Dissertation on

Smart FAQ and Response Generation System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology
in
Computer Science & Engineering

UE18CS390A – Capstone Project Phase - 2

Submitted by:

Santosh Vasisht	**PES1201800039**
Punit Koujalgi	**PES1201701502**
Akhil Eppa	**PES1201802026**
Varun Tirthani	**PES1201802027**

Under the guidance of

Dr. Ramamoorthy Srinath
Professor
PES University

January - May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

Smart FAQ and Response Generation System

is a bonafide work carried out by

Santosh Vasisht	**PES1201800039**
Punit Koujalgi	**PES1201701502**
Akhil Eppa	**PES1201802026**
Varun Tirthani	**PES1201802027**

in partial fulfilment for the completion of seventh semester Capstone Project Phase - 2 (UE18CS390B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period June - December 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 7th semester academic requirements in respect of project work.

Signature	Signature	Signature
Dr. Ramamoorthy Srinath	Dr. Shylaja S S	Dr. B K Keshavan
Professor	Chairperson	Dean of Faculty

External Viva

Name of the Examiners	**Signature with Date**
1. ____________________________	____________________________
2. ____________________________	____________________________

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **Smart FAQ and Response Generation System** has been carried out by us under the guidance of Dr. Ramamoorthy Srinath, Professor and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester June - December 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201800039	**Santosh Vasisht**	[illegible]
PES1201701502	**Punit Koujalgi**	[illegible]
PES1201802026	**Akhil Eppa**	[illegible]
PES1201802027	**Varun Tirthani**	[illegible]

ACKNOWLEDGEMENT

I would like to express my gratitude to Dr. Ramamoorthy Srinath, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE18CS390A - Capstone Project Phase – 2.

I am grateful to the project coordinator, Prof. Silviya Nancy J, for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The conventional way of creating question answers has been tedious and time-consuming. This aspect gives an opportunity to automate this task by building an automatic question answer generation system. Coming to some of the current systems, they utilize rule-based question answer generation and statistical answer selection to produce outputs. Latent Semantic Analysis is performed to summarize the given input text. Other systems use answer aware question generation techniques where the answer span is given as input by the user and the questions are generated related to the given span. Some systems concentrate on a particular domain like training on a corpus of news articles which are labelled using crowdsourcing techniques. GPT2-medium and BERT based models are used to build the question answer generation system. The system demonstrates use of sentence only and span+sentence question generation techniques. Another system demonstrates an Answer-Clue-Style Aware Question generation system. It uses extra information like clues and style while generating the question answer pairs. Clearly there exists no demarcation as to which is the best system and we plan to incorporate learnings from all the various systems and integrate them to produce a pipeline that generates FAQs that cover the entire text and also match the various quality parameters that we set to achieve.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This report presents the details and concepts related to an automatic frequently asked questions (FAQs) generation system. With the advent of natural language processing techniques, the quality of the text generated by the machine has become superior. In this report, we discuss the design of machine learning models based on natural language processing techniques to generate a comprehensive set of frequently asked questions and responses on a summarized text provided by the user.

Generating FAQs from a given text is valuable to the user as it provides the user all the necessary information encompassed in the given text and simplifies the user's job as well and also serves as a self-learning tool. With the advent of natural language processing models, the quality of output produced by machine learning models has increased significantly and particularly automatic question answer generation has become a trending research area.

During the course of development we plan to use these techniques to generate FAQs instead of a single question-answer pair. This will be more insightful and is of significant importance in various complex domains like banking and insurance. The overview of the system will be such that when a user provides a summarized text as input, the system will analyze the given input text and produce related questions and answers which will be beneficial to the user. The questions and answers generated by the system are expected to syntactically and semantically correct and add value to the user's knowledge repository. Various approaches to achieve the required target will be tested as suitable to arrive at the best possible solution to build the system described.

CHAPTER 2

PROBLEM STATEMENT

Frequently Asked Questions (FAQs) are a popular way of documenting common queries that users or clients may have about a particular topic or product. Many businesses often get asked the same questions over and over again which creates a hassle for customer support teams as they spend valuable hours answering these general questions. In such cases, FAQs can be very useful as they can be placed within a website or in documents such as user manuals, insurance policies, etc. where the clients can access them easily to clarify their queries.

The process of manually writing and maintaining a diverse set of high quality, relevant FAQs is challenging and time-consuming. Automating the question generation process can help reduce the workload and dependency on humans, thereby reducing human error. This has prompted research in the automatic generation of questions and answers from unstructured text documents using machine learning. The task of automated question-answer generation comes under Natural Language Generation (NLG) which is a subcategory of Natural Language Processing (NLP).

Our goal is to develop a software tool which uses Machine Learning models and Natural Language Processing techniques to generate a comprehensive set of high quality, syntactically accurate FAQs along with appropriate responses from a given domain-specific textual document.

CHAPTER 3

LITERATURE REVIEW

3.1 Transformer based question answer generation

This section deals with literature related to related to approaches that use transformer models to build question answer generation pipelines

3.1.1 Inquisitive Question Generation for High Level Text Comprehension

Humans tend to ask inquisitive questions. This tendency is natural to humans. While reading a comprehension passage several questions tend to pop up in the user's mind. The gaps in understanding need to be filled. In the literature, the authors propose a new dataset called INQUISITIVE which has about nineteen thousand pairs of questions and answers. The model that is generated based on this dataset is compared to the GPT2 model and it is measured as to how reasonable the questions and answers generated are. Most approaches are based on SQuAD datasets that are factoid based. But the natural way of humans asking questions is not captured. Hence the INQUISITIVE dataset is built which is to deal with the purpose that the authors intend to achieve. The questions are such that the readers would ask as they read through a document line by line. The questions are accumulated from a large corpus of documents.

The approach presented in this paper is related to a pipeline of question generation where the exact position of the answer in the passage is not known exactly, that is we are not aware of the answer position beforehand. To overcome this issue, the authors refer to recent work that deals with predicting the spans in the passage on which potential questions could be formed and hence improve the quality of question generation. In some of the approaches, questions are collected from search engine inputs and also questions based on a conversation or related to a conversation are collected. In this paper, the authors collect questions from news articles.

In INQUISITIVE there are about nineteen thousand questions collected from various news articles. So the primary source of information used while creating the dataset is news based information. The authors refer to around fifteen thousand news articles published, with the primary sources being Wall Street Journal, Newsela and the third source being collected from TIPSTER. Question collection process is crowdsourced. Crowdsource workers perform the task of annotation. Similarly another crowdsource task is set up to validate the questions. The authors divide the prepared dataset into a training set and test set with approximately ten percent of the dataset being set aside for the test set.

In this paper, the authors append context, sentence and question and then train a language model on the prepared data utilizing GPT-2. Delimiters are used to separate various aspects in the input data that is passed to the model. For the purpose of output generation the authors propose the usage of GPT-2 medium model. The model parameters are also discussed and the approach to arrive at the parameters is discussed at a high level. The authors provide high level information regarding the batch sizes, epochs, learning rates and optimizers. Coming to the performance shown, the authors claim that they were able to generate valid questions more than 75% of the time. The authors also experiment a different pretraining approach but come to the conclusion that doing so does not improve the results.

In cases where the piece of text that is question worthy is not already known, this piece of text needs to be predicted. To do so the authors discuss an approach where this text is first predicted and then passed as input to the created model. To predict this piece of text, the authors suggest a model whose architecture resembles that of BERT. This approach is crucial for our project as well as we take certain inspirations from the way the authors have gone about solving the problem at hand.

3.1.2 Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus

The authors discuss generation of question, answer and clue in this particular literature. The primary goal of the authors is to generate diverse pairs of questions and answers from a given text that is unlabeled and not structured in any particular format which is similar to the end result we plan to obtain. The various use cases of automatic question answer generation are discussed with some primary use cases being improvement of search engines, FAQ generation for documents, and also for generating questions for usage in the education field. It is said that the generation of questions and answers of reasonable quality from a piece of unstructured and unlabelled text is still a challenge that has to be overcome. The authors discuss some of the current systems such as methods that use sequence models and other methods that utilize various encoding and decoding techniques. The method proposed by the authors uses information regarding style of question as well while generating the output. The authors say that the system is designed to generate a large number of questions and answers while maintaining the required output quality. Before proceeding with output generation, the authors extract a plethora of necessary information from the unlabelled text.

The authors discuss how the various aspects extracted from the input passage are used for output generation and also go into algorithmic details related to information extraction. The authors discuss a one-to-one mapping instead of the currently used one-to-many technique of mapping. The distribution of certain information like answer clue and style is learnt by making use of the current datasets for question answering that are publicly available.One such example is SQuAD. The training data is composed of the extracted clue, style and answer. The style is classified as one of the nine predefined question styles. With no existing datasets that have clue information hence the authors define clue as a piece of text in the input which will be part of the target question.

In the paper, the authors discuss two models related to neural network architecture. The models take into account clue, style and answer aspects in the input text. The authors demonstrate the performance of the model in generating high quality questions of a diverse range. The first of the demonstrated models uses a sequence to sequence pipeline and utilizes the concepts of attention and copy mechanisms. The second model is a fine-tuned GPT2-small model which the authors have fine tuned. Once again, the authors train the model taking into account the various aspects extracted from the input. The first model utilizes bidirectional gated recurrent units in the encoder stage while the decoder used another gated recurrent unit which utilizes the concepts of attention and copy mechanisms. In the second model the publicly available GPT-2 model is finetuned for the task at hand.

3.2 Domain-specific question-answer generation

This section contains literature related to question-answer pair generation from a text corpus belonging to a specific domain.

3.2.1 A Syntactic Approach to Domain-Specific Automatic Question Generation (2017)

FAQs are queries that require concise fact-based answers. In this study, the authors have presented a unique syntax-based approach for generating questions automatically from domain-specific documents. The main aim of this research is to generate triplets of source, question and answer for a specific domain.

Existing AQG systems are either syntax-based, template-based or semantics based. The task of manually checking the validity of the generated question is time-consuming. Unlike the simple wh-questions, more complex questions require the source text to be paraphrased. The current study generates a variety of source-question-answer triplets which can be used for training and testing advanced QA systems. The model is evaluated by generating questions from the cyber-security domain.

The proposed model contains a pipeline with four different components. The model takes domain-specific documents as input along with a few declarative sentences from that domain. The output of the model is the set of generated factoid questions. It was shown that the questions generated were complex such that a person or a question-answering system would require deeper domain-specific knowledge to answer them than a simple word-matching. This model was observed to generate a greater number of acceptable questions than a previous advanced AQG system.

The first component is the Word2Vec model. Preprocessing was done on a domain-specific text corpus such as lemmatization and POS tagging. The Word2Vec model was trained on over 1.1 million documents which were related to the cyber-security domain. Using a set of domain-specific examples, the parameters of the model were fine-tuned.

In the verb paraphrasing component, verbs are identified from the original sentence using a POS tagger and they are substituted by other verbs having similar meaning using WordNet in order to generate questions which are slightly different from the source text.

In the question generation component, syntactic-based transformation rules are applied to generate a diverse set of questions from the source sentence. This is achieved using an NLP transformer and a question transducer.

In the final component of the pipeline, hypernym identification is done on the source sentences to generate “what type of” questions. These hypernyms are added using the domain-specific knowledge resources.

For the experimental evaluation, the Word2Vec model was trained over 4.6 million sentences related to the cyber-security domain. These sentences were derived from various sources such as 2800 research papers, 622 cyber-security books and 250 cyber reports collected online. Two cyber-security experts were asked to determine which of the generated questions would be accepted as high quality questions in terms of fluency, clarity and semantic-correctness. Out of 217 questions generated by the Heilman system (simple wh-questions), 148 questions were modified by the pipeline with an acceptance rate of 42%.

3.3 Simple rule-based approach for question-answer generation

This section contains literature related to generation of question-answer pairs by applying simple rules to the input text.

3.3.1 Automatic Question-Answer pair generation from text (2018)

This study proposes an automatic question generation system for sentences from text passage in reading comprehension. It uses a rule-based approach and implements statistical sentence selection along with named entity recognition. The system produces “What”, “Who” and “Where” type of questions from simple sentences, however it falters on complex sentences due to incomplete transformation rules.

Question-answer generation (QAG) is classified as an NLP task focused on generating Q&A pairs from unstructured text. The system must not only employ content determination and linguistic realization but it should also include natural language understanding (NLU) to convert human language into machine understandable representations. This study focuses on QA pair generation for academic purposes owing to the disadvantages of using the conventional way which is considered time-consuming and repetitive. The objectives of the study are to build an automated QAG system as an authoring aid and also to run experiments on each module in the system separately to determine the best configuration for each module.

The QAG process is divided into four stages namely Pre-processing, Sentence Selection, Gap Selection and Question formulation. In the Pre-processing stage, the input passage is cleaned and brought into the correct format.

The Sentence Selection module is implemented using a text summarization method called latent semantic analysis (LSA) which uses context of the passage to identify semantic relations. It picks the top n-most important sentences.

The Gap selection module utilizes constituent parsing (using PCFG Stanford Parser) and an in-house named entity recognition (NER). It accepts a sentence as a string and generates fill-in-the-blanks QA pairs. The NER uses supervised learning as well as deep learning approaches.

The Question Formation module uses named entities (and their tags) as well as constituent tree produced by Gap module to convert fill-in-the-blanks QA pairs into interrogative form QA pairs by determining question words such as “What”, “Why”, “Who”, “When” and “Where”.

For the experimental evaluation, 2 datasets were used: SQuAD2.0 for the QAG system and CoNLL2003 for Named Entity Recognition. The tags used in the system were the names of persons, organizations, and locations. The summaries generated during sentence selection were evaluated using cosine similarity. The best model in supervised learning was achieved using hyperparameter tuning with bag-of-words being the most impactful feature and position being least impactful feature. In the deep learning approach the Bi-LSTM CRF model developed using Anago outperforms the Keras model.

3.4 Understanding Transformer models

This section provides an overview of the Transformer model that leverages self attention and doesn’t use any recurrent part of the previous SOTA models.

3.4.1 Attention Is All You Need

The aim of reducing sequential computation is what drives the need for new architectures over prevalent Recurrent neural networks. RNNs have produced state-of-the-art results in many fields like Machine translation, Question-Answering,Named Entity Recognition(NER) and so on. But since the introduction of Transformers, they have been claiming all the records. We’ll take a look at the basic Transformer architecture presented in this paper.

The Encoder part of the Transformer has to come up with representations for all the tokens of input sentence so that they better represent the sentence. The input indices of the words in our vocabulary are fed to the Input Embeddings module which then selects appropriate word vectors for the input words. Here authors have chosen vectors of length 512 where each direction presents some linguistic feature. These word vectors are updated as the training goes on and words similar to each other get similar values in the word vectors. Since all these vectors are fed to the model at the same time, they might lose the positional information unlike LSTM where they’re fed sequentially so they implicitly maintain positional information. So authors decided to add positional vectors to word vectors. This positional information is read off the sinusoidal curves for different positions and different dimensions. Now the word vectors along with positional information are fed to the multihead attention module. The embeddings matrix is replicated to form the Q, K and V matrices. Now these are passed through a linear layer. This serves the purpose of mapping input vectors to some output vectors in another dimension. Self-Attention is one of the main innovations of the paper. Simple-Attention finds a weighted sum of the vectors based on some external query. In Self-Attention on the other hand each vector pays attention to every other vector, intuitively it's understanding the meaning of a word by paying attention to words nearby. Now we've the attention filter and the V matrix which are multiplied to get the weighted sum of all vectors for each position. In reality authors use multiple Linear layers to produce multiple filters so that they could capture different features in the input. That's why it's called multi-headed attention. Different attention vectors are then concatenated and passed through the linear layer to maintain their dimension. These layers are repeated many times to capture dependencies between not just two words but different pairs and phrases.

