TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Major Project Report

On

Question-Answer Generation using T5 transformer

Submitted By:

Abhishek Chaudhary (Exam Roll No. 41752)

Adhip Bhattarai (Exam Roll No. 41753)

Arahanta Pokhrel (Exam Roll No. 41758)

Rishav Subedi (Exam Roll No. 41784)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

April, 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Major Project Report

On

Question-Answer Generation using T5 transformer

Submitted By:

Abhishek Chaudhary (Exam Roll No. 41752)

Adhip Bhattarai (Exam Roll No. 41753)

Arahanta Pokhrel (Exam Roll No. 41758)

Rishav Subedi (Exam Roll No. 41784)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Computer Engineering

Under the Supervision of

Er. Anku Jaiswal

April, 2024

DECLARATION

We hereby declare that the report of the project entitled **"Question-Answer Generation using T5 transformer"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus,** in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Computer Engineering,** is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Abhishek Chaudhary (THA076BCT003) _____

Adhip Bhattarai (THA076BCT004) _____

Arahanta Pokhrel (THA076BCT009) _____

Rishav Subedi (THA076BCT036) _____

Date: April, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus,** a major project work entitled **"Question-Answer Generation using T5 transformer"** submitted by **Abhishek Chaudhary**, **Adhip Bhattarai**, **Arahanta Pokhrel** and **Rishav Subedi** in partial fulfillment for the award of Bachelor's Degree in Computer Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Computer Engineering.

_____

Project Supervisor
Er. Anku Jaiswal
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner
Er. Sanjivan Satyal
Department of Electronics and Computer Engineering, Pulchowk Campus

_____

Project Co-ordinator
Mr. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Mr. Kiran Chandra Dahal
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

April, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We are deeply grateful to everyone who was instrumental in the completion of the report.

First and foremost, we would like to extend our gratitude to our project supervisor**, Er. Anku Jaiswal**, for her continued support and direction in the duration of our project. We could not have accomplished our goal without her input and direction. Our sincere appreciations also go to the Department of Electronics and Computer Engineering at Thapathali Campus for giving us the chance to carry out this project.

Additionally, we would like to extend special thanks to those individuals who provided technical advice assistance during the development stages. Their expertise and support were indispensable in ensuring the project's successful completion.

Sincerely,

Abhishek Chaudhary (THA076BCT003)

Adhip Bhattarai (THA076BCT004)

Arahanta Pokhrel (THA076BCT009)

Rishav Subedi (THA076BCT036)

ABSTRACT

In general, this study presents a new technique for solving question generation and answer generation tasks. We use two T5 models to generate question-answer pairs from the given paragraph. The first T5 model generates questions, and the second T5 model generates answers to these questions. The motivation is to present a natural technique that could generate multiple questions and answers. Therefore, we have gathered a dataset with three question and three answer for each collected paragraph. We have collected paragraphs in 3 domains namely, “Renewable Energy”, “Environmental Science” and “Pollution”. Using these datasets with these two steps would provide essential and concise questions using transformer architecture in the T5 model. We hope that using two T5 models together would help to increase the general quality of question-answer pair and assist to maintain common logic in the passage – the information should flow from the paragraph to questions and back to the answers. The user can also provide answers to the questions generated by the first model and check the similarity between the answer provided by him and the model’s output. For the first model, we were able to obtain an average BLEU score, ROUGE-1 score, ROUGE-2 score, ROUGE-L score, and cosine similarity score of 0.10, 0.33, 0.14, 0.31 and 0.73 respectively. For the second model BLEU, ROUGE-1, ROUGE-2, ROUGE-L, and cosine similarity scores were 0.05, 0.23, 0.09,0.19, and 0.74 respectively.

Keywords: Answer, Context, Generation, Questions, Transformers

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BART	Bidirectional Auto-Regressive Transformers
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Understudy
CNN	Convolutional Neural Network
COVID	Coronavirus Disease
DG	Distractor Generation
DG-RACE	Dataset for Gap Reasoning in Multiple Choice Questions
DM	Daily Mail
ESSK	Entity Subtopic Similarity Kernel
GPU	Graphics Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
LDA	Latent Dirichlet Allocation
LSTM	Long Short-Term Memory
MS MACRO	Microsoft Machine Reading Comprehension Dataset
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RNN	Recurrent Neural Network
SQuAD	Stanford Question Answering Dataset
T5	Text-to-Text Transfer Transformer
TPU	Tensor Processing Unit
QA	Question Answering
QG	Question Generation

1. INTRODUCTION

Question answering has been a fundamental challenge within the field of natural language processing (NLP) and has long been taken into consideration a giant milestone in artificial intelligence. QA systems play a crucial role in permitting users to pose questions in ordinary language and get hold of accurate answers. In this mission, our goal is to create a version that not only provides answers but also generates both questions and their corresponding solutions from a given paragraph in the fields of Renewable Energy, Pollution, and Environmental Science. Such models need to possess a deep understanding of language shape, semantics, context, and the capacity to perceive applicable answer terms within the textual content.

Generating questions and solutions in renewable energy, pollution, and environmental sciences enhances active learning, supports research, enables knowledge transfer, is simple, and acts as a valuable tool for increasing understanding and enhancing critical thinking and discovery in these works.

The emergence of e-learning has transformed the acquisition of knowledge in the modern world. With so many learning resources available, e-learning provides students with a dynamic and ever-changing environment. There is great potential for question generation in this context. A key function is the ability to quickly generate questions and their corresponding solutions based on each paragraph. This ability will greatly help teachers create chapter-specific worksheets or quizzes, reduce their workload and allow them to focus on effective personalized instruction.

Developing a question generation model and an answer generation model is a challenging task in the field of Natural Language Processing The model must be able to understand the underlying linguistic structure, capture the semantic nuances of both the context and the questions being generated, identify the appropriate location within the paragraph to derive the answer. Our goal is to improve language generation models and their useful applications in education by tackling these complexities.

1.1 Background

In recent years, the field of natural language processing (NLP) has witnessed significant advancements. It has enabled machines to understand and generate human language more effectively. One of the key challenges in NLP is question answering, which involves the development of systems that can grasp questions posed in natural language and provide accurate answers. Traditional question-answering systems were primarily focused on retrieving answers from existing knowledge bases or structured data. However, the generation of both questions and answers directly from unstructured text, such as paragraphs, is more complex.

The development of question-and-answer generation models has gained considerable attention in the research community. This is due to its potential applications in various domains, including education, conversational agents, and information retrieval systems. The COVID-19 pandemic era has brought significant changes to our daily lives, including the way we approach education. In the education sector, generating questions from given content can support instructors in creating assessment materials, designing personalized learning experiences, and promoting active engagement among students. By automating the question generation process, educators can save time and effort, allowing them to focus on other crucial aspects of teaching and learning.

Generating questions and answers from paragraphs requires models to possess a deep understanding of language structure, semantics, and context, and the ability to locate relevant information within the text. This involves capturing the underlying meaning, identifying key entities and relationships, and formulating appropriate questions based on the content. Additionally, the models must have the capability to generate high-quality answers that are relevant and concise.

While significant progress has been made in question generation research, there are still several challenges to overcome. These include the ambiguity and variability of human language, the need for robust semantic representation, the incorporation of domain-specific knowledge, and the evaluation of the generated questions and answers. Therefore, our project aims to contribute to the advancements in question-and-answer

generation from paragraphs by developing a model that can effectively generate contextually relevant questions and their corresponding accurate answers.

By addressing these challenges, our project seeks to explore the potential of question-answer generation models in the e-learning domain and contribute to the ongoing research efforts in NLP. The developed model will be evaluated using appropriate metrics and human assessment to measure its performance, effectiveness, and usability.

1.2 Motivation

Continuous feedback is essential for effective learning, and examinations serve as a valuable source of feedback for students. However, the manual generation of questions for assessments is a time-consuming and labor-intensive task that requires expertise in the subject matter. To address this challenge, there is a need for an automated system that can generate questions directly from provided texts, streamlining the process. Such a system would enable students to engage in self-paced learning, reinforce their understanding of the material, and facilitate efficient assessment preparation. By developing an automated question-answer generation model, our project aims to enhance the learning experience by providing educators and students with an accessible and comprehensive assessment tool.

1.3 Problem Definition

Regular and systematic examination is crucial for evaluating students' progress and understanding. However, the process of manually generating questions for examinations is both laborious and time-consuming. So, we aim to address this challenge by making a system to automatically generate diverse question-answer pairs directly from paragraphs. By doing so, we aim to streamline the exam preparation process and provide students with a comprehensive set of questions that assess their understanding of the subject matter. Also, by implementing this approach, we strive to optimize the exam preparation process for educators and equip them with an extensive collection of questions that effectively evaluate students' grasp of the topic at hand.

1.4 Objectives

The following are the objectives of our project: -

- To enhance the performance of question-answer pair generation by collecting the quality dataset
- To develop a novel approach for question-answer pair generation using two T5 models

1.5 Scopes and Applications

Our project mainly focused on question-and-answer generation under the domain of science and technology. It aims to facilitate the automatic generation of relevant questions and concise answers based on a given paragraph. This technology can greatly benefit various applications as below:

- Automation of assessment material: It can automate the creation of assessment material for educational platforms. By generating relevant questions and answers, it reduces the time and effort required by educators to develop quizzes, exams, and other assessment materials.

- Creation of study guides: It can be utilized to automatically generate study guides. By extracting key information from a given paragraph and formulating questions, it provides students with a comprehensive and structured resource for their learning process.

- Interactive content development: Our project enables the creation of interactive content for online learning systems. Generating questions related to a specific topic, increases active engagement and participation, enhancing the overall learning experience for students.

- Support for knowledge-based applications: The technology can be integrated into knowledge-based applications, providing users with instant access to relevant information. By generating questions and answers, it assists users in quickly finding the information they need, promoting efficient knowledge retrieval and exploration.

While talking about the scope of our project, we specifically target the Renewable Energy, Pollution, and Environmental Science domain. By training our model on renewable energy and Environmental science, we ensure that it generates questions and answers that are accurate and relevant to this field. The project's scope also encompasses the extraction of three questions and three answers from a given paragraph, limiting each answer to a maximum of 50 words. Furthermore, our model has a word limit of 512 words, which sets the boundary for the length of the input paragraph.

Since it focuses on the Renewable Energy, Pollution, and Environmental Science domains, the model can provide more accurate and contextually appropriate questions and answers. Its ability to provide users with multiple questions and answers, another strength is our approach to extractive generation, which ensures that the answers are derived directly from the given paragraph, increasing their reliability. Additionally, by allowing users to input their answers and comparing them with the model-generated questions, we promote active learning and engagement. While our project offers valuable capabilities, it also has certain limitations. The major weakness is the word limit of 512 words, which restricts the length of the input paragraph. Moreover, the extractive generation approach may sometimes result in answers that are less comprehensive or cannot generate original insights. while our question-and-answer generation system demonstrates strengths in providing multiple questions and answers within the science and technology domain, it is important to consider its limitations in handling complex inference tasks, input size constraints, and potential domain-specific limitations.

1.6 Report Organization

In the introduction section, we emphasized the significance of “Automatic Question Answer Generation”. We also highlighted that our model has been trained in three key domains: “Pollution”, “Environmental Science”, and “Renewable Energy”.

A literature review is a critical analysis and evaluation of existing research studies, articles, books, and other sources related to a particular research topic. It provides an overview of the current state of knowledge on the topic and helps researchers identify gaps in the existing literature. The literature review also serves as a foundation for the

research, providing a context and rationale for the study. By analyzing and synthesizing the findings of previous studies, researchers can identify patterns, inconsistencies, and areas for further investigation. Additionally, a literature review allows researchers to gain insights into the methodologies and research designs used by others in the field, which can inform their research approach.

The requirement analysis section of a project is an essential component that provides critical information about the feasibility of the project on multiple fronts, including operational, financial, schedule and legal. This section serves as a foundation for the development of a successful project by outlining the objectives and goals of the project, assessing the available resources, and identifying the constraints and challenges that may arise during the implementation process. The hardware and software requirements of the project are also identified in this section, ensuring that the project team has a clear understanding of the tools and technologies needed to meet project objectives. By conducting a comprehensive requirement analysis, we have ensured that our project is well-planned, well-structured, and well-resourced, setting the stage for a successful outcome.

The section titled “System Architecture and Methodology” presents the approach utilized for the development of our project, including a block diagram, flowchart, algorithms, and the model employed. The methodology involved training and fine tuning two models. The first model would then generate questions and the second model would generate answers.

In implementation details, we provided a comprehensive and detailed description of how the project was designed and executed, and how the different components of the project were integrated to achieve the desired outcome.

The presentation and interpretation of the data and conclusions produced by the project are normally the main goals of the results and analysis part of a project report. To aid readers in understanding the findings, this section frequently includes tables, graphs, and charts.

Future enhancements focus on further improvements or iterations of the project. These enhancements for future work are based on insights gained during the project's

implementation and analysis, and they help to identify areas where the project could be improved or expanded.

The conclusion provides a summary of the project's key findings, outcomes, and contributions.

Appendices are typically used to provide additional detail or supplementary information that is not included in the main text of the document, but which may be of interest or relevance to certain readers.

References include detail about sources and materials that were used or consulted during the research and development phases.

2. LITERATURE REVIEW

For the survey, databases of IEEE Xplore, Google Scholar, and Articles were searched manually, by using various keywords like “Text Summarization”, “Question Generation”, “Question Answering”, “Distractor Generation”, etc. Research papers of various Text Processing Approaches were studied from the mentioned repositories like Journals, Conferences, and Articles. Along with Research Papers, certain existing systems were also reviewed. The research papers were studied and a few comparisons were made based on certain parameters.

In 1995 a paper about the Long Short-Term Memory (LSTM) model proposed the LSTM model as an efficient solution to the problem of storing information over extended time intervals. It addresses the issue of decaying error backflow in recurrent backpropagation, which leads to slow learning. The authors review Hochreiter's analysis from 1991, highlighting the limitations of traditional methods. LSTM overcomes these limitations by employing a novel gradient-based approach. It introduces “constant error carrousels” within specialized units to ensure constant error flow and uses multiplicative gate units to control access to this flow. The LSTM model is characterized by its local spatial and temporal nature, with computational complexity per time step and weight at O(1). The paper presents experiments with artificial data, demonstrating LSTM's success in learning local, distributed, real-valued, and noisy pattern representations. Comparisons with other recurrent network algorithms such as RTRL, BPTT, Recurrent Cascade-Correlation, Elman nets, and Neural Sequence Chunking show that LSTM achieves a higher number of successful runs and faster learning. Additionally, LSTM solves complex long-time lag tasks that previous algorithms were unable to solve. This research showcases the effectiveness and efficiency of LSTM in overcoming the challenges of information storage over extended time intervals [1].

The Transformer architecture proposed in the “Attention is All You Need” research paper [2] revolutionizes sequence transduction models by introducing a simple yet powerful network structure based solely on attention mechanisms, eliminating the need for recurrent or convolutional neural networks. Through rigorous experiments on machine translation tasks, the Transformer model showcases its superiority by

achieving exceptional translation quality with a BLEU score of 28.4 on the WMT 2014 English-to-German task and establishing a new state-of-the-art BLEU score of 41.8 on the English-to-French task, all while requiring significantly less training time and offering improved parallelization capabilities. Additionally, the Transformer's adaptability is demonstrated through a successful application to English constituency parsing, further validating its effectiveness and versatility across various NLP tasks. This groundbreaking research contributes substantially to the existing literature on neural network architectures, offering a highly efficient and high-performing model for sequence transduction that surpasses traditional approaches.

In the paper titled “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer,” the authors delve into the effectiveness and landscape of transfer learning techniques in the field of natural language processing (NLP). They introduce a unified framework that converts diverse text-based language problems into a text-to-text format, enabling a systematic exploration of various transfer learning approaches. The study comprehensively examines pre-training objectives, architectures, unlabeled datasets, transfer methods, and other factors across numerous language understanding tasks. Leveraging their new “Colossal Clean Crawled Corpus” and insights gained from the exploration, the authors achieve state-of-the-art performance on various benchmarks, encompassing summarization, question answering, text classification, and more. To facilitate future research in NLP transfer learning, they generously release their dataset, pre-trained models, and code, encouraging the community to build upon their work and further advance the field [3].

2.1 Previous Works

In a paper published by Yllias Chali, et. al, they build a QG on mainly four steps. In the first step, complex sentences (from the given body of texts) related to a topic were simplified, as it was easier to generate questions from simple sentences. In the next step, named entity information and predicate-argument structures of the sentences were extracted and then used to generate questions. In the third step, LDA was used to identify important subtopics from the given body of texts, and then ESSK was applied to find their similarity with the generated questions. In the final step, a syntactic tree kernel was used, and syntactic similarity between the generated questions and the

sentences present in the body of texts determined the syntactic correctness of the questions. Questions were then ranked by considering the ESSK similarity scores and the syntactic similarity scores [4].

In 2017, [5] proposed two types of question generation approaches, one was a retrieval-based method using convolution neural network (CNN), and the other was a generation-based method using recurrent neural network (RNN) and collected 1,984,401 <A, Qp, Qt> pairs from Yahoo Answers as the training set of the question pattern prediction model, used the dev sets and 871 test sets of SQuAD, MS MARCO, and WikiQA, to evaluate question generation method by performing a vanilla sequence-to-sequence method using the original training sets of these three datasets, where BLEU 4 score was used as the metric obtained as for SQuAD - 12.28, MS MARCO - 10.46 and WikiQA - 2.04.

In 2019, [6] introduced three neural architectures built on top of BERT for question generation tasks. The first one was a straightforward BERT employment, which revealed the defects of directly using BERT for text generation. Another two models were proposed by restructuring BERT employment into a sequential manner for taking information from previous decoded results. Models were trained and evaluated on the recent question-answering dataset SQuAD. Experiment results showed that the model yields state-of-the-art performance which advances the BLEU 4 score of the existing best models from 16.85 to 22.17.

In 2021, [7] proposed an automatic question-answer pairs generation system using pre-trained weights of a state-of-the-art question generation system called ProphetNet to generate the questions, and the BERT model to generate the answers for the generated questions. For the experiment, the SQuAD 2.0 dataset was used which consists of 50,000 additional questions to that of the SQuAD 1.1 dataset, which has 100,000 answerable questions. The question similarity scores for the answerable questions were experimented and found that the question similarity scores were in the range of 0.85 to 1.00. The threshold values were set to be in the range of $0.00 - 0.50$ if the posed question is Irrelevant, $0.50 - 0.85$ if the posed question is Unanswerable, and $0.85-1.00$ if the posed question is Answerable question and found that the answerable questions get Question Similarity scores above 0.90.

In 2022, [8] proposed an interactive reading platform where the user can upload an E-Book get a textual summary, and generate questions like MCQs, fill in the blanks, and one word by performing transformer-based approaches namely the pre-trained T5 model, DistilBERT and DistilBART model. For text summarization, the performance of the two models (T5 and DistilBART) was evaluated based on the evaluation benchmarks like CNN/DM-ROUGE-1, CNN/DM-ROUGE-2, and CNN/DM-ROUGE-L were 28.0, 42.05, 20.34, and 39.40(for T5) and 37.96, 44.16, 21.28 and 40.90(for DistilBART) respectively. For question answering, T5 and DistilBERT were used. Evaluation benchmarks like GLUE, SQuAD-EM, and SQuAD-F1 was recorded as 82.7, 85.44, 92.08(for T5), and 77.0, 77.7, 85.8(for DistilBERT). For question generation, it was found that when 20 samples were tested, a maximum accuracy of 79.435% was achieved.

In 2022, [9] focused on using neural language models for the generation of questionnaires composed of multiple-choice questions, based on English Wikipedia articles as input. The problem was addressed using three dimensions: Question Generation (QG), Question Answering (QA), and Distractor Generation (DG). A processing pipeline based on pre-trained T5 language models was designed and a REST API was implemented for its use. The DG task was defined using a Text-To-Text format and a T5 model was fine-tuned on the DG-RACE dataset, showing an improvement to the ROUGE-L metric compared to the reference for the dataset. The implementation consisted of a set of pre-trained T5 models on QG and QA tasks in conjunction with a processing pipeline, based on the open-source development for QG published on GitHub [10] by using the Transformers Python library [11]. The performance of a text generation system, BLEU (from 1 to 4n-grams) [12] and ROGUE-L (based on the longest common sequence present on the text) [13] were used as metrics, showing 14.91 for ROUGE-L and 14.80, 7.06, 3.75, 2.16 for BLEU (1 - 4) for avg. distractor.

In a paper published by Haoliang Cao, in his thesis, a complete deep learning model for generating questions was proposed. He suggested that the model could generate a simple question using a supplied text segment as the answer, based on an English-language Wikipedia article and a corresponding text passage. The foundation of the model was built upon an encoder-decoder architecture. Through their tests, it was found that the best performance was achieved by using a model with a fine-tuned BERT

encoder and a self-attention decoder. Furthermore, the researchers introduced an evaluation metric specifically designed for assessing the relevance and grammatical accuracy of the generated questions in the question-generating task. Upon analyzing the collected data, they found that this new metric outperformed existing widely used metrics for sequence-to-sequence activities, providing a more effective evaluation method [14].

In the paper [15], the authors made a significant contribution to the field of Automatic Question Generation (AQG) by introducing a novel data-driven approach that combined traditional linguistic methodologies with machine learning techniques. They drew inspiration from the advancements in machine translation and leveraged large multilingual parallel corpora to train the AQG system. Their proposed framework involved preprocessing, feature extraction, and the creation of a specialized data structure to calculate sentence transformation rules for question generation. The generated questions were evaluated using a question estimator module, while reinforcement learning was employed to enhance the system based on explicit feedback. This integrated approach held promise for improving the efficiency and efficacy of AQG systems.

Suzan Verberne proposed a method for answering why-questions by focusing on arguments in texts and using Rhetorical Structure Theory (RST). In his study, he tested this strategy on two sets of why-questions: one from native speakers and another from questions submitted to answers.com. The method successfully found answers for approximately 60% of the why-questions. Verberne discovered that certain relation types had strong predictive power in selecting responses. However, he also found that many questions required an entire paragraph to be answered, leading him to shift the study focus to passage retrieval. He outlined a three-step technique for identifying passages likely to contain the answer: query generation, document retrieval, and paragraph retrieval and rating. Traditional information retrieval techniques were ineffective in ranking paragraphs due to their small size. Verberne also emphasized the importance of incorporating information about the presence of RST relations in the language model used for ranking [16].

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirements

3.1.1 GPU Support

A free environment like Collab or Kaggle with powerful GPU or TPU support is required for coding and executing the model.

3.2 Software Requirement

3.2.1 Python

Python is a high-level programming language that is commonly used to create a wide range of software applications. Python supports a variety of programming paradigms including functional and object-oriented. It has a large and active community that contributes to its extensive ecosystem of libraries and frameworks making it versatile for various applications such as web development, data analysis, and AI.

3.2.2 Natural Language Processing (NLP) Libraries

NLP libraries like spaCy, or Hugging Face's Transformers library perform functions such as tokenization, part-of-speech tagging, named entity recognition, text classification, machine translation, and text generation. These libraries provide pre-trained models and algorithms that enable them to perform language processing tasks and build intelligent applications.

- spaCy: spaCy offers pre-trained models for various NLP tasks including tokenization, named entity recognition, part of speech tagging, and syntactic parsing.

- Hugging Face Transformers: It is a library by Hugging Face that specializes in state-of-the-art models for natural language understanding and generation. It offers pre-trained models that can be fine-tuned for specific tasks like text classification, question answering, and more.

3.2.3 Machine Learning Framework

A machine learning framework such as PyTorch, or scikit-learn to build and train our question-answer generation model. These frameworks provide tools and algorithms for training and deploying machine learning models. It facilitates tasks such as data preprocessing, feature engineering model selection, training, evaluation, and deployment. These frameworks offer APIs and interfaces to handle complex machine-learning tasks efficiently.

- PyTorch: PyTorch is an open-source machine learning library that focuses on providing a dynamic and intuitive approach to building and training neural networks. It offers a more imperative and flexible programming style compared to TensorFlow. PyTorch uses dynamic computation graphs, allowing users to define and modify models on the fly, making it easier to debug and experiment with different network architectures. It provides seamless integration with Python and supports automatic differentiation, making it suitable for tasks such as deep learning research, prototyping, and building custom models. PyTorch also has a strong and active community, with extensive documentation and resources available for learning and development.

- Scikit-learn: Scikit-learn is an open-source machine-learning library in Python that provides a wide range of algorithms and tools for various machine learning tasks. It offers a consistent API, a comprehensive collection of algorithms, data preprocessing and feature engineering capabilities, model evaluation, and validation techniques, pipelines for creating complex workflows, seamless integration with NumPy and SciPy, and extensive documentation and community support.

3.2.4 Data Processing and Visualization Libraries

Libraries such as pandas, NumPy, and Matplotlib are crucial for tasks related to data preprocessing, analysis, and visualization.

- Pandas: Pandas is a powerful library for data manipulation and analysis, providing data structures like DataFrame and Series to efficiently handle

structured data. It offers a wide range of functionalities, including data reading and writing, data cleaning and preprocessing, data transformation, and data analysis. Pandas are widely used for tasks related to data preprocessing, making it easier to clean and structure raw data for further analysis.

- Numpy: Numpy stands for Numerical Python. It is a fundamental library for numerical computing. Its main data structure is the ndarray, which allows fast and vectorized operations on multi-dimensional arrays. NumPy provides a vast collection of mathematical and logical functions, making it ideal for numerical computations.

- Matplotlib: Matplotlib is a comprehensive plotting library that enables the creation of various types of static, animated, and interactive visualizations. Matplotlib supports a wide range of plot types and provides tools for creating high-quality, publication-ready visualizations. It seamlessly integrates with Pandas and NumPy, making it easy to visualize data stored in DataFrame or ndarray objects.

3.2.5 UI Development Frameworks

For deploying the model react is used as the front end and Django is used as the back end.

- React: React is a popular JavaScript library used for building user interfaces. It was developed by Facebook and has gained widespread adoption in the web development community. React allows developers to create reusable UI components and efficiently update and render them based on changes in application data.

- Flask: Flask is a micro web framework for Python, providing tools to build web applications quickly and with minimalistic code. It follows the WSGI (Web Server Gateway Interface) standard and is lightweight, making it easy to learn and use. With Flask, we can create dynamic web pages, handle requests, and integrate various libraries and extensions for added functionality.

3.3 Feasibility Study

3.3.1 Operational Feasibility

Since the project will have an easy-to-use GUI, any person without technical knowledge related to machine learning can access it. Our system will generate the questions and their corresponding solutions, but all the details will be hidden from the end user. The end user will just have to provide the paragraph. Therefore, almost anyone can operate the system. So, we can say that our system is operationally feasible.

3.3.2 Financial Feasibility

All the libraries we need for the completion of our project are open-source. So, there is no significant cost for the completion of our project. Also, the hardware requirements are minimal, so we do not have to buy any additional hardware equipment for the completion of our project.

3.3.3 Technical Feasibility

The availability of suitable technologies and resources necessary for the development and implementation of the question-answer generation model is the main focus of the technical feasibility investigation. It was found that the necessary machine learning models, like transformers and methods for natural language processing, are readily available. The project also made sure that the right computing resources and infrastructure were available to support efficient operation.

3.3.4 Schedule Feasibility

The project can be completed within the allocated period for the major project which is around 9-10 months as per the timelines shown in the Gantt chart.

3.3.5 Legal Feasibility

In the context of the question-answer generation project, potential legal issues may arise if copyrighted material is used as part of the training data or if the generated answers contain copyrighted content. As we have utilized freely accessible books, websites, journals, and other online resources, there are no copyright issues associated with our project.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 Transformers

Before the emergence of Transformers, RNNs were the conventional network architecture used for translation tasks. RNNs process text in a sequential manner, either from left to right or right to left. This sequential processing approach required RNNs to undergo multiple steps to make decisions based on words that were distantly positioned. Choosing the most likely meaning and proper representation of the word “bank” in the phrase “I arrived at the bank after crossing the...”, for example, necessitates knowing whether the sentence ends in “... road” or “... river”.

For instance, when examining words such as “bank” and “river,” an RNN would need to scan through each intermediate word step by step to infer that “bank” likely refers to the bank of a river. Previous research has demonstrated that as the number of such intermediary steps increases, it becomes increasingly challenging for a recurrent network to effectively learn how to make such determinations.

On the other hand, the Transformer algorithm stands out with its efficient and powerful approach to language modeling. Unlike traditional models that rely on sequential processing, the Transformer takes a constant number of steps, which is determined through empirical evaluation. This innovative architecture leverages a self-attention mechanism at each stage, enabling it to establish direct connections between all words in a sentence, regardless of their positional order. This capability proves particularly useful in resolving contextual ambiguities. For instance, consider the sentence, “I arrived at the bank after crossing the river.” In this scenario, the Transformer can quickly learn to focus its attention on the word “river” and instantaneously make the inference that the word “bank” refers to the shore of a river, rather than a financial institution. This ability to capture intricate relationships in language and make complex decisions in a single step contributes to the Transformer's remarkable effectiveness in various natural language processing tasks.

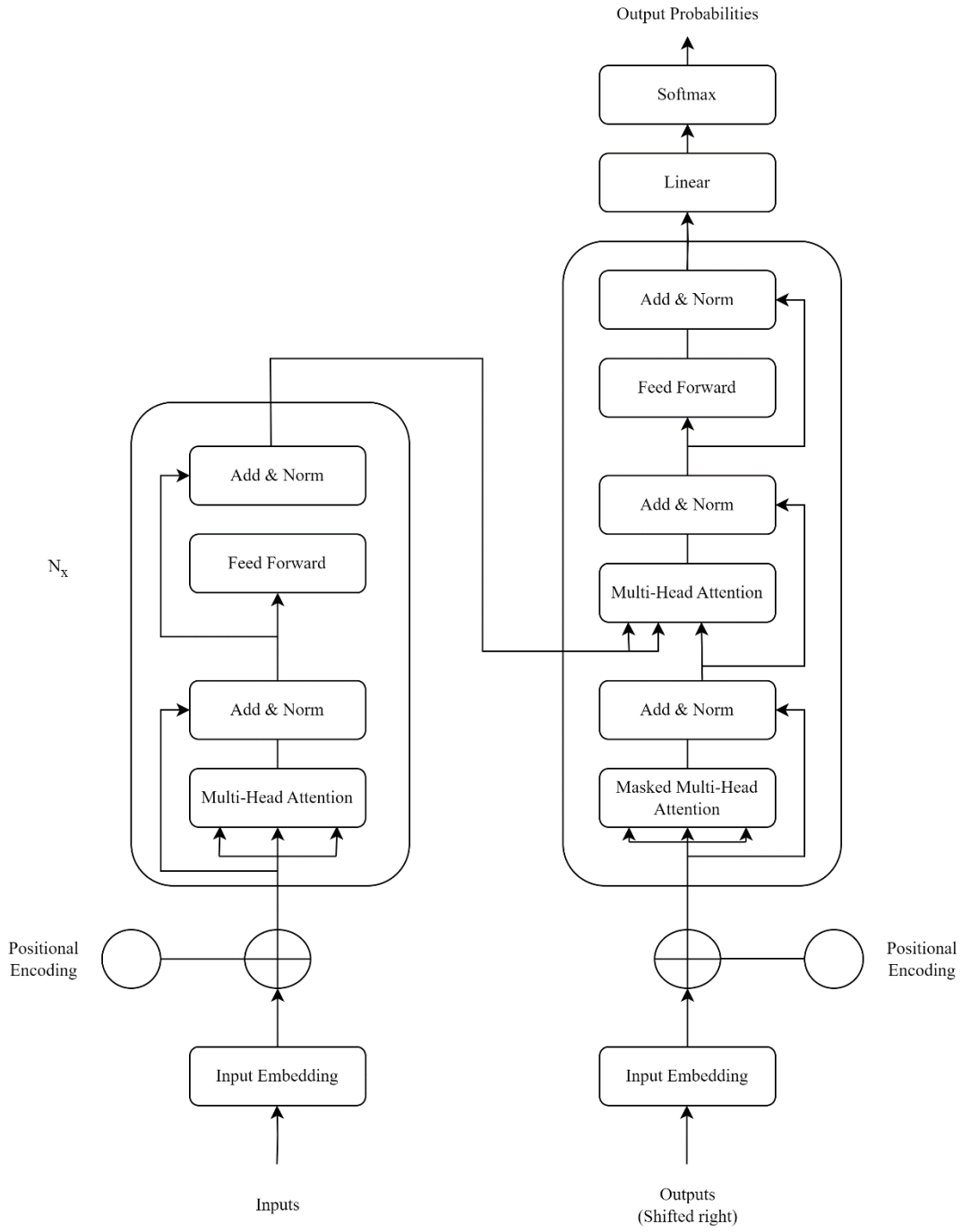


Figure 4-1. Transformers Architecture

4.1.1 Embedding Layer

In the domain of natural language processing (NLP), an initial step involves the conversion of individual words into vectors through the utilization of an embedding algorithm. This embedding process exclusively occurs within the lowermost encoder. Common to all encoders is the fundamental concept that they are provided with a

collection of vectors, each possessing a dimensionality of 512. Within the lower encoder, these vectors correspond to the word embeddings, while in subsequent encoders, they relate to the output derived from the encoder immediately beneath. The size of this vector collection represents a hyperparameter that can be adjusted and is typically determined by the longest sentence present within the training dataset. Notably, the output yielded by the embedding layer is subsequently transmitted to the self-attention layer within the encoder.

4.1.2 Positional Encoding

Positional encoding is a technique that assigns a unique representation to each position or location of an entity in a sequence. In transformer models, a single number like the index value is not employed to depict an item's position for several reasons. One of these reasons is that for long sequences, index values can become very large. Attempting to normalize the index values to a range between 0 and 1 can pose difficulties for sequences of varying lengths, as they would end up being normalized differently.

Transformers adopt a clever positional encoding scheme in which every position or index is associated with a vector. Consequently, the outcome of the positional encoding layer is a matrix in which each row signifies an encoded object from the sequence combined with its positional information.

Diving right into this, let's consider an input sequence of a certain length and the need to determine the position of the kth object within this sequence. The positional encoding in this context is achieved through the utilization of sine and cosine functions with varying frequencies:

$$P(k, 2i) = sin\left[\frac{k}{n^{2i/d}}\right] \tag{4-1}$$

$$P(k, 2i+1) = cos\left[\frac{k}{n^{2i/d}}\right] \tag{4-2}$$

Here,

k: Position of an object in the input space, $0 \leq k \leq L/2$

d: Dimension of output embedding space

P(k,j): Position function for mapping a position k in the input sequence to index (k,j) of the positional matrix

n: User-defined scalar, set to 10000 initially.

i: Used for mapping to column indices $0 \leq i \leq \frac{d}{2}$, with a single value of i maps to both sine and cosine functions

Here, equation (4-1) is used for even positions and equation (4-2) is used for odd positions.

4.1.3 Encoder

A series of encoders function as the encoding component. It takes positional encoding output and embedding layer output as its input. These encoders share a consistent structure, each comprising two layers: a feed-forward neural network and self-attention. The initial step in the encoding process involves passing the inputs of the encoder through a self-attention layer. This layer facilitates the encoder's ability to consider other words within the input phrase while encoding a single word. The outputs of the self-attention layer are subsequently forwarded to a feed-forward neural network. The same feed-forward network is applied to each point. The encoder receives a list of vectors as input, which it analyzes by passing these vectors through a 'self-attention' layer, followed by a feed-forward neural network, and ultimately to the subsequent encoder.

4.1.4 Self-Attention

The key aspect of the Transformer architecture lies in its utilization of self-attention, which enables a more comprehensive understanding of relevant words within the context of the currently processed word. This self-attention mechanism involves the creation of three vectors for each input vector of the encoder: the Query vector, the Key vector, and the Value vector. These vectors are derived by multiplying the input embedding with three specifically trained matrices. These new vectors have a smaller dimensionality compared to the embedding vector, specifically set at 64, while the embedding and encoder input/output vectors maintain a dimensionality of 512. This decision to reduce the dimensionality is not mandatory, but rather a deliberate

architectural choice aimed at facilitating computationally efficient multiheaded attention computations. By incorporating self-attention and optimizing the vector dimensions, the Transformer architecture achieves impressive performance and scalability in a wide range of natural language processing tasks.

