture of Pegasus allows it to capture intricate linguistic structures and relationships, leading to high-quality summaries across various domains [13].

These LLM models are used to develop a document summarization model. To obtain better results the FLAN T5 is future finetuned on four different datasets WikiSum, DialogSum, Lamini datasets, as mentioned in [3][8][9]. The Pegasus model is finetune on the SamSum dataset to obtain the better results [13]. This fine-tuning of models makes the performance of the models better in generating fast and better results.

F. Development of Question Generator

The second phase of building this pipeline involves developing a document-based question generator using GPT-3.5-Turbo model.

GPT-3.5-Turbo, an advanced natural language processing model which is developed by the OpenAI, uses a transformer architecture with the layers of self-attention mechanisms and the feedforward neural networks. Pre-training is a critical stage in its development in which the model learns to predict the next word in the sequence based on previous the context. With a huge parameter count of 175 billion [2], GPT3.5- Turbo is one of the largest language models created to date, allowing it to detect the subtle patterns and the correlations in language. The ability of GPT3.5Turbo model to do the zero-shot and few-shot learning enables it to generate coherent responses to prompts the tasks that it has not been trained on. This is accomplished by presenting the model with a question and a few instances of the desired activity.

In this question answering application, the GPT-3.5-Turbo model can be fine tuned using a smaller dataset that contains the samples of the questions and the responses [2]. This fine tuning process changes the parameters of the model for the better suit for the tasks at hand, allowing it to generate the answers based on the content of the given document. After fine tuning, the careful assessment of the model's performance assures the correctness, with the potential revisions to improve the model's capabilities to overcome bias in training data and prevent potential exploitation. The fine tuning also ensure that the GPT-3.5-Turbo can be used to generate accurate and appropriate results but, fine tuning the GPT-3.5-turbo requires high computation.

The question generator developed in this pipeline uses the GPT-3.5-turbo without fine-tuning. This model takes the documents as inputs and generated questions based on the contents of the documents. The GPT-3.5 -Turbo model is implemented in such as what that the user can also define some features like the number questions to be generated, the level of the questions (Simple, Medium, Hard). When the user defines the values for these features these values are given to the model's prompt which helps to generate the output according to the user's inputs.

G. Application Interface Development with Streamlit

Once the document summarization and question generation models have been developed and finetuned, the next step is to deploy these models into a user-friendly web application, in this phase, the goal is to create an intuitive interface where users can easily upload their documents and interact with the pipeline to obtain summarized content and generated questions.

After accessing the web application, users can upload their documents using a simple file upload interface. The given documents are then processed before giving to the summarization and question generation models in the backend. The output generated by these models is displayed to the users in an organized manner, providing summarized content along with relevant questions generated based on the document contents.

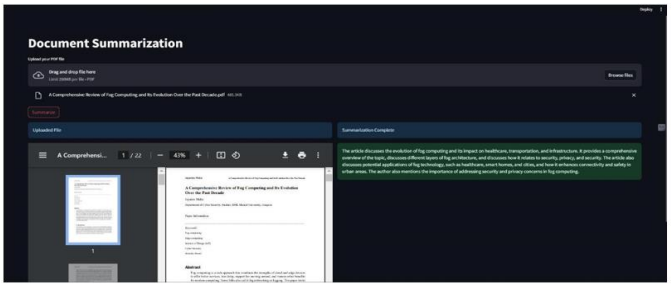

Figure 3: Interface of Document summariser.

The web application offers additional features such as the ability to customize the number and complexity of generated questions, this enhances the user experience. The deployment phase also involves testing to ensure the reliability and the performance of the web application under various user interactions.