The Decoder takes two inputs, the matrix computed by Encoder and the output that has been generated thus far. The embedding matrix is masked so that Self-Attention only works for the words generated by the decoder. Now the Encoder matrix is replicated to form Q and K matrices and V matrix is the output of masked multi-headed attention layer. Again the attention filter and weighted sum of value vectors is computed. Now these are fed to some linear layers and passed through softmax to produce probability distribution of words. Now newly generated word along with previously generated words form the input to decoder which then produces next output word. This process repeats until the decoder generates an end token.

This Transformer architecture changes the landscape of NLP and is the motivation for many amazing neural network models BERT and GPT-2.

CHAPTER 4

SYSTEM REQUIREMENT SPECIFICATION

4.1 Overview

This project deals with the generation of comprehensive FAQ for a given summarized text using machine learning models. The purpose of this project is to aid in various domains while generating FAQs based on a given text summary. For example, a domain where the importance of FAQs is high is the banking and insurance fields. Automatic generation of FAQs would be of great benefit in these domains.

The objective during the course of undertaking this project is to generate fairly relevant FAQs from a given text summary that would aid the end users to a great extent and reduce human intervention while generating FAQs. The goal is to generate the FAQs accurately for a given context that takes the entire text corpus into consideration while generating these FAQs and provide performance which would be on par with the developed current state of the art question answer generation models.

During the course of training the models, text corpus from various domains would be used. During the testing phase, the plan is to limit to a few domains for better human evaluation of the generated FAQs.

4.2 Product Perspective

The task of automatically generating questions and answers which are syntactically and semantically valid from a summarized text corpus is useful in many applications. The process of manually generating questions and answers is a task that involves intensive labour as one has to first read and understand the passage and then start parsing the input passage.

This product helps overcome this hurdle of manually generating pairs of questions and answers from a given text passage by utilizing state of the art methods to automatically generate questions and answers of reasonable quality without any human intervention.

4.2.1 Product Features

The following features would comprise the product:

- Preprocessing of input text before passing it to the necessary pipeline
- Generating Questions and corresponding answers based on understanding of text corpus. The main focus would be on generating FAQs
- Additionally a feature to filter out questions that are out of context could potentially be included.

4.2.2 General Constraints, Assumptions and Dependencies

- The cost of implementing this model will be minimal as there are not many hardware dependencies.
- Each component in the pipeline will get its input in a suitable format for easier processing and prevent generating errors.
- The input corpus text must be large enough to generate a diverse set of FAQs.
- The computational resources provided by the user will be sufficient to run the model on the appropriate platform to generate the FAQ set.

4.2.3 Risks

- The system may generate questions which are out of context or ambiguous questions.
- Though the questions generated may be syntactically accurate, they may not be relevant for the specified domain.
- Human evaluation may be required to filter the FAQs to get the most appropriate FAQ set for the required use case.
- The FAQs will not be very diverse if the input corpus text is limited.

4.3 Functional Requirements

4.3.1 Input Validation

The product accepts input from the user. Before passing the input to the pipelined model, necessary validation has to be performed on the input text corpus whether the given corpus is in the necessary format that is suitable to be passed to the preprocessing module. If the given input passes the necessary checks then the module returns a true boolean value, else a false boolean value is returned.

4.3.2 Input Preprocessing

Once the input is validated the input needs to be preprocessed to be encoded and passed to the machine learning models. The machine learning models. Preprocessing involves natural language processing techniques like stop word removal, stemming etc. that ensures the quality of input passed is maintained at the required standard.

4.3.3 Input Encoding

Input encoding involves representing the given text corpus in a matrix or vector form as suitable for the machine learning models that would be implemented. The model architecture defines the input vector length and other necessary details.

4.3.4 Pipelined Model

The pipelined model is a machine learning model that takes in a given vector as input and after performing the necessary operations gives an output vector that has to be decoded and presented in the required format that would suit the needs of the end user.

4.3.5 Output Decoding

The output vector generated by the pipelined model has to be decoded to be presented in an understandable format to the end user. The decoding of the output involves transforming the output vector into a corpus of words or in this case question answer pairs to be displayed to the user based on the input corpus text that had been passed.

4.3.6 Display of FAQs

The FAQs are displayed in a neat and easily understandable format to be viewed by the end user. The User interface built has to ensure good readability of the question answer pairs in a predefined format.

4.4 External Interface Requirements

4.4.1 User Interfaces

- GUI for login/registration
- GUI for uploading required text document/paragraph
- GUI for generating relevant FAQ’s

4.4.2 Hardware Requirements

- Minimum 256 MB of RAM
- Minimum 500 MB of Disk Space
- Pentium IV 1.7 GHz class or better processor

4.4.3 Communication Interfaces

As it would be an offline application, i.e purely desktop-based, there would be no need for any communication interfaces.

4.5 Non-Functional Requirements

4.5.1 Performance Requirement

- **Reliability:** Under ideal conditions as described by the hardware and software requirements, the system is required to generate the questions and answers in a bound time frame as would be recommended during the testing phases. Besides the time constraints, it is to be made sure that the output delivered by the system is reliable in terms of accuracy and usability.

- **Quality:** The questions and answers generated should be semantically correct. The expected precision and recall we plan to achieve is 80%. The generated question-answer pairs would be compared to existing human generated question answer pairs to output the quality metric.

- **Robustness:** The system should be robust to different kinds of text corpus structures that would be fed in as input to ensure the necessary output standards are maintained.

Robustness reduces the impact of operational mistakes, erroneous input data, and hardware errors. The error rate should be kept at a minimum.

- **Adequacy:** The input required of the user should be limited to only what is necessary. The software system should expect information only if it is necessary for the functions that the user wishes to carry out. The software system should enable flexible data input on the part of the user and should carry out plausibility checks on the input.

- **Producing Results:** The outputs presented by the system should be in a clear and well defined formatted structure and should be easy to interpret. Certain flexibility can be afforded by the system with respect to the degree of the details and output format.

- **Maintainability and Testability:** The system should prioritize maintainability with respect to ease of debugging and scope for modifications and extensions. Testability should also be given priority to allow for easy testing and also allow for separate testing of various modules.

- **Efficiency:** The system should be efficient and should fulfill the purposes with best possible utilization of the resources presented.

- **Portability:** The system should be designed keeping portability in mind. Adapting the software system to run on systems it was not designed to run should be cumbersome, that is effort involved in portability should be lesser than implementation in the new environment from scratch.

4.5.2 Safety Requirements

Recovery systems have to be put in place based on the criticality of the systems.
Triplicate recovery systems must be in place for mission-critical systems.

4.5.3 Security Requirements

A unique User ID and Password would be assigned to authorized individuals who will be given the complete right to access the software. Unauthorized users would not gain any access.

CHAPTER 5

HIGH LEVEL DESIGN

5.1 Introduction

The High Level Design Document presents the concepts and necessary details to implement machine learning models based on natural language processing techniques to generate a comprehensive set of frequently asked questions(FAQs) and responses on a summarized text provided by the user. Generating FAQs from a given text is valuable to the user as it provides the user all the necessary information encompassed in the given text and simplifies the user's job as well and also serves as a self learning help. With the advent of natural language processing models, the quality of output produced by machine learning models has increased significantly and particularly automatic question answer generation has become a trending research area. During the course of development we plan to use these techniques to generate FAQs instead of a single question pair. This will be more insightful and is of significant importance in various domains like banking and insurance. Hence, we plan on exploring research opportunities in this particular topic.

5.2 Current System

The conventional way of creating question answers has been tedious and time-consuming. This aspect gives an opportunity to automate this task by building an automatic question answer generation system. Coming to some of the current systems, they utilize rule based question answer generation and statistical answer selection to produce outputs. Latent Semantic Analysis is performed to summarize the given input text. Other systems use a technique in which the answer is already known, that is a span of text is highlighted by the user indicating the answer span. Some systems concentrate on a particular domain like training on a corpus of news articles which are labelled using crowdsourcing techniques. GPT2-medium and BERT based models are used to build the question answer generation system. The system demonstrates use of sentence only and span+sentence question generation techniques. Another system demonstrates a Answer-Clue-Style Aware Question generation system. It uses extra information like clue and style while generating the question answer pairs. Clearly there exists no demarcation as to which is the best system and we plan to incorporate learnings from all the various systems and integrate them to produce a pipeline that generates FAQs that cover the entire text and also match the various quality parameters that we set to achieve. We plan to research another methodology and test the results produced.

5.3 Design Considerations

5.3.1 Design Goals

- We believe that generating FAQs using an answer-agnostic or answer-unaware technique is more relevant compared to generated questions from a given passage and span labelled by the user.
- The answer unaware question answer generation pipeline also involves predicting span before passing it as input to the trained model.
- This helps generate question answer pairs in an as human way as possible and also improves the quality and diversity of question answer pairs generated by the pipeline.
- For the purpose of span prediction, BERT based transformer models have shown good results which we wish to emulate for this purpose and possibly make further improvements.
- Our aim is to implement a question-answer generation model based on the GPT2-medium or similar model that is finetuned to match the needs of the task at hand and also be optimized to generate results relevant to the user.
- We believe that incorporating context of the given input passage into the question-answer generation model will produce better quality of results and hence plan to incorporate this aspect as well.
- Our aim is to generate valid question-answer pairs consistently across a variety of inputs and show optimum performance in varying scenarios.

5.3.2 Architecture Choices

As mentioned previously(under Current System), our model will be a combination of all the positive aspects from each of the architectures under consideration.

1. Seq2Seq Model

Sequence-to-Sequence Model is a neural network model that converts a set of words in a sentence into another set of words. In basic terms, it is translation. It can be understood with the help of a diagram:-

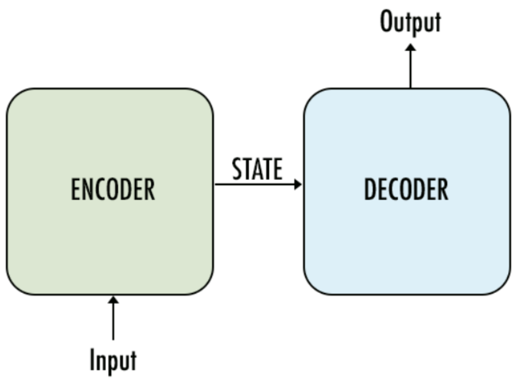


Figure 1: Seq2Seq Architecture

This model first captures and stores all the information of the input sequence, i.e details of the objects, how they are related to each other, etc in an intermediate state and then uses this information for the subsequent translation. The encoders and the decoders in this are Recurrent Neural Networks. These networks take a word vector from the entered set of words and a hidden context vector from the previous unit of time. This combination is later passed to the decoder which generates the target sequence. The vector size for this intermediate vector is however fixed which plays a huge role in this entire process. The ability to remember all the information passed in the input sentence is critical to the quality of the translated output. The biggest advantage of the model is that it can easily store important object-related features and information. However, at the same time, in cases with very long sentences as input, the intermediate state fails and is not able to capture all the information. It reads input sequentially due to which parallel computation cannot be performed on this model. It will often forget the details in the initial part of the sentence while trying to process the output after going through the entire input due to the exploding gradients condition. The model's performance can be improved by using the concept of "attention" to eliminate the possible bottlenecks caused by the intermediary vector. In this mechanism, a weighted sum of all the hidden states is passed, in parallel and not sequentially, as the context vector to the decoder unlike the case where the context vector of only the previous unit of time was passed. This ensures that each hidden state can be accessed independently by the decoder states without any predefined bias. Experimentally, it has been proven that models with the attention mechanism outperform the conventional method of Seq2Seq Model irrespective of the length.

2. Transformer Model for NLP

The Transformer is an architecture that helps to solve seq2seq models handling long dependencies. It uses the attention mechanism discussed in seq2seq models but it does not employ RNN's in comparison. The following is the flowchart diagram of the entire transformer architecture:-

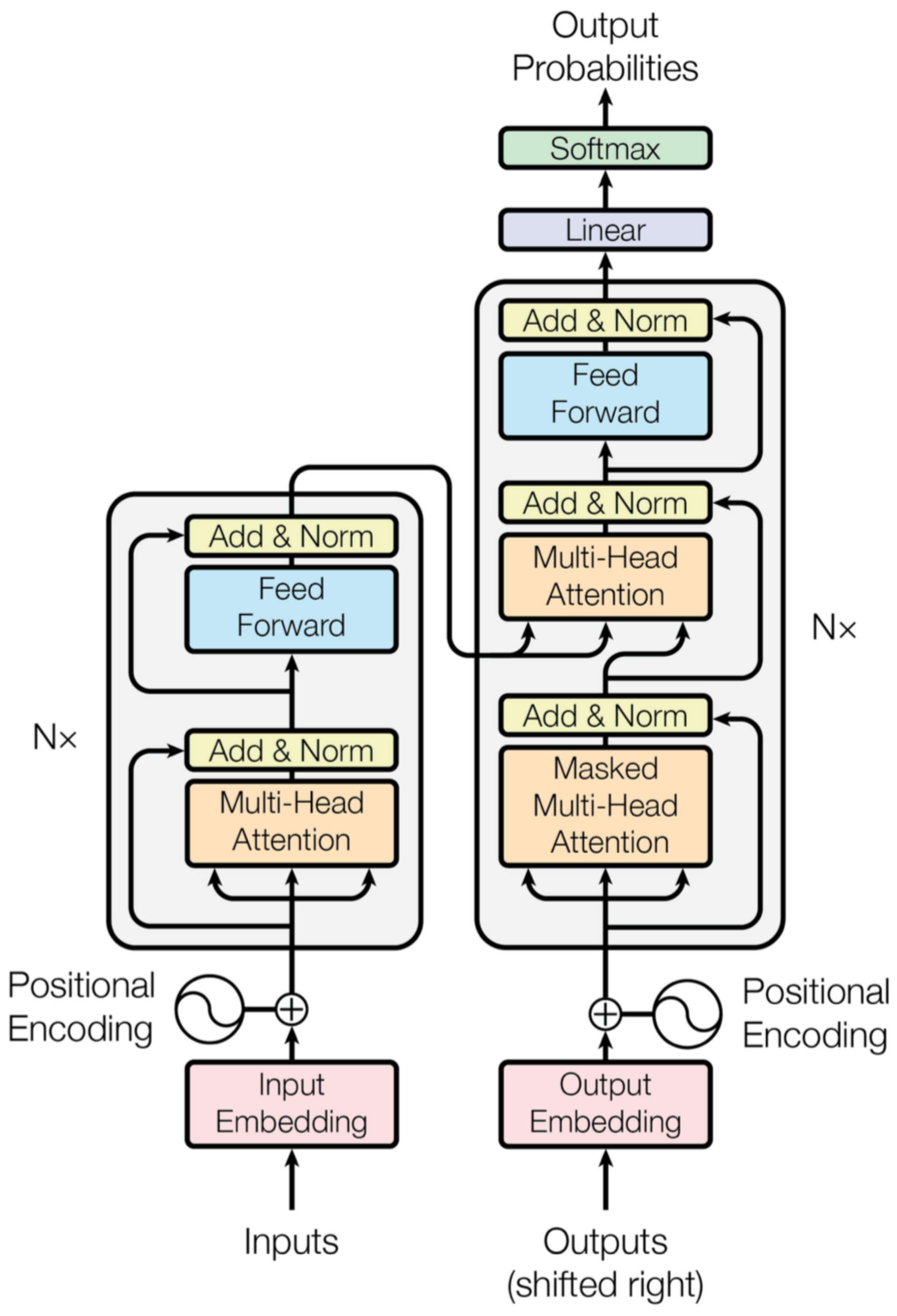


Figure 2: Transformer Architecture 1

In the initial stage, the input embedding ensures that words with similar meanings have similar representations and are mapped to vectors accordingly with the help of the pre-trained model. The next stage of the model encompasses the context of the word through the encoding of the locations of the set of words. Through this the model gets an understanding of the context of the word in the sentence. This encoding is done using trigonometric functions and is as per the embedded vector dimensions. The encoder block then consists of multi-head attention and feed forward modules. The attention level calculates and checks the relevance of a particular input word with the other words in the sequence. The feed forward modules then perform computations on these individual vectors. This block is followed by the decoder block wherein the target sequence is embedded and positionally encoded like the input sequence. The only difference between this block and the encoder block is that this block includes masking of the words in the target sequence for learning purposes. The attention vectors generated are compared and mapping of words between input and target sequence takes place next. This can be done as the attention vectors now have relational comparisons with words in its own language as well as that of the other language. In the event of a masked word, the decoder is stopped from having a look at the words ahead of the mask and tests its prediction based on the previous words as context. The prediction of the masked word by the decoder is done based on the encoder output and the target sequence attention vector calculated. The softmax function helps in calculating human-understandable probability of the choices for the predicted word. The word with the highest probability is then taken to be the predicted word.