The scoring process plays a crucial role in the second stage of self-attention computation within the Transformer architecture. In this stage, each word in the input sentence is compared to the specific term being considered for the self-attention score calculation. As a word undergoes encoding at a particular position, the score determines the level of significance assigned to other segments of the input text. To calculate the score, the dot product of the query vector and the key vector associated with the word being evaluated are computed.

- Query: The query represents a vector that is used to seek relevant information from the input sequence.

- Key: The key represents a vector associated with each input token in the sequence. It serves as a mechanism to encode the input sequence's information.

- Value: The value vectors are associated with each input token as well, and they represent the information that will be used to produce the output of the attention mechanism.

For instance, when performing self-attention for the word at index #1, the initial score is obtained by taking the dot product of the query vector q1 and the key vector k1. Similarly, the second score is generated by computing the dot product of q1 and k2. This scoring mechanism allows the Transformer to assign attention weights to different words in the input, enabling it to capture meaningful relationships and dependencies among them during the encoding process.

The third and fourth phases of self-attention computation in the Transformer involve a crucial normalization step. In these phases, the calculated scores are divided by the square root of the dimension of the key vectors being utilized. This normalization process helps to stabilize the gradients and prevent them from exploding or vanishing.

The resulting normalized scores are then passed through a softmax operation, which transforms them into a probability distribution. This ensures that all scores are positive and sum up to one, reflecting the degree to which each word contributes in the given context.

Moving to the fifth step, the softmax scores obtained are multiplied by their corresponding value vectors. By doing so, the Transformer effectively preserves the values associated with the words that are the focus of attention while minimizing the influence of irrelevant words. This weighting process allows the model to emphasize the most relevant information.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4-3}$$

Where,
Q, K, and V are Query, Key, and Value matrices respectively.
k and d represent the key and dimensionality for each word respectively.

In the final step of this self-attention computation, the weighted value vectors are added together. This aggregation results in the output of the self-attention layer at this stage, specifically for the first word. The entire process of scoring, normalizing, weighting, and aggregating enables the Transformer to capture the contextual relationships between words and produce meaningful representations for each word in the input sequence.

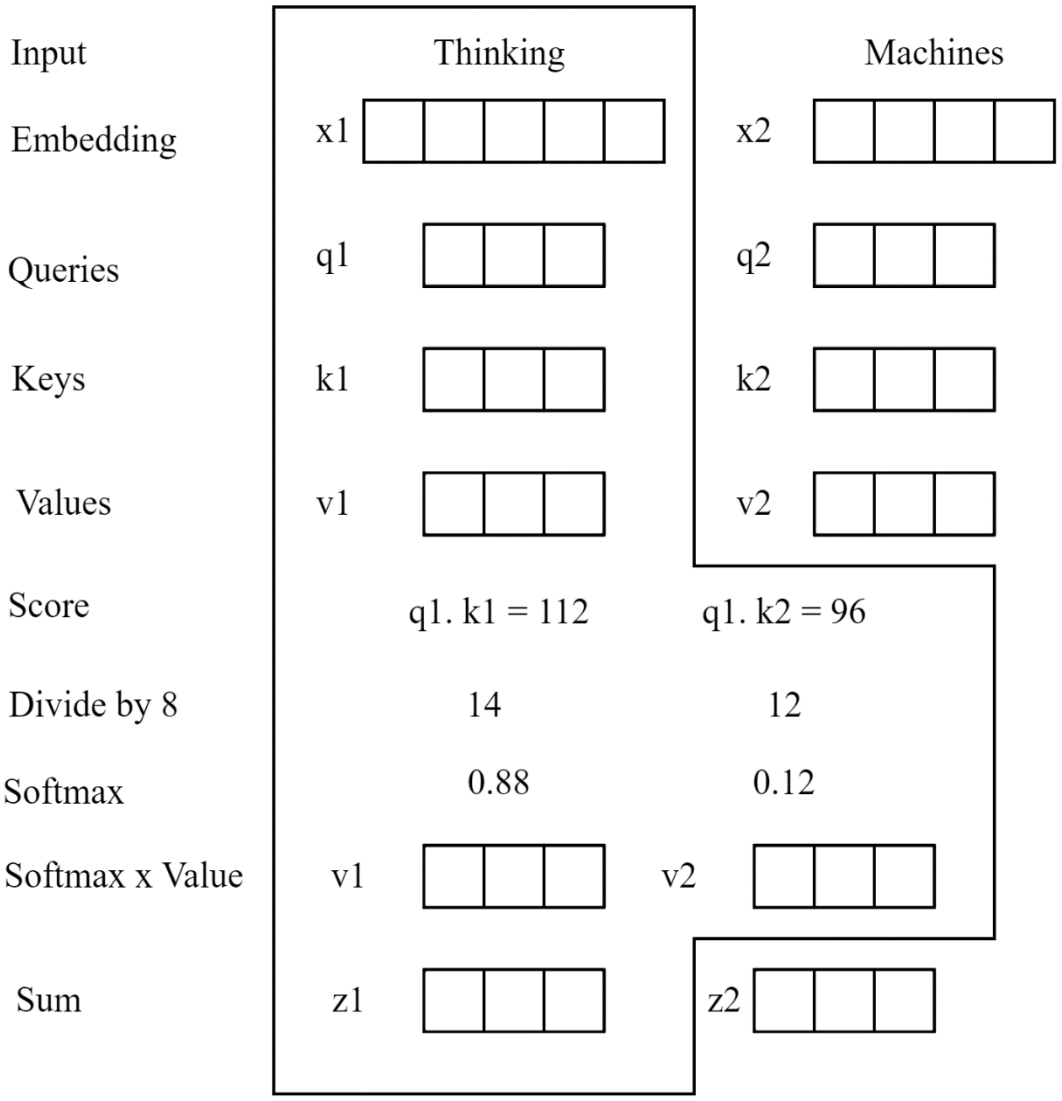


Figure 4-2. Self-Attention for the first word

4.1.5 Matrix calculation for self-attention

First and foremost, it is necessary to calculate the Query, Key, and Value matrices. The embeddings are then packed into a matrix X, which is subsequently multiplied by the weight matrices obtained through training (WQ, WK, WV). Lastly, when dealing with matrices, it is possible to consolidate steps two through six into a singular formula for the computation of the self-attention layer outputs.

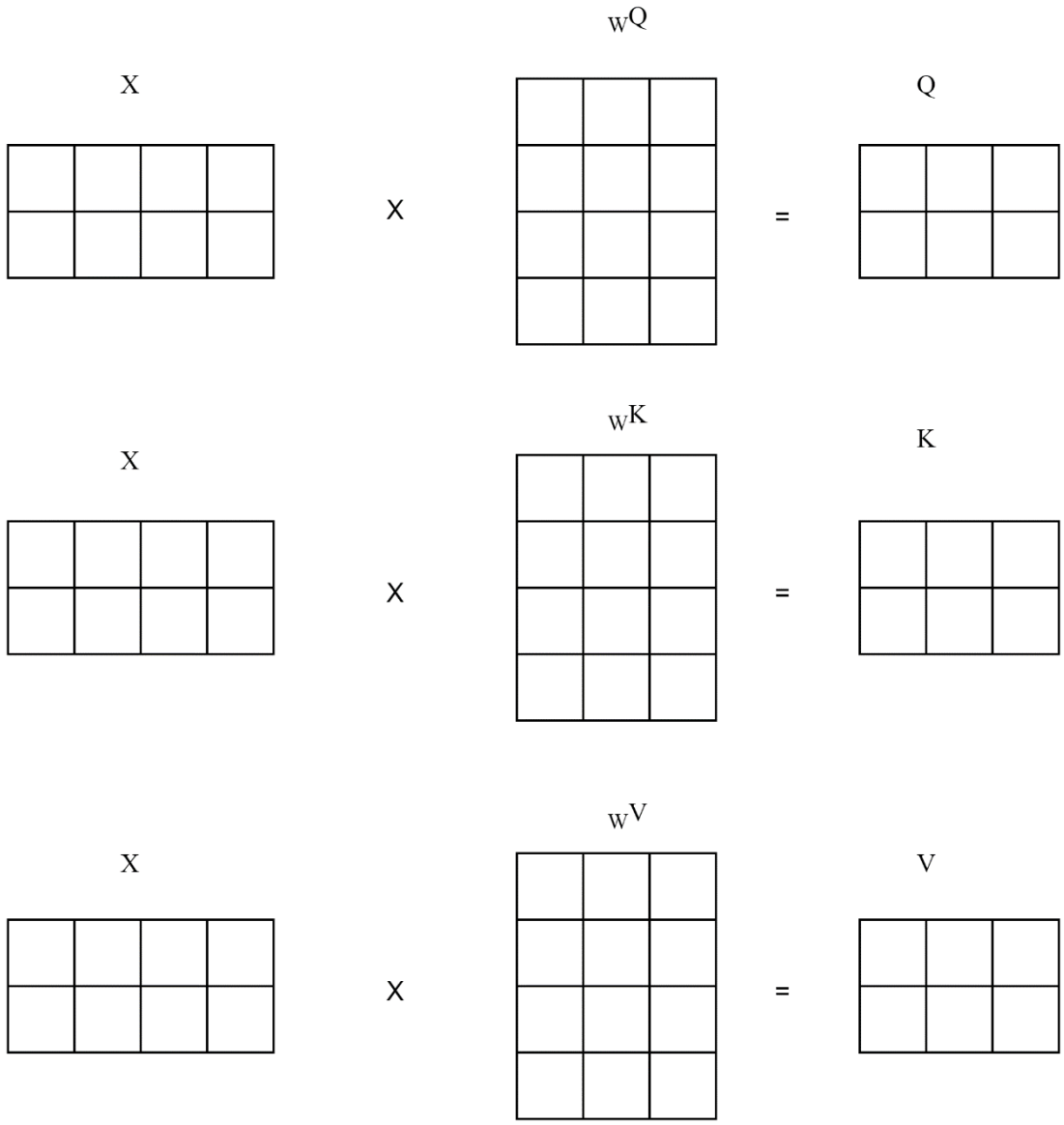


Figure 4-3. Weight, Key, and Value Matrices

4.1.6 Multi-head Attention

It expands the model's ability to focus on different positions. In the example above, z1 contains a little bit of every other encoding, but it could be dominated by the actual word itself. When translating a sentence such as "The animal didn't cross the street because it was too tired," it becomes important to determine the referent of the word "it."

The attention layer plays a crucial role in creating different "representation subspaces" through the use of multi-headed attention. In multi-headed attention, multiple sets of

weight matrices for Query, Key, and Value are employed. In the Transformer model, eight attention heads are used for each encoder and decoder, resulting in eight sets of weight matrices. These sets are initialized randomly and are utilized during the training process to transform the input embeddings or vectors from lower encoders/decoders into distinct representation subspaces.

The self-attention computation, as described above, is iterated eight times with different weight matrices, resulting in eight distinct Z matrices. However, this poses a challenge as the feed-forward layer is specifically designed to process a single matrix (comprising vectors for each word) rather than eight matrices. Consequently, there arises a need for a mechanism to consolidate these eight matrices into a singular matrix.

To address this, the matrices are concatenated and subsequently multiplied by an additional weight matrix WO. To address this, the matrices are concatenated and subsequently multiplied by an additional weight matrix WO.

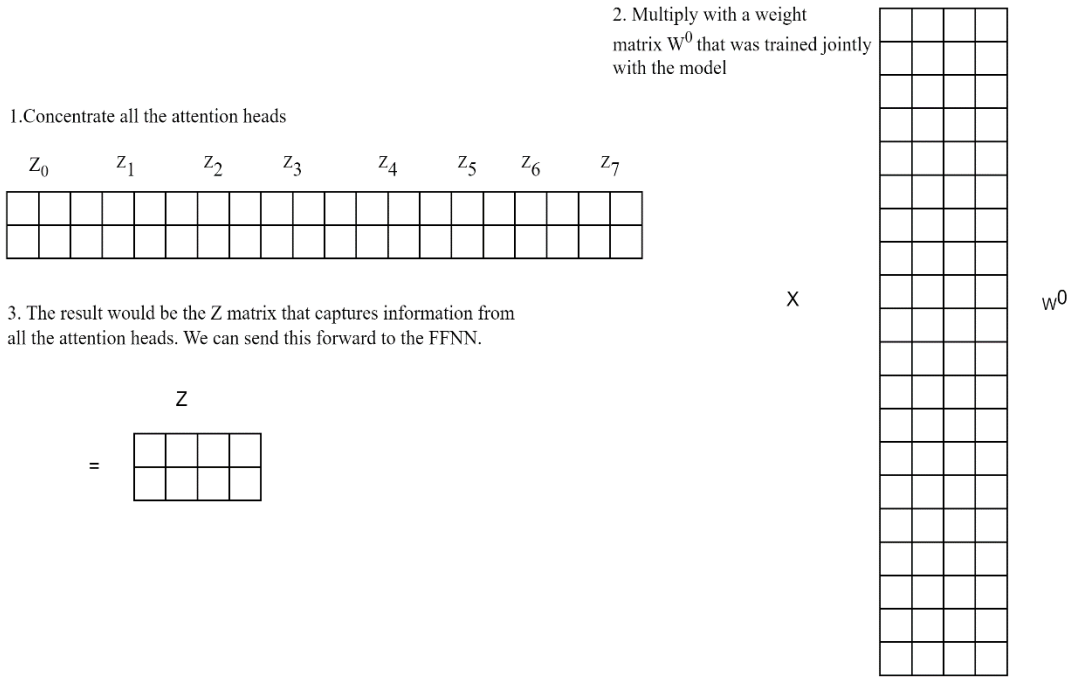


Figure 4-4. Multi-head attention

Positional encoding: In the Transformer, vectors are appended to the input embeddings. These vectors follow a predefined pattern that the model learns, allowing it to calculate

the location of each word in the sequence or the distance between distinct words in the sequence. The idea is that incorporating these values into the embeddings results in significant distances between the embedding vectors when projected into Q/K/V vectors and during dot-product attention.

4.1.7 Layer Normalization

Before progressing further, it is essential to mention a crucial feature in the design of the encoder: each sub-layer in every encoder incorporates a residual connection surrounding it, which is subsequently followed by a layer-normalization step. When depicting the vectors and the associated layer-norm operation related to self-attention, they are presented as follows:

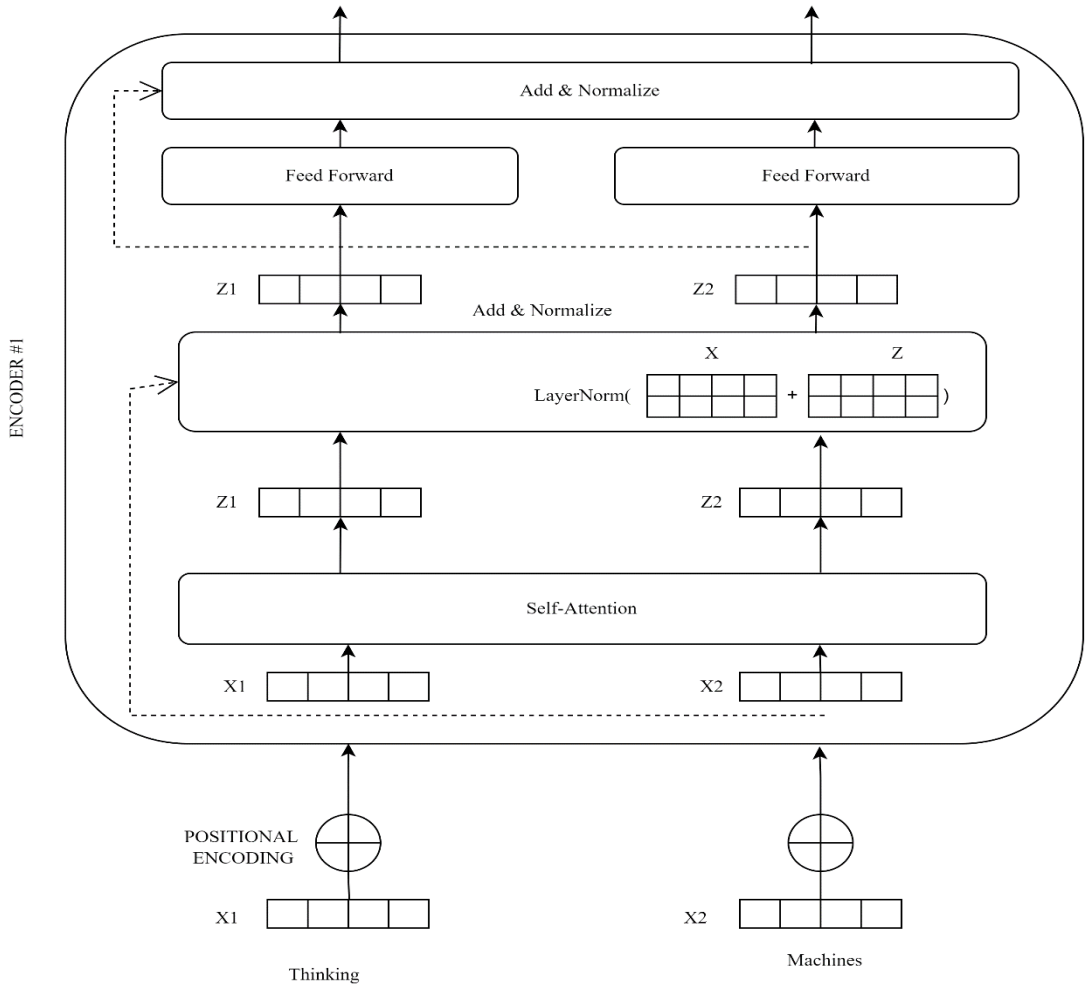


Figure 4-5. Layer Normalization

The process is iterated until a distinct symbol is observed, indicating the completion of the output by the transformer decoder. Subsequently, the output of each step is forwarded to the subsequent time step as input to the lower decoder, thereby propagating the decoding outcomes in a manner consistent with the behavior of the encoders. Similar to the encoder inputs, the decoder inputs are endowed with positional encoding to signify the positional information of each word.

Nevertheless, the self-attention layers in the decoder operate in a distinctly different manner compared to those employed in the encoder:

The self-attention layer in the decoder restricts its attention solely to preceding positions in the output sequence. This restriction is enforced by masking future positions before the softmax stage of the self-attention computation.

4.1.8 Final and Softmax Layer

The decoder stack outputs a vector of floats. The job of the final Linear layer is to turn the output of the decoder into a word.

The Linear layer serves as a fully connected neural network responsible for projecting the vector generated by the decoder stack onto a significantly larger vector referred to as the logits vector.

These scores are then transformed into probabilities through the softmax layer. The softmax function processes a vector z containing K real numbers, transforming it into a probability distribution comprising K probabilities that are proportional to the exponential values of the input numbers. Mathematically, the softmax function is defined as,

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4-4}$$

The function's input consists of the outcomes of K separate linear functions, and the estimated probability for the j^{th} class when provided with a sample vector x and a weight vector w is calculated as follows:

$$P(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_j}} \quad (4\text{-}5)$$

The cell with the highest probability is subsequently selected, and the word associated with it is generated as the output for that particular time step.

4.2 T5 model

T5 is a state-of-the-art natural language processing (NLP) model developed by Google Research. It is based on transformer architecture and has since become the foundation for many other NLP models. The T5 transformer model has the same standard encoder-decoder structure as standard transformer models. It consists of 12-pair blocks of encoder-decoder. Each block contains self-attention, a feed-forward network, and optional encoder-decoder attention. It's trained on a massive amount of text data, which allows it to understand and generate a wide range of natural language. One of the key innovations of T5 is its "prefix" approach to transfer learning, where the model is fine-tuned for a specific task by training it with a prefix added to the input text.

The architecture shown below in Figure 4-6. T5 Transformer Architecture is the structure of the T5 transformer model, which is used to translate sentences. The transformer consists of an encoder block and a decoder block. The encoder block compresses information about the sentence structure and sends it to the decoder block. The decoder block uses this information as a reference to translate the input sentence. The transformer model solves the limitations of the Seq2Seq model by utilizing contextual information with an attention mechanism.

To process the input sentence, it goes through an embedding process and positional encoding. There are three embedding layers defined:

- shared: An embedding layer shared between encoder and decoder, mapping tokens to continuous vector representations of size 768.

- Encoder, embed_tokens: An embedding layer specific to the encoder, mapping input tokens to continuous vector representations of size 768.

- Decoder, embed_tokens: An embedding layer specific to the decoder, mapping output tokens to continuous vector representations of size 768.

The encoder consists of a stack of T5 blocks, each containing:

- A self-attention mechanism with linear transformations (without bias) followed by layer normalization and dropout with a rate of 0.1.

- A feed-forward neural network with ReLU activation and linear transformations (without bias) followed by layer normalization and dropout with a rate of 0.1.

- The first block also includes a relative attention bias embedding (relative_attention_bias) for handling positional information.

The decoder also consists of a stack of T5 blocks, each containing:

- A self-attention mechanism with linear transformations (without bias) followed by layer normalization and dropout with a rate of 0.1.

- Cross-attention mechanism with linear transformations (without bias) followed by layer normalization and dropout with a rate of 0.1.

- A feed-forward neural network with ReLU activation and linear transformations (without bias) followed by layer normalization and dropout with a rate of 0.1.

Both encoder and decoder end with a layer normalization and dropout layer with a dropout rate of 0.1.

Each word token in the input sentence generates three types of vectors: representation (Query), Keys for identifying relevant words, and Values for the actual word representation. These vectors undergo self-attention processes, where the query is multiplied by the keys to calculate a score indicating significance. The attention weights are then multiplied by the values to determine the relevance among words in the input sentence. To aid model learning, the input and output vectors are added together as a residual connection, and the scores are normalized.

The lm_head linear layer generates the output sequence. It maps the final output of the decoder to a vector of size 32128 (vocabulary size) without bias.

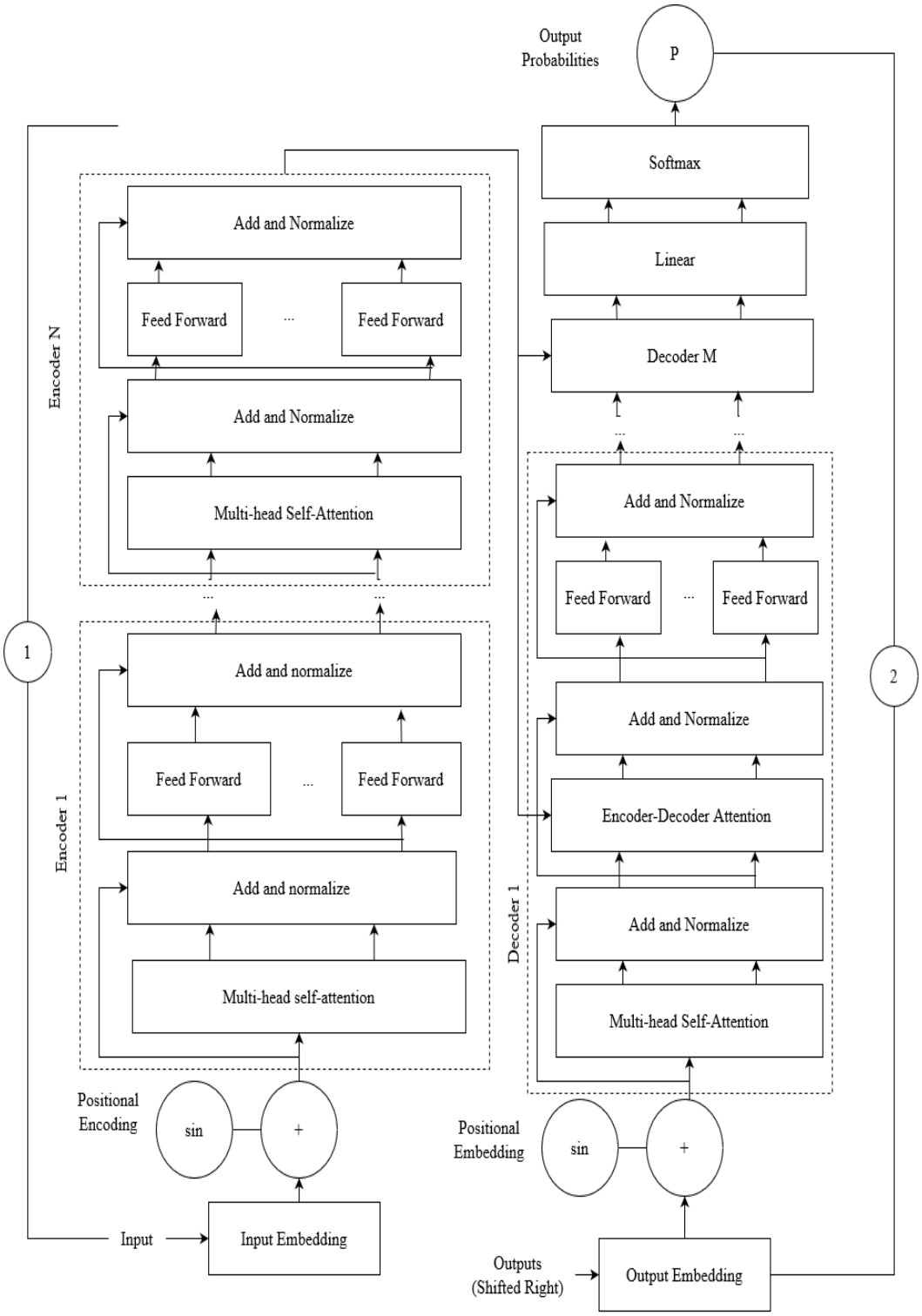


Figure 4-6. T5 Transformer Architecture

The decoder block translates the input sentence or modifies its structure using information from the encoder block. It consists of two types of self-attention processes: masked self-attention within the decoding block and attention between the encoder's output and the decoder. Unlike the self-attention process, which includes all word tokens in the input sentence, the decoding block only incorporates the attention score of word tokens positioned before each target word token. This means that predictions for successive words are made without information about words following the target word during the decoding process. The encoder's output is used to determine the significance and correlation of words within the input sentence of the decoder block. The decoder attention score is influenced by the attention within the encoder. Finally, the output values from multiple decoder blocks pass through a linear layer and softmax to produce the final probability value for the word order.

4.3 System Architecture

The proposed methodology aims to implement a system for generating questions from paragraphs and checking the similarity between user-provided answers and generated answers. The system will be using a special language model called the T5 transformer, a state-of-the-art language model known for its effectiveness in natural language processing tasks. By fine-tuning the T5 model, the system can efficiently navigate potential word choices and generate relevant questions and answers. We will also use a combination of both Top-K and Top-p sampling techniques to generate questions with our T5 model. This strategy typically leads to more natural-looking text. Additionally, the system incorporates text similarity metrics to quantitatively evaluate the similarity between user-provided answers and generated answers. This comprehensive approach offers a user-friendly solution for question generation and answer similarity checking, contributing to improved accuracy and user satisfaction.

In our project, the first step will involve data collection, where we will gather a dataset containing paragraphs, corresponding questions, and their respective answers. To prepare the data for training, we will perform preprocessing tasks such as removing stop words, handling missing values, etc. After this, we will perform a train-test split to create separate sets for training and evaluation. We will also employ tokenization to convert the textual data into a suitable format for the models to process.

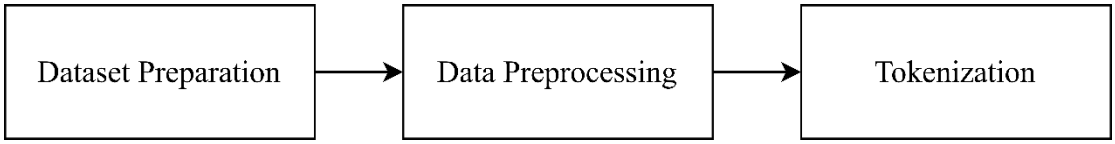


Figure 4-7. Initial Steps

Next, we will train two distinct models. The first model will be trained using paragraph-question pairs as input. This training will enable the model to learn the relationship between a paragraph and the associated questions. Subsequently, when provided with a paragraph, the model will be able to generate questions related to that particular context.

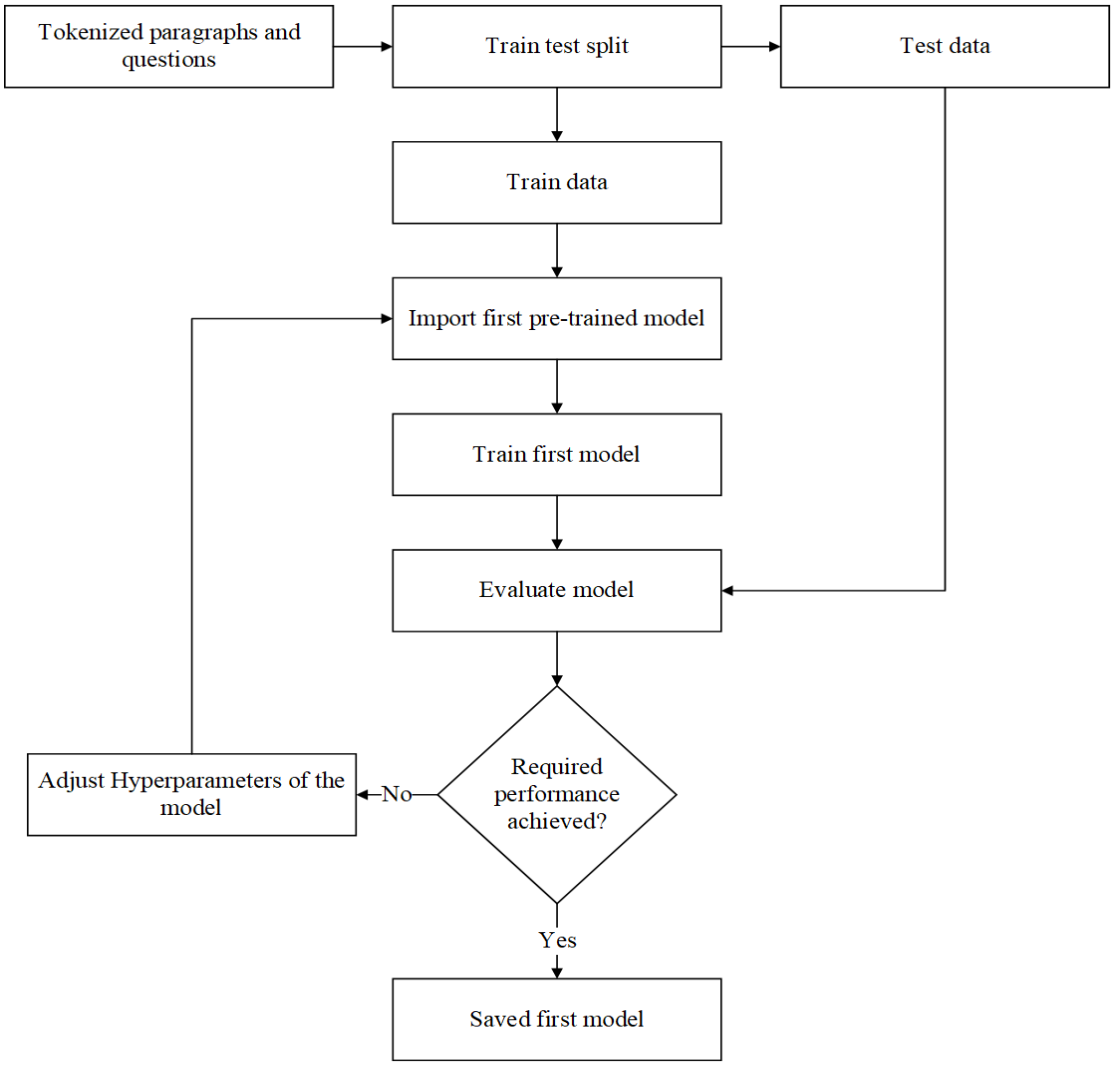


Figure 4-8. Block diagram for training first model

On the other hand, the second model will be trained using paragraph-question-answer triplets. This training process will allow the model to understand the connections between a paragraph, its associated question, and the corresponding answer. Consequently, when given a paragraph and a question, this model will generate an answer based on the context provided.

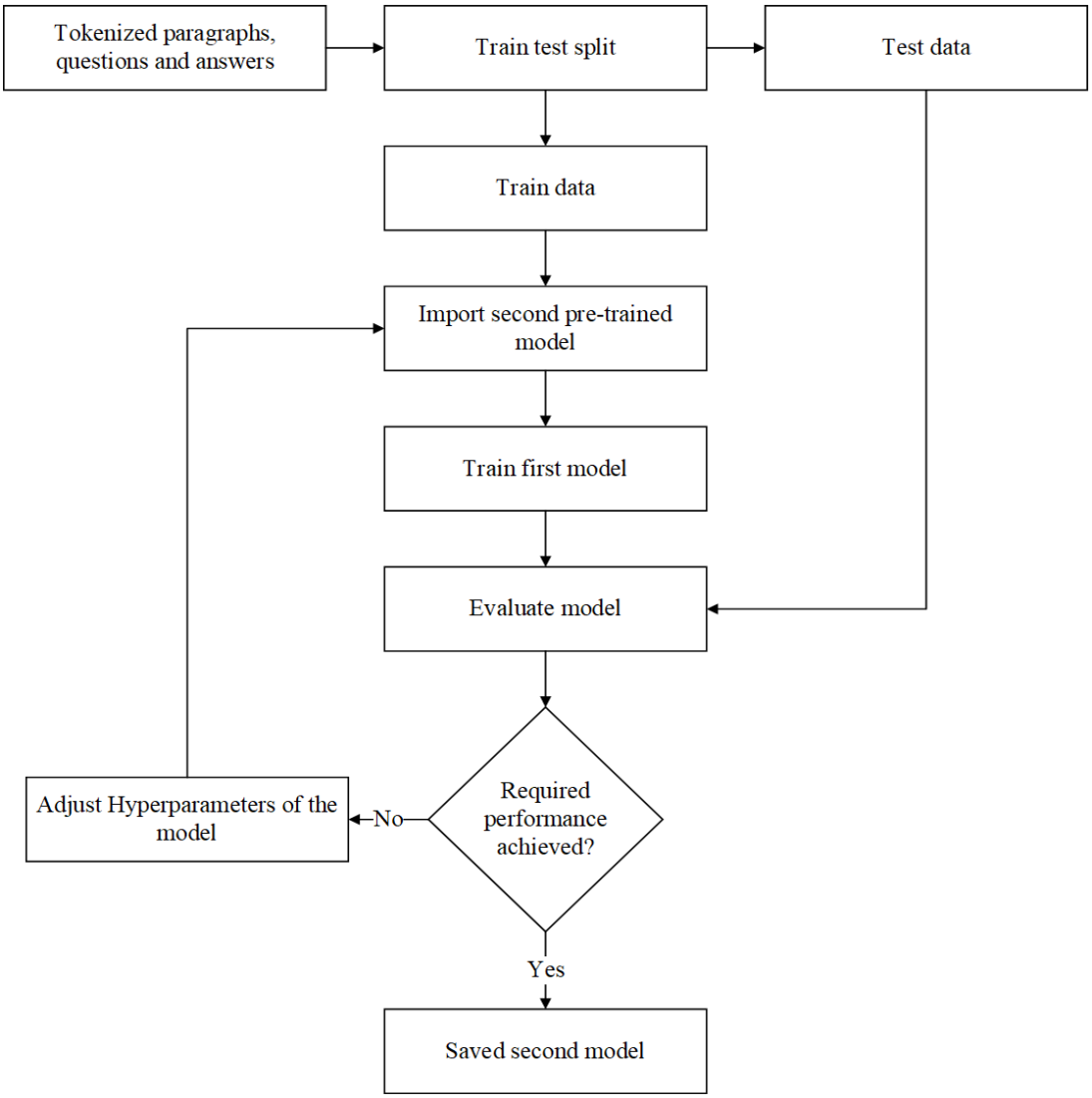


Figure 4-9. Block diagram for training the second model

During the user interaction phase, the user will be prompted to write a response to a specific question generated by the first model. The system will utilize this input to generate an answer, leveraging the question from the first model. Subsequently, a comparative analysis will be conducted between the user's answer and the answer generated by the second model, enabling an assessment of the accuracy and relevance

of the user's response. Ultimately, the system will present both the questions generated by the first model and the corresponding answers generated by the second model, delivering a comprehensive and interactive solution for question generation and answer evaluation in line with the provided paragraph.