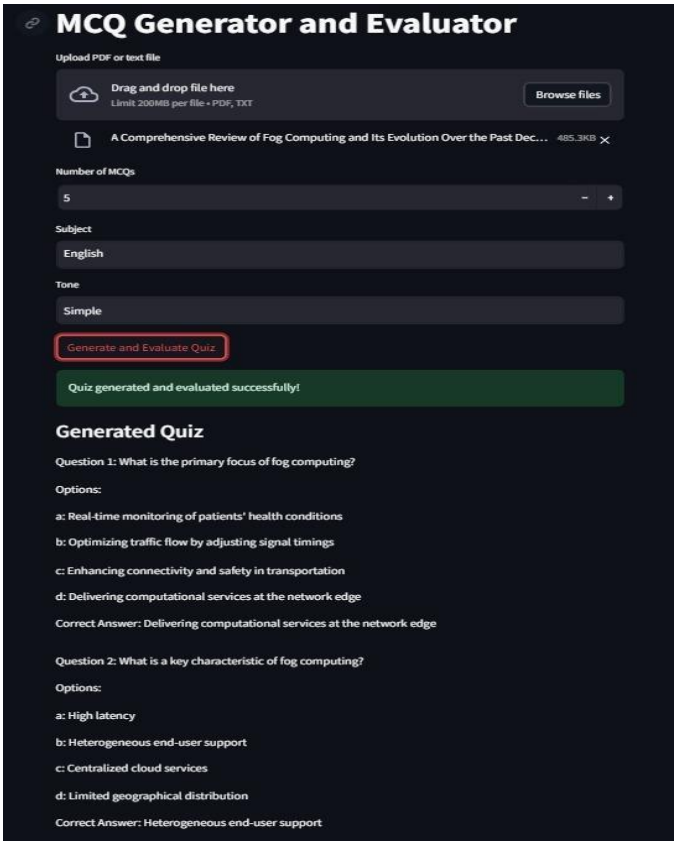

Figure 4: Interface of Question Generator

The deployment of the document summarization and the question generation models into a web application helps the users by building an efficient tool for quickly extracting the useful information and generating the insightful questions form the given documents.

H. Constraints

- **Model Complexity:** The use of powerful natural language processing models such as FLAN T5 and Pegasus for document summarization places limits on computing resources and model complexity. These models necessitate substantial processing

power and memory resources for training and finetuning, rendering them unavailable to users with minimal hardware or computational capabilities.

- **Fine-tuning Overhead:** The process of finetuning LLM models like as FLAN T5 and Pegasus on various datasets imposes restrictions on the time, effort, and skill necessary for data preparation, model training, and assessment. Finetuning across several datasets demands considerable data collecting, annotation, and curation activities, which may be time-consuming and resource intensive.
- **Computation Cost:** The implementation of GPT3.5-Turbo for question generation without finetuning imposes limits on computing cost and scalability. GPT-3.5-Turbo's huge parameter count, and complex design necessitate significant computational resources for inference, restricting its practical application in resource-constrained contexts or scenarios with a high demand for real-time processing.
- **Testing Requirements:** Testing requirements are limited to ensure the web application is reliable, accurate, and performs well under a range of user interactions and document kinds. Extensive testing approaches, including as user acceptability testing, unit testing, and integration testing, are necessary to find and fix any possible problems or defects in the programme.

IV. Results

The results show the performance of various models in both summarization and question generation tasks. Among the summarization models, Flan T5 variants demonstrate robust capabilities, with Flan T5Dialogsum achieving the highest rouge scores across Rouge 1, Rouge 2, Rouge L, Rouge Lsum metrics. Notably, Flan T5-Lamini and Flan T5-Wikkisum also exhibit competitive performance, highlighting the versatility of the Flan T5 architecture. However, Pegasus models lag significantly in summarization tasks, suggesting potential areas for improvement.

Task	Model	Rouge 1	Rouge 2		RougeL	RougeLsum
Summarization	Flan T5	0.209883	0.075420		0.185412	0.188644
	Flan T5-DialogSum	0.286477	0.134974		0.236190	0.239307
	Flan T5-LaMini	0.284482	0.043478		0.181030	0.181034
	Flan T5-WikkiSum	0.252980	0.079613	0.179282		0.178056
			0.000297	0.015541		
	Pegasus	0.015554				0.015530
	Pegasus-SamSum	0.018570	0.000296		0.018431	0.018448
Question generation	GPT-3.5-Turbo	0.234412	0.139393		0.235534	0.235534

Table 1: Rouge Matrix of different models for Document Summarization & Question generation.

In question generation, Gpt-3.5-turbo emerges as a strong performer, showcasing impressive rouge scores across all metrics, indicating its effectiveness in generating diverse and contextually relevant questions. These results highlight the diverse capabilities of different models in natural language processing tasks and provide insights into their strengths and weaknesses.

V. Conclusions and Future Work

A. Conclusion

The Document Summarizer & Question Generator tool offers a powerful solution to challenges of extracting insights from documents. By providing users with accurate summaries and contextually relevant questions based on the user's requirement, the tool enhances information retrieval. This study demonstrates the effectiveness of fine-tuning FLAN T5 model on various datasets for document summarization task and also shows the effectiveness of GPT-3.5-Turbo model in question generation task, highlighting their potential for real-world applications.