The brief architecture of attention calculation is as follows:-

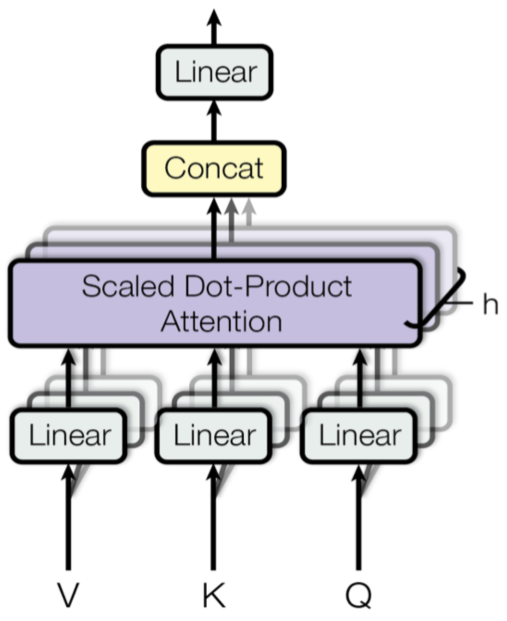


Figure 3: Transformer Architecture 2

Q is a query matrix, K contains all the keys whereas V has all the values. Simply put, these are identical copies of embedding vectors along with the positional encodings for a single word for different components of the word.

The final step in the prediction of the attention is calculation based on the mathematical formal of the same:-

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

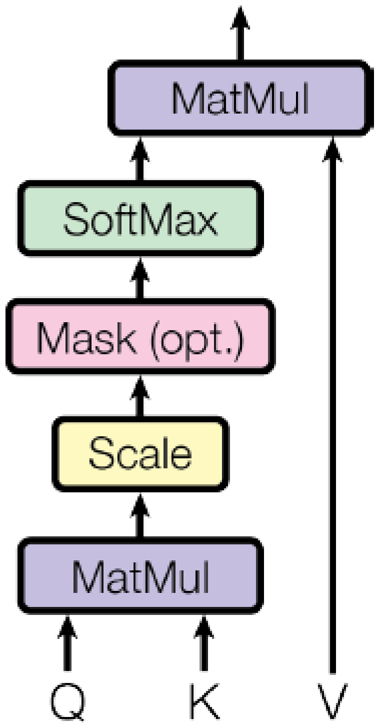


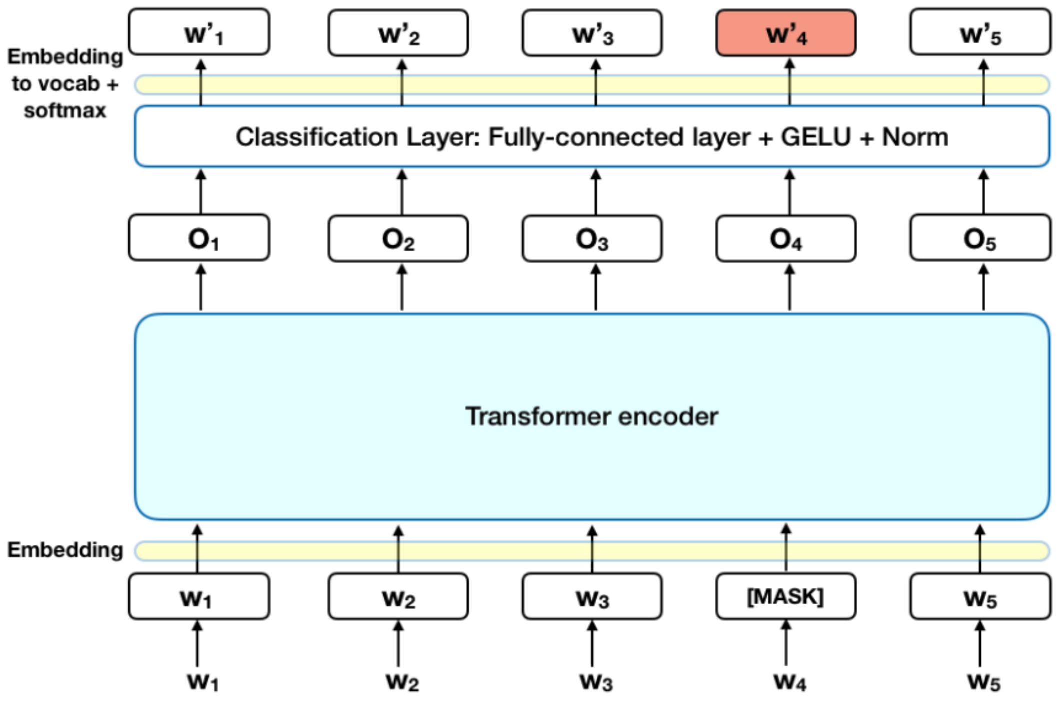
Figure 4: Transformer Architecture 3

One of the biggest advantages is that it overcomes the length-related dependency issues plagued in seq2seq models and one of the disadvantages is that it is not bidirectional like BERT Model.

3. BERT Model

BERT (Bidirectional Encoder Representations from Transformers) is a model that has completely revolutionized the art of Natural Language Processing. BERT makes use of the attention mechanism, though bidirectionally, as discussed in the previous model. It has already been trained on Wikipedia corpus. It is basically utilised to understand language and context. Bert Model has been pre-trained using 2 training strategies:-

1) Masked Language Model:- Few words,chosen at random, in the input sequence are masked. The model tries to determine the contents of the masked words based on the context provided by the unmasked words. This uses the bidirectional effect in the best possible manner using fill in the blanks strategy. This can be explained more clearly with the help of the following flowchart diagram:-



w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 5: BERT Architecture 1

2) Next Sentence Prediction:- The model receives a pair of sentences as input and learns to ascertain whether the sentences are connected or not. A CLS token is added at the beginning of the first sentence and a SEP token is inserted at the end of every sentence within the same context. This is very useful for our project since it generates an understanding which sentences relate to another sentence. The following is a simple diagram that depicts the training:-

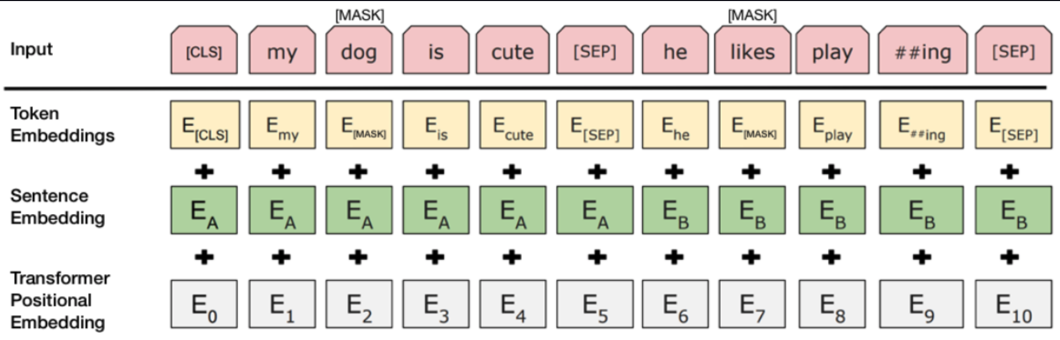

Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 6: BERT Architecture 2

After pre-training the model, the next phase consists of the finetuning of the model based on the specific task needed. In our case, the Bert Model is used in FAQ Generators wherein it is used to mark the answer in the given text sequence given a question based on the same sequence of text. That is, input is question and passage(unlike a pair of sentences with masked words) and output is the predicted span of the answer.

One of the biggest advantages of the BERT Model is that it is the largest model of its kind, is bidirectional and trained on millions of parameters.The higher the training data the higher is the accuracy of this model. Major disadvantage is that if the training corpus is less, then desired results may not be achieved.

5.3.3 Constraints, Assumptions and Dependencies

- Interoperability requirements

  The deep learning libraries that will be used will be compatible with the version of the python programming language that will be used to implement the application. It has to be ensured that the various modules do not have any compatibility issues with each other.

- Data repository and distribution requirements

  Compiled datasets like SQuAD 1.1 that cover a wide range of domains will be utilized. Besides this, more question answer pairs are expected to be scraped from webpages for further enhancement of models or for testing purposes across different parameters to ensure optimal performance.

- Performance related issues as relevant

  The larger the document corpus during training, the more the computational resources are required to fulfill the tasks. To perform more operations in parallel with a faster speed the number of GPUs will need to be increased to meet the demands. Incase of higher computational resource requirements tools like Google Colab will be used.

- End-user environment.

  The end user will be provided with a graphical user interface. The user will have to provide a summarized text for which FAQs will be generated automatically and displayed in a neat and understandable format.

- Availability of Resources.

  Data corpus is required for training purposes and testing purposes. Pretrained models which are available will be fine tuned to suit the needs of the question answer generation model or the span prediction model.

- Hardware environment

  Training machine learning models is a hardware intensive task. Dedicated GPUs will be used to accelerate the training time and performance of the machine learning models.

- Software environment
  - ❏ Python - Application will be coded in the python programming language. Setting up a python environment is necessary for execution.
  - ❏ Tensorflow - Deep learning framework will be used for implementation of the models.
  - ❏ Keras - Deep learning framework will be used for the implementation of the models.
  - ❏ Google Colab - Tool will be used when there is requirement for higher GPU capabilities for training purposes.
- Target environment is a desktop or web based application. The NLP technologies we plan to use are state-of-the-art and are expected to have certain issues in their infancy and hence maintainability is crucial. Scalability to larger inputs from the user needs to be taken into account and ensure that equivalent performance is demonstrated across a variety of input conditions.
- Assumptions

❏ The input corpus text will be large enough to generate a variety of FAQs.

❏ The computational resources provided by the user will be sufficient to run the model to generate the required FAQs.

❏ The texts used for input and training will be in standard English.

CHAPTER 6

LOW LEVEL DESIGN

6.1 Introduction

6.1.1 Overview

Generating FAQs from a given text is valuable to the user as it provides the user all the necessary information encompassed in the given text and simplifies the user's job as well and also serves as a self-learning help. With the advent of natural language processing models, the quality of output produced by machine learning models has increased significantly and particularly automatic question answer generation has become a trending research area. The low level design document delves deeper into the implementation details and provides reasoning for design strategies.

6.1.2 Purpose

The low level design document covers in detail the various modules that compose the entire system. The implementation details along with their diagrammatic representations are presented in this document. Details are also provided as to how the various modules interact with each other and how the integration is performed.

6.1.3 Scope

The low level design document provides the various class diagrams, deployment diagrams and the sequence diagrams. Besides this, certain implementation details and early test results are also discussed to provide evidence that the design strategies are yielding the right results.

6.2 Design Constraints, Assumptions and Dependencies

- Interoperability requirements

  The deep learning libraries that will be used will be compatible with the version of the Python programming language that will be used to implement the application. It has to be ensured that the various modules do not have any compatibility issues with each other.

- Data repository and distribution requirements

  Compiled datasets like SQuAD 1.1 that cover a wide range of domains will be utilized. Besides this, more question answer pairs are expected to be scraped from webpages for further enhancement of models or for testing purposes across different parameters to ensure optimal performance.

- Performance

  The larger the document corpus during training, the more the computational resources are required to fulfill the tasks. To perform more operations in parallel with a faster speed, the number of GPUs will need to be increased to meet the demands. In the event that higher computational resources are needed, tools like Google Colab will be used.

- End-user environment.

  The end user will be provided with a graphical user interface. The user will have to provide a summarized text for which FAQs will be generated automatically and displayed in a neat and understandable format.

- Availability of Resources.

  Data corpus is required for training and testing purposes. Pretrained models which are available will be fine tuned to suit the needs of the question answer generation model or the span prediction model.

- Hardware environment

  Training machine learning models is a hardware intensive task. Dedicated GPUs will be used to accelerate the training time and performance of the machine learning models.

- Software environment

  - ❏ Python - Application will be coded in the python programming language. Setting up a python environment is necessary for execution.
  - ❏ Tensorflow - Deep learning framework will be used for implementation of the models.
  - ❏ Keras - Deep learning framework will be used for the implementation of the models.
  - ❏ Google Colab - Tool will be used when there is a requirement for higher GPU capabilities for training purposes.

- Target environment is a desktop or web based application. The NLP technologies we plan to use are state-of-the-art and are expected to have certain issues in their infancy and hence maintainability is crucial. Scalability to larger inputs from the user needs to be taken into account and ensure that equivalent performance is demonstrated across a variety of input conditions.
- Assumptions

1. The input corpus text will be large enough to generate a variety of FAQs.
2. The computational resources provided by the user will be sufficient to run the model to generate the required FAQs.
3. The texts used for input and training will be in standard English.

6.3 Design Description

The FAQ Generation System is composed of four fundamental modules. These modules are span generator/extractor, question generation, answer generator and finally the Question-Answer Ranker. First the span is extracted and based on the extracted span and context, the questions are generated. Once the questions are generated, the answer generator module takes the questions along with the relevant context to arrive at the answer. Finally, all the generated questions and answers are ranked according to how semantically accurate they are and are finally presented to the user. All these modules are integrated together to ensure that the FAQ generation system works seamlessly.