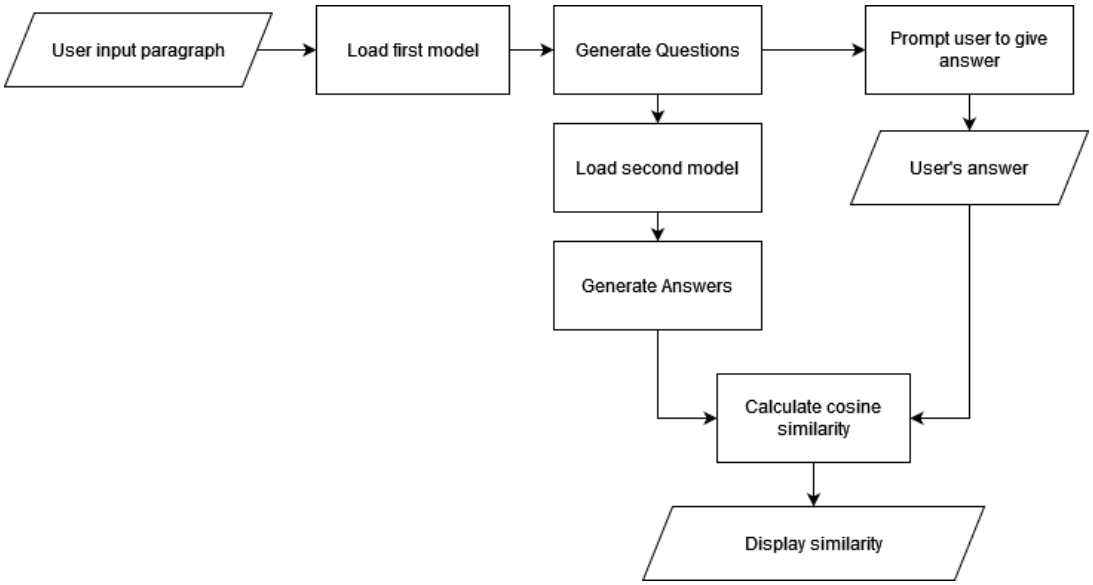


Figure 4-10. User Interaction Phase

4.4 Description of Working Principle

4.4.1 Dataset preparation

The initial phase of our project will involve collecting a diverse range of paragraphs from various reliable sources within the domain of science and technology. We will ensure that the collected paragraphs cover a wide range of topics, providing a comprehensive understanding of the subject matter. Alongside the paragraphs, we will prepare respective questions and answers to create a high-quality dataset specifically tailored for training our question generation and answer evaluation models. The primary focus during this phase will be to guarantee the dataset's quality and relevance, recognizing its main role as the foundation for effective model training.

To ensure a well-rounded dataset, we will design the questions to cover various angles. Each question will be thoughtfully created to extract specific insights from the corresponding paragraph. Additionally, we will carefully provide accurate and

comprehensive answers to these questions. We will also collect various lengths of paragraphs to ensure the model can generate question-answers paired from paragraphs of different lengths.

4.4.2 Data preprocessing

The collected dataset will need to undergo preprocessing to prepare it for training the T5 model. Firstly, we will check if any value is null in the dataset. If a null value is present, we will remove the entire row containing it from the dataset. After this, we will remove punctuations like ':', '?', '&', etc. from our dataset. We will perform this process because we will have a limited number of tokens available for giving input to the model. Therefore, we need to remove those characters from the text that do not have significant meaning to efficiently use the available number of tokens. Additionally, we will also perform the removal of stop words like 'is', 'an', 'the', etc. from our dataset. This process will also be performed for the same reason as punctuation removal.

4.4.3 Tokenization

This will involve tokenizing the paragraphs, questions, and answers. Tokenization will involve splitting the text into smaller units called tokens. This breaking down of text into manageable chunks will help the model to understand and process effectively. Additionally, any special characters or formatting in the data will need to be handled appropriately. In the T5 architecture, the "vocab_file" will be used to provide the mapping between tokens and their respective IDs.

4.4.4 Train test split

Subsequently, we executed a train-test split on the dataset. Furthermore, the training dataset underwent an additional division, resulting in separate training and validation data.

- Train Data: The training data is used iteratively to update the model's parameters, adjusting its internal mechanisms to minimize the error between its predictions and the true labels.

- Test Data: The test data serves as a final checkpoint to ensure that the model has not overfit the training data or been fine-tuned too much based on the validation data. Using this data helps determine the model's real-world applicability and provides insights into its strengths and weaknesses.

- Validation Data: Regular performance measurements are taken during the model training using validation data. This helps monitor progress, detect overfitting, and optimize performance by adjusting hyperparameters. The validation data remains unbiased, as it is not used for training, providing an indicator of the model's generalization ability.

4.4.5 Model Fine-tuning

Fine-tuning will involve training the model on the dataset to adjust its parameters specifically for question-and-answer generation. One T5 base model will be fine-tuned to generate questions. The T5 base model will take the paragraph as input and generate questions as the output. Additionally, another T5 base model will be fine-tuned to generate answers from the given paragraph and question. This step will enable the model to learn the patterns and structures necessary for generating relevant questions and answers from the paragraph. Here, we even calculated the cosine similarity between the questions generated from the model and questions from the training dataset.

The T5 model uses cross-entropy loss for backpropagation. Suppose for input token X, the actual output token is Y but the model generates an output token of Z, then we calculate cross-entropy loss between Y and Z during backpropagation.

If the model generates a token $\hat{Y}$ but the actual token in training data is Y, then the cross-entropy loss is given as:

$$L(\hat{Y}, Y) = -\sum_{k=1}^{K} Y^{(k)} \, log \, (\hat{Y}^{(k)}) \tag{4-6}$$

Where,

$Y^{(k)}$ represents 0 or 1 for k^{th} token as presence (1) or absence (0). It is ground truth.

$\hat{Y}^{(k)}$ represents the probability of predicted token.
K represents the total number of tokens.

4.4.6 Grammar Correction Model

We also imported a grammar correction module to correct the grammar of our generated answers.

4.4.7 Answer Similarity Checking

The generated questions will be presented to the user along with the original paragraph. The user will be allowed to answer each question. To evaluate the similarity between the user's answer and the grammatically correct generated answer, text similarity metrics such as cosine similarity can be utilized. These metrics will calculate the similarity between two text strings based on their content. The user's answer will be compared with the generated answer, and a similarity score will be obtained.

If A and B be two matrices containing the self-attention values of the user's answer and model's answer, then the cosine similarity between them is given by,

$$cos(\theta) = \frac{A.B^T}{|A|.|B|} \tag{4-7}$$

Here, θ represents the angle between A and B. As cos(θ) lies in the range [-1,1]:

- cos(θ) = -1 will suggest A and B are opposite in meaning (θ = 180°)
- cos(θ) = 0 will suggest A and B are contextually independent (θ = 90°)
- cos(θ) = 1 will suggest A and B are similar in meaning (θ = 0°)

4.4.8 Evaluation Method

The performance of the system will be evaluated using a combination of automated and manual evaluation methods. Automated evaluation metrics such as ROUGE and BELU can be employed to compare the generated questions and answers against the reference questions and answers in the dataset. These metrics will provide numerical scores that indicate the similarity between the generated and reference text.

4.4.8.1 ROUGE Score

ROUGE Score is a suite of metrics widely applied in text summarization tasks, where the objective is to automatically condense lengthy text into a summary. ROUGE was created to evaluate the quality of machine-generated summaries by comparing them with reference summaries provided by humans. It measures the similarity between the machine-generated summary and the reference summaries using overlapping n-grams, word sequences that appear in both the machine-generated summary and the reference summaries. The most common n-grams used are unigrams, bigrams, and trigrams. ROUGE score calculates the recall of n-grams in the machine-generated summary by comparing them to the reference summaries.

The formula for the ROUGE score is as follows:

$$ROUGE = \sum(Recall\ of\ n-grams) \tag{4-8}$$

Here, Recall of n-grams is the number of n-grams that appear in both the machine-generated summary and the reference summaries divided by the total number of n-grams in the reference summaries.

The ROUGE score ranges from 0 to 1, with higher values indicating better summary quality. Like the BLEU score, a perfect summary would have a ROUGE score of 1, while a completely incorrect summary would have a ROUGE score of 0.

ROUGE scores are branched into ROUGE-N, ROUGE-L, and ROUGE-S.

- ROUGE-N: ROUGE-N measures the overlap of n-grams (contiguous sequences of n words) between the candidate text and the reference text. It evaluates the similarity between the generated and reference texts.. For example, ROUGE-1 (unigram) measures the overlap of single words, ROUGE-2 (bigram) measures the overlap of two-word sequences, and so on. ROUGE-N is often used to evaluate the grammatical correctness and fluency of generated text.

  $$ROUGE - N = \frac{X}{Y} \tag{4-9}$$

  Where,

  X – Total number of overlaping n-grams in general text and reference text

  Y – Total Number of n-grams in reference text

- ROUGE-L: ROUGE-L measures the longest common subsequence (LCS) between the candidate and reference texts. It employs the concept of Longest Common Subsequence (LCS) to measure the longest matching sequence of words. It takes into account the word order and does not require consecutive matches. As a result, it can capture sentence-level word order and does not rely on a predefined n-gram length. The LCS-based approach of ROUGE-L ensures a more comprehensive evaluation.

$$ROUGE - L = \frac{X}{Y} \tag{4-10}$$

Where,

X – Length of longest common subsequence in generated text and reference text

Y – Total number of words in reference text

The ROUGE score offers flexibility by permitting the adjustment of n-gram lengths to suit specific task needs. However, ROUGE has its drawbacks. It might not completely grasp the semantic significance or flow of the summary and solely depends on n-gram overlap, which might not always precisely reflect the quality of the summary.

4.4.8.2 BLEU Score

The BLEU score is a tool used to measure how well machines translate text from one language to another. Instead of just looking at individual words, it considers groups of words, like two-word sequences or three-word sequences, which are called n-grams. It then compares these groups in the machine's translation to similar groups in human-made translations.

It evaluates how similar the machine-translated text is to the reference translations by looking at groups of words called n-grams. These n-grams can be single words (unigrams), two-word sequences (bigrams), three-word sequences (trigrams), and so on.

The BLEU score calculates the accuracy of these n-grams in the machine-generated translation by comparing them to the reference translations. It also adjusts this accuracy with a penalty if the machine translation is shorter than the reference.

The BLEU score is calculated using a formula:

$$BLEU = BP.e^{\sum P_n} \quad (4\text{-}111)$$

BP (Brevity Penalty) is a penalty term that adjusts the score for translations that are shorter than the reference translations. It is calculated as min(1, (reference_length / translated_length)), where reference_length is the total number of words in the reference translations, and translated_length is the total number of words in the machine-generated translation.

P_n represents the precision of n-grams, calculated as the number of common n-grams between the machine-generated and reference translations divided by the total number of n-grams in the machine-generated translation.

The BLEU score ranges from 0 to 1. Higher scores indicate better translation quality; a perfect translation would score 1, while a completely incorrect one would score 0. The BLEU score is extensively employed in machine translation tasks due to its straightforward and efficient method of evaluating machine-generated translations against reference translations. However, it does have drawbacks. Because it mainly focuses on n-grams, it might not fully capture the overall meaning or fluency of the translated text. Additionally, it could penalize translations that exceed the length of reference translations, which might be unjust in certain situations.

Evaluating a language generation model will be trickier than evaluating a classification model. The reason will be that there won't be a definitive correct answer to compare against, unlike in classification models. Therefore, one of the top ways to evaluate a language generation model will be to produce text and have a manual evaluation, which will involve an actual person rating the quality and relevance of the generated questions and answers based on factors like correctness, relevance, and fluency.

4.4.8.3 Cosine Similarity

Here, we calculated the cosine similarity between the model generated output and reference text for both questions and answers.

4.4.9 Model Deployment

We have developed a web application that enables users to input a paragraph and receive question-answer pairs as the output. To accomplish this, we utilized the React

and Flask frameworks for web application development. To design the website we used Figma.

4.4.9.1 UI Design

Creating a website is crucial because it shapes users' perceptions and influences their decision to stay engaged. When a website is visually appealing and user-friendly, it encourages people to explore and interact more. Leveraging Figma tools, we crafted an attractive and intuitive design for the Edu pulse site. By grasping user needs and behaviors, we've optimized the site for efficiency and enjoyment.

4.4.9.2 React

React, also called React.js or ReactJS, is a free and open-source JavaScript library for creating user interfaces using components. With React we can build different types of applications like single-page apps, mobile apps, or server-rendered apps using frameworks like Next.js. React mainly focuses on handling the user interface and displaying components on the web page. React applications often rely on other libraries for tasks like navigation (moving between pages) and other client-side features. One of the main benefits of React is its ability to update only the parts of the page that have changed, instead of re-rendering the entire page every time there's an update. This helps in making React applications faster and more efficient.

4.4.9.3 HTML

HTML, known as the HyperText Markup Language, is the basic language used for creating web pages that we see in our browser. It's like the skeleton or structure of a webpage. Web browsers receive HTML files from the internet or our computer and turn them into the web pages we view. HTML tells the browser how to arrange the content on the page, like text, images, and links. HTML uses tags, which are like instructions, to define different parts of the page. For example, <p> tags indicate paragraphs, <img> tags show images, and <a> tags make links. We can also add styling to HTML using CSS (Cascading Style Sheets), which controls how the page looks, and add interactivity using JavaScript.

4.4.9.4 Styled component

Styled-component is a tool used with React that lets developers to write CSS code directly inside their JSX components. Its big advantage is that it helps create custom-styled components that can be used again and again across the application. We've used Styled Components in our project to make it easier to customize and build upon our components. With Styled Components, we can define styles right where we need them, which helps us manage our CSS rules better and avoid clashes in naming.

4.4.9.5 React router dom

React Router DOM is a widely used library for managing navigation in React applications. It helps developers control how users move around within a single-page application. With React Router DOM, we can easily define routes, manage redirects, and share data between different parts of our app. This tool also lets us create routes that change dynamically and nest within each other.

4.4.9.6 Flask

Flask is a small web framework made with Python. It's called “micro” because it's simple and doesn't need special tools or extra libraries to work. Unlike bigger frameworks, Flask doesn't come with built-in tools for managing databases, validating forms, or other common tasks. However, Flask can be expanded using extensions. These extensions let us to add features to our Flask app as if they were built into Flask from the start.

4.5 Flowchart

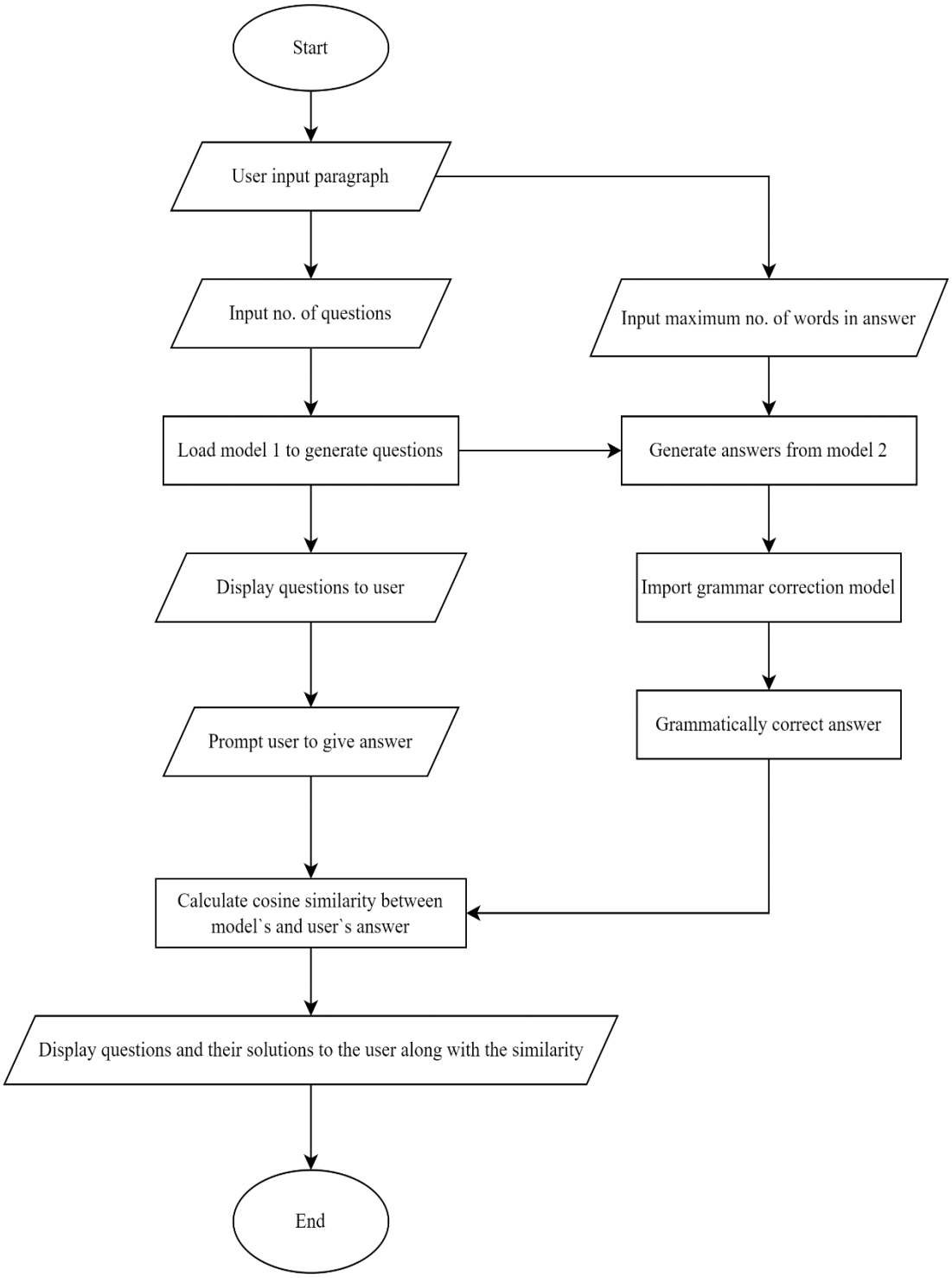


Figure 4-11. System Flowchart

4.6 Decoding Algorithm:

Transformer models have transformed NLP tasks like translation, text generation, and question answering. Key decoding techniques, such as Top-K and Top-P, excel in generating diverse outputs. These algorithms will be employed in our task, and we'll delve into their workings and examples below.

4.6.1 Top-K Sampling

Top-K sampling is a decoding algorithm that ensures diversity in the generated text by sampling from the top-K most likely tokens at each decoding step. The value of K determines the number of tokens considered for sampling. This approach allows the model to explore various possibilities and avoid over-reliance on only the most probable tokens.

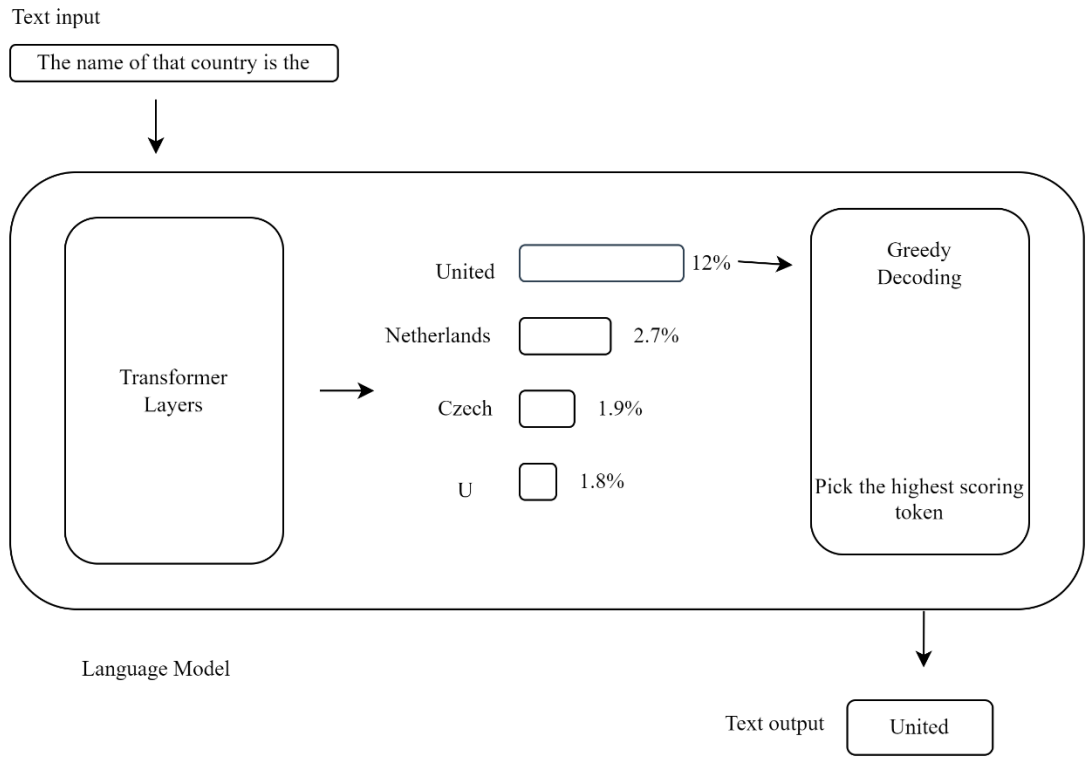


Figure 4-12. Top-K Sampling

In the above figure, a language model generates text based on the prompt, “The name of that country is the ____.” Suppose the model's vocabulary consists of the words: [United, Netherlands, Czech, u]. If K is set to 3, the model will consider the three most

likely words based on their probabilities. Let's assume the model assigns the following probabilities: P(United) = 0.12, P(Netherlands) = 0.027, P(Czech) = 0.019, P(u) = 0.018. The model will randomly sample from the top-K probabilities (0.12, 0.027, 0.019) and select the word “United” to continue the sentence.

4.6.2 Top-P Sampling

Top-P sampling, also known as nucleus sampling or the “penalty” method, ensures diversity in the generated text by sampling from the smallest set of tokens whose cumulative probability exceeds a predefined threshold, P. This technique allows for dynamic control of the generated output's diversity based on the value of P.

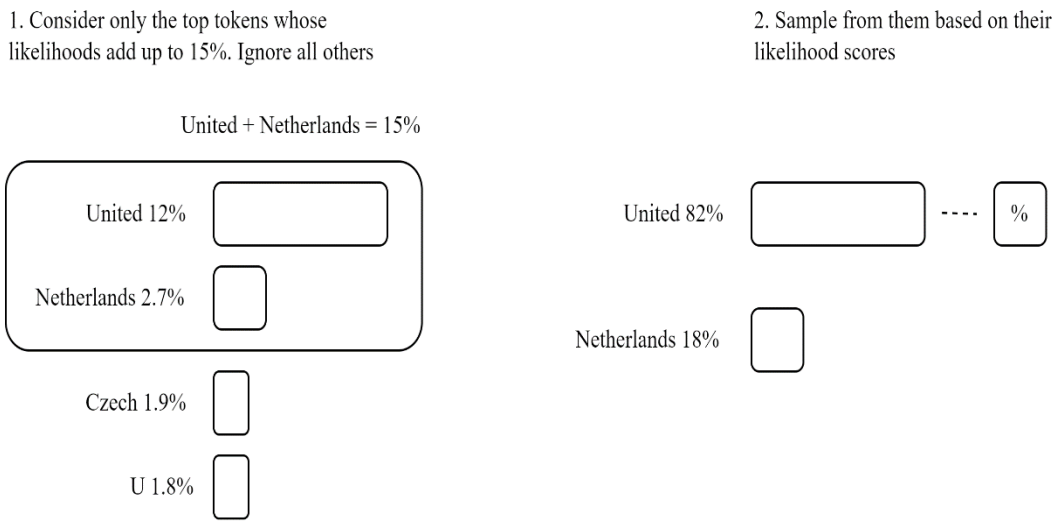


Figure 4-13. Top-P Sampling

In the above figure, the same language model generates text with the vocabulary [united, Netherlands, Czech, u]. Let's assume the model assigns the following probabilities: P(United) = 0.12, P(Netherlands) = 0.027, P(Czech) = 0.019, P(u) = 0.018. If the threshold probability, P, is set to 0.15, the algorithm will sample from the smallest set of tokens whose cumulative probability exceeds 0.15 (United, Netherlands).

We can first consider the most likely words using Top-K sampling before combining these algorithms. Next, we use Top-P sampling with a probability threshold from this smaller collection. Combining Top-K and Top-P sampling gives the model the

advantage of control and diversity. By taking into account a small group of likely tokens (Top-K), it investigates alternate options, and the probability threshold (Top-P) gives fine-grained control over the output's diversity. This combination produces a variety of text outputs that are contextually relevant.

4.7 Use Case Diagram

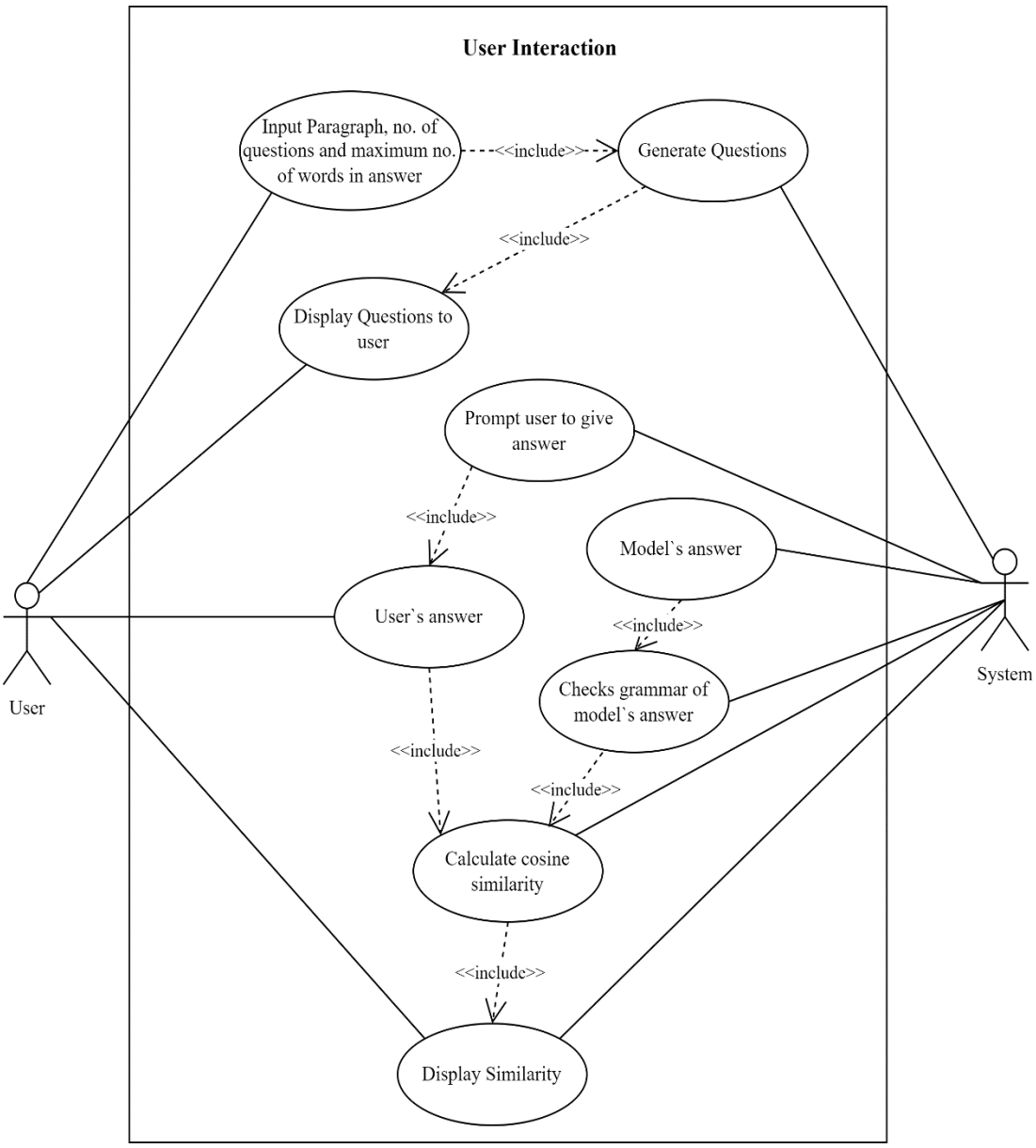


Figure 4-14. Use Case Diagram

In this scenario, the user provides an input paragraph, number of questions to be generated and maximum length of answer to the system. Subsequently, the system generates a set of questions as specified by the user based on the given paragraph. The

system then presents these questions to the user and requests answers from the user for each question. Additionally, the system generates answers to the questions it posed. The system ensures that the length of answers generated are less than or equal to the maximum number of words in answer as specified by user. The generated answer is passed to the grammar correction module to check it’s grammar. Once the user provides answers to all the questions, they have the option to view the system-generated solutions for each question, along with a similarity score that compares the user's answers to the system's answers.

This process aims to facilitate an interactive and educational experience, allowing users to engage with the content of the paragraph by actively responding to the generated questions. The system's ability to generate questions and evaluate the user's answers provides a comprehensive learning environment, while the similarity score offers insight into the alignment between the user's understanding and the system's expected answers.

4.8 Software Development Life Cycle (SDLC)

In the project of implementing Question Answer Generation using the T5 Transformer, we would follow an Agile development approach, breaking down the process into iterative phases. Here's how the Agile model aligns with this project:

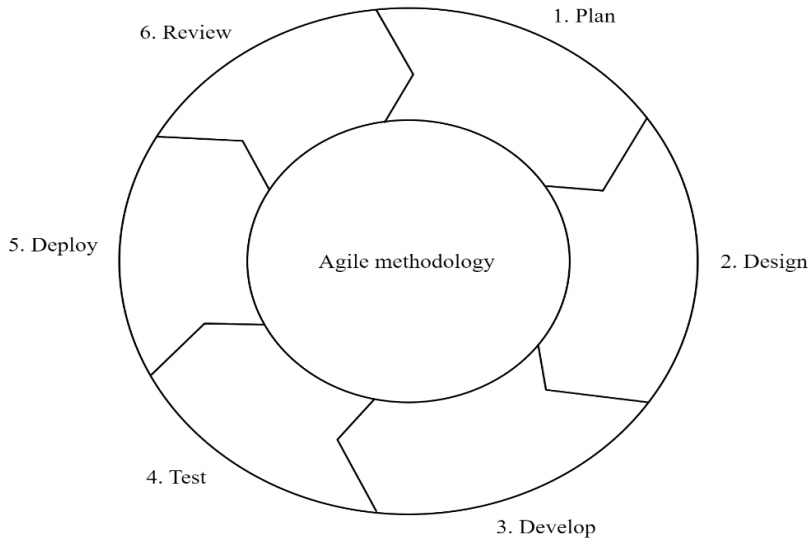


Figure 4-15. Agile Method

4.8.1 Plan

- At first, we defined the scope of the project, including the input and output requirements.
- Then we created a backlog of tasks, such as data preprocessing, model training, and evaluation.
- After that we prioritized the tasks based on their importance and dependencies.

4.8.2 Design

- Then we designed the data preprocessing pipeline to convert paragraphs into input-output pairs suitable for training the T5 Transformer.
- We also defined the architecture of the T5 model, including the input format, fine-tuning strategy, and output generation process.

4.8.3 Develop

- We implemented the data preprocessing pipeline to prepare the training data.
- We also fine-tuned the pre-trained T5 Transformer using the prepared data.
- Then we developed scripts or APIs to interact with the trained model for question answering.

4.8.4 Test

- We also conducted unit tests to ensure the correctness of individual components, such as data preprocessing functions and model training scripts.
- Then we performed integration tests to validate the end-to-end functionality of the question answering system.
- We also evaluated the quality of generated questions and answers using human evaluators or automated metrics.

4.8.5 Deploy

- We deployed the question answering system to a production environment, such as a web server.
- We also monitored the performance and usage of the deployed system, collecting feedback from users.

- We even iterated on improvements and updates based on user feedback and changing requirements.

4.8.6 Review

- We also held regular sprint reviews to assess the progress of the project and identify any issues or bottlenecks.
- We also demonstrated the functionality of the question answering system to stakeholders, gathering feedback for future iterations.
- We finally reflected on lessons learned and incorporate improvements into the development process for subsequent sprints.

Throughout each phase, the Agile model encourages collaboration and adaptability, allowing the project team to adjust course as needed based on feedback and changing priorities. By following this iterative approach, we can efficiently develop and refine the Question Answer Generation system using the T5 Transformer, ultimately delivering a high-quality solution that meets our needs.

4.9 State Diagram

In this system, users begin by giving input a paragraph. Then, first model creates questions based on the input paragraph. The number of questions it generates is based on the input given by the user. Next, second model uses both the paragraph and the questions to make answers. The length of answer generated by second model is less than or equal to the maximum number of words given as input by the user. The answer generated from the model is passed to the grammar correction model to generate grammatically correct answers. After the system produces these answers, users give their own answers. The system checks how similar the user's answer is to the generated one using cosine similarity, which measures how alike two answers are. Finally, the system shows the questions made and their answers, along with a score indicating how similar they are, for the user to see.

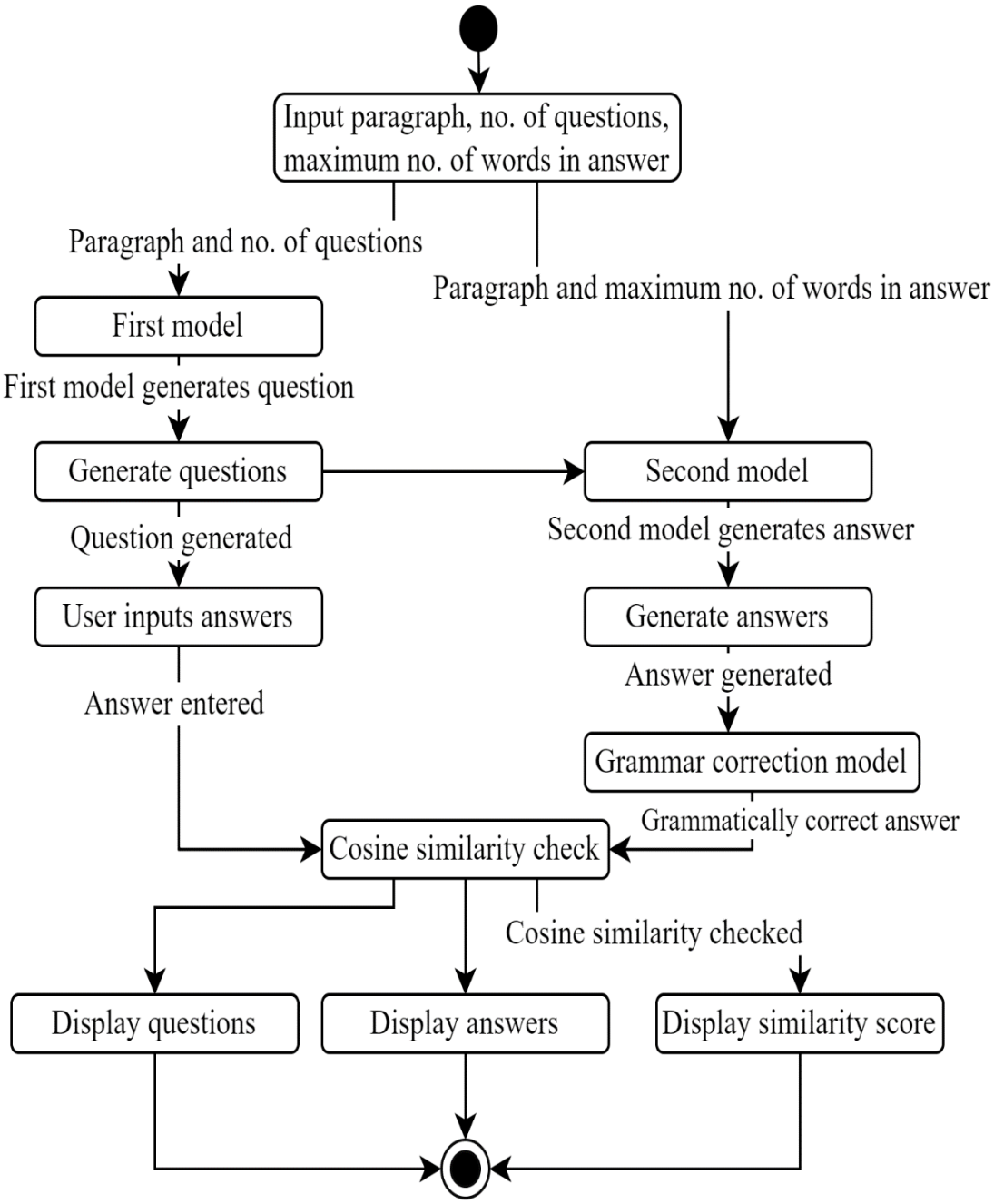


Figure 4-16. State Diagram

4.10 Data Flow Diagram (DFD)

4.10.1 DFD Level 0

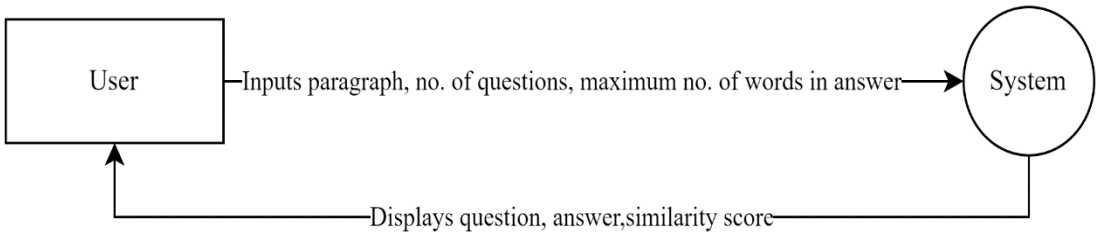


Figure 4-17. DFD Level 0

In our system, users input paragraphs, and then the system processes this input. After processing, the system presents questions, answers, and a similarity score to the user.