From the results it is quite clear that the FLAN T5 model finetuned on the Dialog Sum dataset is giving great results of rouge metrics like rouge 1, Rouge 2, RougeL, RougeLsum with values 0.209883, 0.075420, 0.185412, 0.188644 respectively and hence is deployed into the interface for generating better results. This tool uses the GPT-3.5-Turbo model for question generation that generated the rouge values 0.234412, 0.139393, 0.235534, 0.235534 for rouge metrics like rouge 1, Rouge 2, RougeL, RougeLsum respectively.

The user-friendly interface and customizable features of the tool ensure accessibility represents a valuable resource for unlocking the wealth of knowledge contained within the PDF documents, empowering users to access and comprehend information effectively in various domains.

B. Future Work

The project's future scope includes several possibilities for development. Firstly, the models must be improving their accuracy and performance, resulting in improved question development and summaries. Furthermore, investigating the integration of several LLM models opens up new options, perhaps improving the quality of output produced. Furthermore, including other user inputs, such as the opportunity to set the tone of the summary (formal, professional, or simple), improves the entire user experience by increasing customisation and adaptability.

Furthermore, adding support for multilingual papers is critical to increasing the project's application and usefulness. This improvement allows for a wider diversity of document formats and language preferences among users, promoting inclusivity and accessibility. As the project progresses in these areas, it will be better able to meet the different demands and preferences of its users, therefore increasing its usability and effect in the field of natural language processing.

Acknowledgement References

[1] Cekinel, R. F., & Karagoz, P. (2024). Explaining Veracity Predictions with Evidence Summarization: A Multi-Task Model Approach. Middle East Technical University. Retrieved from arXiv:2402.06443v1 [cs.CL]

[2] Wang, Q., Rose, R., Orita, N., & Sugawara, A. (2024). Automated Generation of Multiple-

Choice Cloze Questions for Assessing English Vocabulary Using GPT-turbo

3.5. Center for English Language Education, Faculty of Science and Engineering, Waseda University. Retrieved from arXiv:2403.02078v1 [cs.CL]

[3] Laban, P., Krysci´ nski, W., Agarwal, D., Fabbri, A. R., Xiong, C., Joty, S., & Wu, C.S. (2023). SUMMEDITS: Measuring LLM Ability at Factual Reasoning Through The Lens of Summarization. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[4] Yao, Jg., Wan, X. & Xiao, J. Recent advances in document summarization. Knowl Inf Syst 53, 297–336 (2017).

[5] Duan, N., Tang, D., Chen, P., & Zhou, M. (2017). Question Generation for Question

Answering. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), 866–874. Copenhagen, Denmark, September 7–11, 2017. Copyright 2017 by the Association for Computational Linguistics.

[6] Heilman, M., & Smith, N. A. (2009). Question Generation via Overgenerating

Transformations and Ranking. Technical Report CMU-LTI-09-013, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213. Retrieved from www.lti.cs.cmu.edu. Copyright 2009 by Michael Heilman and Noah A. Smith.

[7] Ahmed, T., & Devanbu, P. (2022). Few-shot training LLMs for project-specific codesummarization. In ASE ’22: Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (pp. 1-5)

[8] Italiani, P., Frisoni, G., Moro, G., Carbonaro, A., & Sartori, C. (2024). Evidence, my Dear Watson: Abstractive dialogue summarization on learnable relevant utterances. Department of Computer Science and Engineering (DISI), University of Bologna, Italy. Retrieved from Neurocomputing, Volume 572, 127132.

[9] Wu, M., Waheed, A., Zhang, C., Abdul-Mageed, M., & Aji, A. F. (2024). LaMini-LM: A Diverse Herd of Distilled Models from Large-Scale Instructions. Mohamed bin Zayed

University of Artificial Intelligence, Monash University, The University of British Columbia. Retrieved from arXiv:2304.14402v3 [cs.CL]

[10] Kumar, R., Kim, Y., Ravi, S., Sun, H., Faloutsos, C., Salakhutdinov, R., & Yoon, M. (2024). Automatic Question-Answer Generation for Long-Tail Knowledge.
Cornell University. Retrieved from arXiv:2403.01382 [cs.CL]

[11] Jingqing Zhang, Yao Zhao, Mohammad Saleh, Peter J. Liu. (2020). PEGASUS: Pre-training with Extracted Gap sentences for Abstractive Summarization. the 37 th International Conference on Machine Learning. PMLR 119.