6.3.1 Master Class Diagram

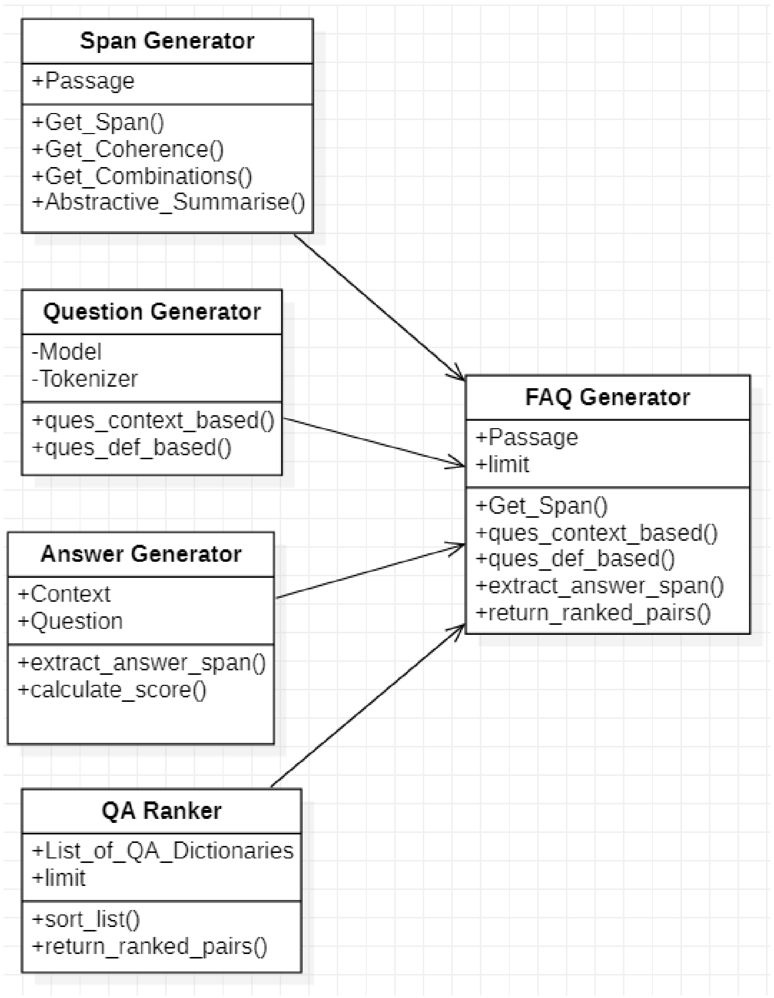


Figure 7: Master Class Diagram

6.3.2 Span Generator

6.3.2.1 Description

The Span Generator module is the entry point into the entire question answer generation pipeline. Span Generation deals with identifying the part of the text in the given sentence on which potential questions can be generated. Spans are those texts based on which we can ask questions. The span generator module generates spans based on identifying various entities and also using a transformer model for this task. Another additional task undertaken by the Span Generator Module is to group coherent sentences together and summarise it to get a single meaningful sentence. Later on spans are extracted from these newly created sentences. This ensures that connections are maintained across sentences and questions that connect concepts across multiple sentences can also be generated in the later phases.

6.3.2.2 Use Case Diagram

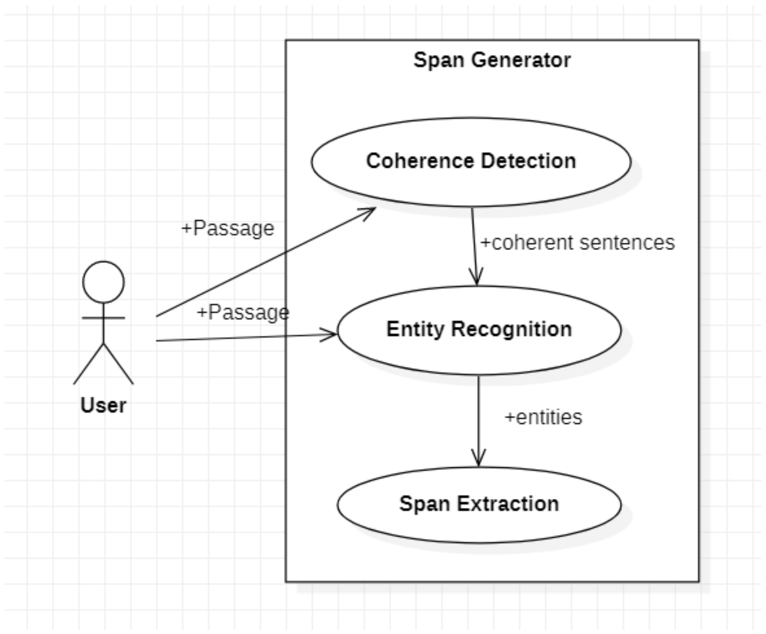


Figure 8: Span Generation Use Case Diagram

6.3.2.3 Class Diagram

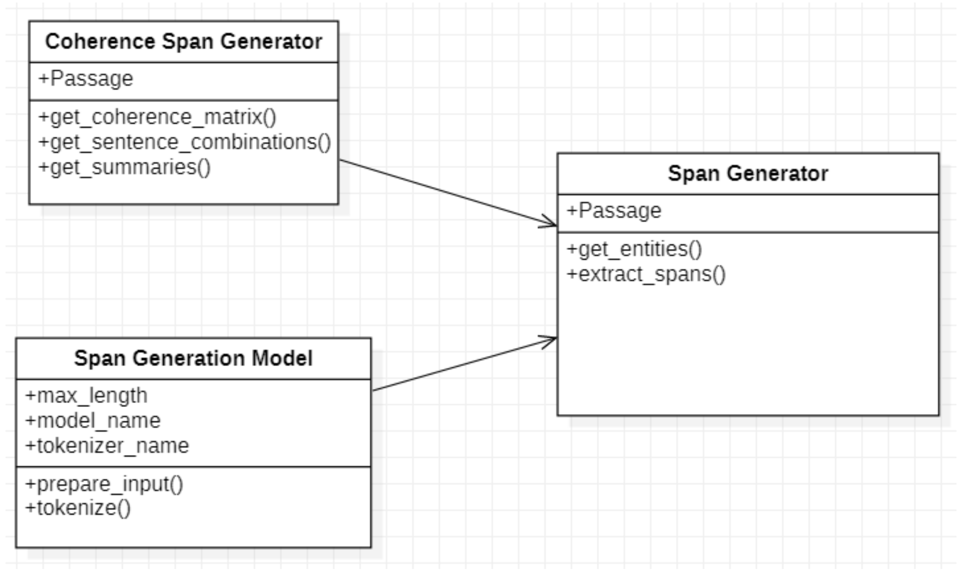


Figure 9: Span Generation Class Diagram

6.3.2.3.1 Coherence Span Generator

6.3.2.3.2 Class Description

The coherence span generator class, takes in a passage of english text, constructs a matrix based on how coherent sentences are, combines the coherent sentences and abstractively summarises the group of sentences.

6.3.2.3.3 Data members

The coherence span generator class has one data member which is the passage of text that is passed as input. This passage is a string data type and contains text on which the user wants to generate various FAQs.

6.3.2.3.4 Method - get_coherence_matrix()

This method serves the purpose of checking the similarity or coherence between sentences. The method takes in the passage tokenized as sentences as input. A universal sentence encoder is used for generating vectors for each sentence and cosine similarity is used to get a similarity score. The method gives a symmetric square matrix as output with each element in the matrix representing the similarity between each row element and each column element. No specific parameters need to be passed to the method besides the passage itself. A nested loop is used to compare sentences and the matrix is populated by calculating the similarity scores between the corresponding sentences.

Pseudocode:

```
    m=initialise_matrix()
    for x each sentence i:
```

```
                    for y each sentence i+1 to n:
                            m[x][y]=similarity_score
```

6.3.2.3.5 Method - get_sentence_combinations()

This method serves the purpose of grouping similar sentences together. This method takes the coherence matrix and sentences and groups similar sentences together. For each sentence, a sentence with highest similarity before and after it is found and combined. These combinations are stored as a list which can be used in the later phase to abstractedly summarise the groups of sentences. The parameters that are taken are the sentences and the coherence matrix.

Pseudocode:

```
l<-empty_list
for x each sentence 1 to n-1:
        before<-sentence before given sentence with highest similarity
        after<-sentence after given sentence with highest similarity
        c=before+x+after
        append c to l
```

6.3.2.3.6 Method get_summaries()

This method takes in the sentence combinations and performs abstractive summarisation. A pretrained model is used for abstractive summarisation. The list of grouped sentences is taken as input and a list of summarised sentences is given as output. There are no other specific parameters that are required.

Pseudocode:

```
l<-empty_list
for x each in sentence_group:
        t=summarise(x)
        append t to l
```

6.3.2.3.7 Span Generation Model

6.3.2.3.8 Class Description

This class has the pretrained span generation model. The pretrained model is used to get the spans in a given sentence which can be further used for question generation in the further phases of the pipeline. The model is pretrained on the SQuAD dataset and suitably fine tuned for the task at hand.

6.3.2.3.9 Data Members

This class has three data members namely, maximum_length, model_name and tokenizer_name. The maximum length represents the maximum window size of text that can be passed to the model at a time. The model name and tokenizer name represent the names of the pretrained model that have to be used that are already predefined.

6.3.2.3.10 Methods - prepare_input()

This method performs certain operations on the input text to make it suitable to be passed to the machine learning model. For example, certain token tags are inserted into the text to aid in span generation. This method takes in the passage as the input, and tokenises it as sentences and also inserts certain token tags into the sentences. There are no other parameters required for this method.

Pseudocode:

```
sents<-tokenize_passage
l<-empty_list()
for x each in sents:
        t=add_tokens(x)
        append t to l
```

6.3.2.3.11 Method tokenize()

This method, tokenizes the sentences into vectors based on the vocabulary on which the model was trained. The sentences are passed as input and the tokenized sentences are given as output. There are no other specific parameters that are required.

Pseudocode:

```
inputs=ans_tokenizer(text)
return inputs
```

6.3.2.3.12 Span Generator

6.3.2.3.13 Class Description

This class uses the coherence span generator class and the pretrained span generator model to get the actual spans in the given sentences. Besides using the pretrained model, certain entities are also identified in the sentences upon which questions could be generated. So finally, we get all the possible spans in a given sentence.

6.3.2.3.14 Data Members

The coherence span generator class has one data member which is the passage of text that is passed as input. This passage is a string data type and contains text on which the user wants to generate various FAQs.

6.3.2.3.15 Method get_entities()

This method identifies additional pieces of text in the sentence that can be used as span besides those spans that are identified by the pretrained model. From a predefined vocabulary, parts of speech like nouns are identified in the given sentences. The inputs are the individual sentences and the output produced are the entities identified in the sentence. There are no additional specific parameters that are required to be passed to this method.

Pseudocode:

```
l<-empty_list()
b<-Textblob(sent)
for each x in b:
        if x is noun phrase and x not in l:
                append x to l
```

6.3.2.3.16 Method extract_spans()

This method uses the various already defined methods and provides an end to end pipeline for span extraction. The pretrained model is instantiated, coherent sentences are formed, other entities are identified and finally spans are extracted using the pretrained model and also the identified entities. The passage of text is taken in as input and a list of dictionaries is given as output with each dictionary containing a sentence and the corresponding extracted spans. There are no other specific parameters that need to be passed to the method.

Pseudocode:

```
get_coherent_sentences()
get_entities()
d<-empty_dict
l<-empty_list
for each x in sentences:
        t=get_span(x)
        d["sentence"]<-x
        d["spans"]<-t
        append d to l
```

6.3.2.4 Sequence Diagram

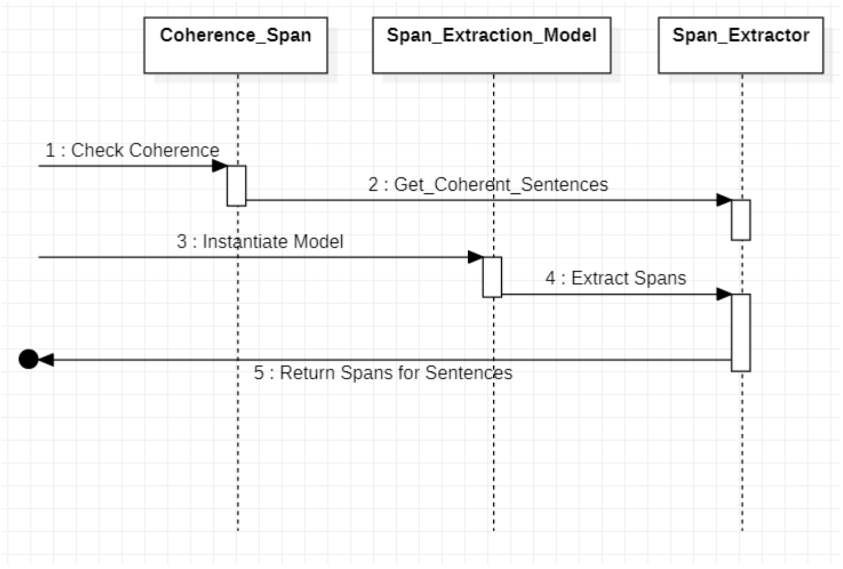


Figure 10: Span Generation Sequence Diagram

6.3.3 Question Generator

6.3.3.1 Description

The Question generator module is responsible for generating Questions given the context and extracted spans. We generate two kinds of Questions. First kind contains Questions, answers to which are present in the context. These are generated with a T5 model that is trained on SQuAD dataset. Second kind of Questions don't have answers in the context and intend to explain domain terms in the context with the help of Knowledge graph. The Domain questions are about defining an entity and some background about that entity where entity can Person, Organization or Place. We use Named Entity Recognition(NER) to tag the words as various entities and then we extract information with the help of a Knowledge graph. A knowledge graph, also known as a semantic network, represents a network of real-world entities—i.e. objects, events, situations, or concepts—and illustrates the relationship between them. This information is usually stored in a graph database and visualized as a graph structure, prompting the term knowledge "graph."

6.3.3.2 Class Diagram

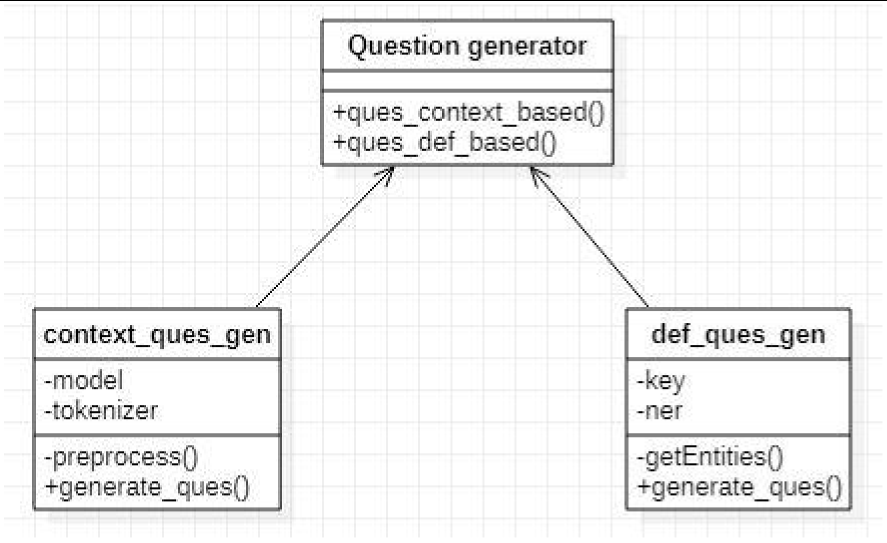


Figure 11: Question Generation Class Diagram

6.3.3.2.1 Class context_ques_gen

6.3.3.2.2 Class Description

This class is responsible for generating context based Questions given the context and spans. So these are the Questions which have their answer present in the context itself. We loop through the spans and generate question for each. We use a T5 model and T5 tokenizer loaded from the transformers library for Question generation. First

6.3.3.2.3 Data member : Tokenizer

We use a T5 Tokenizer loaded from the transformers library that is imported in the module. The Tokenizer output is an encoding of the input that is fed to the Model.