4.10.2 DFD Level 1

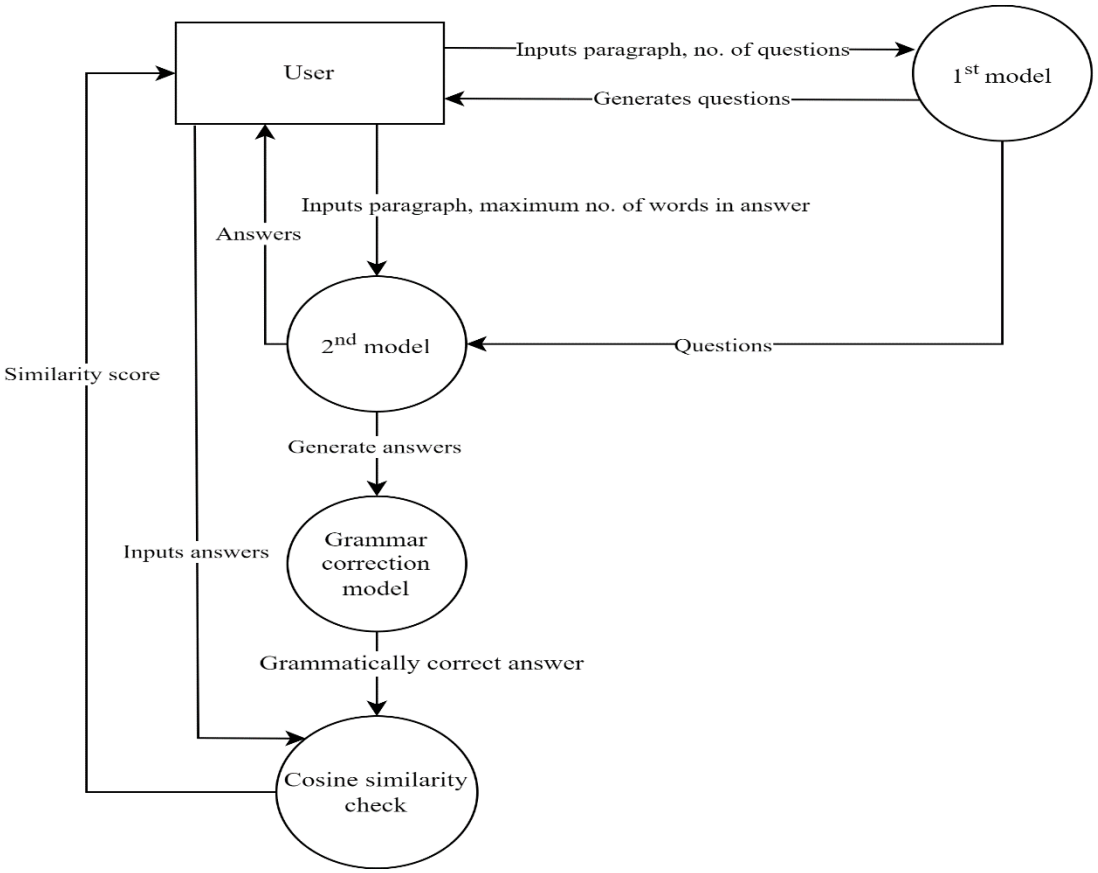


Figure 4-18. DFD Level 1

At first, user inputs a paragraph. They also specify the number of questions to be generated and the maximum length of answers in words. The first model generates questions based on the user's input, with the number of questions determined by the user. Then, another model utilizes both the paragraph and the questions to craft answers, ensuring they adhere to the maximum word limit provided by the user. The generated answers undergo grammar correction. Once the system produces these answers, users provide their own responses. The system evaluates the similarity between the user's answer and the generated one using cosine similarity, which measures their likeness. Finally, the system presents the questions along with their answers and a score indicating their similarity for the user's review.

4.10.3 DFD Level 2

- **For Question Model**

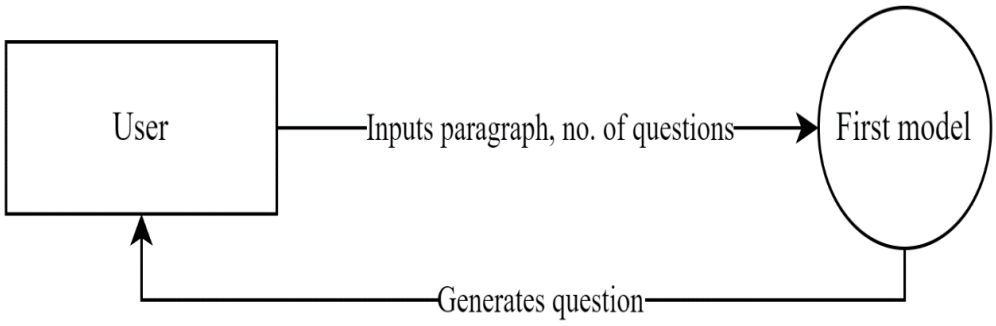


Figure 4-19. Level 2 DFD for question model

Users input a paragraph and number of questions into the system. Then, the first model in our system processes at it and generates questions. The number of questions varies according to the input from the user. These questions are then given back to the user as the output.

- **For Answer Model**

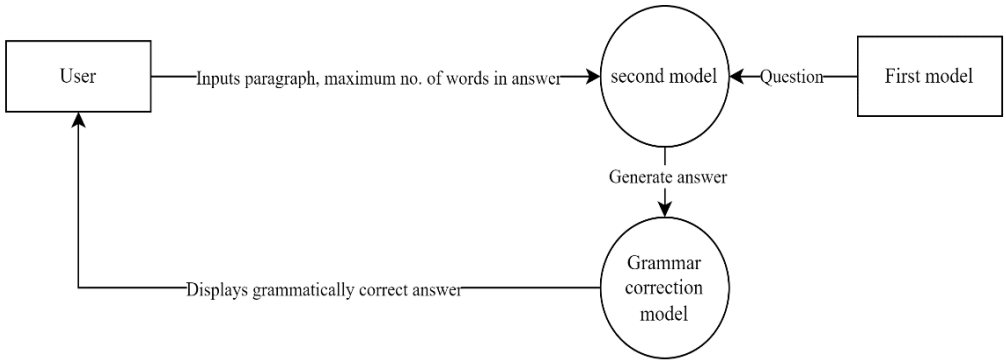


Figure 4-20. DFD Level 2 for Answer model

The user enters a paragraph and maximum length of answer into the system. Then, the second model of our system takes that paragraph and the questions generated by the first model. It uses both of these to create answers, which are then fed to the grammar correction model. Those grammatically correct answers are given back to the user as the output. The length of answer generated by the model is always less than or equal to the specified length given by user.

- **For Similarity Checking**

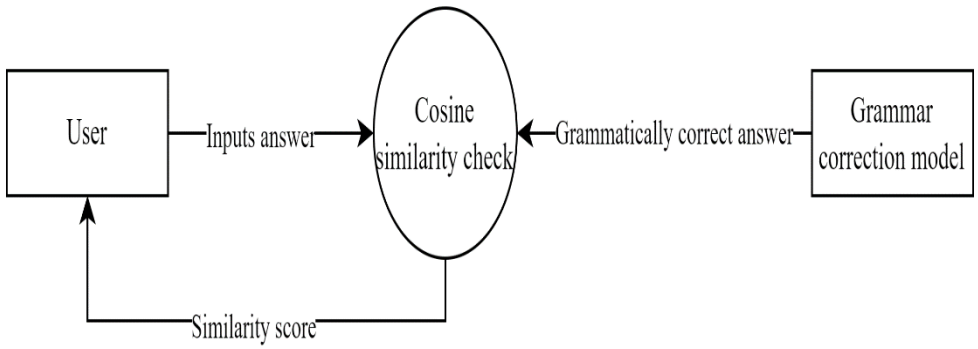


Figure 4-21. Level 2 DFD for Similarity Checking

The user inputs answers to the system. Grammar correction models also gives grammatically correct answer. Then, the system compares these two answers using cosine similarity to see how similar they are. Finally, it gives a similarity score as the output.

4.11 Class Diagram

In our system, we have four main classes: User, Paragraph, Question Model, and Answer Model. The User class holds attributes about users, such as their name, username, contact number, and email. Users can perform actions like inputting paragraphs and submitting answers. The Paragraph class contains attributes about paragraphs, like the category name and length of the paragraph. Operations like fetching answers and generating questions are carried out within this class.

The Question Model class includes attributes such as the number of questions and the type of questions. Its main function is to generate questions based on provided paragraphs. Similarly, the Answer Model class has attributes like the number of answers and the length of answers. Its primary task is to generate answers, typically using both the paragraph and the generated questions.

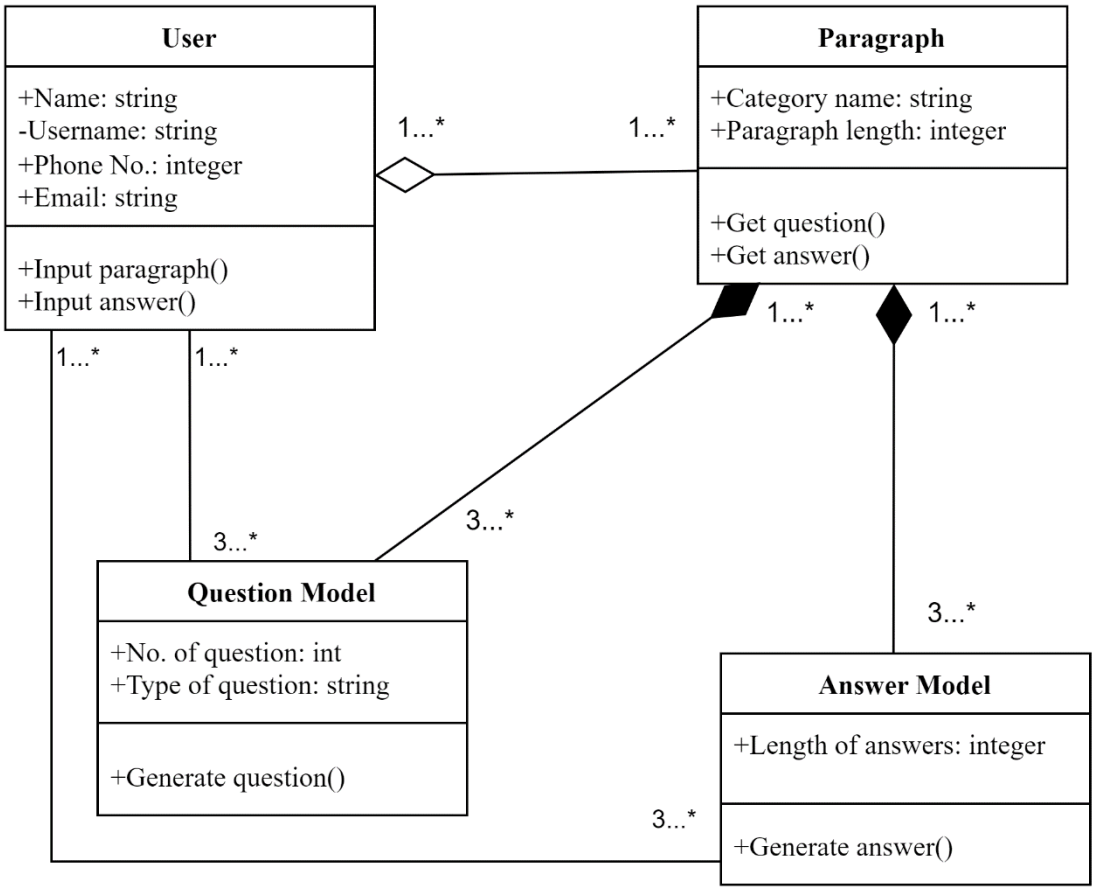


Figure 4-22. Class Diagram

4.12 Activity Diagram

In this system, users start by giving a input paragraph, number of questions and maximum length of answer into the system. Then, a model generates questions based on the paragraph provided. The number of questions to be generated is determined by the user. Following that, another model uses both the paragraph and the generated questions to come up with answers. The length of answer is also specified by the user. The answer generated is passed to grammar correction model which generates grammatically correct answers. Once the system generates these answers, users provide their own responses. The system then compares the user's response with the grammatically correct generated answer using cosine similarity, which measures how similar those two answers are. Finally, the system presents the generated question and its corresponding answer, along with a similarity score, to the user for review.

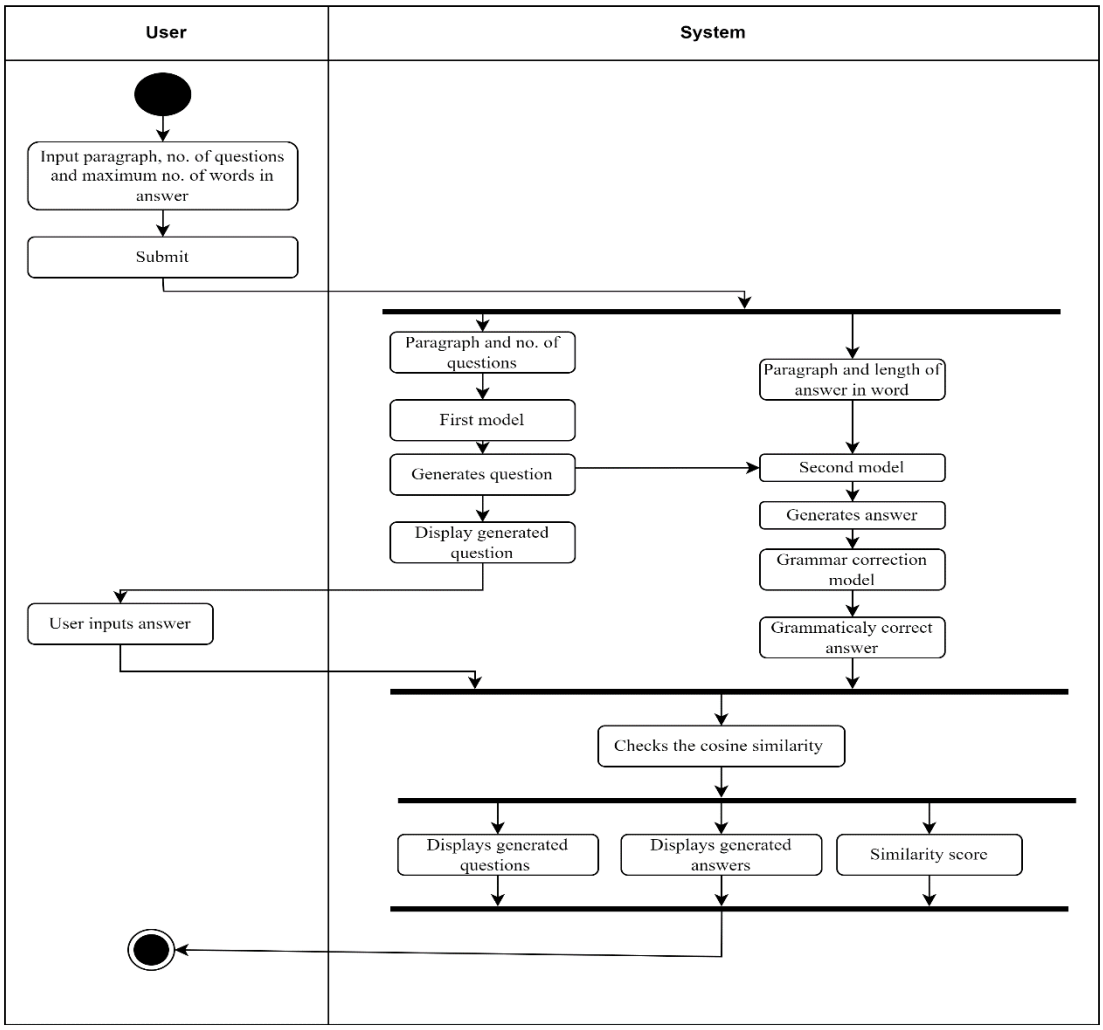


Figure 4-23. Activity Diagram

5. IMPLEMENTATION DETAILS

In this project, we use two T5 transformers to generate questions and answers from a given input paragraph. To train the model, we will require thousands of question-answer pairs along with the paragraph. The major goal of the first part of the major project is to collect an accurate and credible dataset to train the model.

5.1 Dataset Collection

Due to the absence of a sufficiently robust dataset for question-answer generation from paragraphs, we encountered a need to undertake a thorough collection process from various sources such as books, articles, and websites. This procedure involved the diligent collection and subsequent manual annotation of datasets. The initial phase of data collection required the careful manual selection of appropriate paragraphs from these sources, considering their relevance and informativeness. Following this rigorous selection process, we proceeded to manually generate questions based on the selected paragraphs. We took into account the context, key ideas, and main arguments presented in each paragraph to ensure the questions captured the essence of the text. Once the questions were generated, we moved on to providing corresponding answers. This phase involved careful analysis and extraction of information from the paragraphs. We aimed to provide accurate and concise answers that directly addressed the questions.

5.2 Data Augmentation

The Data Augmentation (DA) Technique allows us to artificially enhance the quantity of training data by producing multiple copies of genuine datasets without actually gathering the data. To improve classification performance, the data must be altered to preserve the class categories.

Data augmentation is a technique that contributes to the improvement of model performance by expanding the available training data. The augmentation process results in a larger dataset, which is beneficial for enhancing the model's effectiveness. As the amount of data increases, the performance of the model tends to improve. It is crucial, however, to maintain a balanced distribution in the augmented data, ensuring that it neither closely resembles nor significantly deviates from the original dataset. Striking

the right balance in terms of similarity between the original and augmented data is essential for optimal model performance.

Considering that our goal is to improve the performance of the question-and-answer generation model, it would be more effective to apply data augmentation techniques specifically to the questions rather than the entire paragraph or the answers. Augmenting the questions can introduce variations in wording, phrasing, and structure, which can help the model learn to generate diverse and accurate questions. Augmenting the answers might not provide as much benefit since the answers are expected to be concise and specific to the given paragraph. For augmenting paragraphs sentence swap can be used.

Some of the data augmentation techniques used in the field of NLP are:

5.2.1 Back Translation

In this methodology, the process involves translating the text data into a different language and subsequently translating it back to the original language. This technique proves valuable in generating textual data that encompasses diverse terminologies while preserving the underlying context of the original text.

To facilitate the translation, widely used language translation APIs such as Google Translate, Bing Translator, and Yandex Translator are utilized.

Let us consider an example:

English original question:

“What potential byproduct can be formed during ozonation and what conditions contribute to its formation?”

French translated question:

“Quel sous-produit potentiel peut se former lors de l'ozonation et quelles conditions contribuent à sa formation ?”

English (translated from French) question:

“What potential by-product can form during ozonation and what conditions contribute to its formation?”

5.2.2 Easy Data Augmentation

Easy Data Augmentation (EDA) is a data augmentation technique that relies on classical and fundamentally straightforward methods. EDA consists of four primary procedures that effectively address overfitting concerns and aid in the training of more resilient models.

- Synonym Replacement: In this approach, a certain number, n, of non-stop words are randomly selected from the given phrase. Each of these chosen words is then replaced with a randomly selected synonym.

  Let's consider the following original sentence:

  “What potential byproduct can be formed during ozonation and what conditions contribute to its formation?”

  For this example, let's randomly select two words from the question, excluding any stop words. Suppose we choose to replace “potential” and “formed” with synonyms. Here, n=2 as we selected 2 words.

  Assuming we select “possible” as the synonym for “potential” and “created” as the synonym for “formed,” the augmented question would be:

  “What possible byproduct can be created during ozonation and what conditions contribute to its formation?”

- Random Insertion: This approach entails selecting a synonym for a randomly chosen non-stop word within the given phrase. Subsequently, the identified synonym is inserted into a randomly determined location within the phrase. This

process is repeated a total of n times, thereby enhancing the sophistication of the paragraph.

Let's say we have the following original question:

“What potential byproduct can be formed during ozonation and what conditions contribute to its formation?”

For this example, let's randomly select two words from the question, excluding any stop words. Suppose we choose to replace “potential” and “formed” with synonyms.

Assuming we select “possible” as the synonym for “potential” and “created” as the synonym for “formed,” the augmented question would be:

“What byproduct possible can be during ozonation created and what conditions contribute to its formation?”

- Random swap: Two words from the provided phrase are selected at random and their positions are. This process is then repeated n times, introducing further variation and dynamism to the paragraph.

“What potential byproduct can be formed during ozonation and what conditions contribute to its formation?”

For this example, let's randomly select two words from the question, excluding any stop words. Suppose we choose to replace “potential” and “formed” with synonyms.

Assuming we select “potential” and “formed” for swapping. Then augmented question looks like this:

“What formed byproduct can be potential during ozonation and what conditions contribute to its formation?”

- Random Deletion: With a given probability p, each word within the phrase is randomly eliminated.

  Let's say we have the following original question:

  “What potential byproduct can be formed during ozonation and what conditions contribute to its formation?”

  To apply random deletion data augmentation, we randomly select a word and delete the word from the phrase. Suppose the selected word is “be”.

  Augmented sentence: “What potential byproduct can formed during ozonation and what conditions contribute to its formation?”

5.2.3 Using Questions templates

One effective technique for data augmentation in question generation from a paragraph involves utilizing question templates. These templates serve as a structured framework for generating new questions based on the content of the paragraph. For each question type, we create a template that specifies the position of the question word and provides a placeholder for the specific information from the paragraph.

Here's an example of inserting specific question templates as a data augmentation technique for question augmentation:

Original Question: “What are biofuels made from, as mentioned in the paragraph?”

Augmented Question with Template Insertion: “What are the primary ingredients used to create biofuels, as mentioned in the paragraph?”

In this example, a specific question template, “What are the primary ingredients used to create [topic],” is inserted into the original question. By utilizing question templates, you can introduce variations in the phrasing and structure of the questions while keeping the main focus and context intact. This technique helps the model learn to

handle different question formats and generalize its understanding of the paragraph's content.

5.2.4 Sentence swap

In this method, we take 2 sentences at random from the paragraph and swap their position.

Take an example:

Original Paragraph:

"The environment refers to the surroundings in which life exists on Earth. The environment comprises components like animals, humans, sunlight, water, trees, and air. They are the earth's living and non-living components. Living organisms include trees, humans, and animals. Non-living components such as the sun, water, and air are essential for man's life. Both living and non-living organisms are dependent on each other to maintain a balanced ecosystem. A disturbance in any of the categories will create chaos in the entire ecosystem and threaten the environment. Just like the growing human population has become a threat to the environment. Global warming is a major threat to the environment. The atmosphere and hydrosphere, on the other hand, are components of the environment that have an impact on the lives of living things. The atmosphere contains gases like oxygen and nitrogen. Every living being is developed according to the characteristics of these components. Aquatic species, for example, are designed to breathe underwater. Aerial animals are designed to live in the air. So even the tiniest change in nature's subtle balance makes it difficult for living organisms to survive in the environment it is designed to live in."

In the original paragraph suppose that the sentences underlined are to be swapped. Then augmented paragraph would look like this:

"The environment refers to the surroundings in which life exists on Earth. Both living and non-living organisms are dependent on each other to maintain a balanced ecosystem. They are the earth's living and non-living components. Living organisms include trees, humans, and animals. Non-living components such as the sun, water, and

air are essential for man's life. The environment comprises components like animals, humans, sunlight, water, trees, and air. A disturbance in any of the categories will create chaos in the entire ecosystem and threaten the environment. Just like the growing human population has become a threat to the environment. Global warming is a major threat to the environment. The atmosphere and hydrosphere, on the other hand, are components of the environment that have an impact on the lives of living things. The atmosphere contains gases like oxygen and nitrogen. Every living being is developed according to the characteristics of these components. Aquatic species, for example, are designed to breathe underwater. Aerial animals are designed to live in the air. So even the tiniest change in nature's subtle balance makes it difficult for living organisms to survive in the environment it is designed to live in."

Here, only the position of underlined sentences is swapped and the other part of the paragraph remains the same.

5.3 Data Preprocessing

The collected dataset will need to undergo preprocessing to prepare it for training the T5 model. The following steps were performed in data preprocessing.

5.3.1 Checking Null values

This step will check if any value is null in the dataset. If a null value is present, we will remove the entire row containing it from the dataset.

5.3.2 Removing Punctuation

After this, we will remove punctuations like ':', '?', '&', etc. from our dataset. We will perform this process because we will have a limited number of tokens available for giving input to the model. Therefore, we need to remove those characters from the text that do not have significant meaning to efficiently use the available number of tokens.

Let us take an example:

Original Text:
"I can't believe it's already Friday! Where did the week go?"

Text after Punctuation Removal:
“I cant believe its already Friday Where did the week go”

5.3.3 Stop words removal

Additionally, we will also perform the removal of stop words like ‘is’, ‘an’, ‘the’, ‘to’, ‘also’ etc. from our dataset. This process will also be performed for the same reason as punctuation removal.

Let us take an example:

Original Text:
“I love to eat pizza, but I also enjoy pasta and burgers.”

Text after Stop Words Removal:
“I love eat pizza, but I enjoy pasta burgers.”

5.3.4 Lemmatization

Lemmatization, also known as lemmatization, is a process in natural language processing (NLP) that involves reducing words to their base or root form, known as the lemma. The lemma is the canonical or dictionary form of a word, and lemmatization aims to normalize different inflected forms of a word to their common base form. Then, we will perform lemmatization, which will help the model understand and converse with end users accurately.

Let us take an example:

Original Text:
“I am running in the park and saw a bunch of beautiful flowers.”

Text after Lemmatization:
“I am run in the park and see a bunch of beautiful flowers.”

5.3.5 Tokenization

For tokenization, we will be using T5 tokenizer which inherits from "PreTrainedTokenizer" which contains most of the main methods. It is based on the SentencePiece tokenization mechanism. SentencePiece is an unsupervised text tokenizer and detokenizer designed primarily for Neural Network-based text creation systems with a preset vocabulary size before neural model training. SentencePiece extends direct training from raw sentences to incorporate subword units (e.g., byte-pair-encoding (BPE) and unigram language model). SentencePiece enables us to create a system that is entirely end-to-end and does not rely on language-specific pre/postprocessing.

In the T5 tokenizer, after tokenization, we will get 2 vectors.

- Input IDs: They represent the tokenized and converted input text into a sequence of integer IDs, which the model can understand. Each ID corresponds to a specific token in the tokenizer's vocabulary.

- Attention Mask: It is a binary mask that indicates which tokens are actual input tokens and which ones are padding tokens. It helps the model know which parts of the input sequence to pay attention to and which parts to ignore.

5.4 Model Creation

The model creation process begins with the utilization of the Text-To-Text Transfer Transformer model, a transformer-based architecture used for various natural language processing tasks. Transformers are deep learning models that capture contextual relationships in data, making them highly effective for sequential tasks like language generation. T5's unique approach treats every NLP task as a text-to-text task, unifying all tasks under a common text generation framework. This simplifies complex tasks into text generation problems, allowing the model to learn patterns in textual data more effectively.

The input text, representing paragraphs, undergoes tokenization, dividing it into smaller units known as tokens, and is encoded by the T5 tokenizer. Attention masks and padding are applied to handle variable-length sequences, aiding the model in

understanding contextual relationships within the input. Attention masks indicate which tokens the model should pay attention to during training, enhancing the model's understanding of the context within the input sequence.

The model is initialized from a pre-trained checkpoint and fine-tuned for question generation using the provided dataset. During training, the AdamW optimizer is employed, combining adaptive learning rates with weight decay for efficient parameter updates. The training loop, spanning multiple epochs, involves predicting questions from paragraphs, computing loss, and adjusting internal parameters through backpropagation. Once trained, the model is saved for future use, providing an efficient means of generating questions without retraining. These processes collectively result in the development of a robust question-generation model capable of generating meaningful questions from input paragraphs, demonstrating the intricate interplay of various techniques in the realm of natural language processing.

The T5 base model consists of ReLU as activation function in feed forward layer. ReLU, also known as the rectified linear activation function, is a piecewise linear function that, in the event that the input is positive, will output the input directly; if not, it will output zero. Because a model that utilizes it is quicker to train and frequently performs better, it has become the default activation function for many different kinds of neural networks.

Mathematically, ReLU can be defined as,

$$y = max(0, x) \qquad (5\text{-}1)$$

where x is input.

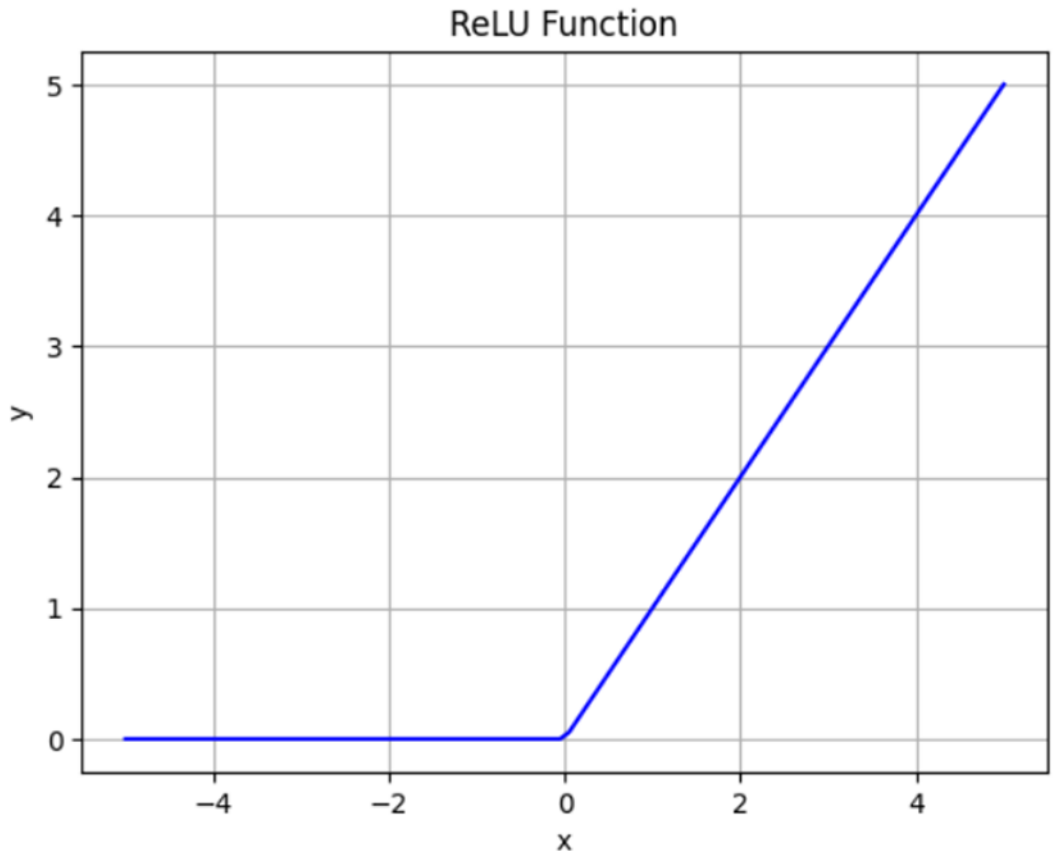


Figure 5-1. ReLU Function

From the graph, we can see that the ReLU activation function is linear for positive values of x which makes it easy to optimize using gradient-based methods.

5.5 Model Training and Validation

Training a neural network involves a complex process of optimizing its parameters to make accurate predictions or generate meaningful outputs. During training, the model generates questions based on input paragraphs and compares them to the actual questions from the dataset. This comparison is done using a loss function, a mathematical measure of the dissimilarity between the generated and actual questions. In this case, the model uses the Cross-Entropy Loss, a common choice for sequence generation tasks. The optimizer, in this case, AdamW, adjusts the model's internal parameters iteratively to minimize this loss. Optimization algorithms like AdamW adjust the learning rates of each parameter individually, ensuring that the model converges to an optimal solution efficiently.

Once the loss is computed, the backpropagation algorithm is employed. Backpropagation calculates the gradients of the loss concerning each model parameter. These gradients represent the direction in which the parameters should be adjusted to minimize the loss. Gradient Descent then uses these gradients to update the model's weights. This iterative process of computing gradients, adjusting weights, and reducing loss is fundamental to training neural networks. It ensures that the model learns the patterns in the data and improves its performance over time.

Training data is processed in batches to make computations more memory-efficient. Batching involves dividing the dataset into smaller chunks. During each iteration (or batch), the model receives multiple input-output pairs simultaneously, updating its parameters collectively. One pass through the entire dataset is called an epoch. The model undergoes multiple epochs, allowing it to learn from the entire dataset multiple times. Each epoch refines the model's understanding of the data, enhancing its predictive capabilities.

In the validation phase, we extended the training process to include the assessment of model performance on a separate dataset. Similar to the training procedure, the model generated questions based on input paragraphs, and the generated questions were compared to the actual ones in the validation dataset. The Cross-Entropy Loss, utilizing the AdamW optimizer, was employed to measure the dissimilarity between the generated and actual questions. This process helped ensure that the model not only learned from the training data but also generalized well to new, unseen data, preventing overfitting and enhancing its overall predictive capabilities. To validate the dataset, we used KFold cross validation with 5 folds.

In transformer-based architectures like T5, attention mechanisms play a crucial role. Attention mechanisms allow the model to focus on different parts of the input sequence when generating an output. It assigns different weights to different input tokens, allowing the model to prioritize relevant information. Attention mechanisms enable the model to capture long-range dependencies in the data, making it effective for tasks involving sequential data like text.

During training, the model might become too specialized for the training data, a phenomenon known as overfitting. Regularization techniques like weight decay,

included in the AdamW optimizer, help prevent overfitting. Weight decay discourages overly complex models by penalizing large weights during optimization. This regularization term ensures that the model generalizes well to unseen data, making its predictions more reliable.

During training, we trained two models. The first model is trained using paragraphs and questions. The second model is trained using paragraphs, questions, and answers.

5.6 Grammar Correction Model

We used a vennify model which was trained on a dataset called JFLEG. This is a t5-base model trained using “HappyTransformers”.

5.7 Question Generation

Generating questions from input paragraphs involves several key steps. First, the input paragraph is broken down into smaller units called tokens and converted into numerical representations using the model's tokenizer. The T5 model, designed for conditional text generation, learns to generate questions based on this encoded input. During the generation process, the model predicts the next token using techniques like greedy decoding (selecting the most probable token at each step) or beam search (exploring multiple potential sequences for varied outputs). The model's attention mechanism allows it to focus on specific parts of the input, ensuring contextually relevant questions. Special tokens, indicating elements like padding and sentence boundaries, help maintain the proper structure of generated questions. Finally, post-processing and decoding steps transform the numerical tokens back into readable text, ensuring the questions are grammatically correct and coherent. These steps collectively enable the generation of meaningful and contextually appropriate questions from input paragraphs.

5.8 Answer Generation

Answer generation is a multi-step process that involves several crucial stages. To begin with, the input paragraph and the questions are dissected into smaller units referred to as tokens, and these tokens are then converted into numerical representations using the model's tokenizer. The T5 model, specifically designed for conditional text generation, excels at crafting answers based on this encoded input. During the generation process,

the model predicts the next token using techniques like greedy decoding (selecting the most probable token at each step) or beam search (exploring multiple potential sequences for varied outputs). The model's attention mechanism allows it to focus on specific parts of the input, ensuring contextually relevant answers. Special tokens, indicating elements like padding and sentence boundaries, help maintain the proper structure of generated questions. Finally, post-processing and decoding steps transform the numerical tokens back into readable text, ensuring the answers are grammatically correct and coherent. These steps collectively enable the generation of meaningful and contextually appropriate questions from input paragraphs.

5.9 Model Evaluation

In the context of assessing the efficacy of the question generation model, two fundamental evaluation metrics, BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores, are employed. These metrics serve as quantitative measures to gauge the quality and accuracy of the generated questions compared to the reference questions.

BLEU scores are calculated to quantify the similarity between the questions generated by the model and the reference questions in the dataset. BLEU is particularly suited for machine translation and text generation tasks, providing a precision-based evaluation metric. It compares n-grams (subsequences of n words) in the generated questions with those in the reference questions. By measuring the overlap of n-grams, BLEU scores offer insights into how well the generated questions align with the reference questions, considering both unigram and higher-order n-grams.

ROUGE scores, including ROUGE-1, ROUGE-2, and ROUGE-L, are utilized for a more comprehensive evaluation of the generated questions' quality. ROUGE metrics assess the overlap of n-grams and subsequences between the generated questions and the reference questions. ROUGE-1 focuses on unigrams, ROUGE-2 on bigrams, and ROUGE-L on the longest common subsequences. These scores provide a nuanced perspective, capturing both exact matches and partial overlaps between the generated and reference questions.

5.10 Hyperparameter Tuning

Different values of hyper-parameters like learning rate, number of epochs, optimizer, batch sizes, and decay factor were tried out for optimization purposes.

5.10.1 Learning Rate

The learning rate hyper-parameter controls how big steps to take while updating the weights of the model with a gradient of the loss function. The model will take a long time to converge to the ideal solution if the learning rate is too small, and it may oscillate or diverge if the learning rate is too high. To train machine learning models for the best performance determining an acceptable learning rate is a crucial task. The learning rate during training can be optimized using a variety of strategies, including learning rate schedules and adaptive learning rate methods. Here, we discovered that the most optimal learning rate was 1e-4 for both models.

5.10.2 Epochs

When training, an epoch is a single iteration of the complete dataset. The model is trained on the full dataset in batches during each epoch, and its weights are adjusted following the error or loss function. The number of epochs is a hyperparameter that controls how frequently the model iterates through the training dataset. A lower value of the number of epochs can lead to underfitting, where the model is unable to capture the underlying patterns in the data and a higher value of the number of epochs lead to over-fitting, where the model is overly complicated and struggles to generalize to new data. The ideal number of epochs for our model was 15.

5.10.3 Batch Size

The term “batch size” describes the number of training examples used in a single training iteration. The model is fed a set of training examples with each iteration, and its weights are adjusted based on the error or loss function. The batch size is a hyperparameter that controls how many samples are processed by the model before the weights are updated. The model updates its weights more frequently with a smaller batch size, but this can slow down training and increase the noise in the updates. The model changes its weights less frequently when the batch size is bigger, which can lead to shorter training times and smoother updates but can also result in the model

becoming stuck in a local minimum. The batch size we discovered to be the most effective was 16 for both models.

5.10.4 Decay Factor

In weight decay, the decay factor, or regularization parameter (λ), significantly influences model training. A higher decay factor intensifies the penalty on large weights, promoting a simpler model and preventing overfitting but risking underfitting. Conversely, a lower decay factor weakens regularization, allowing larger weights and potentially leading to overfitting. The optimal decay factor strikes a balance between fitting the training data and preventing excessive model complexity. The decay factor we found most optimal was 1e-4 for both models.

5.10.5 Optimizer

In our model, we have employed AdamW optimizer. In Adam, L2 regularization is ineffective. Since most deep learning libraries only implement L2 regularization and not the original weight decay, one reason why SGD with momentum might beat Adam and other adaptive gradient algorithms could be AdamW. Adam therefore produces worse outcomes than SGD with momentum (for which L2 regularization performs as expected) on tasks/datasets where the adoption of L2 regularization is advantageous for SGD.

In the picture, we can see implementation of Adam with L2 regularization vs AdamW implementation. Step 6 signifies Adam with L2 regularization and step 12 Signifies Adam with decoupled weight decay (AdamW).

As stated in line 12 of Algorithm 2, we decouple Adam's weight decay and loss-based gradient updates, resulting in our form of Adam with decoupled weight decay (AdamW). Having demonstrated the differences between weight decay regularization and L2 regularization for adaptive gradient algorithms, it remains to be seen how these differences differ and what their implications mean. For intuitive reasons, their equivalency to normal SGD is still highly useful: both algorithms move weights in the direction of zero at the same pace.

1: given $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, $\lambda \varepsilon R$

2: initialize time step $t \leftarrow 0$, parameter vector $\theta_{t=0} \varepsilon R^n$, first moment vector $m_{t=0} \leftarrow 0$, second moment vector $v_{t=0} \leftarrow 0$, schedule multiplier $\eta_{t=0} \varepsilon R$

3: repeat

4: $t \leftarrow t+1$

5: $\nabla f_t(\theta_{t-1}) \leftarrow$ SelectBatch(θ_{t-1}) →select batch and return the corresponding gradient

6: $g_t \leftarrow \nabla f_t(q_{t-1}) + \lambda\theta_{t-1}$

7: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ →here and below all operations are element-wise

8: $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$

9: $m_t \leftarrow m_t/(1-\beta_1^t)$ → β_1 is takenb to the power of t

10: $v_t \leftarrow v_t/(1-\beta_2^t)$ → β_2 is taken to the power of t

11: $\eta_t \leftarrow$ SetScheduleMultiplier(t) → can be fixed, decay or also be used for warm restarts

12: $\theta_t \leftarrow \theta_{t-1} - \eta_t(\alpha m_t / (\sqrt{v_t} + \varepsilon) + \lambda\theta_{t-1})$

13: until stopping criterion is met

14: return optimized parameters θ_t

Figure 5-2. Adam and AdamW implementation

But when it comes to adaptive gradient algorithms, they are different: while with decoupled weight decay, only the gradients of the loss function are adapted (with the weight decay step separated from the adaptive gradient mechanism), with L2 regularization, the sums of the gradient of the loss function and the gradient of the regularizer (i.e., the L2 norm of the weights) are adapted. Weights x with large typical gradient magnitudes s are regularized by a smaller relative amount than other weights because both types of gradients are normalized by their usual (summed) magnitudes under L2 regularization. By regularizing all weights at the same rate λ, decoupled weight decay efficiently regularizes weights x with large s more than normal L2 regularization.

6. RESULTS AND ANALYSIS

6.1 Dataset Analysis

6.1.1 Dataset Collection

Datasets containing 13851 Questions, 13851 Answers, and 4617 paragraphs were collected during data collection. It includes paragraphs consisting of related question-answer pairs. Each paragraph will have 3 questions and 3 answers. The dataset is stored as a Comma Separated Values file(.csv). The dataset has been collected manually and subsequently cleaned and filtered to ensure its suitability for the model. This laborious and time-consuming process was undertaken with the utmost care and dedication to craft a high-quality dataset, specifically designed for generating extractive subjective questions and answers from the provided input paragraphs.

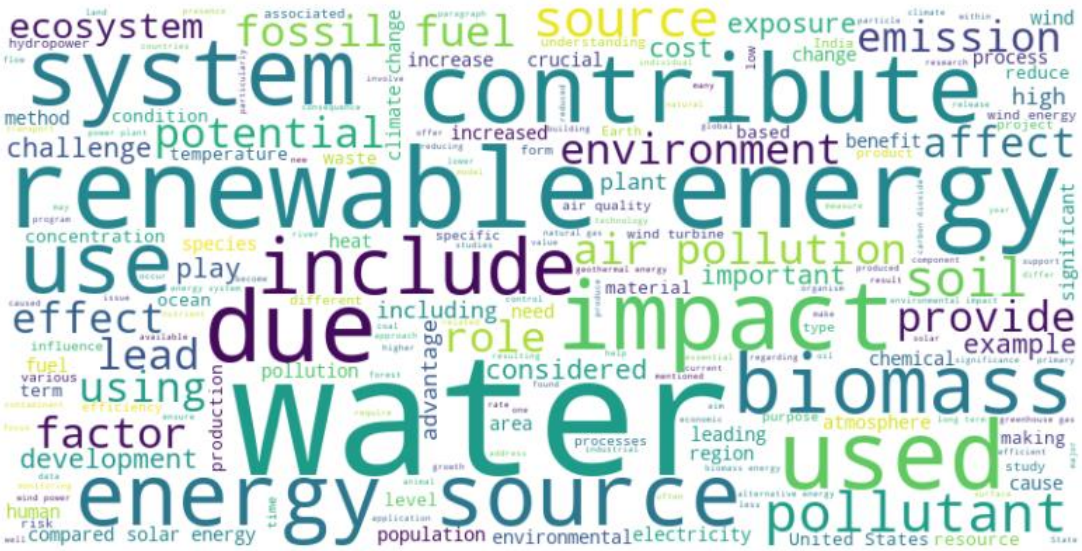


Figure 6-1. Word Cloud without Stop words

These paragraphs, carefully curated for above mention purpose, were predominantly sourced from the domain of renewable energy, pollution, and environmental science. An extensive effort was made to collect information from a wide range of reputable sources to guarantee the dataset's depth and diversity. These sources included articles, journals, reference books, research articles, authoritative websites, and other reliable repositories of knowledge.

During the collection phase, we made sure to gather paragraphs of different lengths. This was important because we wanted our dataset to include text of various sizes. By doing this, we ensured that our model could effectively understand and respond to text inputs of different complexities.

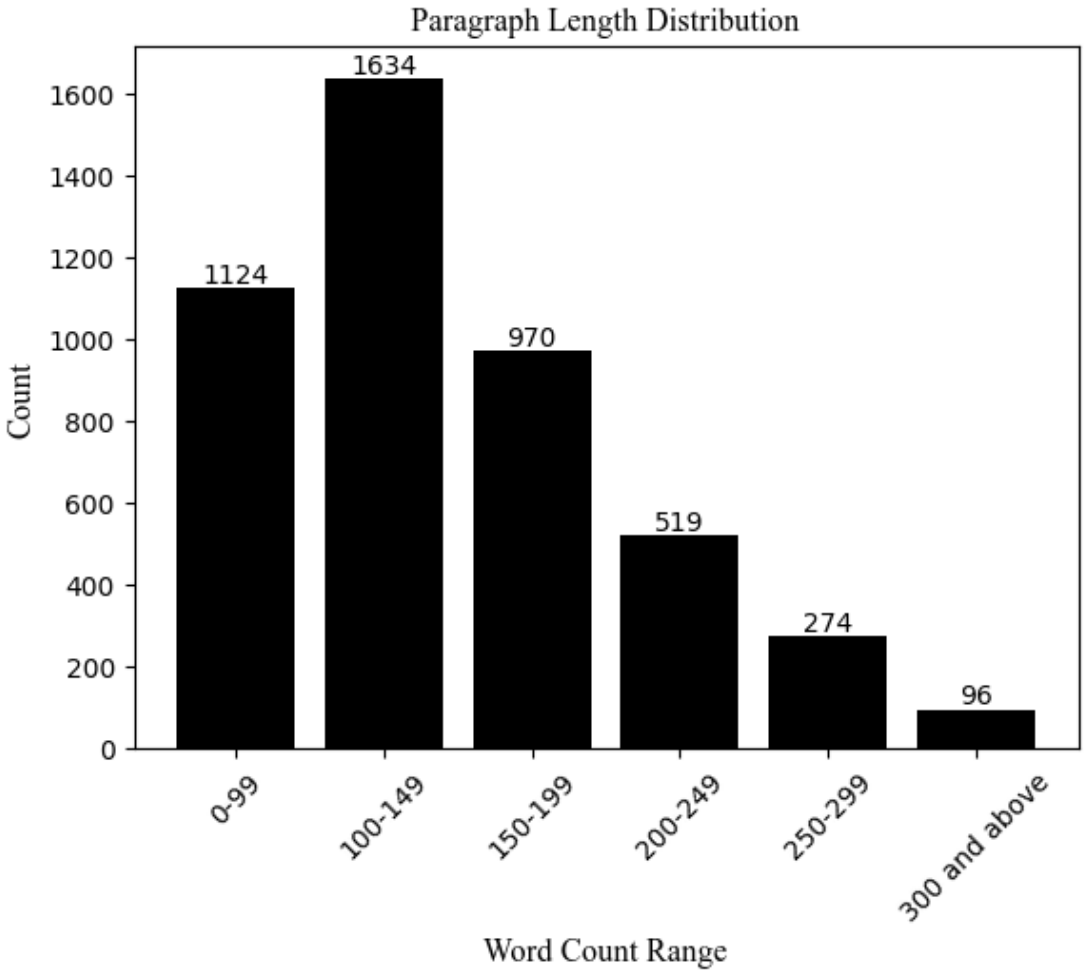


Figure 6-2. Paragraph Length Distribution

The T5 transformer is a powerful language model designed for text processing with a maximum token limit of 512. So, we collected a paragraph range of up to 350 words during our initial data collection phase. We reach this conclusion, to achieve optimal performance during training and inference. Many reasons limit us. firstly, deviating beyond this limit could lead to performance degradation and memory-related challenges during training and inference, hindering the model's overall effectiveness.

secondly, we reduce the computational complexity and training time significantly, which becomes especially crucial when dealing with large datasets. furthermore, by restricting the input length, we encourage the model to concentrate on the most salient and relevant information for the given task, avoiding potential confusion or loss of focus in longer text.

Along with collecting paragraphs, Questions were also collected manually which is extractive. we have designed the question nature to focus on specific information available within a given paragraph. The questions from the paragraphs are factual, descriptive, interrogative, Definition, Enumeration, etc. The reason behind this questioning nature is to efficiently retrieve precise answers, by tailoring the questions to target specific information and should focus on generating answers that would not cross the 50-word limitation.

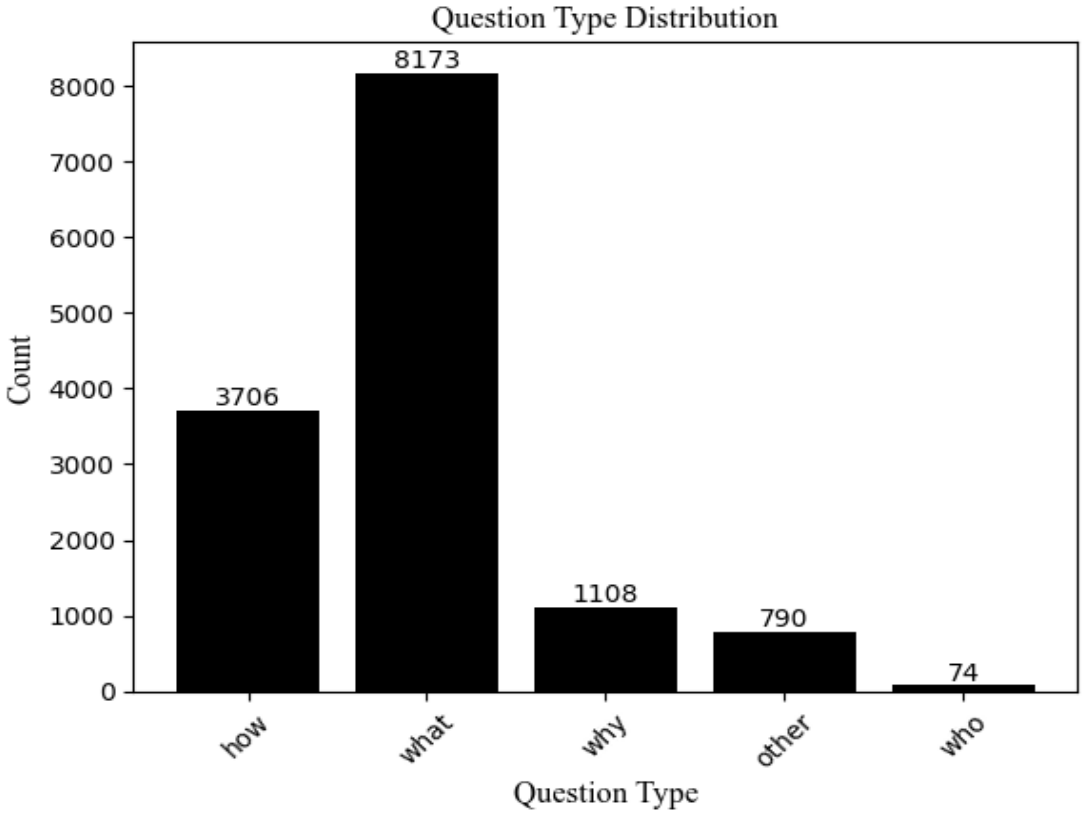


Figure 6-3. Question Type Distribution

The questions are thoughtfully categorized into different types, namely “why,” “how,” “what,” “other,” and “who.” Each category serves a particular purpose in exploring the

paragraph's content from various angles and perspectives, enriching the model's ability to provide insightful and contextually appropriate answers to a wide range of questions.

With paragraphs and questions, their corresponding answers are also collected in our dataset. The T5 transformer can generate outputs of different lengths. However, to ensure that the responses are concise, coherent, and easily understandable, we have established a criterion to limit the model's output to a maximum of 50 words. This enhances the quality of generated output by removing ambiguity which is mostly seen in long responses. Also, we need to consider the model's capacity. Although the T5 transformer can handle longer outputs, it is essential to find a balance between generating informative answers and overloading the model's capacity. Moreover, it helps to generate brief, to-the-point answers inside form paragraphs. lastly limiting output length simplifies the evaluation process.

Below is a scatter plot showing the relationship between paragraph length and answer length. Each data point in the scatter plot represents a specific paragraph and its corresponding answer from the Data Frame. The position of the point is determined by the lengths of the paragraph and its corresponding answer. points are clustered near a certain area of the plot, which suggests that there might be some common paragraph or answer lengths in the Data Frame.

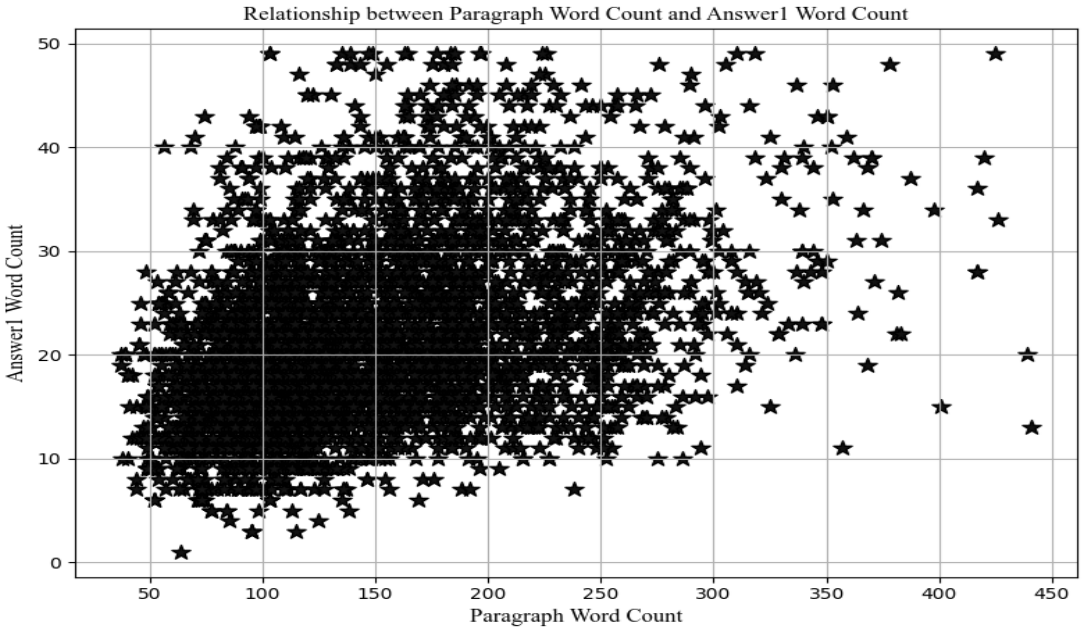


Figure 6-4. Relationship between Paragraph and Answer1 Word Count

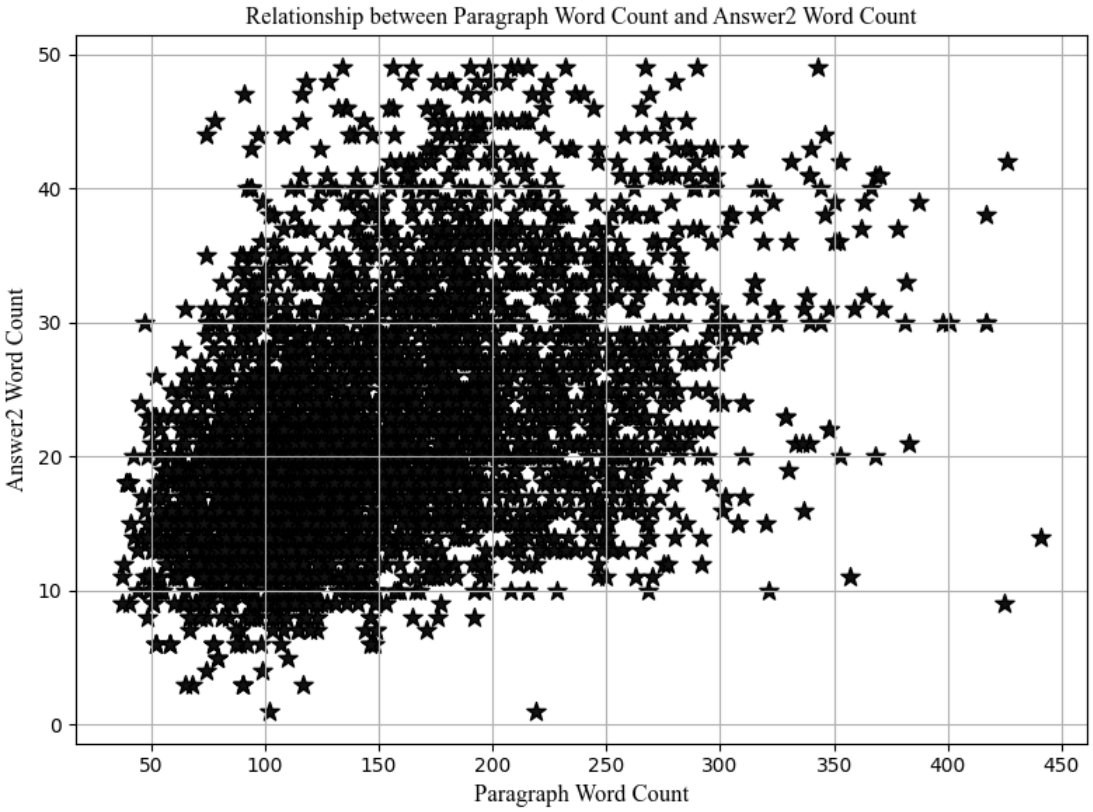


Figure 6-5. Relationship between Paragraph and Answer2 Word Count

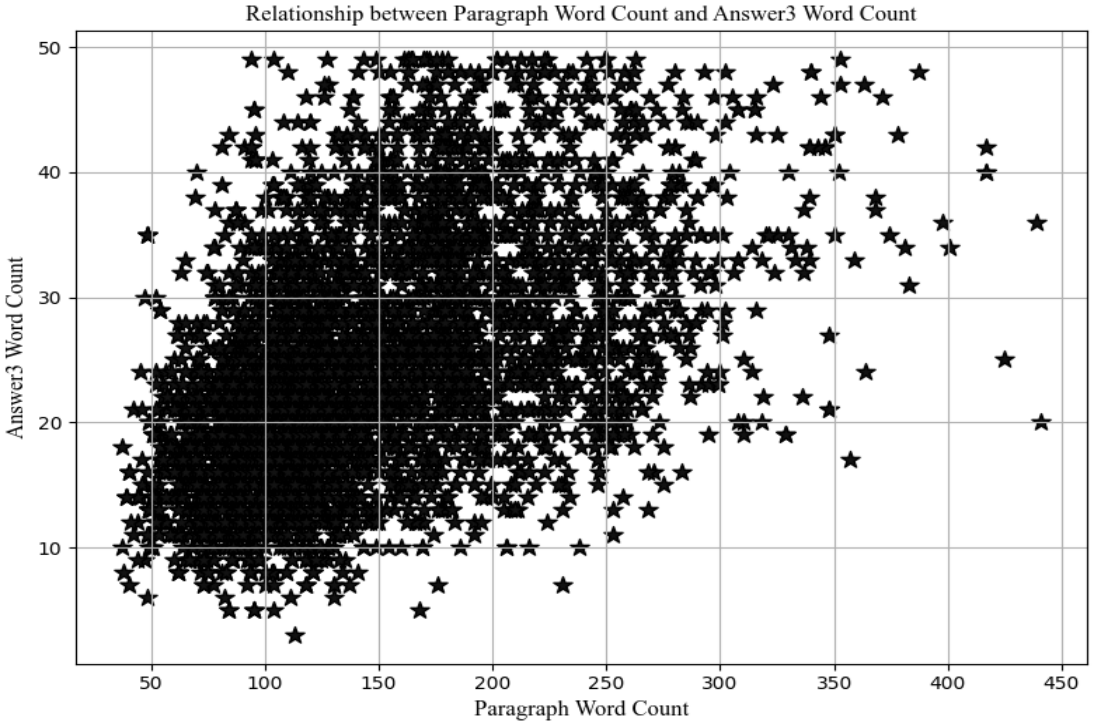


Figure 6-6. Relationship between Paragraph and Answer3 Word Count

The above scatter plot is useful to gain insights into the relationship between two paragraph lengths and Answer1, Answer2, and Answer3 lengths and helps in understanding the data and identifying potential patterns or trends.

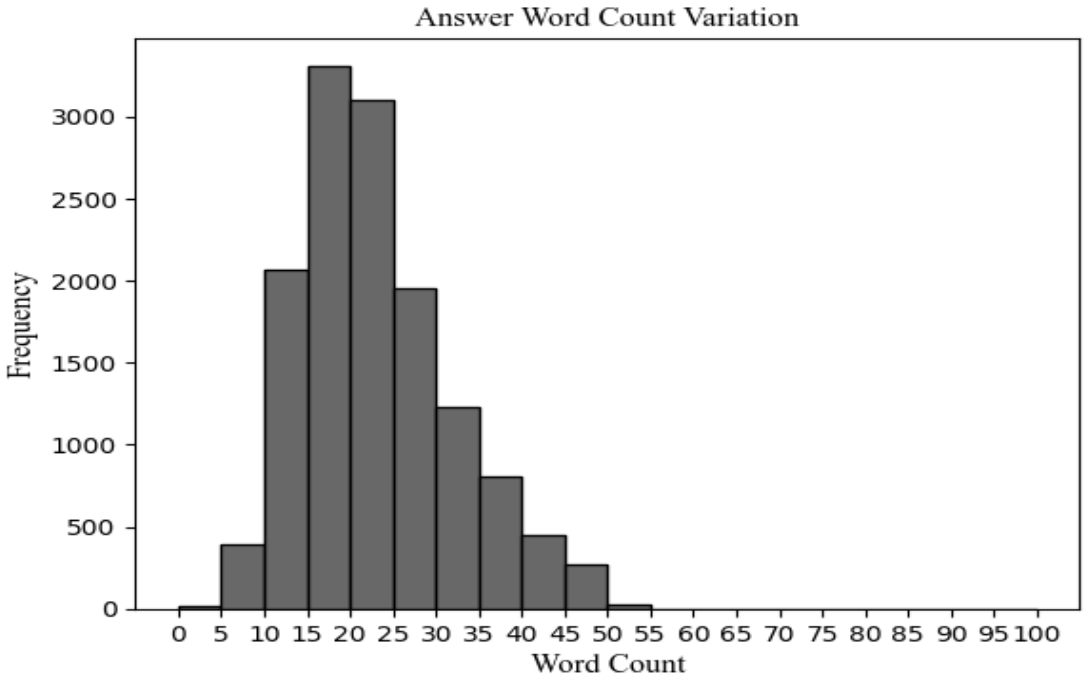


Figure 6-7. Answer Word Count Variation

The histogram above displays the distribution of word counts for the answers in the dataset. The x-axis represents the word count ranges, and the y-axis represents the frequency or the number of answers falling into each range. the histogram indicates that most answers in the dataset are concise and within the limit 0f 50 words, with a peak of around 15 to 20 words. There are fewer answers with extremely short or long word counts, indicating that the majority of responses are of moderate length. The distribution allows us to understand the central tendency, range, and variation of word counts in the answers, which can provide valuable insights into the characteristics of the dataset and the writing patterns of respondents.

6.2 First Model Training (For Question Generation)

For the first model, we were able to obtain an average BLEU score, ROUGE-1 score, ROUGE-2 score, ROUGE-L and cosine similarity score of 0.10, 0.33, 0.14, 0.31 and 0.73 respectively.

6.2.1 Loss vs Epoch Graph

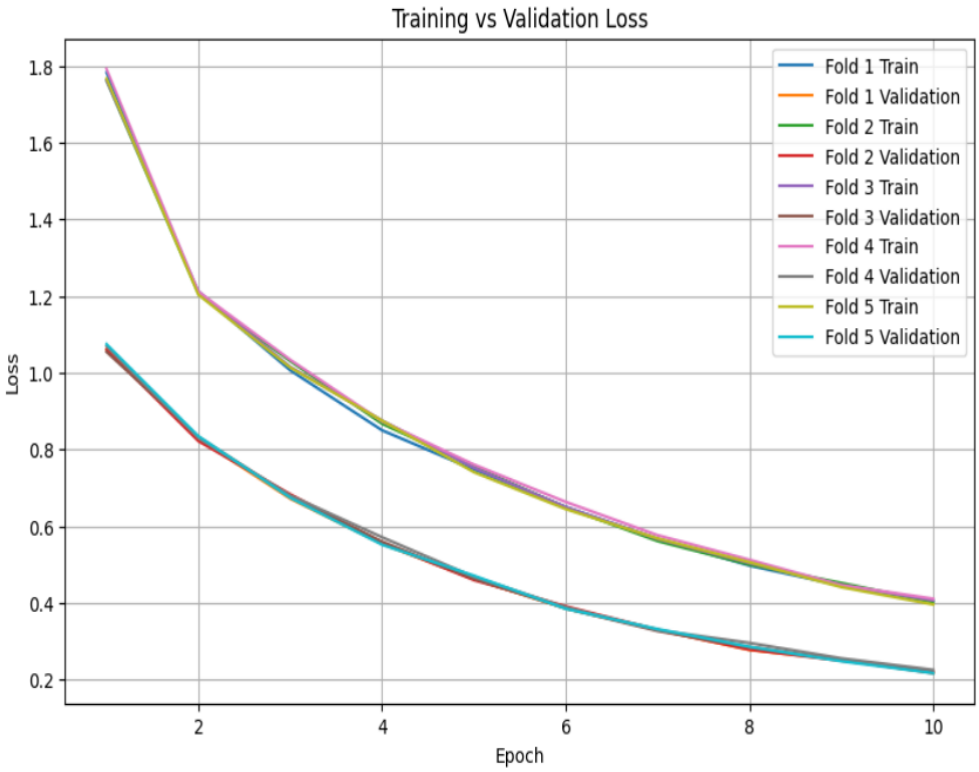


Figure 6-8. 1st Model Loss: Hyperparameter Set 1

Hyperparameters set are:

Table 6-1. 1st set of Hyperparameters for 1st Model

BATCH_SIZE	16
LEARNING_RATE	1e-4
EPOCHS	10

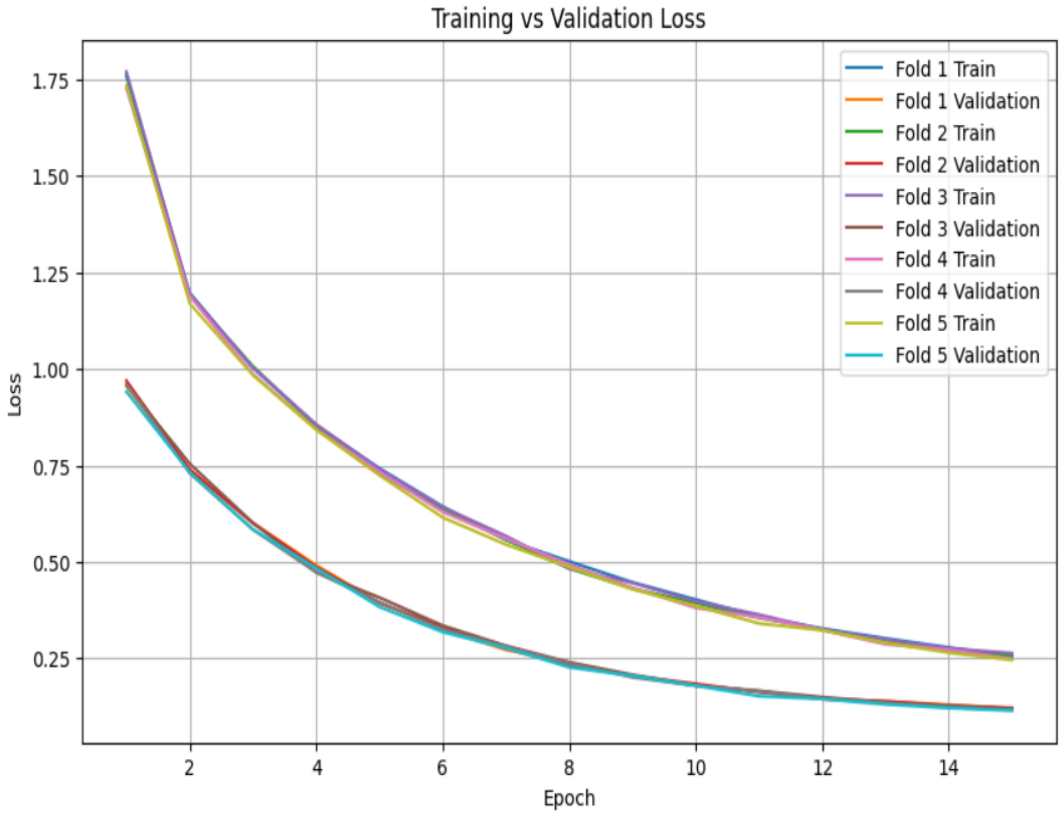


Figure 6-9. 1st Model Loss: Hyperparameter Set 2

Hyperparameters set are:

Table 6-2. 2nd set of Hyperparameters for 1st Model

BATCH_SIZE	16
LEARNING_RATE	1e-4
EPOCHS	15

After conducting training sessions for both 10 and 15 epochs, it was observed that despite the extended training duration, the improvement in the quality of generated questions during evaluation was not substantial. Thus, a decision was made to terminate the training process at the 15-epoch mark.