6.3.3.2.4 Data member : Model

We load the T5 model trained for Question generation from the transformers library. This object contains fundamental functions like generate which generates question.

6.3.3.2.5 Method : get_questions()

This method receives as input a list of dictionaries and each contains “sentence” and “spans” as keys. “spans” key contains a list of spans that are generated on the respective sentence. We loop through the various spans for each dictionary, and generate question for each combination of sentence and span. This method returns a list of dictionaries each with sentence, span and question(generated here) as keys.

pseudo code:

```
for each dict in list:
    for each span in dict[“spans”]:
        sentence= dict[“sentence”]
        generate_question(sentence, span)
```

6.3.3.2.6 Method : generate_question()

This method receives as input a sentence and a span. This first concatenates them and encodes it with the “tokenizer”. The encoding is then fed to generate method of the “model” object. The output can be decoded by “tokenizer” to get the Question which is returned.

pseudo code:

```
text <- concatenate sentence and span
encoding <- tokenizer.encode(text)
output <- model.generate(encoding)
question <- tokenizer.decode(output)
```

6.3.3.2.7 Class def_question_gen

6.3.3.2.8 Class Description

This class is responsible for generating Domain based questions that intend to explain some terms in the context. Named entity recognition is used to tag the entities and a Knowledge graph is used to extract their definitions and information.

6.3.3.2.9 Data member : nlp

“nlp” object holds the language model loaded with Spacy that is imported in the module. This object is crucial for performing Named Entity Recognition(NER).

6.3.3.2.10 Data member : key

This is the Key that helps query the Knowledge graph. The Key serves the purpose of Authentication for the Knowledge graph API.

6.3.3.2.11 Method : get_entities()

This method uses the nlp object to do Named Entity Recognition on the text received as input. So various words in the text get tagged as PER for person, ORG for organization etc. We enclose all the information in a list of dictionaries and return it

6.3.3.2.12 Method : generate_ques()

This method gets the context as input. It uses the method above to get Entities. Then it loops over Entities and for each if it's tagged as PER or ORG, it queries the Knowledge graph with corresponding text to get more information about it. We form a list of dictionaries with question and answer as keys in each and return the list.

pseudo code:

```
entities <- get_entities(context)
for each entity in entities:
    if entity == "PER" or entity == "ORG":
        kg_df = knowledge_graph(key, query)
        question= get_question(entity)
        answer = kg_df["result.description"]
```

6.3.3.3 Sequence Diagram

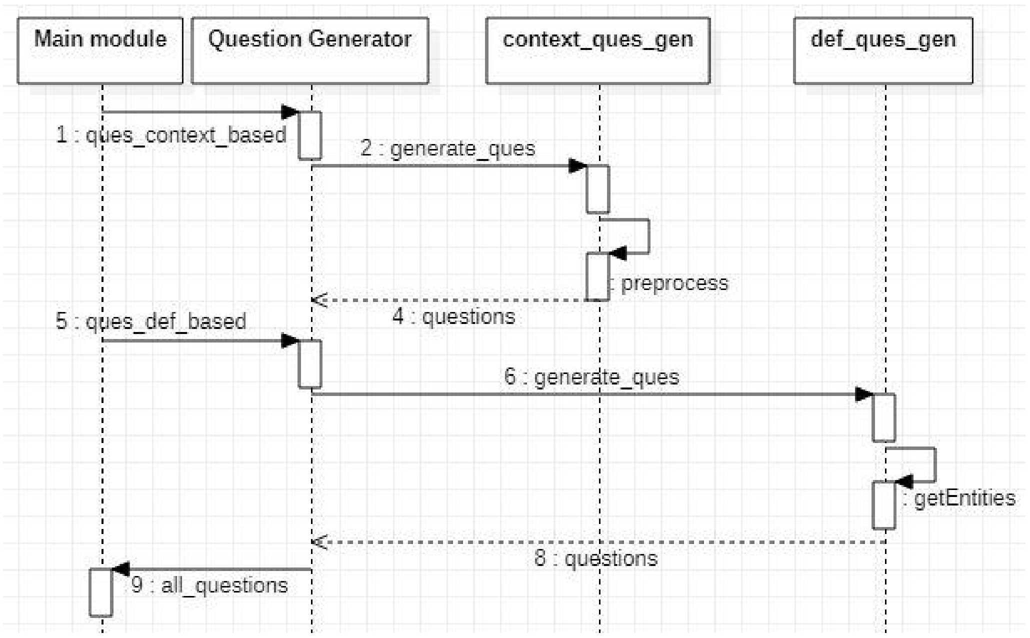


Figure 12: Question Generation Sequence Diagram

6.3.4 Answer Generator

6.3.4.1 Description

The Answer Generation module of the pipeline is responsible for identifying and generating answers from the text based on the questions generated from the Question Generator module. The Answer Generation involves tokenizing the extracted span as well as the generated question. These tokens are fed to a pretrained transformer encoder model in order to generate the relevant answers found in the span. The model also calculates a probability score between 0 and 1 indicating the probability of the generated answer for the given question. After the answer is extracted from the span, it is passed to the knowledge graph as a keyword in order to fetch additional information for the FAQ if required by the user.

6.3.4.2 Class Diagram

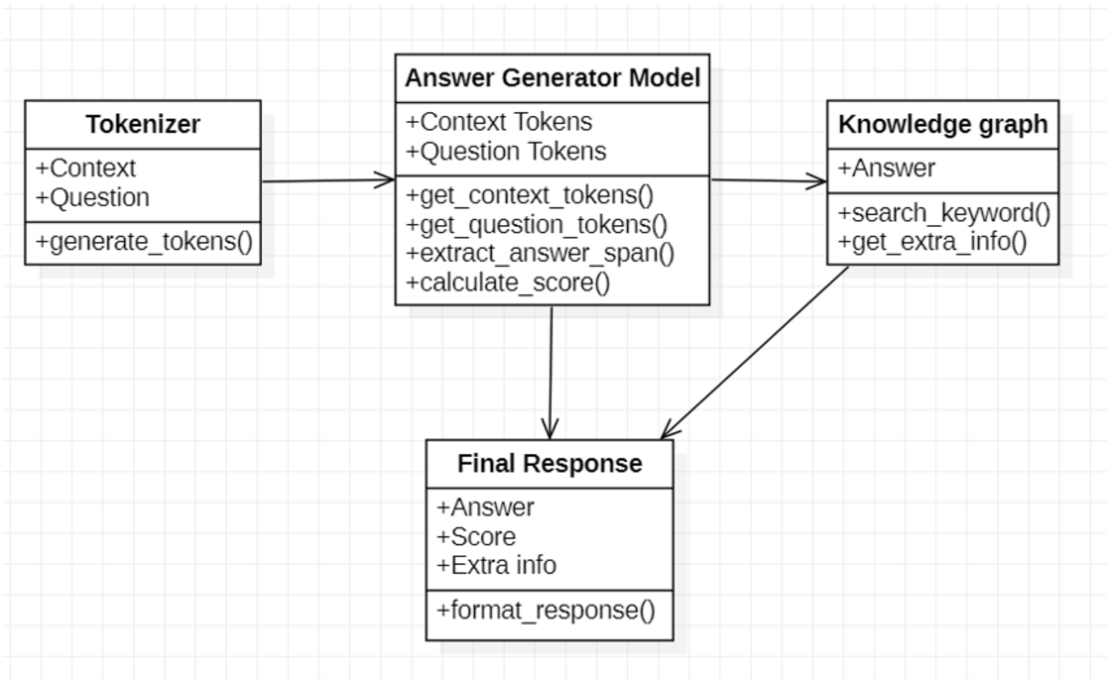


Figure 13: Answer Generator Class Diagram

6.3.4.2.1 Tokenizer

6.3.4.2.2 Class Description

This class takes as input the extracted span (context) as well as the generated question and generates tokens which are passed to the Answer Generator model.

6.3.4.2.3 Data Members

The Tokenizer class has 2 data members, namely Context and Question which are both of type string. Context represents the span of text extracted in the Span Generation module and Question represents the question generated based on the context. They are then converted to individual tokens which are used for answer generation.

6.3.4.2.4 Method generate_tokens()

This method takes the context and question strings and converts them to a list of tokens. These tokens are then passed to the Answer Generation model to extract the set of tokens which form the answer for the given question.

6.3.4.2.5 Answer Generator Model

6.3.4.2.6 Class Description

The Answer Generator Model identifies and generates the answer present in the context based on the given question. It is pretrained on the SQuAD v1 dataset. It uses the SPANBert Model which contains two linear classifiers to mark the start and end of the answer span.

6.3.4.2.7 Data Members

The Data Members of this class are the context tokens and question tokens provided by the tokenizer.

6.3.4.2.8 Method get_context_tokens()

This method is used to fetch the Context tokens provided by the Tokenizer class.

6.3.4.2.9 Method get_question_tokens()

This method is used to fetch the Question tokens provided by the Tokenizer class.

6.3.4.2.10 Method extract_answer_span()

Based on the Question tokens, the relevant Answer tokens are identified from the set of Context tokens and returned. This is implemented using the SPANBert question-answering model.

6.3.4.2.11 Method calculate_score()

The probability of the generated answer is calculated based on the question and context provided and it is returned as a float value between 0 and 1.

6.3.4.2.12 Knowledge Graph

6.3.4.2.13Class Description

The Knowledge Graph class uses Google's knowledge graph API to search for additional information about the generated answer. It uses the answer span as a keyword to search the knowledge graph.

6.3.4.2.14 Data Members

This class uses the answer object which is required by the search_keyword() method.

6.3.4.2.15 Method search_keyword()

This method is used to search the knowledge graph to check whether the keyword passed to it is present in the graph. If it is present, it returns 'true' and tries to fetch the associated information for that keyword.

6.3.4.2.16 Method get_extra_info()

When the keyword is found, the associated information for that keyword is passed back to the answer object in order to display additional information in the FAQ which is not present in the original text passage.

6.3.4.2.17 Final Response

6.3.4.2.18Class Description

This is the final class in the answer generation module. It compiles the generated answer, the calculated score, extra information and the associated question into a convenient format to be displayed to the user.

6.3.4.2.19 Data Members

The data members for this class are Answer, Score, Extra Info. These objects are generated by the extract_answer_span(), calculae_score() and get_extra_info() methods respectively.

6.3.4.2.20 Method format_response()

The final FAQ to be displayed is formatted in a user-friendly way and returned.

6.3.4.3 Sequence Diagram

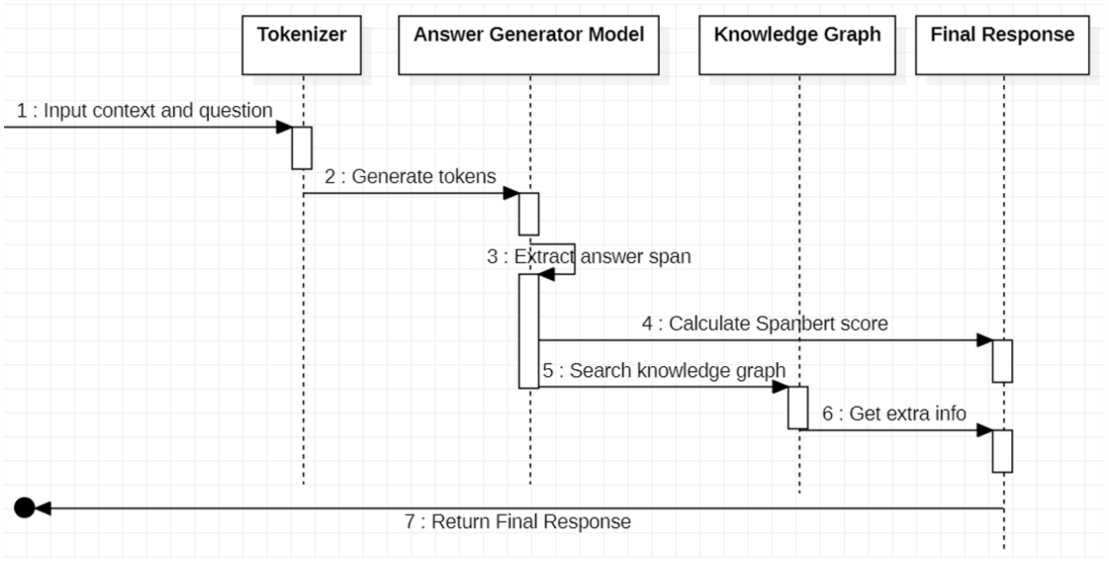


Figure 14: Answer Generator Sequence Diagram

6.3.5 QA Ranker

6.3.5.1 Description

The Question Answer Ranker is the final part of the entire pipeline. This stage involves the comprehensive evaluation and ranking of the question-answer pairs that are being generated in the previous stages of the pipeline. Based on a combination of user preference and the quality of question-answer pairs generated by the pipeline, the top pairs will be returned. Apart from ranking the pairs, the sorting of these pairs is also done so that the best pairs are presented first to the user.