6.2.2 Number of Words in Question vs BLEU score

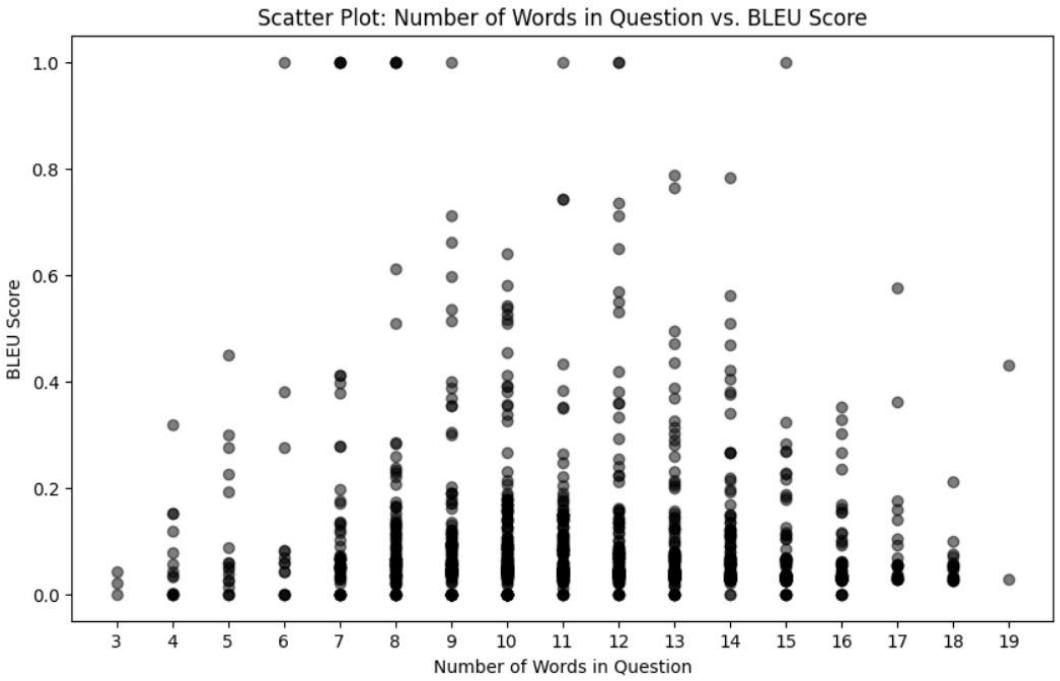


Figure 6-10. Number of Words in Question vs BLEU Score

The results show that there is a positive correlation between the number of model generated questions and the BLEU score. This means that as the number of model generated questions increases, the BLEU score also increases. This suggests that the model is able to generate more fluent and accurate questions as it is trained on more data.

However, the graph also shows that there is a lot of variation in BLEU scores for questions with the same number of words. This is because there are other factors that affect BLEU score, such as the complexity of the question and the quality of the machine question generation system.

6.2.3 BLEU Score Histogram

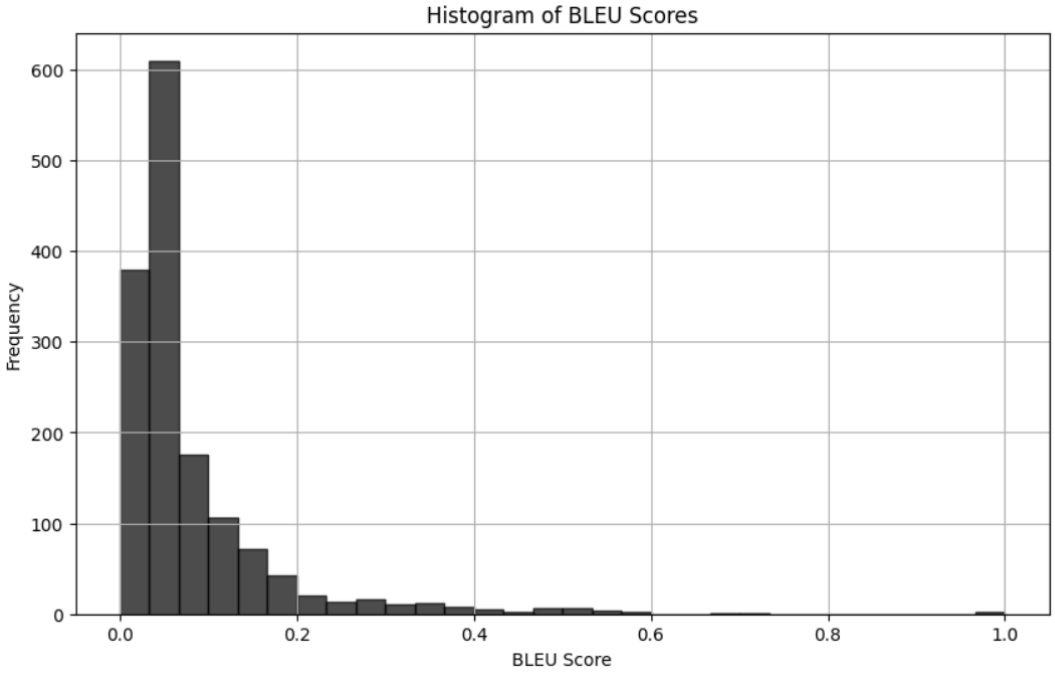


Figure 6-11. Histogram of BLEU scores for the first model

6.2.4 ROUGE-1 Histogram

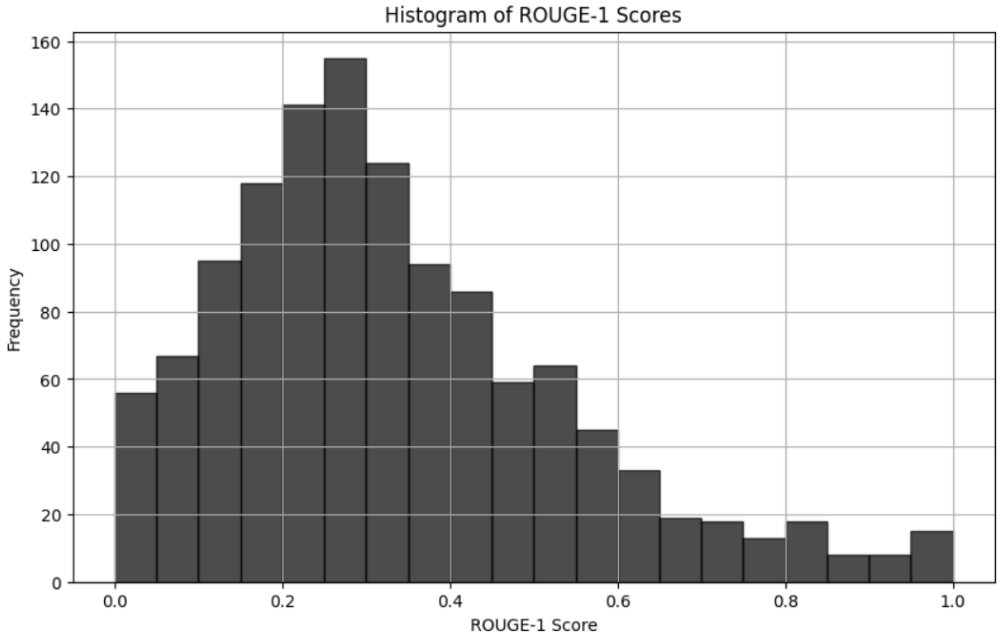


Figure 6-12. ROUGE-1 Score Histogram for the first model

6.2.5 ROUGE-2 Score Histogram

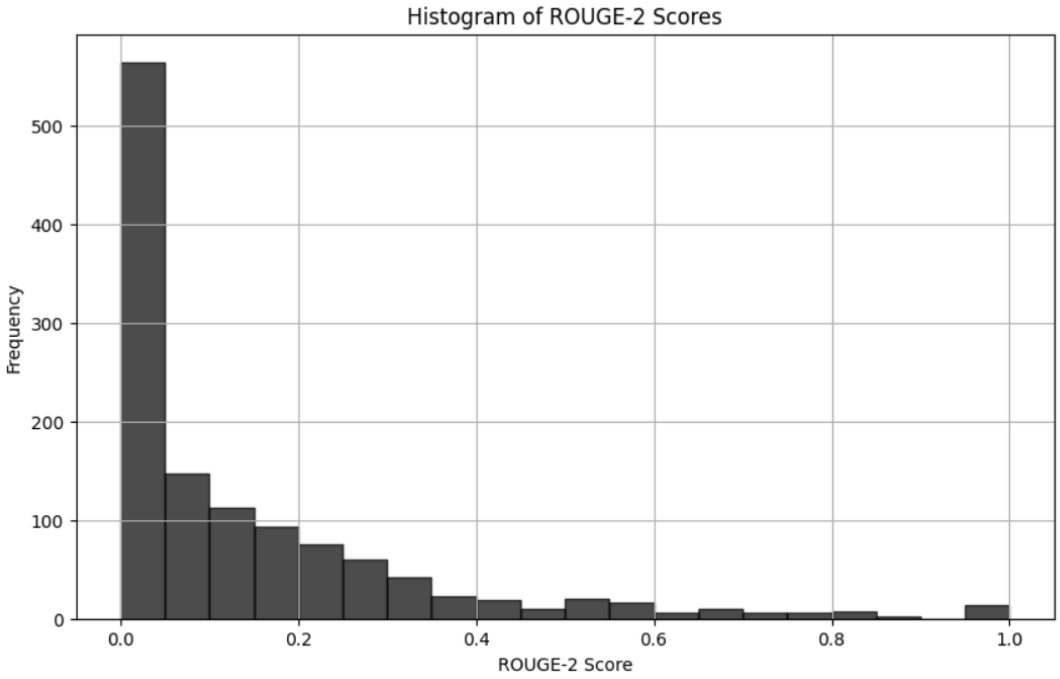


Figure 6-13. ROUGE-2 Score Histogram for the first model

6.2.6 ROUGE-L Score Histogram

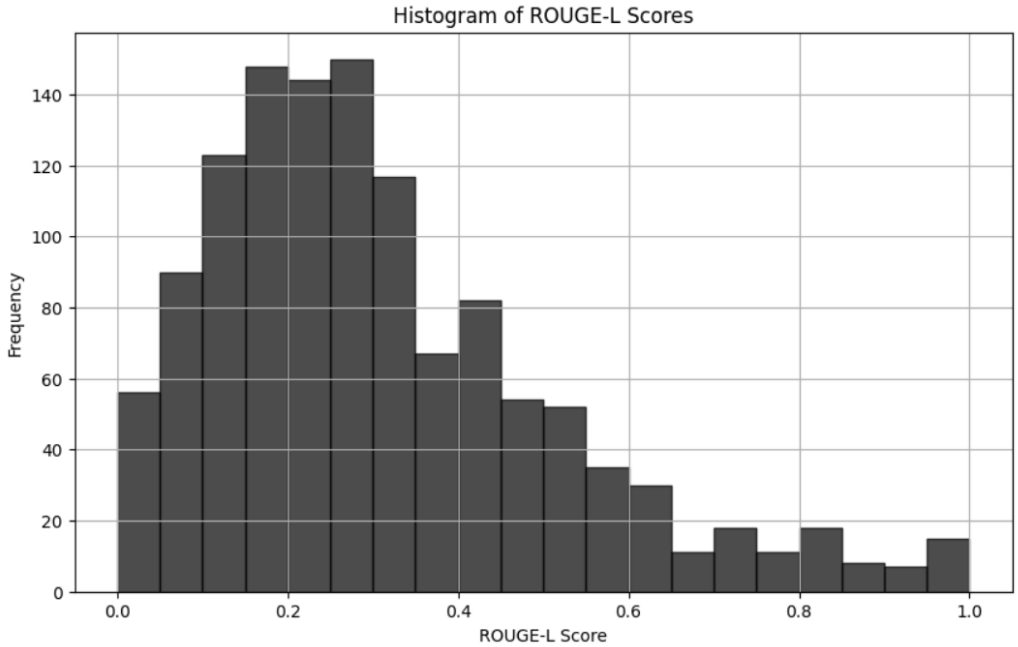


Figure 6-14. ROUGE-L Score Histogram for the first model

6.2.7 Cosine Similarity Score Histogram

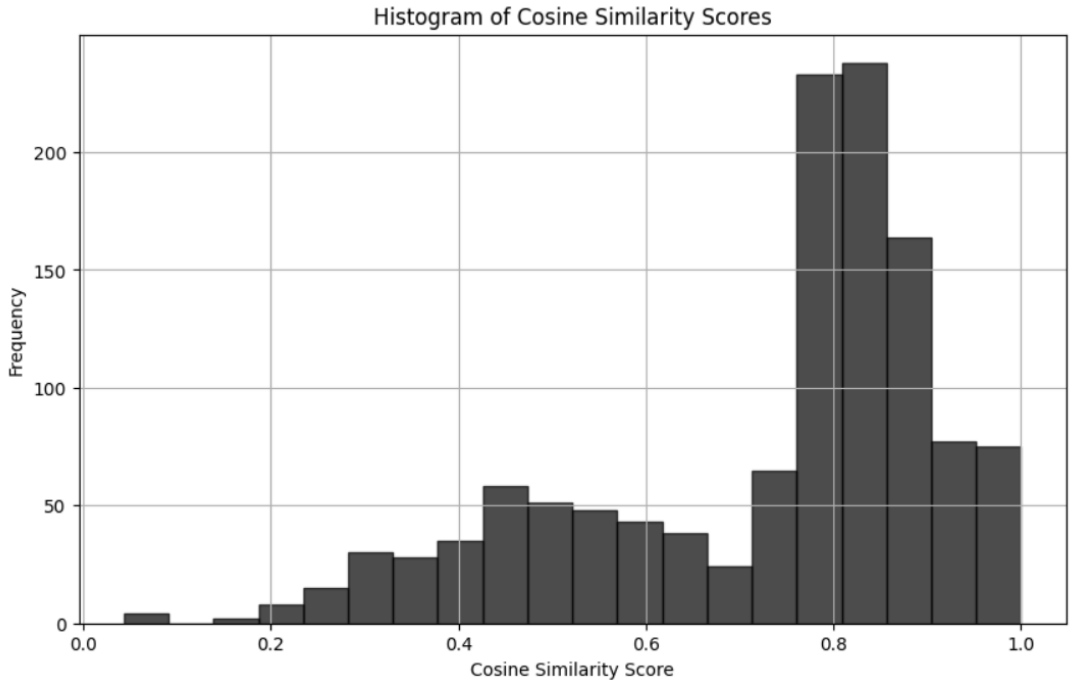


Figure 6-15. Cosine Similarity Score Histogram for 1st model

The cosine similarity scores between the reference questions and the model-generated questions are distributed between 0.5 to 1, with the majority of the scores between 0.8 to 1. This indicates that the model-generated questions are highly similar to the reference questions.

6.3 Second Model Training (For Answer Generation)

For the second model BLEU, ROUGE-1, ROUGE-2, ROUGE-L, and cosine similarity scores were 0.05, 0.23, 0.09,0.19, and 0.74 respectively.

6.3.1 Loss vs Epoch Graph

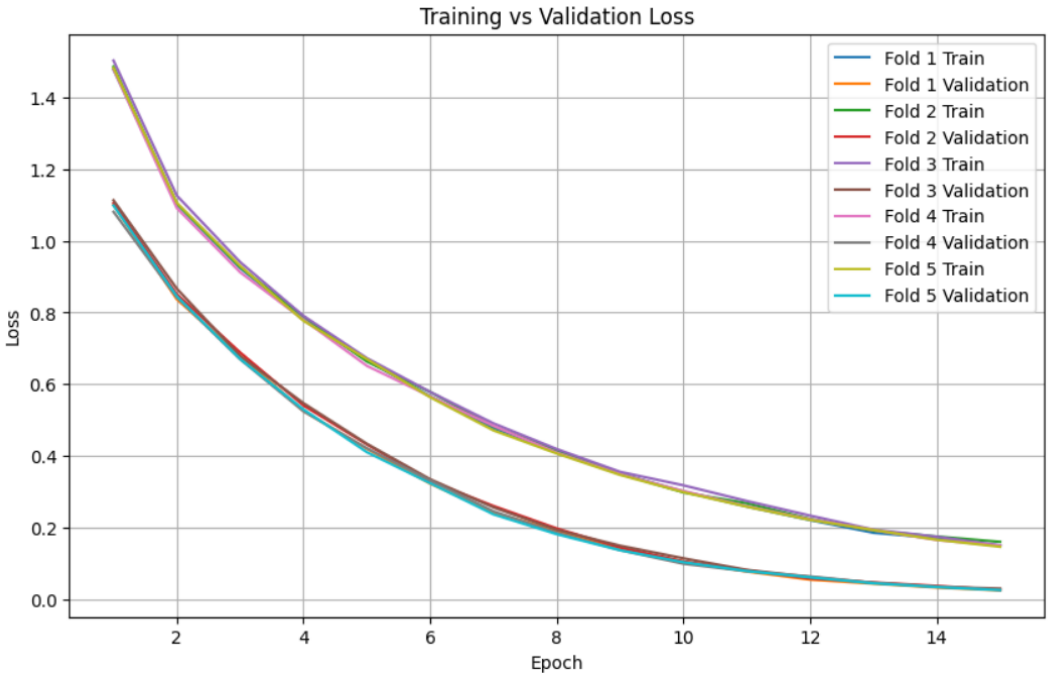


Figure 6-16. 2nd Model Loss: Hyperparameter Set 1

The set of hyperparameters are:

Table 6-3. 1st set of Hyperparameters for 2nd Model

BATCH_SIZE	16
LEARNING_RATE	1e-4
EPOCHS	15

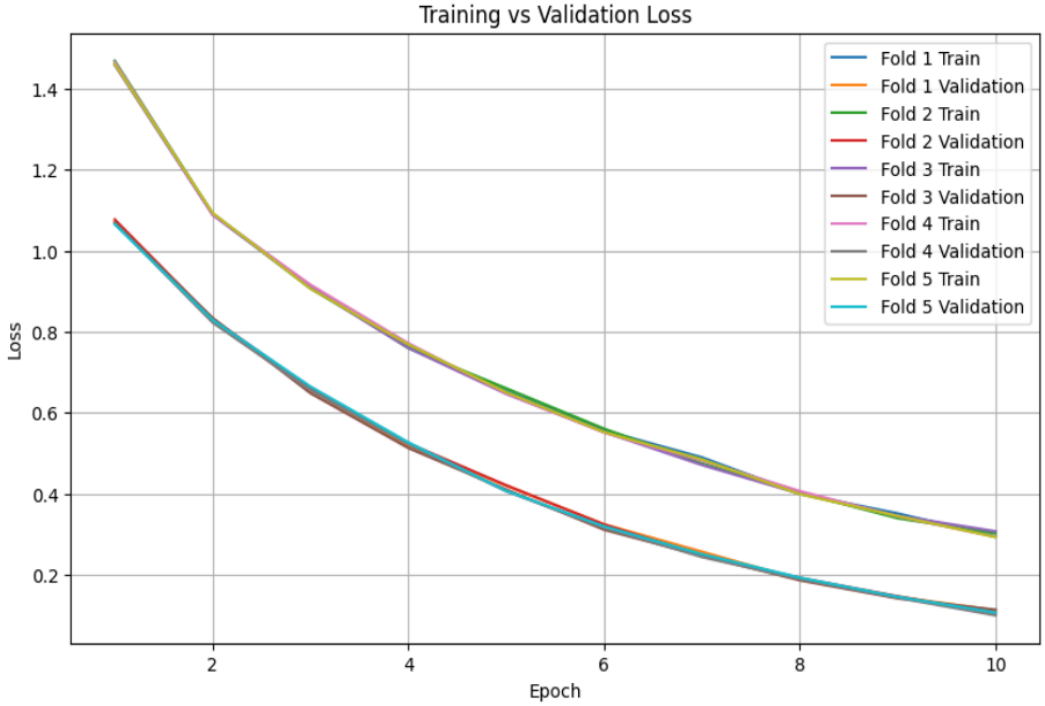


Figure 6-17. 2nd Model Loss: Hyperparameter Set 2

The set of hyperparameters are:

Table 6-4. 2nd set of Hyperparameters for 2nd Model

BATCH_SIZE	16
LEARNING_RATE	1e-4
EPOCHS	10

Its training was also stopped at 15 epochs due to the same reason as the first model.

6.3.2 Number of Words in Answer vs BLEU

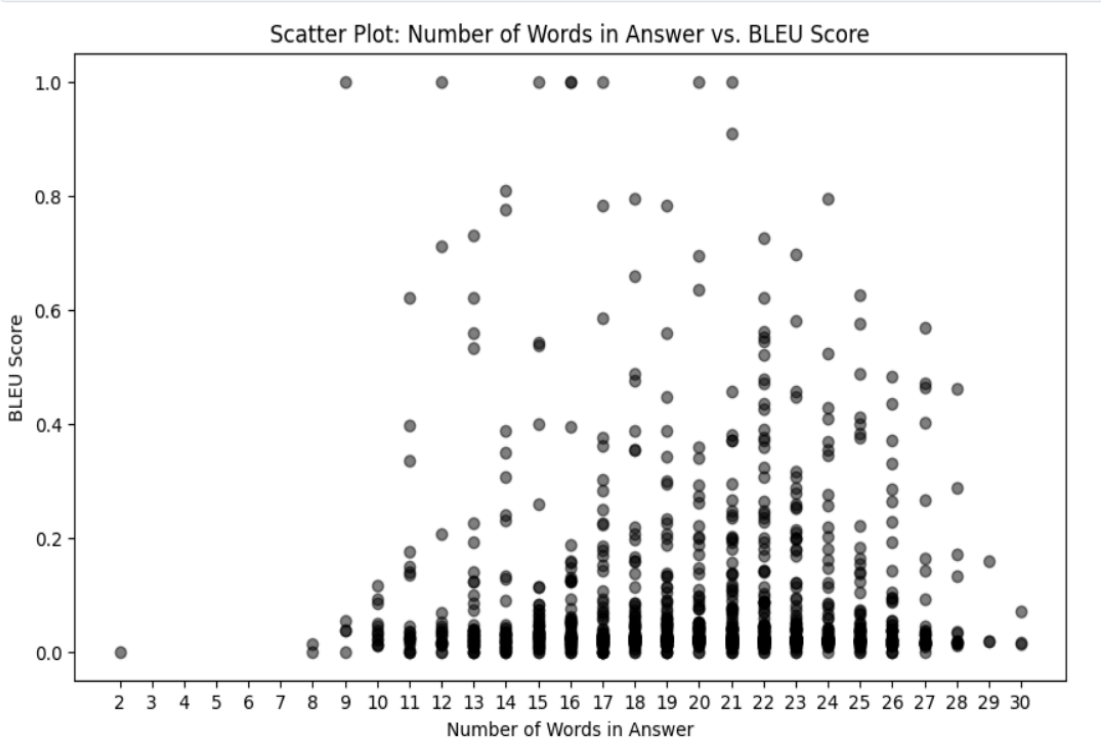


Figure 6-18. Number of Words in Answer vs BLEU Score

The graph shows that there is a positive correlation between the number of words in an answer and its BLEU score. This means that the more words there are in an answer, the more likely it is to be similar to a human-generated answer. This is because longer answers have more n-grams, which increases the chance that they will match the n-grams in the human-generated answer.

However, the graph also shows that there is a point of diminishing returns. After a certain number of words, the BLEU score starts to decrease.

The optimal number of words in an answer will vary depending on the task. For some tasks, a longer answer may be necessary to convey all of the necessary information. For other tasks, a shorter answer may be more effective.

6.3.3 Histogram of BLEU Score

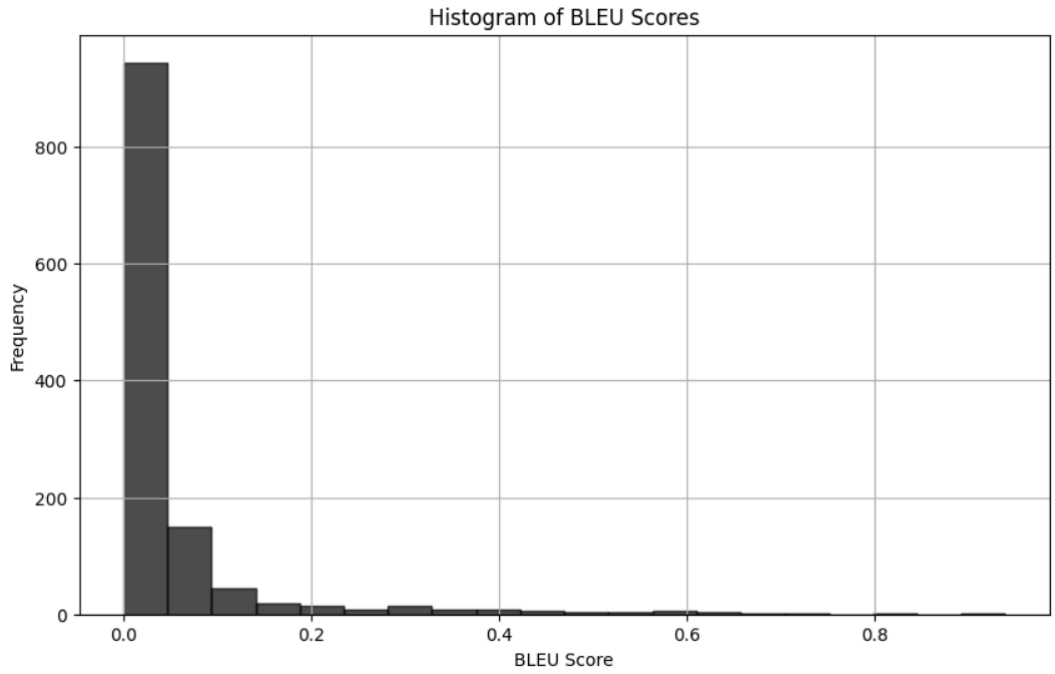


Figure 6-19. Histogram of BLEU Scores for the second model

6.3.4 Histogram of ROUGE-1 Score

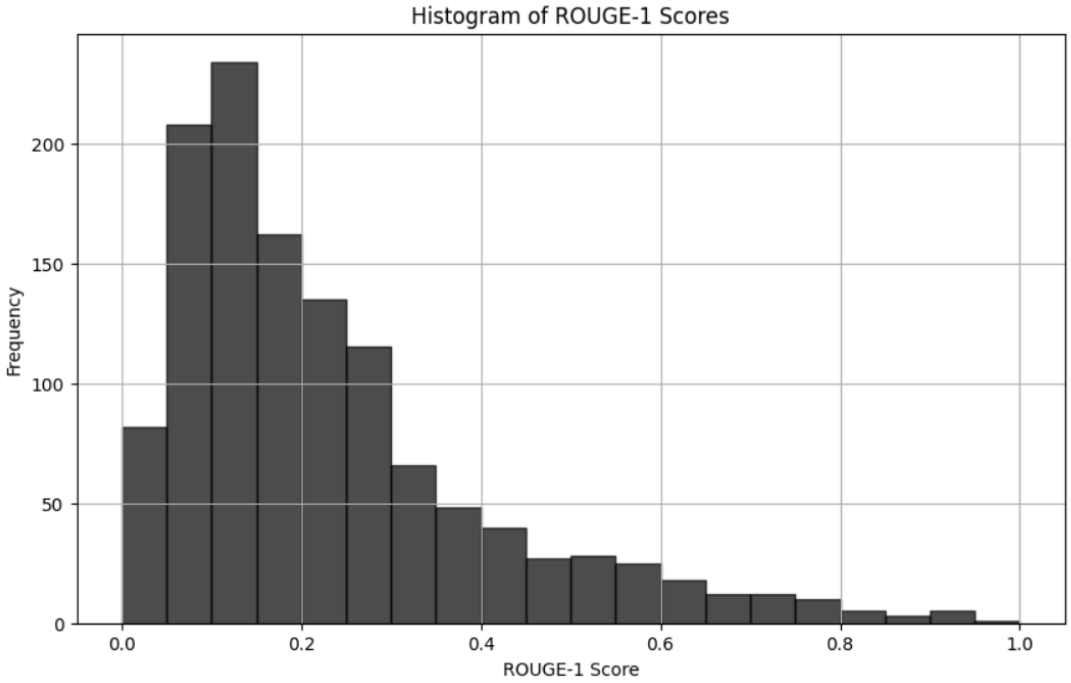


Figure 6-20. Histogram of ROUGE-1 Score for the second model

6.3.5 Histogram for ROUGE-2 Score

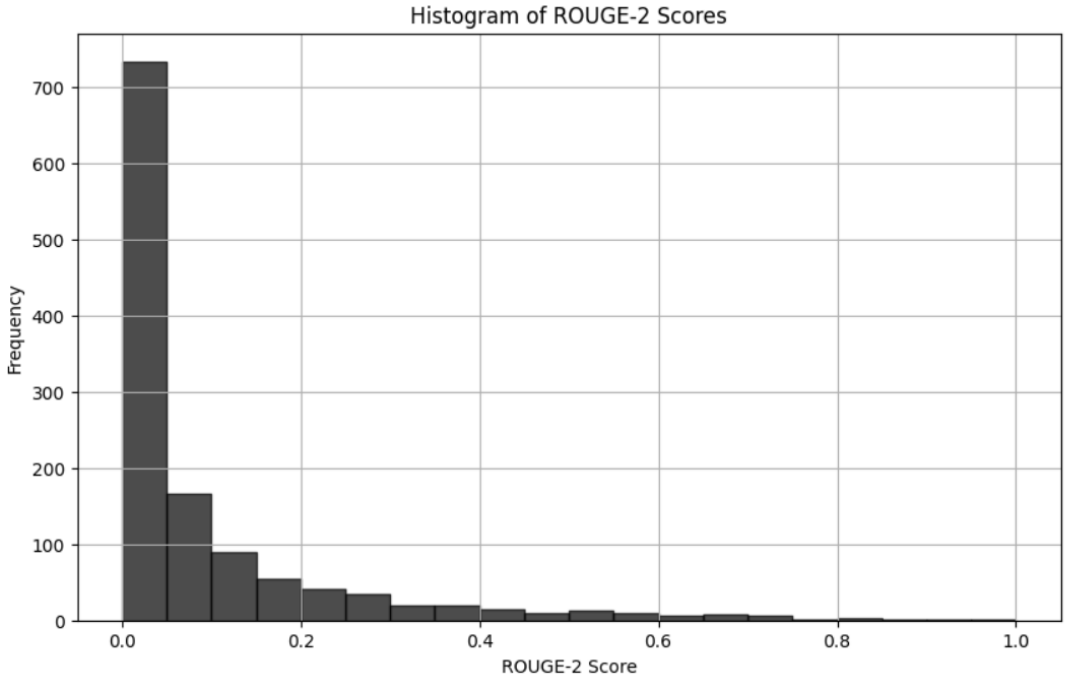


Figure 6-21. Histogram of ROUGE-2 Score for Second Model

6.3.6 Histogram for ROUGE-L Score

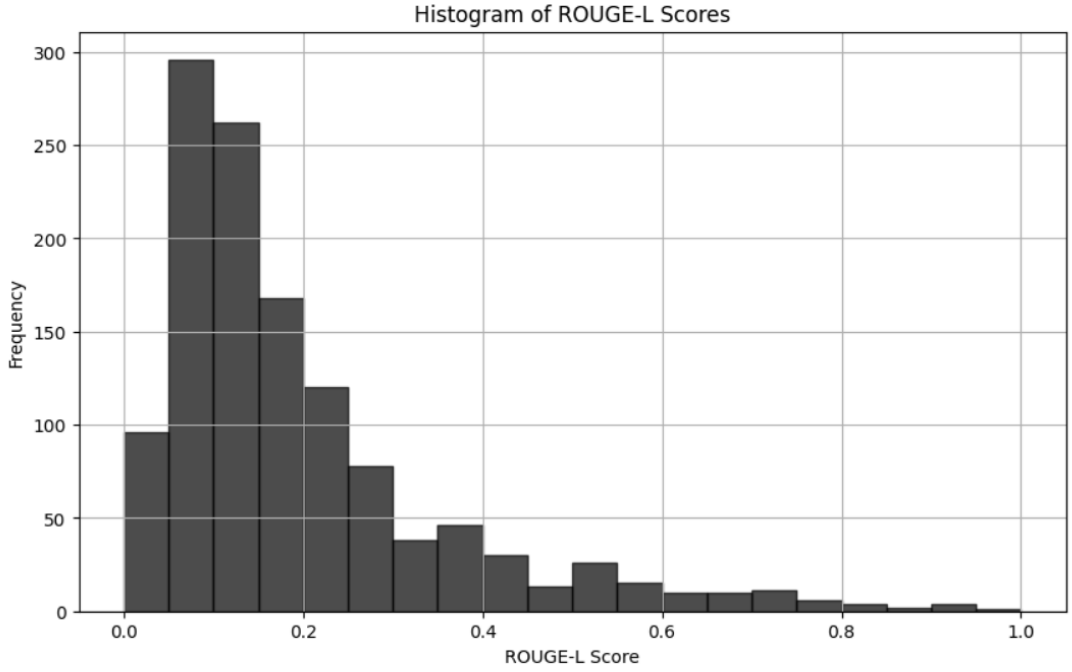


Figure 6-22. Histogram for ROUGE-L Score for the second model

6.3.7 Histogram of Cosine Similarity Score

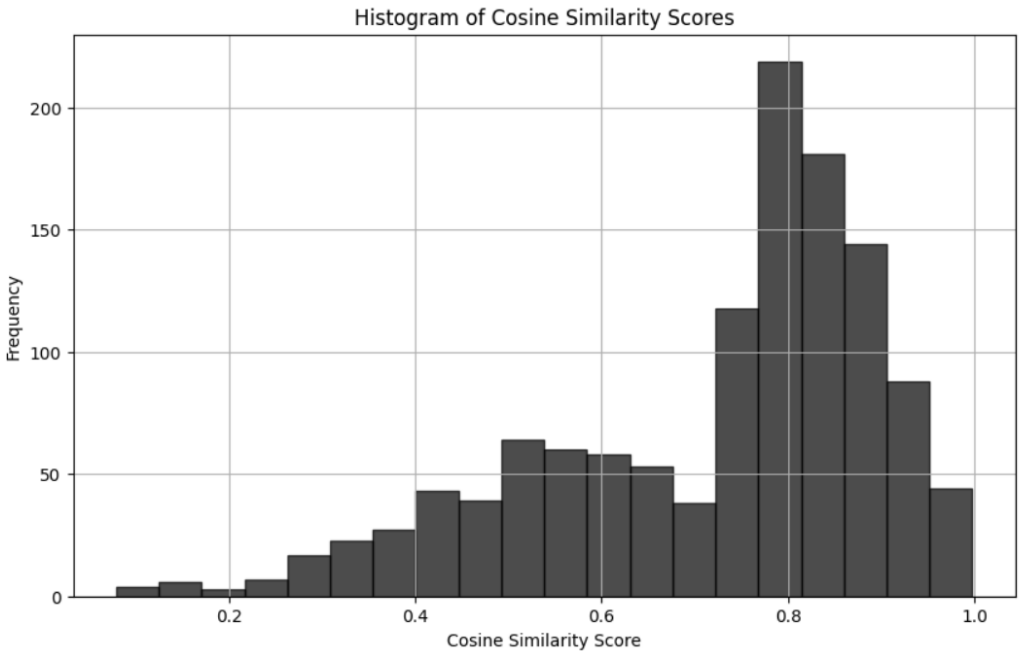


Figure 6-23. Histogram of Cosine Similarity Score for 2nd model

There are few model-generated answers that have a low cosine similarity score. This means that these answers are very different from the reference answers. This could be because the model is not able to understand the question, or because it is not able to generate an answer that is relevant to the question.

Overall, the graph shows that the model is able to generate answers that are similar to the human-written answers. However, there are few model-generated answers that are very different from the reference answers. This could be because the model is not able to understand the question, or because it is not able to generate an answer that is relevant to the question.

6.4 ROUGE Histogram

In the rouge histogram, we can see translation quality for question generation has a lower rouge score. Most of the answer has a low rouge score but shows better than the Belu score.

6.5 BLEU Histogram

The BLEU Score vs Frequency histogram reveals a compelling trend in the translation quality of the generated questions and answers. It reveals that a significant portion of the generated questions and answers falls within the lower BELU score range, suggesting suboptimal translation accuracy for a substantial number of instances. This insight highlights the need for targeted improvements in the translation model, particularly in enhancing the quality of questions and answers falling within the lower BLEU score range.

6.6 Overall Summary

Although the question generated by our model is fine rouge and BLEU score are low. The disparity between the human judgment of generated questions or answers and the low BLEU and ROUGE scores can be attributed to several factors. One key factor is the lack of lexical diversity; even if the output seems appropriate to a human, the model might not be generating sufficiently diverse answers or questions compared to the reference data, affecting the BLEU and ROUGE scores sensitive to specific word choices. Additionally, the model might lack deep contextual understanding, miss subtle cues, or fail to grasp topic intricacies, causing discrepancies between human judgment and automated metrics. Ambiguity handling poses another challenge; if the model struggles with unclear phrases or references commonly found in real-world questions or answers, it may diverge from reference answers, leading to lower scores. Furthermore, the size and diversity of the training dataset play a role; if the dataset isn't comprehensive enough, the model might miss various phrasings and contexts, impacting performance on new or slightly varied data. Addressing these issues involves enriching the training data, refining the model to capture nuances and handle ambiguity, and employing techniques like diverse decoding strategies to enhance lexical diversity in the generated text.