6.3.5.2 Class Diagram

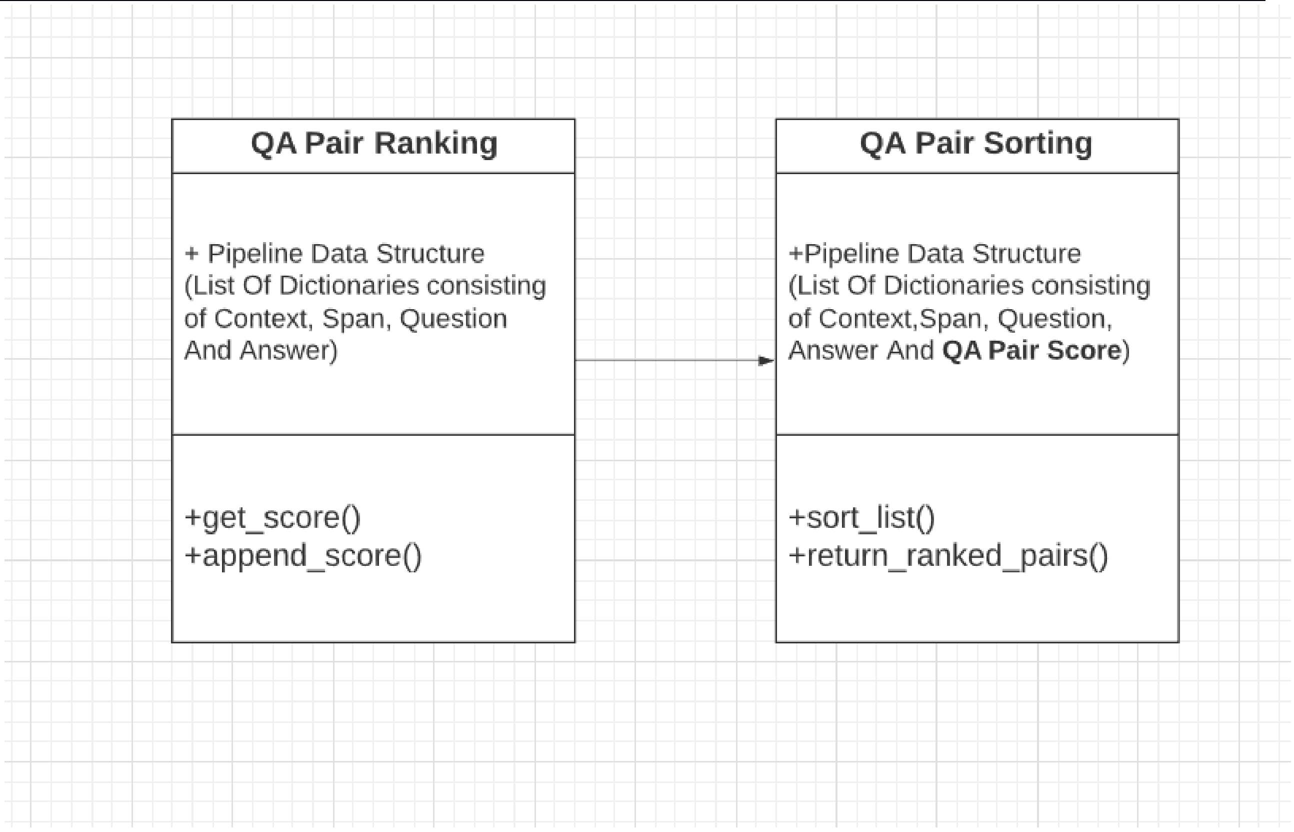

QA Pair Ranking
+ Pipeline Data Structure (List Of Dictionaries consisting of Context, Span, Question And Answer)
+get_score()
+append_score()
QA Pair Sorting
+Pipeline Data Structure (List Of Dictionaries consisting of Context,Span, Question, Answer And QA Pair Score)
+sort_list()
+return_ranked_pairs()

Figure 15: QA Ranker Class Diagram

6.3.5.2.1 QA Pair Ranking

6.3.5.2.2 Class Description

This class entails purely ranking the question answer pairs received from the pipeline. For this purpose, a pre-trained model will be utilized which will score the pairs on their semantic similarity and not based on whether the answer for the given question is appropriate or not.

6.3.5.2.3 Data Members

This class uses only one data member which is a list of dictionaries that is generated along the entire system pipeline. Each dictionary in the list contains the context/sentence, potential spans, generated question and the answer to that question. This list is used for the purpose of this class.

6.3.5.2.4 Method get_score

The list is iterated through each dictionary and the question-answer pair is fed as input into the pre-trained model which checks the semantic similarity between the pair and accordingly outputs the score. (Numerically higher the value, better is the pair)

6.3.5.2.5 Method append_score

The score generated is then appended iteratively to the list of dictionaries. This is required as the score would be needed to be used for the sorting of the pairs. The appending ensures that all the parameters of the FAQ generation right from the prediction of the span to the question and answer generation to the ranking is stored in a single data structure.

Pseudocode:

```
list_of_dict <- list
```

```
for each dict in list_of_dict:
    dict['score'] <- score_model(qa_pair)
```

6.3.5.2.6 QA Pair Sorting

6.3.5.2.7 Class Description

This class is concerned with sorting the pairs generated and returning the optimal number of responses to the users based on their preferences and the quality of question-answer pairs generated.

6.3.5.2.8 Data Members

This class uses only one data member which is the list of dictionaries which is forwarded by the ranking class. This list will have an additional key for each dictionary, which is the score parameter that is added.

6.3.5.2.9 Method sort_list

This method simply uses the built-in sort method to sort the scores using a lambda function. The sorting is done in a decreasing order so that the best pairs are represented to the user.

Pseudocode:

```
list_of_dict <- list
sorted_list <- sorted (list_of_dict, key=lambda k: k['score'], reverse=True)
```

6.3.5.2.10 Method return_ranked_pairs

This method renders the list of pairs to the graphical user interface. Again, the number of pairs depends on user preference and optimal pairs generated by the model.

Pseudocode:

```
list_of_dict <- list
append_score(list_of_dict)
sort_list(list_of_dict)
```

6.3.5.3 Sequence Diagram

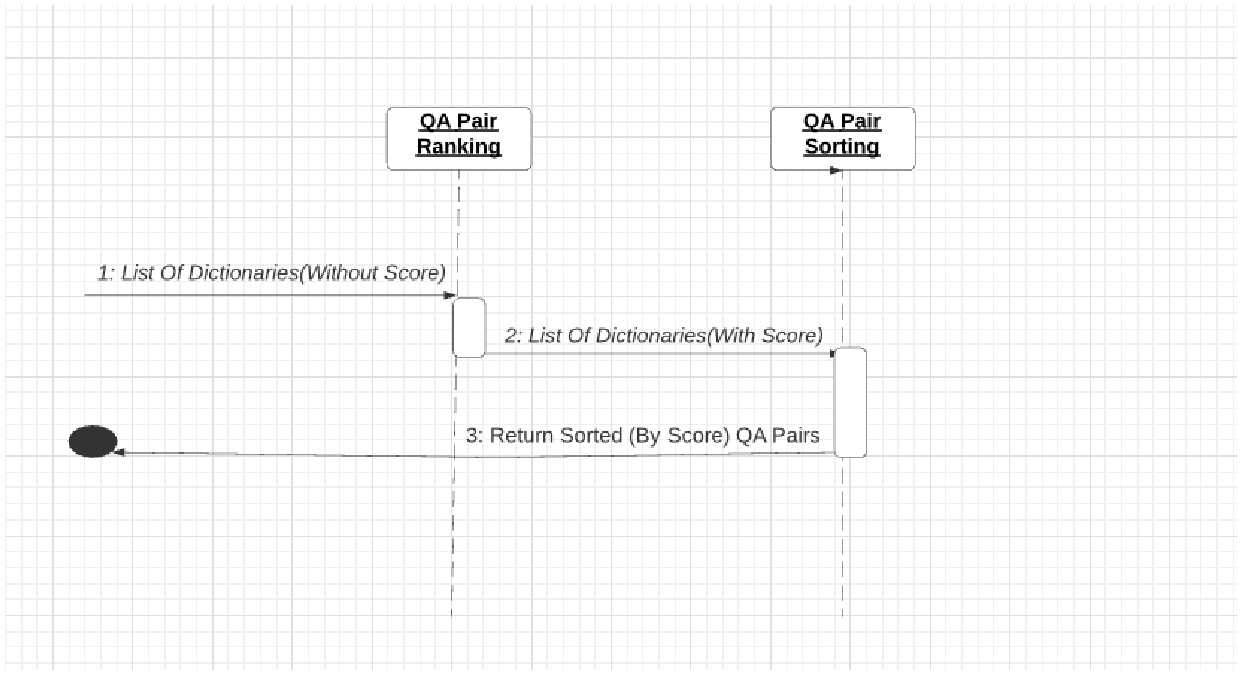

QA Pair Ranking
QA Pair Sorting
1: List Of Dictionaries(Without Score)
2: List Of Dictionaries(With Score)
3: Return Sorted (By Score) QA Pairs

Figure 16: QA Ranker Sequence Diagram

CHAPTER 7

SYSTEM DESIGN

7.1 High Level System Design

7.1.1 Block Diagram

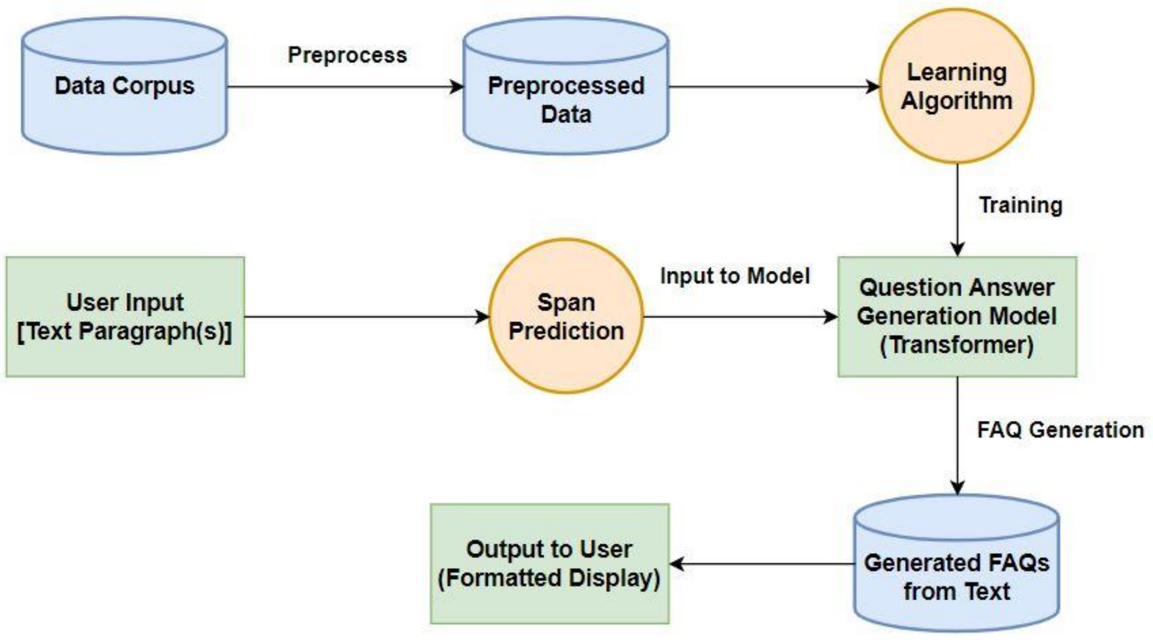


Figure 17: System Design Block Diagram

7.1.2 Logical User Groups and Characteristics

- Consumer Services Industry

  In Consumer service industries the companies have to set up elaborate websites to attract customers and one aspect of these websites is an FAQ page that clears all of the customer's potential doubts that may arise in the user's mind. It is essential to provide clarity to the customer to ensure that the customer's trust in the company is not lost.

- Learners

  Learners are those who try to understand a given piece of text elaborately. Given a paragraph to learn, they can automatically generate question answer pairs from the given paragraph and better their understanding.

- Research Scholars

Research Scholars are those who pursue research in new and upcoming technologies. Question answer generation has been a trending research area. FAQ generation techniques will serve as a base for others looking into conducting further in depth research.

7.1.3 Data Components

- Training Corpus of questions-answers and corresponding text from dataset SQuAD 1.1 which is compiled from various wikipedia articles
- Question answer pairs and corresponding text scraped from relevant websites belonging to suitable domains.
- The training data consists of
    - ❏ segment of text
    - ❏ context
    - ❏ span
    - ❏ questions
    - ❏ answers

7.2 Design Description

7.2.1 Master Class Diagram

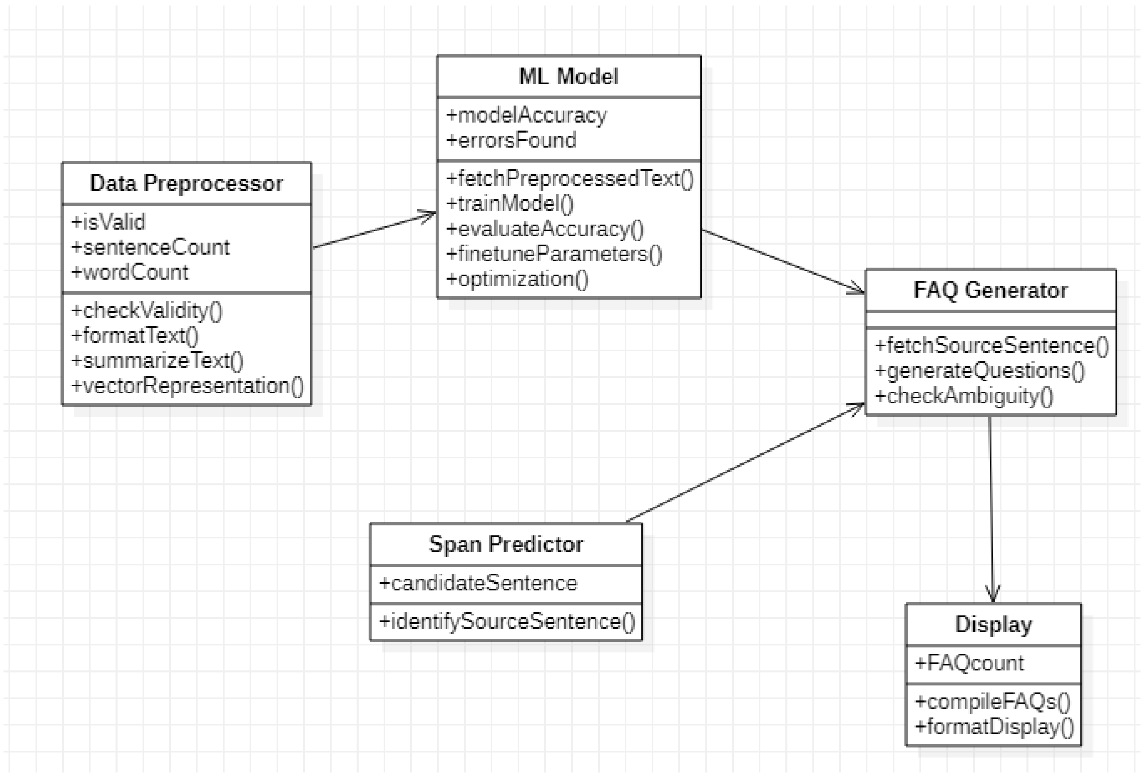


Figure 18: System Class Diagram

7.2.2 Reusability Considerations

- The dataset used during the course of the development is expected to be reusable for a plethora of other language modelling or other tasks related to natural language processing.
- The model that will be constructed for the question answer generation will be built keeping reusability in mind. There will be a possibility to finetune the models and use the model in other specific tasks.

7.3 Swimlane Diagram

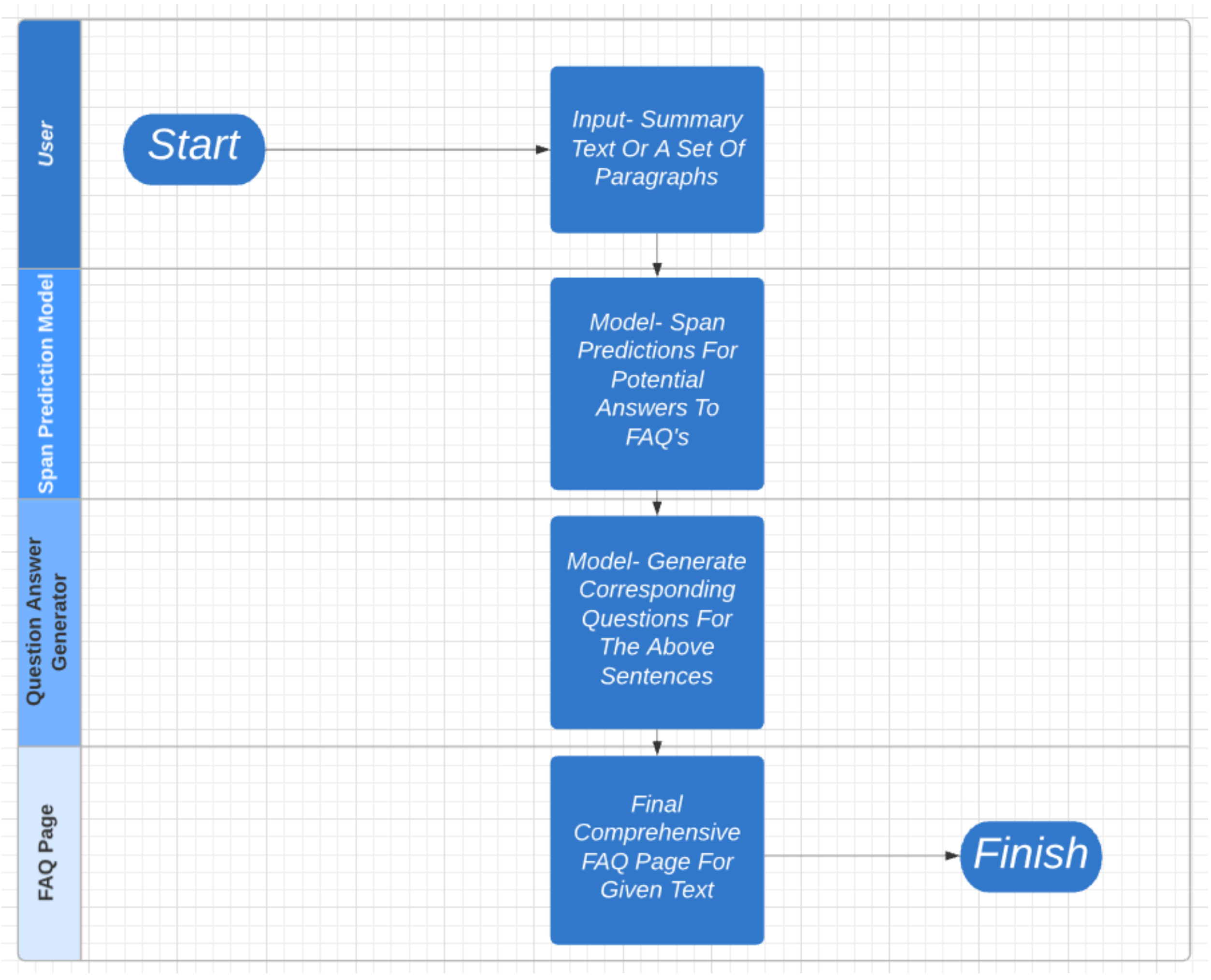

User
Span Prediction Model
Question Answer Generator
FAQ Page
Start
Input- Summary Text Or A Set Of Paragraphs
Model- Span Predictions For Potential Answers To FAQ's
Model- Generate Corresponding Questions For The Above Sentences
Final Comprehensive FAQ Page For Given Text
Finish

Figure 19: Swimlane Diagram

7.4 User Interface Diagrams

The user interface will be designed as follows:

- Provision to accept input from the user in an intuitive manner. The input that has to be entered will be input into a suitable placeholder which is designated to accept text from the user.
- Feature clearly cautioning the user in case an incorrect input entry by the user in case where the input is not suitable for processing further.
- Provision to display automatically generated FAQs in a neat format that can be read easily. The generated FAQs will be displayed in pairs of question-answer with each pair having a clear demarcation.

7.5 Packaging and Deployment Diagram

This entire code and data files will be zipped and shipped. Requirements will be listed and once they are installed the model can be run.

OR

Deployed on a server and the user can utilize the model by accessing the server where it performs intensive calculations and the user can see the results in their browser.

7.6 Help

- A user manual will be provided to the user providing clear instructions regarding the usage and features of the tool at hand. A brief overview will also be provided. All the steps will be clearly mentioned as to how to go about using the application so as to ensure that the user is not stuck at any point. An example usage scenario will also be provided for the user's benefit.
- The user will interact only through the user interface and will have no access to backend workflow and hence there will not arise a need for a technical manual for the user of this application.

7.7 Design Details

7.7.1 Novelty

FAQ generation is an active research area and there is scope for different type of models like transformer model, BERT model**.**

7.7.2 Innovativeness

While implementing the models we will include span prediction and use multiple spans to generate multiple answers instead of using just one span and generate a single question answer pair.

7.7.3 Interoperability

We'll provide a simple intuitive interface to the user to input text and answer span and display the results clearly.

7.7.4 Performance

Dedicated GPUs to accelerate the training time and performance of Neural Networks so that the results obtained are accurate and quick.

7.7.5 Reliability

When the users are displayed the results, they can choose or slightly modify them to reliably present them to their customers.

7.7.6 Maintainability

Improve performance and accuracy of FAQs generated on new domains.

7.7.7 Portability

Users could have various dependencies installed and run the model on any system. Alternatively users could also use different browsers to access the webpage.

7.7.8 Reusability

The base model can be fine tuned to provide better performance on different new domains.

CHAPTER 8

PROPOSED METHODOLOGY

8.1 Algorithms

8.1.1 Span Generation

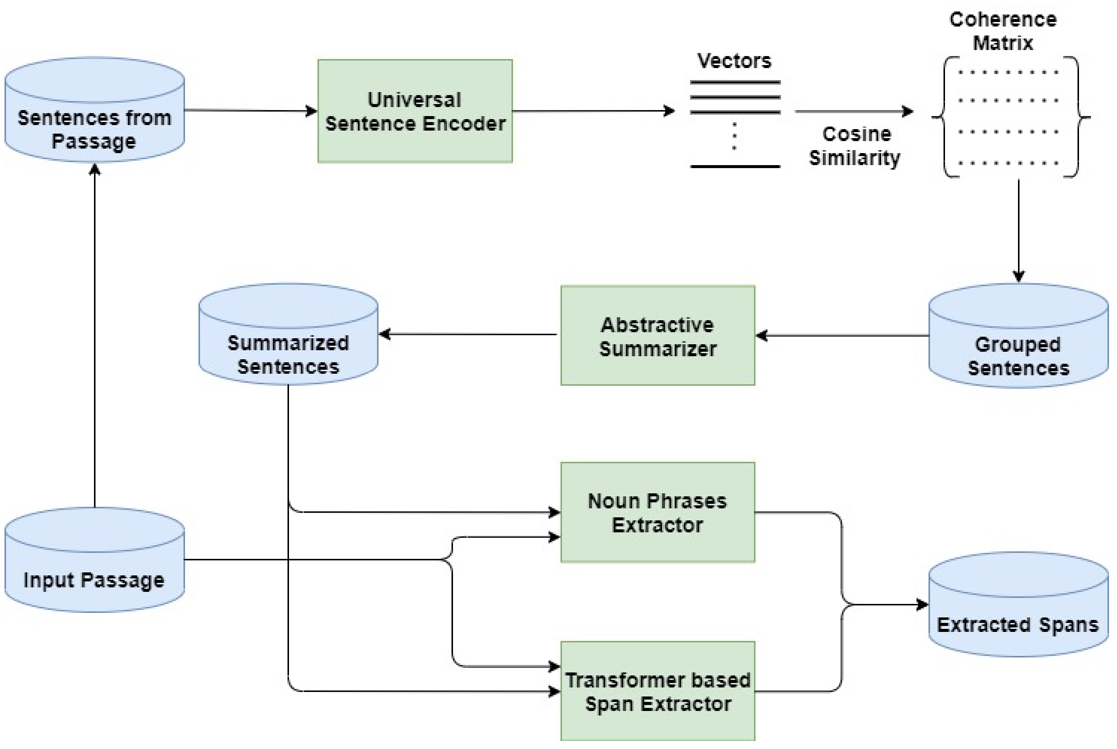


Figure 20: Span Generation Deployment Diagram

- The span generation module is the entry point to the entire FAQ generation pipeline. This module takes in the passage as the input. The passage is broken down into corresponding sentences. Now, there are two separate tasks that are performed.
- The first task deals with encoding the sentences as vectors using the universal sentence encoder. Furthermore, these vectors are compared using the cosine similarity metric and an NxN matrix is generated which holds the coherence values between the sentences. The sentences with high coherence values are paired together so as to be able to relate sentences that are in different parts of the passage and hence be able to link information across sentences. Once all the groups of sentences are obtained, abstractive summarization is performed on each group to get the crux of the information from each group. Later these summarized sentences are passed to the

Noun Phrase Extractor and Transformer Based Span Extractor to generate the suitable spans.

- In the second task, the sentences from the passage are just split up and directly passed to the Noun Phrases Extractor and Transformer based Span Extractor without any prior processing steps. The spans are extracted from individual sentences and stored.
- These two separate tasks help to provide a wider range of spans thereby resulting in production of effective question answer pairs. The sentences and spans are organised as a list of dictionaries for easy access and management in the further modules of the pipeline.

8.1.2 Question Generation

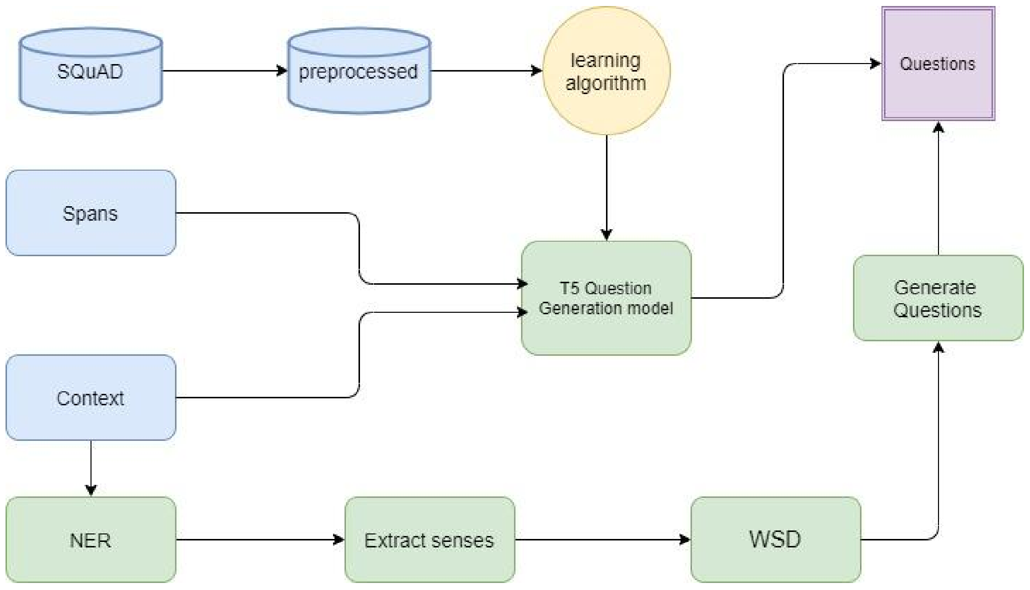


Figure 21: Question Generation Deployment Diagram

- The SQuAD dataset is preprocessed to have only context, span and question as the column. A T5 Question generation model is trained on it to produce questions as output given context and span as input.
- The Spans generated in the previous module are used along with the context to generate questions whose answers are present in the context
- Named Entity Recognition(NER) : The words in the context are tagged as PER for person or ORG for organisation with the help of NER that uses a trained language model.
- Once we have tagged a word, we generate various senses of that word(because a word can have many meanings). Then Word sense disambiguation is used to get the right sense of the word with respect to the context.

- Now generate simple questions with the right sense as the answer. This way we generate questions that give us better understanding of the context and whose answers are not present in the context.

8.1.3 Answer Generation

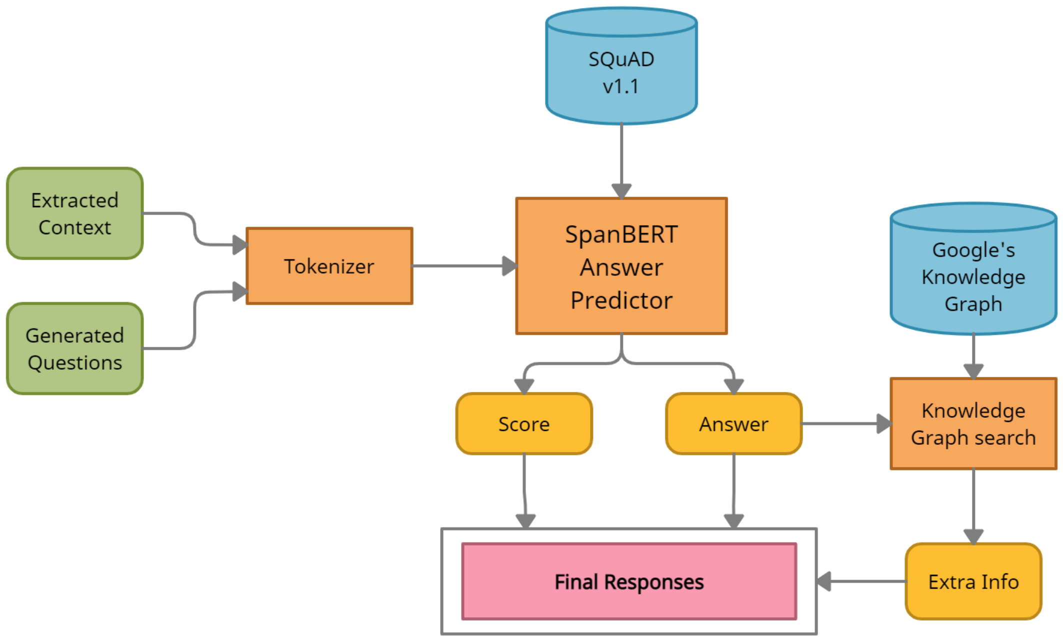


Figure 22: Answer Generation Deployment Diagram

- The Answer Generation Module is used to extract the answer spans from the given passage and also fetch additional information about the answer which is not present in the original passage.
- The SpanBERT Answer Predictor model fine-tuned on the SQuAD v1.1 dataset is used to extract the answer span. It takes as input the context passage along with the questions generated from the previous module.
- The model uses two linear classifiers to predict the start and end of the answer tokens and it also calculates a score between 0 and 1 representing the probability of the predicted answer for the given question.
- The answer generated is then fed to Google's Knowledge Graph in order to fetch extra information about the answer. If the generated answer is present as a keyword in the knowledge graph, the information associated with it is fetched and returned to the answer formatter.
- The Final Response is compiled by combining the Question, Answer, Source sentence, Additional info and Probability score. These FAQs are then forwarded to the QA Ranker module so that the most relevant FAQs can be displayed.