The graph shows the distribution of cosine similarity scores between model generated answers and reference answers. The cosine similarity score measures the similarity between two vectors. In this case, the vectors are the word embeddings of the model generated answers and the reference answers. The cosine similarity score can range

from -1 to 1. A score of 1 means that the two vectors are identical, and a score of -1 means that the two vectors are completely different.

The prevailing metrics such as BLEU and ROUGE, designed for evaluating the quality of automatically generated text, have shown limitations in certain contexts. BLEU and ROUGE rely heavily on lexical overlap and n-gram matches between the generated text and reference text. This approach can be restrictive and may not capture the semantic nuances or overall coherence of the generated text. Conversely, cosine similarity, a metric commonly employed in natural language processing tasks, offers a more nuanced approach. By quantifying the similarity in vector space between the generated text and a set of predefined features, cosine similarity can capture semantic similarities even in the absence of exact lexical matches. This makes it more adaptable to diverse contexts and less reliant on the availability of reference text. Moreover, cosine similarity can be efficiently computed and scaled to large datasets, making it a viable option for evaluating text quality in various applications.

6.7 Output

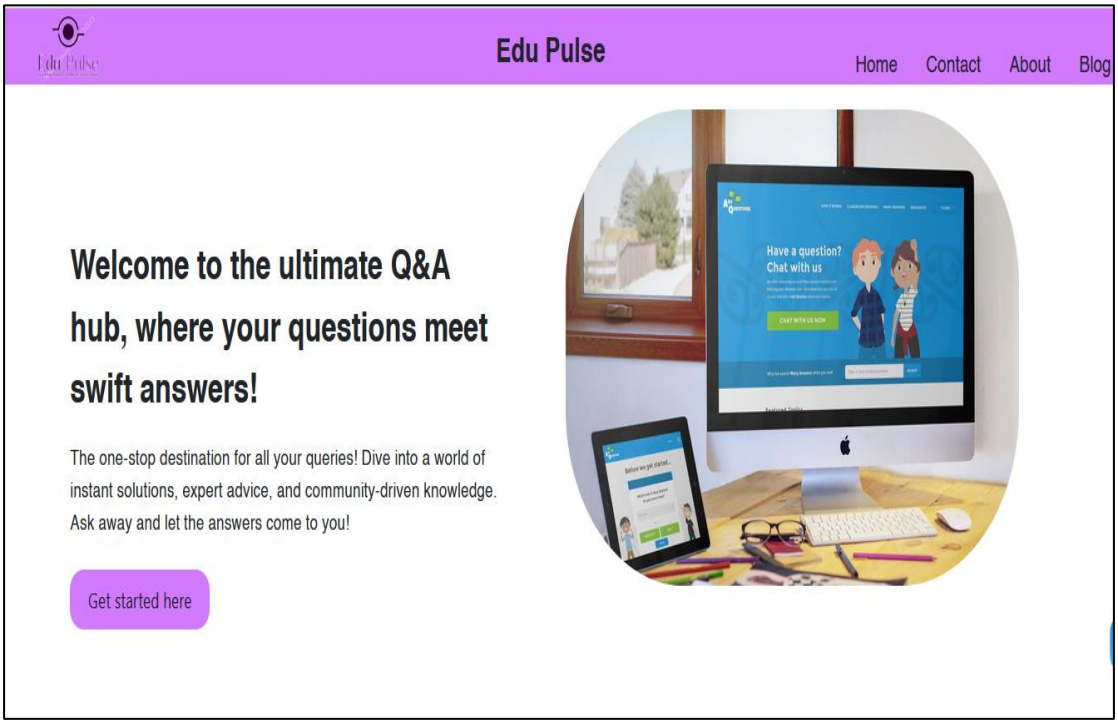


Figure 6-24. Welcome page

This is the welcome page of our website. As visitors navigate through our website, they will encounter a homepage featuring a sleek navigation bar displaying our project's name prominently in bold block letters. The website currently offers several pages, including Home, Contact, and About with the homepage being the sole fully developed section at present.

Beyond the navigation bar, visitors will find a warm, block-lettered greeting along with a succinct introduction to Edu Pulse, setting the stage for the educational journey ahead. To begin exploring, visitors may locate the strategically placed “Get started here” button and join our community.

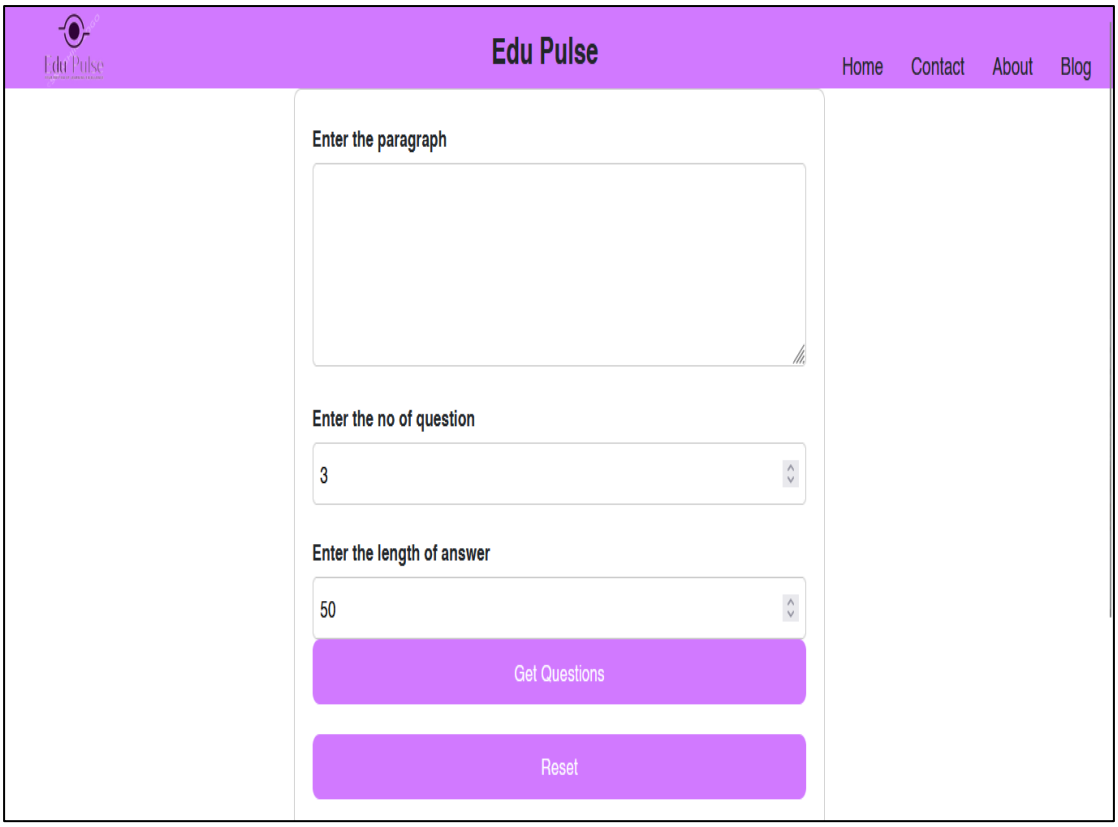


Figure 6-25. UI to enter paragraph

This is the homepage of our website. On the homepage, the website begins with a strategically placed navbar, a component previously discussed on the welcome page. Following the navbar, an input field labeled “Enter the paragraph” is provided, enabling users to enter their paragraphs. Adjacent to this input field, a “Get Questions” button empowers users to generate the desired questions and answers. The “Reset” button is used when user wants to enter a new paragraph after already entering a paragraph.

In the designated input field labeled “Enter the paragraph,” users have the option to input a paragraph from which they will receive questions and answers. Additionally, users are able to specify the number of questions they desire and the desired length of the answers. Following the submission of the input paragraph, users will be prompted to provide their own answers. Subsequently, the cosine similarity between their answers and those generated by the model after it has been grammatically corrected will be computed and displayed.

We have tried to generate questions and generations in different fields. Some of the examples are shown below. The paragraph in first example contains the paragraph in which the model was trained while others don’t.

Example 1: Domain = Renewable Energy

Paragraph used:

Renewable energy stands at the forefront of the global transition towards a sustainable and carbon-neutral future, offering a multifaceted solution to the intertwined challenges of climate change, energy security, and environmental degradation. At its core, renewable energy derives from naturally replenished resources such as sunlight, wind, water, biomass, and geothermal heat, presenting a stark departure from the finite and polluting nature of fossil fuels. Solar power, perhaps the most emblematic of renewable energy sources, harnesses the inexhaustible energy of the sun through photovoltaic cells and solar thermal systems, powering homes, businesses, and entire communities with clean and abundant electricity. Wind energy, propelled by the kinetic force of moving air, has rapidly matured into a mainstream energy source, with towering wind turbines dotting landscapes worldwide and providing a significant portion of electricity generation in many regions. Hydroelectric power, originating from the gravitational energy of flowing water, remains a stalwart of renewable energy infrastructure, offering reliable and dispatchable electricity generation while also providing ancillary benefits such as flood control and water storage. Beyond these well-established sources, emerging technologies such as tidal and wave energy hold promise for tapping into the immense power of the oceans, further diversifying the renewable energy portfolio. Moreover, biomass energy derived from organic materials and geothermal energy

extracted from the Earth's heat provide versatile options for heat and power generation, particularly in sectors like heating, industry, and agriculture. As nations strive to meet ambitious climate targets and transition towards more sustainable energy systems, the importance of renewable energy as a cornerstone of this transformation cannot be overstated. From reducing greenhouse gas emissions and mitigating air pollution to fostering energy independence and creating green jobs, the benefits of embracing renewable energy reverberate across economies and ecosystems alike, paving the way for a cleaner, healthier, and more resilient planet for generations to come.

Number of questions to generate = 5

Maximum length of Answer = 40

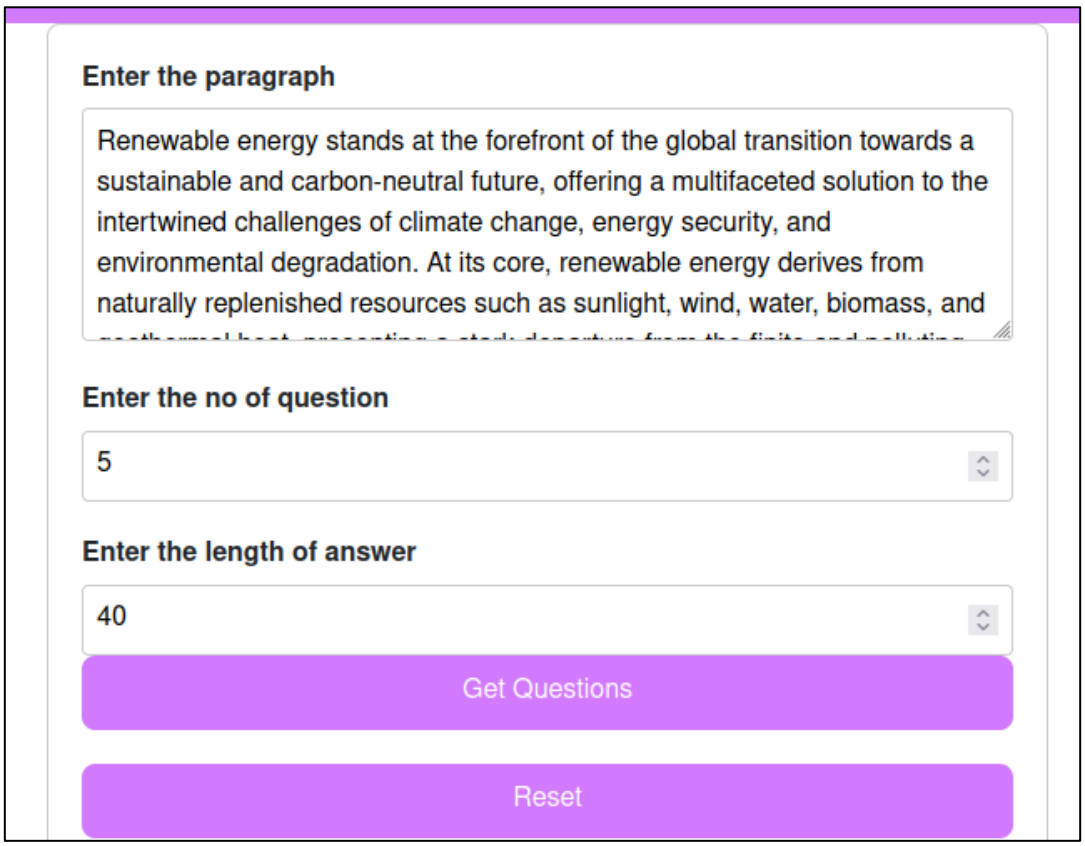


Figure 6-26. Giving paragraph of “Renewable Energy” to the models

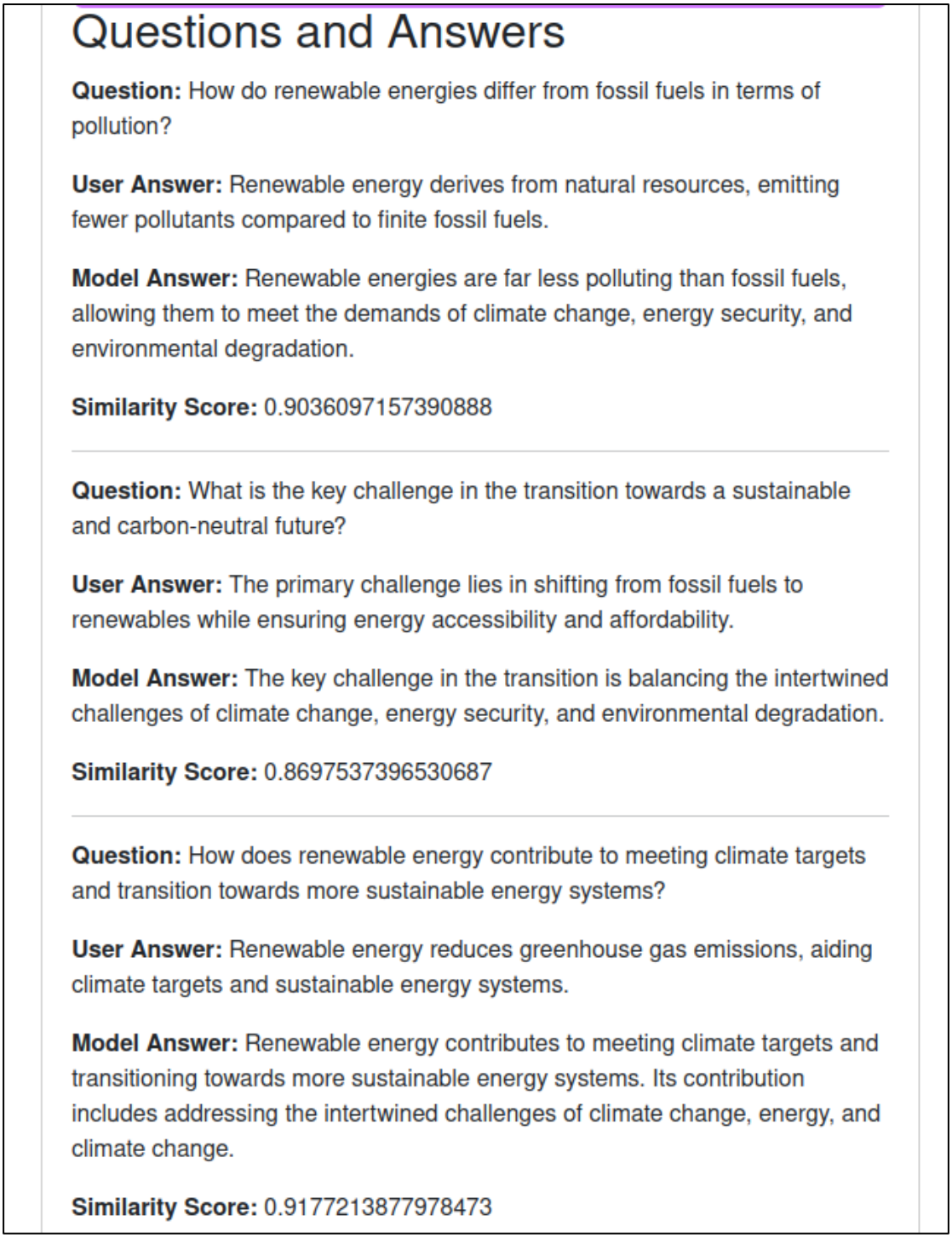


Figure 6-27. Generated Question and Answer in “Renewable Energy” domain-1

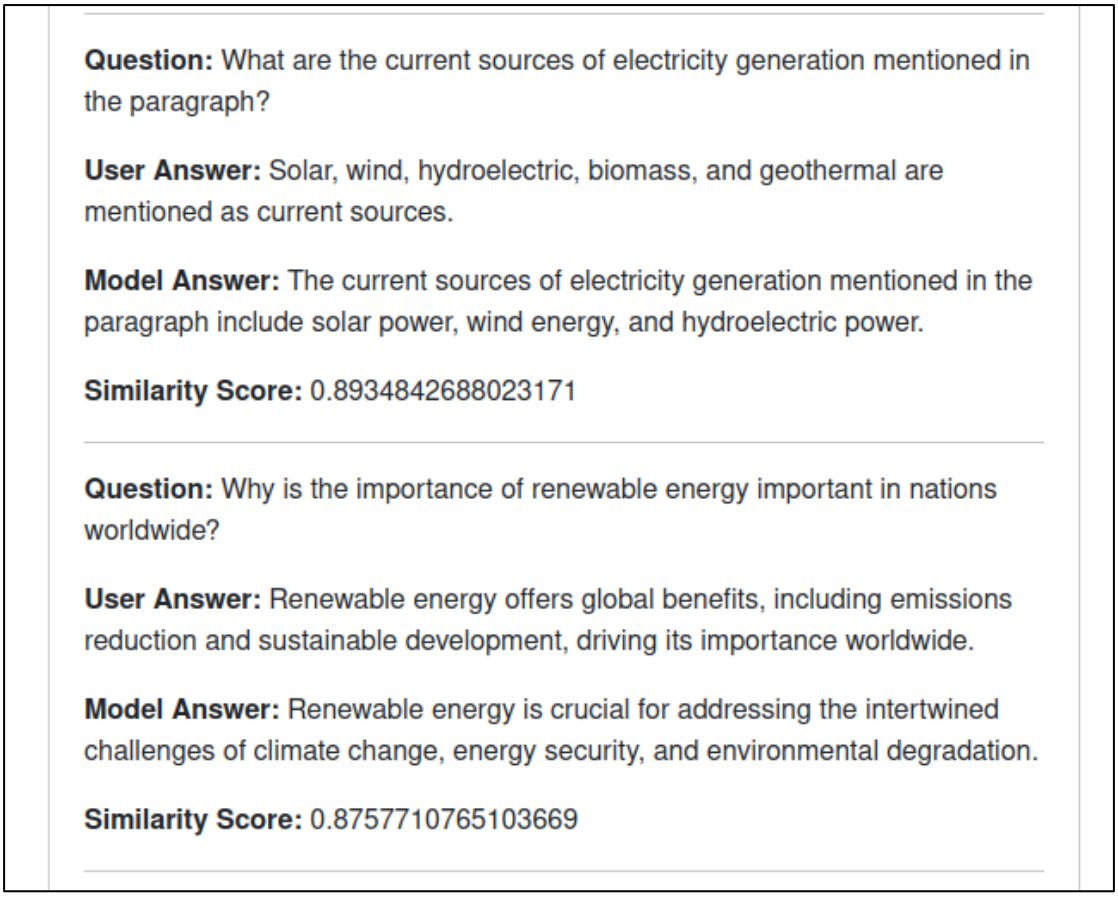


Figure 6-28. Generated Question and Answer in “Renewable Energy” domain-2

Example 2: Domain = Politics

Paragraph used:

Former Maryland Gov. Larry Hogan announced Friday that he is running for US Senate this year, providing Republicans with a strong candidate in their bid to flip a seat in a deep-blue state.

“I am running for the United States Senate – not to serve one party – but to stand up to both parties, fight for Maryland, and fix our nation’s broken politics. It’s what I did as Maryland’s governor, and it’s exactly how I’ll serve Maryland in the Senate. Let’s get back to work,” Hogan said in a message posted to X.

He argued that Washington is “completely broken” and that he’s “completely fed up with politics as usual.”

“Enough is enough. We can do so much better, but not if we keep electing the same kind of typical partisan politicians,” Hogan said.

A moderate Republican, Hogan, 67, has long been critical of former President Donald Trump’s influence on the Republican Party. He said he did not vote for him in 2016 or 2020 and endorsed former South Carolina Gov. Nikki Haley for president this year. Hogan was once seen as a potential challenger to Trump in the 2020 GOP primary but decided to instead focus on his second term as governor.

Number of Questions to generate = 3

Maximum length of answer = 50

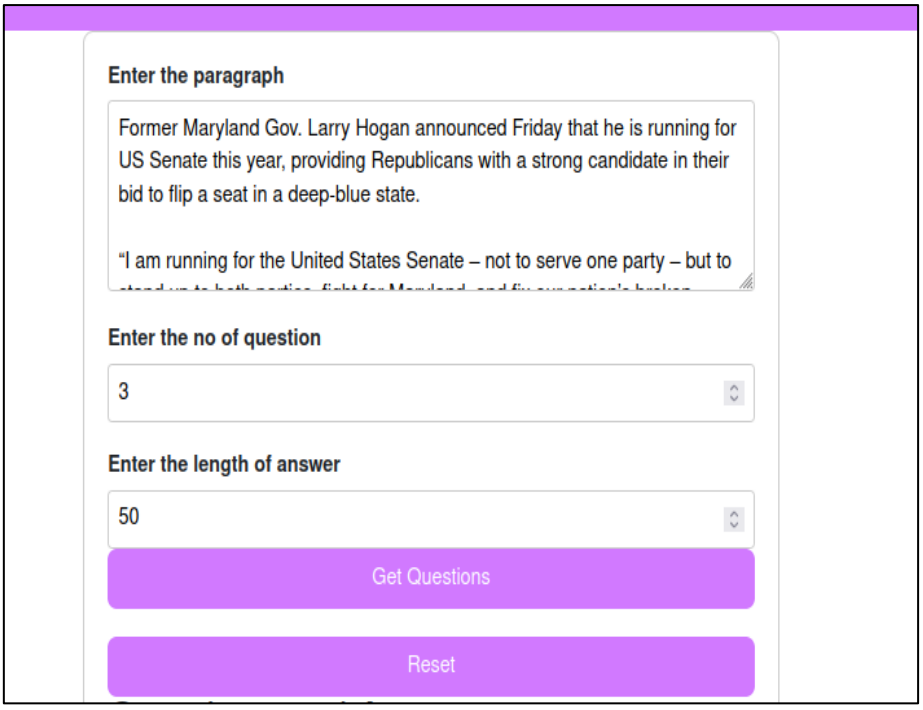


Figure 6-29. Giving paragraph of “Politics” to the models

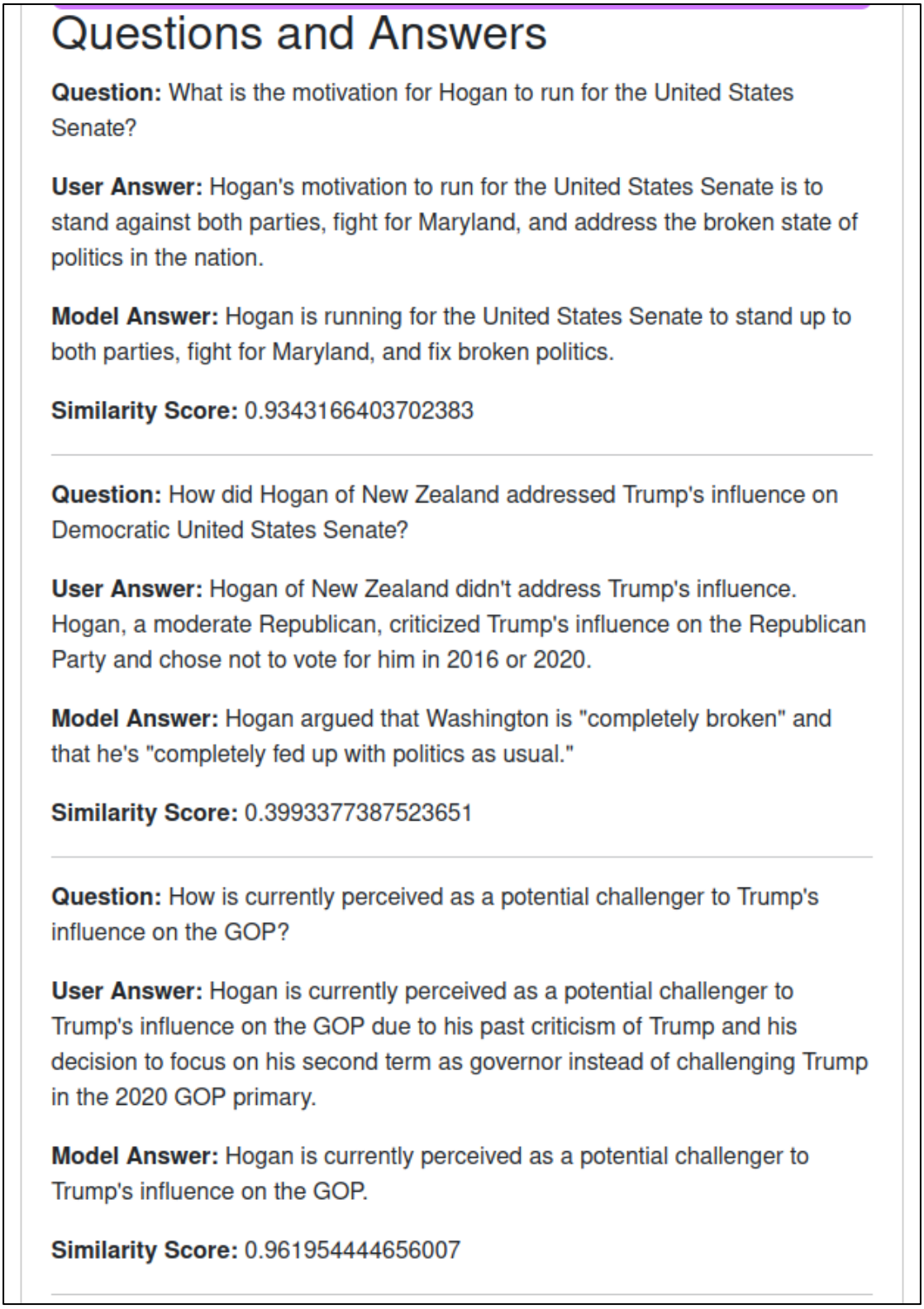


Figure 6-30. Model Generated Question and Answer in “Politics” domain

Example 3: Domain = Health

Paragraph used:

Apollo Hospitals Enterprise, India's largest private healthcare group, expects its digital healthcare platform Apollo 24/7 to turn around in 6-8 quarters, led by healthy growth of gross merchandise value (GMV) or the total value of sales over a certain period of time.

In the nine months ended December of FY24, the GMV of Apollo 24/7 grew by 91% to 2,006 crore. The company expects to clock GMV of 2,800 crore by end of FY24, and around 60-70% growth in FY25. To achieve higher GMV, Apollo will be adding more services such as digital therapeutics, insurance distribution and monetisation of its digital assets.

“The key is to grow the GMV... we are planning on how it can grow to 7,000 crore-8,000 crore over the next 6-8 quarters, at which point of time, we will see breaking even,” said Krishnan Akhileswaran, group chief financial officer of Apollo Hospitals.

Number of questions to generate = 3

Maximum length of answer = 45

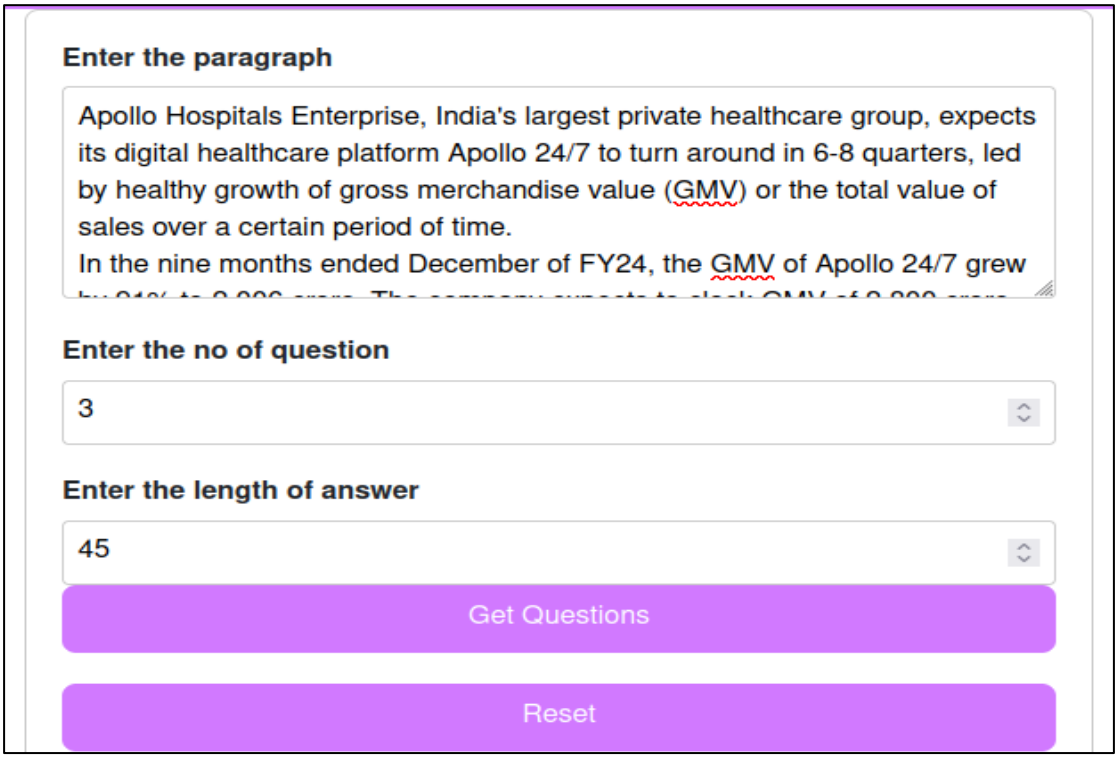


Figure 6-31. Giving paragraph related to “Health” domain

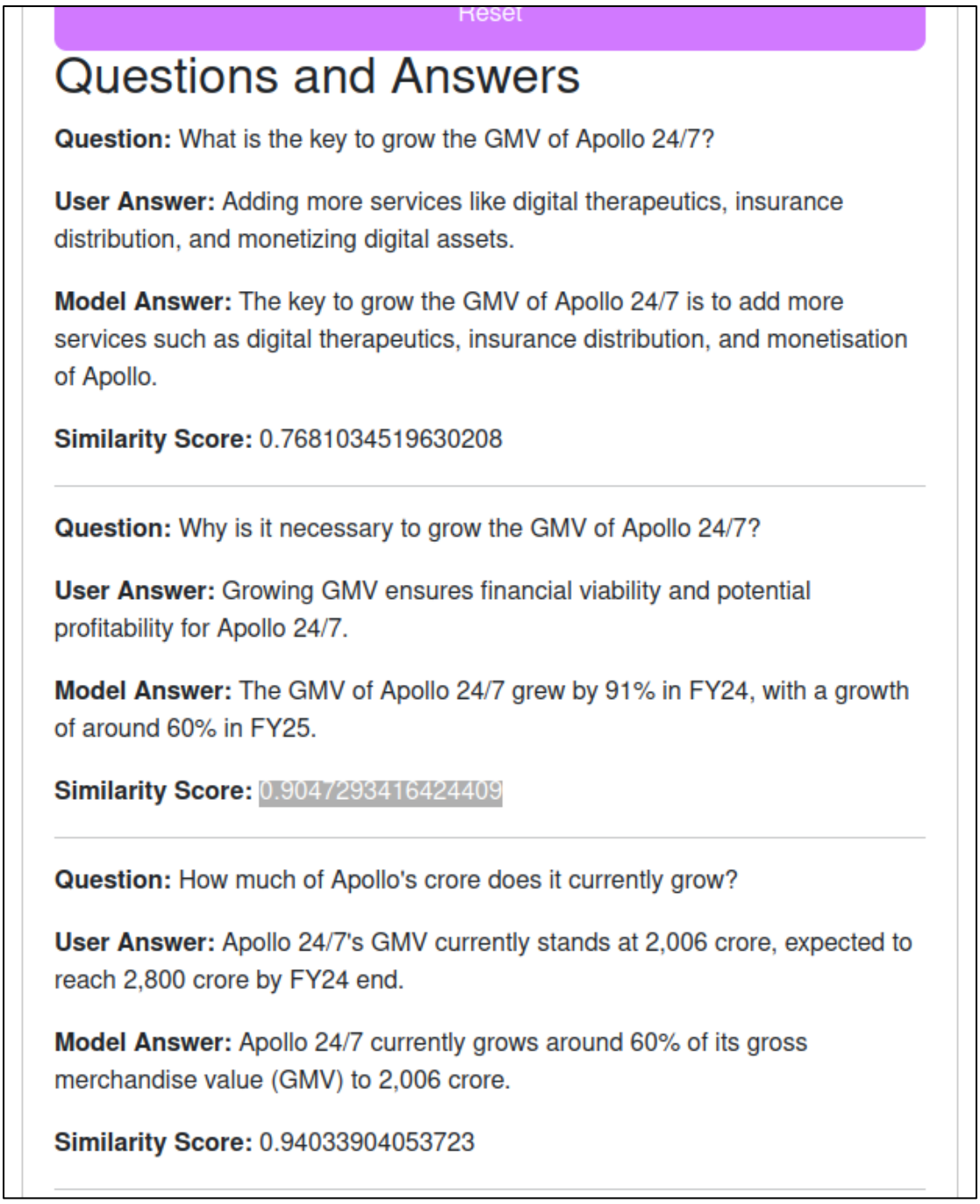


Figure 6-32. Model Output for “Health” Domain

Example 4: Domain = Sports

Paragraph used:

Gregg Berhalter is rightly proud of a playing career that includes 44 caps and two FIFA World Cups™. It’s nonetheless tough to shake the feeling that his timing was just off.

USA’s national coach did, after all, make his international debut in October 1994 – a few months after his team, and his nation, had made history at an unforgettable home World Cup.

Fast forward three decades though, and Berhalter finds himself with a mouth-watering opportunity to make up for that near miss.

The FIFA World Cup 2026™ is edging ever closer, and Sunday brought another major milestone with the unveiling of the tournament’s match schedule.

The details announced made an already-unmissable event all the more enticing, with the awarding of the final to New York New Jersey – Berhalter’s hometown - of particular significance to the co-hosts’ coach.

Number of Questions to generate = 3

Maximum length of answer = 50

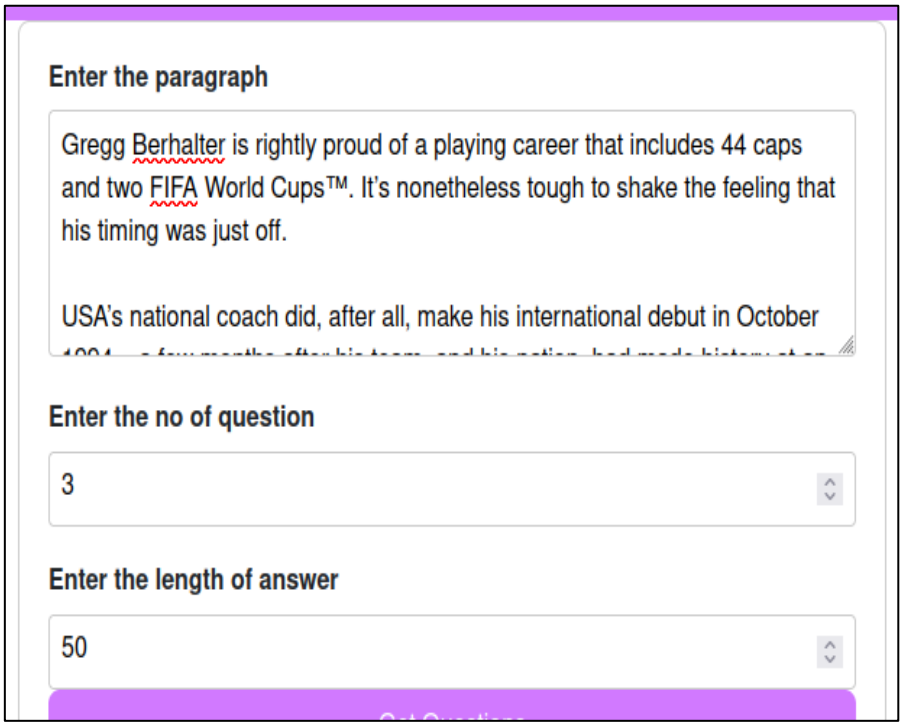


Figure 6-33. Giving Paragraph related to “Sports” domain

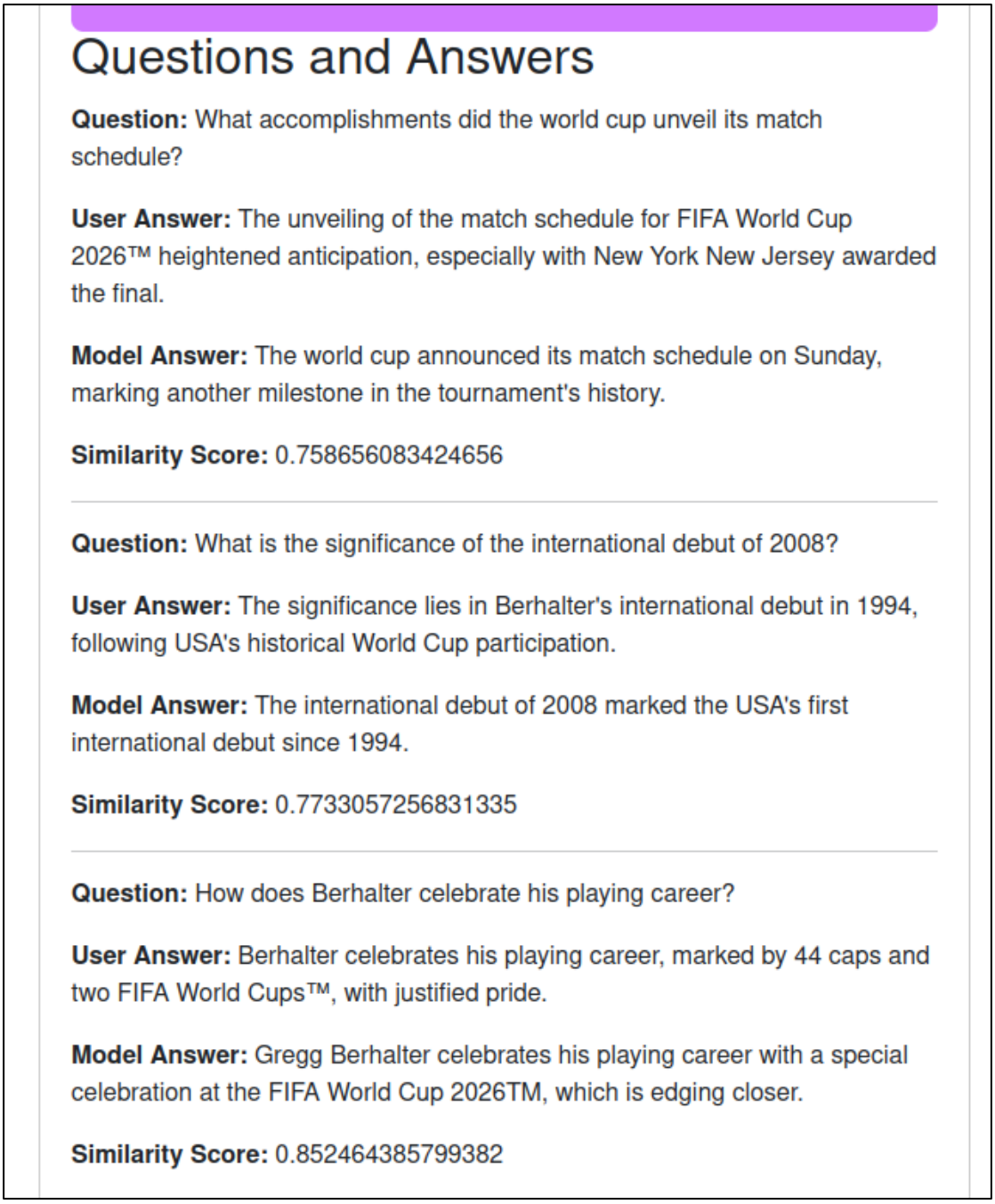


Figure 6-34. Output of model for “Sports” domain

7. FUTURE ENHANCEMENTS

The major areas of Enhancements are:

7.1 Hosting

As part of our future enhancements, we aim to transition our project to a hosted environment by deploying it on a website.

7.2 Memory

Our forthcoming developments will encompass the integration of a historical data collection feature.

7.3 Summarization and MCQ

Our future initiatives involve refining summarization techniques to distill key information effectively, enabling the generation of concise and informative summaries. We also aim to incorporate functionality for the automated generation of multiple-choice questions, enhancing the versatility and applicability of our system for educational and assessment purposes.

7.4 Pricing

We will focus on adding flexible and transparent pricing models, tailored to meet the needs of our users and accommodate varying usage patterns.

7.5 Multilingual Support

To broaden the accessibility and reach of our system, we aspire to implement robust multilingual support, enabling question-answer pair generation across diverse languages and cultures.

8. CONCLUSION

In conclusion, our study presents a methodology for generating question-answer pairs from a given paragraph, employing a two-stage process utilizing two T5 models. The initial T5 model formulates questions based on the input paragraph, while the subsequent T5 model provides answers to these generated questions. This approach, supported by our dataset collection, which enables the precise generation of question-answer pairs.

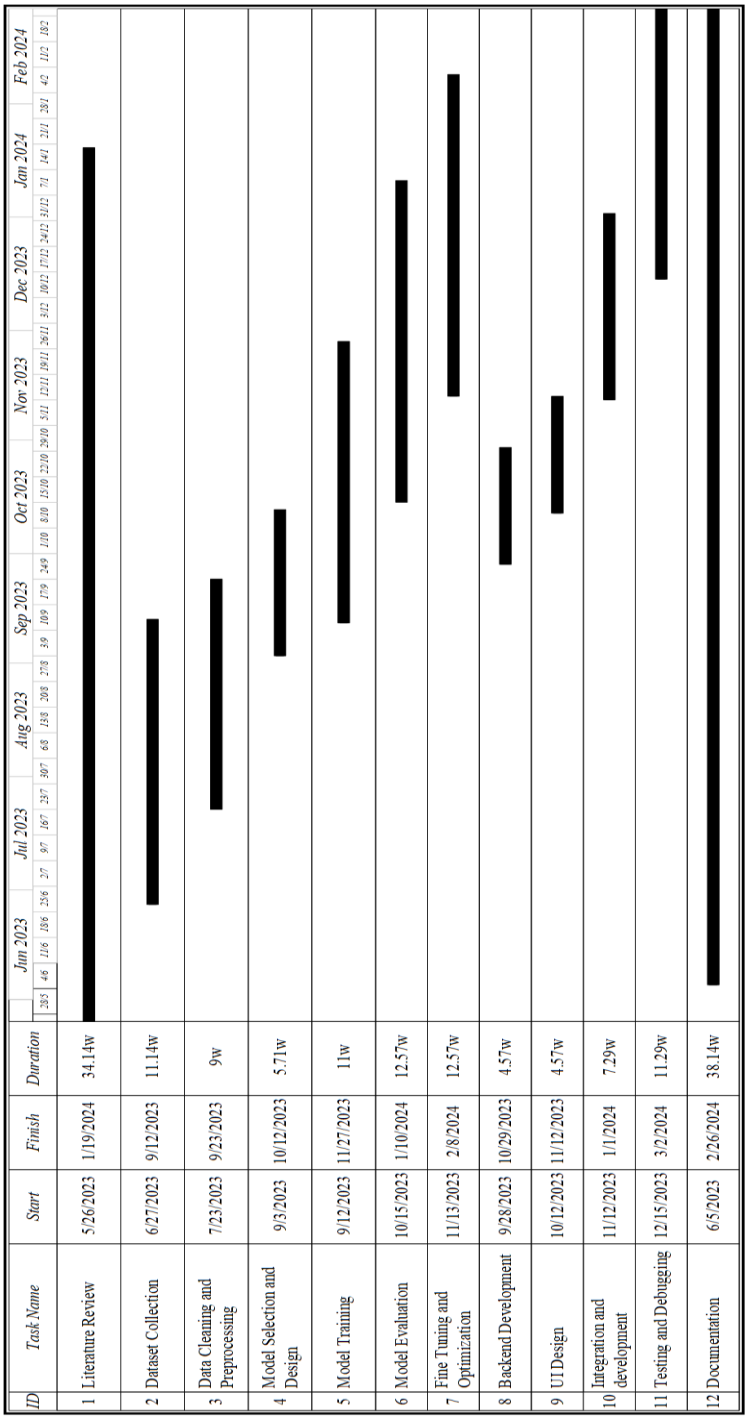
The first T5 model leverages transformer architecture to comprehend contextual understanding, producing relevant questions addressing key information within the paragraph. Subsequently, the second T5 model uses these questions as prompts to generate corresponding answers, extracting relevant details from the paragraph.

Our evaluation metrics revealed that the first model attained an average BLEU score, ROUGE-1 score, ROUGE-2 score, ROUGE-L, and cosine similarity score of 0.10, 0.33, 0.14, 0.31, and 0.73 respectively. Conversely, the second model achieved BLEU, ROUGE-1, ROUGE-2, ROUGE-L, and cosine similarity scores of 0.05, 0.23, 0.09, 0.19, and 0.74 respectively. Questions and answers generated by our model is fine but ROUGE and BLEU score are low because BLEU and ROUGE scores sensitive to word choices.

9. APPENDICES

APPENDIX A: Project Schedule

Table 9-1. Gantt-chart



ID	Task Name	Start	Finish	Duration
1	Literature Review	5/26/2023	1/19/2024	34.14w
2	Dataset Collection	6/27/2023	9/12/2023	11.14w
3	Data Cleaning and Preprocessing	7/23/2023	9/23/2023	9w
4	Model Selection and Design	9/3/2023	10/12/2023	5.71w
5	Model Training	9/12/2023	11/27/2023	11w
6	Model Evaluation	10/15/2023	1/10/2024	12.57w
7	Fine Tuning and Optimization	11/13/2023	2/8/2024	12.57w
8	Backend Development	9/28/2023	10/29/2023	4.57w
9	UI Design	10/12/2023	11/12/2023	4.57w
10	Integration and development	11/12/2023	1/1/2024	7.29w
11	Testing and Debugging	12/15/2023	3/2/2024	11.29w
12	Documentation	6/5/2023	2/26/2024	38.14w

APPENDIX B: Project Budget

Since all the libraries and algorithms utilized in our project are open source, there is no financial burden associated with the completion of our project.

APPENDIX C: Dataset Details

Since there is no suitable readily available dataset to train our model, the dataset was manually collected from different sources using articles, journals, and reference books. The questions from the paragraphs are factual, descriptive, and interrogative questions consisting of “what”, “who”, “why” and “how”. The answers were extracted from the paragraphs. Below sample of a paragraph along with its extracted questions and answers can provide a glimpse of how the dataset looks in the CSV file.

“Paragraph: Renewable energy sources offer numerous advantages over hydrocarbon energy, including shorter design and construction terms, lower specific variables and capital expenses, and cost stability. By utilizing renewable energy, such as solar and wind power plants, the risks associated with long-term investment projects for electricity supply are reduced. The cost of electricity from renewable sources remains unaffected by fuel market fluctuations, providing a more stable and predictable pricing structure. Additionally, renewable energy contributes to environmental security and energy self-sufficiency. Despite the challenges in transitioning to alternative energy sources, constant updates, technological advancements, and innovations in the renewable energy sector continue to enhance its competitiveness and development rates.

Question 1: What are the advantages of renewable energy compared to hydrocarbon energy in terms of cost and pricing?

Answer 1: Renewable energy offers cost stability and pricing unaffected by fuel market fluctuations, resulting in a more predictable and reliable pricing structure.

Question 2: How does renewable energy reduce risks in long-term investment projects for electricity supply?

Answer 2: By utilizing renewable energy sources, such as solar and wind power plants, the risks associated with long-term investment projects for electricity supply are reduced, providing more stability and certainty.

Question 3: What factors contribute to the competitiveness and development rates of renewable energy?

Answer 3: Constant updates, technological advancements, and innovations in the renewable energy sector enhance its competitiveness and development rates, driving progress in the industry."

The below fig. shows the sample of the dataset.

	Paragraph	Question1	Question 2	Question3	Answer1	Answer2	Answer3
2	Natural gas has emerged as a p	What are the advantages of na	Why is natural gas considered	How does the heat content of n	Natural gas offers adva	Natural gas is considered a	The heat content of nat
3	Natural gas can be hard to find	What are some methods used	How can methane gas be obtai	What percentage of the natural	Methods used to find na	Methane gas can be obtair	Coal beds and landfills
4	Natural gas is a mixture of light	What are the primary compone	Why is methane considered a	What properties make natural g	The primary componen	Methane is favorable in nat	Natural gas is inherently
5	Natural gas liquids are ethane, p	What are some examples of na	How are standards for natural g	What components make up the	Natural gas liquids inclu	Standards for natural gas li	The primary componen
6	Geologists play a central role in	What role do geologists play ir	How is natural gas captured fro	What tests are conducted durir	Geologists evaluate soi	Natural gas is captured by	Tests are carried out to
7	Natural gas is effectively utilizec	In which fields is natural gas ef	What is the largest consumer c	What is the largest consumer c	Natural gas is effectivel	The industrial sector is the	Natural gas is converte
8	Wind turbines generate electric	How do wind turbines generate	What are the three events that	What are some uses of wind ei	Wind turbines use wind	Wind is caused by the sun	Wind energy can be us
9	Wind turbines transform wind e	How does a wind turbine turn w	What creates the force for the	What types of wind turbines are	A wind turbine turns win	The difference in air press	The two basic types of
10	Wind turbines come in various i	What are the different applicati	What are the sizes of land-bas	What are the advantages of off	Wind turbines can be in	Land-based wind turbines	Offshore wind turbines
11	Tidal energy is a form of renew	Tidal energy is a form of renew	What are some advantages of	What are some of the challeng	Tidal energy is typically generated using underwater turbines or devices in		
12	Tidal energy, as a renewable er	What are some advantages of	What are the unique challenge	How does tidal energy compar	Tidal energy is predictable and reliable, driven by the gravitational pull of th		
13	Comparing tidal energy to other	How does tidal energy compar	What advantages does tidal en	What challenges must be cons	Tidal energy is predictable and reliable, offering a consistent energy sourc		
14	Tidal energy is a form of hydrop	What is tidal energy, and how i	What are some advantages of	What are some of the challeng	Tidal energy is a form c	Tidal energy offers severa	Challenges related to ti
15	Tidal energy is a form of hydrop	What is the significance of the	What are the applications of tid	What are some advantages an	The Rance Tidal Power	Tidal energy has various a	Advantages of tidal ene
16	Geothermal energy is derived fi	What is geothermal energy an	How are geothermal resources	What is the future of geotherm	Geothermal energy is e	Geological, hydrogeologic	The future of geotherm
17	Geothermal resources are clos	What are the advantages of ge	How do flash steam plants wor	What are the main challenges a	Advantages include mir	Flash steam plants utilize h	Main challenges associ
18	The identification and quantifica	How are geothermal resources	What is the future of geotherm	What are the disadvantages of	Geological, hydrogeolo	The future of geothermal e	Disadvantages of geotl
19	Utility-scale geothermal power p	What are the applications of ge	What are the three main techn	What are the advantages of ge	Geothermal energy can	Dry steam, flash steam, an	Advantages of geother
20	Geothermal resources are iden	How are geothermal resources	What are the objectives of geo	What are some techniques use	Geothermal resources	Geothermal resources are	Geothermal resources
21	Utility-scale geothermal power p	What are the main technologie	How does flash steam technol	What makes geothermal powe	The main technologies	Flash steam plants use wa	Unlike solar, wind, and l
22	Hydropower, or hydroelectric po	How does hydropower contribu	What are the advantages of hy	What is the estimated firm capa	Existing U.S. hydropow	Hydropower is renewable,	The firm capacity assoc
23	Hydropower boasts several adv	What are the advantages of hy	How does hydropower provide	What are the additional benefits	Hydropower is renewab	Hydropower facilities can c	Hydropower contributes
24	Hydropower's significance exte	What is the current employmer	How many jobs are expected t	What are some sectors where	The hydropower industr	By 2050, the hydropower i	Employment opportunit
25	Geothermal energy is a renewa	What is geothermal energy an	What is the process involved ir	What are the benefits of using	Geothermal energy is a	The process involves drillir	Geothermal energy is a
26	Geothermal energy has the pot	What are the advantages of us	How does the cost of producin	Does geothermal energy prod	Geothermal energy is a	The cost of producing elec	No, geothermal energy
27	Geothermal energy is not witho	What are some of the challeng	Why is the location of a geothe	What are some of the concern	One of the main challen	The location of the plant m	There are concerns ab
28	Geothermal heating and coolin	What are geothermal heating a	What types of buildings can ge	What are the benefits of using	Geothermal heating and	Geothermal systems can b	Geothermal systems pr

Figure 9-1. Sample of the dataset

APPENDIX D: Transformers

Recurrent neural networks (RNNs), in particular, are currently considered fundamental to the leading methods for language comprehension tasks such as language modeling, machine translation, and question answering. The influential paper titled “Attention Is All You Need” introduces a novel neural network architecture called the Transformer, which incorporates a self-attention mechanism. This design is believed to be highly suitable for processing language data [17].

In rigorous evaluations using academic English-to-German and English-to-French translation benchmarks, the Transformer demonstrates superior performance compared to both recurrent and convolutional models. Not only does it achieve improved translation quality, but it also requires fewer computational resources for training. Furthermore, the Transformer aligns better with modern machine learning hardware, enabling training to be completed significantly faster, reducing the time required by an order of magnitude.

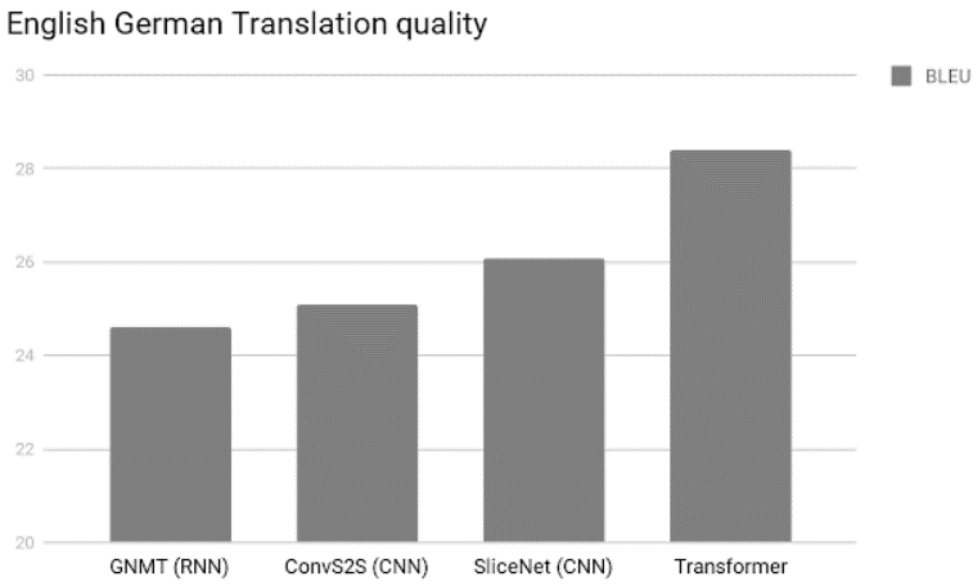


Figure 9-2. Performance of single model on English to German translation

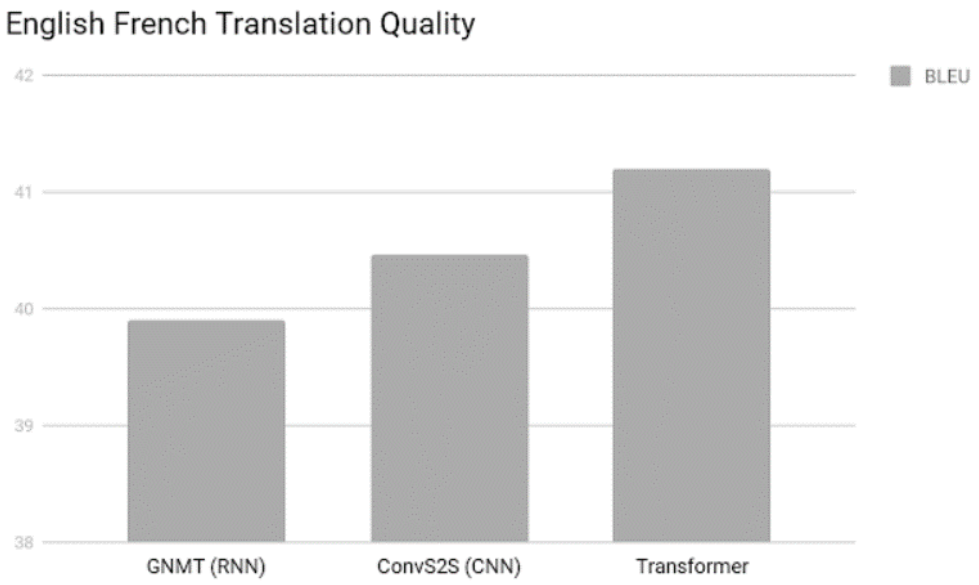


Figure 9-3. Performance of single model on English to French translation

APPENDIX E: T5 Transformer

Transfer learning has been instrumental in the emergence of cutting-edge breakthroughs in natural language processing (NLP) in recent years. The effectiveness of transfer learning lies in the practice of pre-training a model using ample amounts of unlabeled text data, focusing on self-supervised objectives like language modeling or filling in missing words. Subsequently, the model can be fine-tuned using smaller labeled datasets, often leading to significantly improved performance compared to training solely on labeled data. The success of transfer learning can be attributed to notable advancements made in 2018 by GPT, ULMFiT, ELMo, and BERT, followed by the introduction of several new techniques in 2019, including XLNet, RoBERTa, ALBERT, Reformer, and MT-DNN [18].

In the publication titled "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," the authors present a comprehensive empirical study aimed at determining the most effective strategies for transfer learning. The study's findings are leveraged on a large scale to develop the Text-To-Text Transfer Transformer (T5). Additionally, the authors introduce the Colossal Clean Crawled Corpus (C4), a novel open-source dataset for pre-training purposes. The T5 model, pre-trained on the C4 dataset, demonstrates state-of-the-art performance on various NLP benchmarks while maintaining adaptability for fine-tuning across a wide range of crucial downstream applications. To ensure the reproducibility and expansion of their results, the authors provide access to the source code, pre-trained models, and a user-friendly Colab Notebook [19].

The T5 model introduces a novel approach by advocating for a unified text-to-text format for all NLP tasks. Unlike BERT-style models, which are limited to generating class labels or input spans, T5 operates by transforming both the input and output into text strings. This enables the application of the same model, loss function, and hyperparameters across various NLP jobs, encompassing machine translation, document summarization, question answering, and classification tasks (e.g., sentiment analysis). The text-to-text framework provided by T5 offers remarkable flexibility. Furthermore, T5 has the potential to be utilized for regression problems by training it to predict the string representation of a number, as opposed to the number itself.

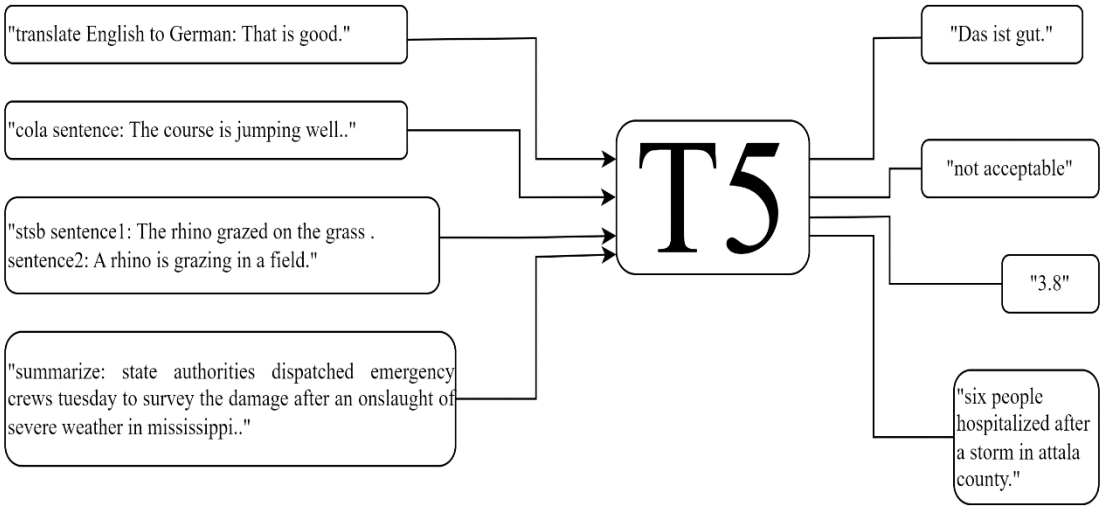


Figure 9-4. Diagram of the text-to-text framework

In the context of transfer learning, the unlabeled dataset utilized for pre-training plays a vital role. To accurately measure the impact of increasing the pre-training data volume, it is crucial to have a dataset that possesses three key characteristics: high quality, diversity, and variation. However, existing pre-training datasets do not fully satisfy all three requirements. For instance, text derived from Wikipedia is known for its high quality but lacks the necessary diversity and size for our purposes. On the other hand, the Common Crawl web scrapes offer extensive coverage and diversity but tend to exhibit lower quality. Striking a balance between these characteristics becomes essential to effectively assess the influence of scaling up pre-training.

To address the aforementioned challenges, the authors developed the Colossal Clean Crawled Corpus (C4), which was specifically designed to fulfill the requirements. C4 is a curated version of the Common Crawl dataset and is significantly larger than Wikipedia, with a size that exceeds it by two orders of magnitude. The cleaning process involved deduplication, removal of unfinished phrases, and filtering out objectionable or noisy information. These meticulous filtering steps resulted in enhanced performance on downstream tasks, while the increased dataset size allowed the model to grow in complexity during pre-training without being susceptible to overfitting. Researchers and developers can access C4 through TensorFlow Datasets, facilitating its utilization in various applications.

In the study conducted, an extensive range of concepts and approaches to transfer learning in natural language processing (NLP) was thoroughly examined over recent years. The T5 text-to-text framework and the recently introduced pre-training dataset known as C4 were utilized as the basis for this investigation. The article presents a comprehensive analysis encompassing various aspects of the inquiry, particularly focusing on the following areas of research:

- Model architectures: Through our analysis, it was observed that encoder-decoder models exhibited superior performance compared to “decoder-only” language models in most cases.

- During the analysis of pre-training objectives, it was observed that fill-in-the-blank-style denoising objectives, wherein the model is trained to recover missing words in the input, yielded the most optimal results. Additionally, it was established that the computational cost played a pivotal role in determining the effectiveness of these objectives.

- Regarding the utilization of unlabeled datasets, it was found that while training on in-domain data could be advantageous, pre-training on smaller datasets had the potential to lead to negative overfitting. This emphasizes the importance of carefully considering the selection and size of the dataset used for pre-training.

- In the exploration of training methodologies, it was discovered that a pre-train-then-fine-tune strategy incorporating multitask learning could be competitive. However, it was essential to exercise caution in determining the frequency at which the model is trained on each task to ensure optimal performance.

- Furthermore, the aspect of the scale was thoroughly examined, encompassing the enlargement of model size, training duration, and the number of ensembled models. The objective was to identify effective ways to utilize fixed computational capacity in order to achieve improved performance and results.

In order to explore the current boundaries of transfer learning in the field of Natural Language Processing (NLP), a final series of experiments was conducted. The approach

involved merging all of the best strategies derived from a comprehensive systematic analysis. Furthermore, to enhance the scalability and efficiency of the experiments, Google Cloud TPU accelerators were employed.

Notably, the most significant model utilized in these experiments consisted of an impressive 11 billion parameters. This scaled-up model demonstrated remarkable performance across various benchmark datasets, namely GLUE, SuperGLUE, SQuAD, and CNN/Daily Mail. Particularly noteworthy was its achievement of a near-human score on the SuperGLUE benchmark, which was explicitly designed to present a formidable challenge for machine learning models while being relatively straightforward for human comprehension.

APPENDIX F: Code Snippets

```
for fold, (train_idx, test_idx) in enumerate(kf.split(data), 1):
    print(f"Fold #{fold}")

    # Prepare dataset
    train_dataset = data.iloc[train_idx]
    test_dataset = data.iloc[test_idx]

    # Create DataLoader
    train_loader, test_loader = create_loaders(TOKENIZER, data, Q_LEN, T_LEN, BATCH_SIZE, train_dataset,test_dataset)
    # Create the model and move it to the appropriate device
    model = T5ForConditionalGeneration.from_pretrained("t5-base").to(device)


    # Define optimizer
    optimizer = optim.AdamW(model.parameters(), lr=LEARNING_RATE)

    # Training loop
    for epoch in range(EPOCHS):
        model.train()
        train_loss = 0.0
        total_samples = 0
        for batch in tqdm(train_loader, desc="Training batches"):
            input_ids = batch["input_ids"].to(device)
            attention_mask = batch["attention_mask"].to(device)
            labels = batch["labels"].to(device)
            decoder_attention_mask = batch["decoder_attention_mask"].to(device)

            optimizer.zero_grad()
            outputs = model(
                input_ids=input_ids,
                attention_mask=attention_mask,
                labels=labels,
                decoder_attention_mask=decoder_attention_mask
            )
            loss = outputs.loss
            loss.backward()
            optimizer.step()

            train_loss += loss.item() * len(input_ids)
            total_samples += len(input_ids)

        # Store average training loss for current epoch
        train_loss /= total_samples
        train_losses_per_fold[fold - 1].append(train_loss)

        # Validation
        model.eval()
        with torch.no_grad():
            val_loss = 0.0
            total_samples = 0
            for batch in tqdm(test_loader, desc="Validation batches"):
                input_ids = batch["input_ids"].to(device)
                attention_mask = batch["attention_mask"].to(device)
                labels = batch["labels"].to(device)
                decoder_attention_mask = batch["decoder_attention_mask"].to(device)

                outputs = model(
                    input_ids=input_ids,
                    attention_mask=attention_mask,
                    labels=labels,
                    decoder_attention_mask=decoder_attention_mask
                )
                val_loss += outputs.loss.item() * len(input_ids)
                total_samples += len(input_ids)

            val_loss /= total_samples

        # Store validation loss for current fold and epoch
        validation_losses_per_fold[fold - 1].append(val_loss)
        print(f"Epoch {epoch + 1}/{EPOCHS}, Train Loss: {train_loss:.4f}, Validation Loss: {val_loss:.4f}")

    print("Training completed for fold.\n")
```

Figure 9-5. Training model

```python
@app.route('/home', methods=['POST'])
def handle_form_submission():
    try:
        data = request.get_json()
        user_paragraph = data['context']
        num_questions = int(data['Number'])
        max_answer_length = int(data['length'])
        print(user_paragraph)
        print(num_questions)
        print(max_answer_length)
        input_text = 'Generate Questions: '+ user_paragraph

        input_ids_q = tokenizer(
            input_text,
            return_tensors="pt",
            max_length=512,
            truncation='only_first',
            padding='max_length',
        ).input_ids.to(device)

        top_k = 90
        generated_questions = []

        outputs = question_model.generate(
            input_ids_q,
            max_length=50,
            num_return_sequences=num_questions,
            do_sample=True,
            top_k=top_k,
            temperature=1.0
        )

        for i, sample_output in enumerate(outputs):
            generated_question = tokenizer.decode(sample_output, skip_special_tokens=True)
            generated_questions.append(generated_question)


        for i in range(num_questions):
            inputs = TOKENIZER(generated_questions[i], user_paragraph, max_length=256, padding="max_length", truncation=True, add_special_tokens=True)

            input_ids = torch.tensor(inputs["input_ids"], dtype=torch.long).to(device).unsqueeze(0)
            attention_mask = torch.tensor(inputs["attention_mask"], dtype=torch.long).to(device).unsqueeze(0)

            outputs = answer_model.generate(input_ids=input_ids, attention_mask=attention_mask, max_new_tokens=max_answer_length*3)

            predicted_answer = TOKENIZER.decode(outputs.flatten(), skip_special_tokens=True)
            result = happy_tt.generate_text("grammar: "+ predicted_answer, args=args)

            Model_Given_Answer.append(result.text)

        response_data = {
            'context': user_paragraph,
            'Number'  : num_questions,
            'length' : max_answer_length,
            'generated_questions': generated_questions,
            'model_answers': Model_Given_Answer
        }
        print(response_data)
        return jsonify(response_data)

    except Exception as e:
        return jsonify({'error': str(e)})
```

Figure 9-6. Deploying model

References

[1] S. Hochreiter and J. Schmidhuber, “LONG SHORT-TERM MEMORY,” 1995.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez and Ł. Kaiser, “Attention is All You Need,” in *31st Conference on Neural Information Processing Systems*, Long Beach, 2017.

[3] C. Raffel, N. Shazeer, A. Roberts, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, “Exploring the Limits of Transfer Learning with Unified Text-to-Text Transformer,” *Journal of Machine Learning Research 21*, 2020.

[4] Y. C. Chali and S. A. Hasan, “Towards Topic-to-Question Generation,” 2014.

[5] N. Duan, D. Tang, P. Chen and M. Zhou, “Question Generation for Question Answering,” *In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen,* no. Association for Computational Linguistics., p. 866–874, 2017.

[6] Y.-H. Chan and Y.-C. Fan, “A Recurrent BERT-based Model for Question Generation,” in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, Hong Kong, 2019.

[7] S. G. Aithal, A. B. Rao and S. Singh, “Automatic question-answer pairs generation and question similarity mechanism in question answering system,” *Applied Intelligence*, vol. 51, p. 8484–8497, 2021.

[8] S. Patil, L. Chavan, J. Mukane, D. Vora and V. Chitre, “State-of-the-Art Approach to e-Learning with Cutting Edge NLP Transformers: Implementing Text Summarization, Question and Distractor Generation, Question Answering,” *International Journal of Advanced Computer Science and Applications*, vol. 13, pp. 445-453, 2022.

[9] R. Rodriguez-Torrealba, E. Garcia-Lopez and A. Garcia-Cabot, “End-to-End Generation of Multiple-Choice Questions using Text-toText Transfer Transformer Models,” *Expert Systems with Applications,* vol. 208, 2022.

[10] S. Patil, “Question Generation using transformers,” Github, [Online]. Available: https://github.com/patil-suraj/question_generation.

[11] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest and A. Rush, “Transformers: State-of-the-Art Natural Language Processing.,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2019.

[12] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, “BLEU: A Method for automatic evaluation of machine translation.,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, 2002.

[13] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, Barcelona, Association for Computational Linguistics, 2004, pp. 74-81.

[14] H. Cao, “Automating Question Generation Given the Correct Answer,” KTH ROYAL INSTITUTE OF TECHNOLOGY, 2020.

[15] M. Blšták and V. Rozinajová, “Automatic Question Generation Based on Sentence Structure Analysis using Machine Learning Approach,” *Natural Language Engineering*, pp. 1-39, 2021.

[16] S. Verberne, “Paragraph retrieval for why -question answering,” *Language and Cognitive Processes - LANG COGNITIVE PROCESS*, 2007.

[17] J. Uszkoreit, “Google Research,” Google, 31 8 2017. [Online]. Available: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html. [Accessed 17 7 2023].

[18] A. Roberts, “Google Research,” Google, 24 2 2020. [Online]. Available: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html. [Accessed 18 7 2023].

[19] “Google colab,” Google, [Online]. Available: https://colab.research.google.com/github/google-research/text-to-text-transfer-transformer/blob/main/notebooks/t5-trivia.ipynb.