8.1.4 Question Answer Ranking

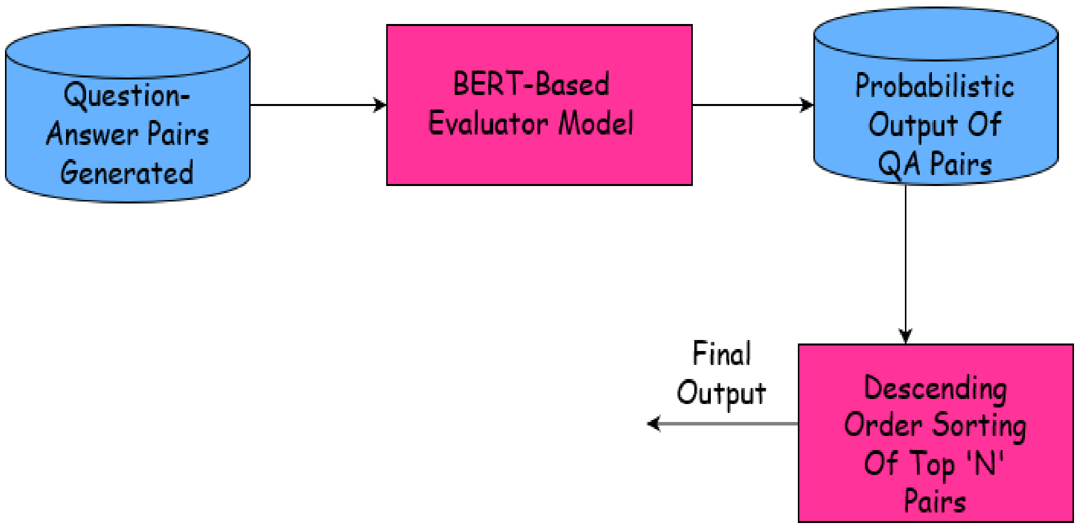

Question-Answer Pairs Generated
BERT-Based Evaluator Model
Probabilistic Output Of QA Pairs
Descending Order Sorting Of Top 'N' Pairs
Final Output

Figure 23: Question Answer Ranking Deployment Diagram

- The Question Answer Ranking Module is the final step of the entire pipeline. This module takes as input the list of dictionaries containing the span, question and answer pairs along with the SpanBERT probabilistic score. (The score is about how relevant the answer is to the question and not whether the answer is right or not)
- The ranker module initially segregates and restructures the list of dictionaries such that in the new list of dictionaries, each dictionaries contains the question answer pair and the score only.
- This unsorted list of dictionaries is then sorted in the descending order of their scores into a sorted list.
- Based on the user input and the number of quality question-answer pairs generated (above a certain threshold of score), the pairs are returned to the user.

CHAPTER 9

IMPLEMENTATION AND PSEUDOCODE

9.1 Span Extraction

```
import nltk
nltk.download('punkt')
from pipelines import answer_pipeline
from nltk import sent_tokenize
nltk.download('averaged_perceptron_tagger')
from textblob import TextBlob
from coherence_span import get_coherent_sentences
def get_entities(sent):
    ans=[]
    b=TextBlob(sent)
    for n in b.noun_phrases:
        if len(n)>4 and n not in ans:
            if n[0] not in ["'"]:
                ans.append(n)
    return ans
x=answer_pipeline("answer-extraction",ans_model="valhalla/t5-small-qa-qg-hl",
ans_tokenizer="valhalla/t5-small-qa-qg-hl")
def extract_answers(context):
    pres_ans=[]
    gen_ans=[]
    c=sent_tokenize(context)
    d=dict()
    for i in x(context):
        pres_ans=[]
        sent=i[0]
        ans=i[1]
        if sent not in d:
            d[sent]=[]
        for a in ans:
            new_a=(a.strip("<pad>")).strip()
            if new_a not in d[sent]:
                d[sent].append(new_a)
                pres_ans.append(new_a)
        for j in x(sent):
            for ans in j[1]:
                a=(ans.strip("<pad>")).strip()
                if a not in d[sent]:
                    d[sent].append(a)
                    pres_ans.append(a)
        ent=get_entities(sent)
        temp_pres_ans=[i.lower() for i in pres_ans]
        for n in ent:
            if n not in temp_pres_ans:
                temp_pres_ans.append(n)
                pres_ans.append(n)
                d[sent].append(n)
    combs=get_coherent_sentences(c)
      for i in combs:
        for j in x(i):
            pres_ans=[]
            sent=j[0]
            ans=j[1]
            if sent not in d:
                d[sent]=[]
            for a in ans:
                new_a=(a.strip("<pad>")).strip()
                if new_a not in d[sent] and abs(len(new_a)-len(sent))>5:
                    d[sent].append(new_a)
                    pres_ans.append(new_a)
            ent=get_entities(sent)
            temp_pres_ans=[i.lower() for i in pres_ans]
            for n in ent:
                if n not in temp_pres_ans:
                    temp_pres_ans.append(n)
                    pres_ans.append(n)
                    d[sent].append(n)
    ind_answer={}
    for i in d:
        ind_answer={}
        ind_answer["sentence"]=i
        ind_answer["spans"]=d[i]
        ind_answer["questions"]=[]
        ind_answer["answers"]=[]
        gen_ans.append(ind_answer)
    return gen_ans
```

9.2 Question Generation

```
import torch
from transformers import T5ForConditionalGeneration,T5Tokenizer

model =
T5ForConditionalGeneration.from_pretrained('ramsrigouthamg/t5_squad_v1')
tokenizer = T5Tokenizer.from_pretrained('ramsrigouthamg/t5_squad_v1')
device = torch.device("cpu")
model = model.to(device)
```

```
def get_questions(qa_dict):
      for qa in qa_dict:
            for span in qa['spans']:
                  text="context: {} answer: {}".format(qa['sentence'],span)

encoding=tokenizer.encode_plus(text,max_length=384,pad_to_max_length=False,tr
uncation=True,return_tensors="pt")
                  input_ids, attention_mask = encoding["input_ids"],
encoding["attention_mask"]
                  outs = model.generate(input_ids=input_ids,
attention_mask=attention_mask, early_stopping=True, num_beams=5,
num_return_sequences=1, no_repeat_ngram_size=2, max_length=72)
                  dec = [tokenizer.decode(ids,skip_special_tokens=True) for
ids in outs]
                  Question = dec[0].replace("question:","")
                  Question = Question.strip()
                  qa['questions'].append(Question)
```

9.3 Answer Generation

```
from transformers import pipeline

qa_pipeline = pipeline(
    "question-answering",
    model="mrm8488/spanbert-finetuned-squadv1",
    tokenizer="mrm8488/spanbert-finetuned-squadv1"
)

def get_gold_answer(qa_dict):
    for qa in qa_dict:
        for question in qa['questions']:
            qa['answers'].append(extract_gold_ans(qa['sentence'],question))

def extract_gold_ans(context,question):
    response = qa_pipeline({'context': context,'question': question})
    return response
```

9.4 Question Answer Ranking

```
'''
Helper functions that when given a list of dictionaries that contains
sentence(context),span,question, answer and score of that particular QA pair,
it sorts the dictionaries in the descending order of the score (i.e highest
score will be first and lowest score will be last)
'''
```

```
def sort_list (list_of_dictionaries):
    unsorted_list=list()
    for item in list_of_dictionaries:
        for i in range (len(item["questions"])):
            d=dict()
            d["question"]=item["questions"][i]
            d["answer"]=item["answers"][i]["answer"]
            d["score"]=item["answers"][i]["score"]
            unsorted_list.append(d)
    sorted_list = sorted(unsorted_list, key=lambda k: k["score"], reverse=True)
    if len(sorted_list)<=5:
        return sorted_list
    else:
        filtered_list = [d for d in sorted_list if d["score"]>=0.8]
        return filtered_list

'''
Given list of dictionaries with sentence(context),span,question and answer, it
focuses only on QA and using a pre-trained model scores them and also appends
the score as one of the keys of the dictionary
'''
def rank_qa_pairs(list_of_dictionaries):
    for i in list_of_dictionaries:
        rank_score = qapair_rank_pipeline("[CLS]" + i["questions"] + "[SEP]" +
i["answers"] + "[SEP]")
        i["score"] = rank_score["LABEL_0"]
    sort_list(list_of_dictionaries)
```

CHAPTER 10
EXPERIMENTATION RESULTS AND DISCUSSION

10.1 Experimentation Outcome

Once the pipeline was constructed, we wanted to compare the results generated by our pipeline with that of human generated results. We picked 2 random paragraphs from the SQuAD dataset and 3 different text paragraphs from the website of PES University. We also prepared some human generated question answer pairs for the paragraphs that were picked from websites. The human generated results were considered the ground truth and we passed the same passages to our question-answer generation pipeline and noted the results. The performance of the question answer generation pipeline was on par with the human generated results. The sample passages and corresponding human generated and machine generated output is shown below.

Passage 1:

Today, the programs at PES University are sought after by students from around the country. Leading industries choose PES University when they need the right talent. One of the key reasons for this is the University's focus on admitting the best talent in India and abroad. We are proud about the tradition of crème de la crème of CET, PESSAT and other national level entrance exams consistently preferring a PES Institution. The University is committed to excellence in education that meets the expectations of the students aspiring for enriching academic and professional careers.	
Human Generated Results	**Machine Generated Results**
Question: Programs at PES University are sought after by whom? Answer: Students from around the country Question: When do leading industries choose PES University? Answer: When they need the right talent. Question: What is one of the key reasons why leading industries choose PES University? Answer: The University's focus on admitting the best talent in India and abroad Question: What is the University proud about? Answer: crème de la crème of CET, PESSAT consistently preferring a PES Institution Question: What is the University committed to?	Question: What institution does leading industries choose when they need the right talent? Answer: PES University Question: Leading industries choose PES University when they need what? Answer: the right talent Question: What is the University committed to? Answer: excellence in education Question: The university is committed to excellence in education that meets the expectations of students aspiring for enriching academic and what else? Answer: professional careers Question:What is the tradition of CET, PESSAT and other national entrance exams?

Answer: Excellence in education that meets the expectations of the students	Answer: crème de la crème

Table 1: Sample Outputs for Paragraph 1

Passage 2:

People's Education Society (PES) was founded in 1972 and started PUC course with just over 40 students in a rented gymnasium at Bangalore. Today, PES has more than 15,000 students and 1000 staff spread across four different campuses in Karnataka and Andhra Pradesh offering programs ranging from Pre-University to Post Graduation. PES is focused on four main educational areas: Engineering, Medicine, Management and Life Sciences. PES offers both foundation and specialization programs leading to Bachelors, Masters and Ph.D Degree. PES is contributing to the field of education through its institutions.	
Human Generated Results	**Machine Generated Results**
Question: When was People's Education Society (PES) founded? Answer: 1972 Question: How many different campuses does PES have across Karnataka and Andhra Pradesh? Answer: 4 Question: How many students and staff does PES have today? Answer: 15000 students and 1000 staff Question: Which 4 main educational areas does PES focus on? Answer: Engineering, Medicine, Management and Life Sciences. Question: How is PES is contributing to the field of education? Answer: Through its institutions Question: What does PES offer? Answer: Foundation and specialization programs leading to Bachelors, Masters and Ph.D Degree.	Question: Along with Engineering, Medicine and Life Sciences, what is PES's other major educational area? Answer: Management Question: Along with Andhra Pradesh, in what state is PES located? Answer: Karnataka Question: Who contributes to the field of education through its institutions? Answer: PES Question:Along with Masters and Ph.D, what type of degree is offered by PES? Answer: Bachelors Question: When was the People's Education Society founded? Answer: 1972 Question: How many main educational areas does PES focus on? Answer: four

Table 2: Sample Outputs for Paragraph 2

Passage 3:

Students a chance to participate in the process of discussing new ideas, developing them into

research concepts and workable products. Some students consider this experience as a turning point in their careers. A chance encounter opens up a new world of possibilities. At PES, students pursue their interests through diverse communities: from debates to the famed PES band, the various clubs and teams reflect the wide range of personal and professional interests of our vibrant, student community. Our air-conditioned auditorium provides a platform for the events.

Human Generated Results	Machine Generated Results
Question: What do students consider as a turning point in their careers? Answer: Discussing new ideas and developing them into research concepts and products. Question: What can a chance encounter do? Answer: It can open up a new world of possibilities. Question: How can students pursue their interests? Answer: Through diverse communities such as the debate and the band Question: What do the various clubs and teams reflect? Answer: Wide range of personal and professional interests of the student community Question: What is a platform for the events? Answer: Air-conditioned auditorium	Question: Along with personal interests, what do PES clubs reflect? Answer: professional Question: What do students have the chance to discuss? Answer: new ideas Question: How do students pursue their interests at PES? Answer: through diverse communities Question:What do students develop ideas into? Answer: research concepts and workable products Question: What opens up a new world of possibilities? Answer: A chance encounter Question: What provides a platform for the events? Answer: air-conditioned auditorium Question: What do some students consider this experience as in their careers? Answer: a turning point

Table 3: Sample Outputs for Paragraph 3

Passage 4:

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.

Human Generated Results	Machine Generated Results

Question: Which NFL team represented the AFC at Super Bowl 50? Answer: Denver Broncos Question: Which NFL team represented the NFC at Super Bowl 50? Answer: Carolina Panthers Question: Where did Super Bowl 50 take place? Answer: Santa Clara, California Question: Super Bowl 50 decided the NFL champion for what season? Answer: 2015 Question: What year did the Denver Broncos secure a Super Bowl title for the third time? Answer: 2015 Question: What was the final score of Super Bowl 50? Answer: 24–10	Question: In what year did Super Bowl 50 take place? Answer: 2015 Question: What was the name of the American football game that decided the 2015 national football league champion? Answer: super Bowl 50 Question: What does NFC stand for? Answer: National Football Conference Question:Who did the Denver Broncos defeat to win their third Super Bowl? Answer: Carolina Panthers Question: Where are the Broncos located? Answer: Denver Question: Where is Levi's Stadium located? Answer: San Francisco Bay Area

Table 4: Sample Outputs for Paragraph 4

Passage 5:

Many faults are able to produce a magnitude 6.7+ earthquake, such as the San Andreas Fault, which can produce a magnitude 8.0 event. Other faults include the San Jacinto Fault, the Puente Hills Fault, and the Elsinore Fault Zone. The USGS has released a California Earthquake forecast which models Earthquake occurrence in California.	
Human Generated Results	**Machine Generated Results**
Question: Which fault can produce a magnitude earthquake of 8.0? Answer: San Andreas Question: What magnitude of earthquake can many faults produce? Answer: 6.7 Question: Other than the San Jacinto Fault, and the Elsinore Fault, name one other fault. Answer: Puente Hills Question: Which organization released a California Earthquake forecast? Answer: USGS Question: The earthquake forecast models what features	Question: Where is the Earthquake forecast from? Answer: California Question: What type of occurrence does the USGS model? Answer: Earthquake Question: What is the name of the Hills Fault? Answer: Puente Hills Fault Question:What fault can produce a magnitude 8.0 earthquake? Answer: San Andreas Fault Question: What type of earthquake can many faults produce? Answer: magnitude 6.7+

of earthquakes in California? Answer: occurrence	Question: What is the name of the fault zone? Answer: Elsinore Fault Zone Question: Many faults are able to produce what magnitude earthquake? Answer: 6.7+

Table 5: Sample Outputs for Paragraph 5

10.2 Testing Results

We picked 100 different passages from the SQuAD test set and generated question answer pairs for these passages. The test set contained ground truth human generated question answer pairs. These were stored in a pickled list. The question answer pairs generated by our pipeline were also stored in a separate pickled list. BLEU Score is a standard testing metric when it comes to language synthesis tasks. We utilized the BLEU Score to measure the quality of the generated questions and answers respectively with respect to the ground truth questions and answers. We have also shown the interpretation of the various BLEU Score ranges to prove that our results are very promising. The results are as shown below.

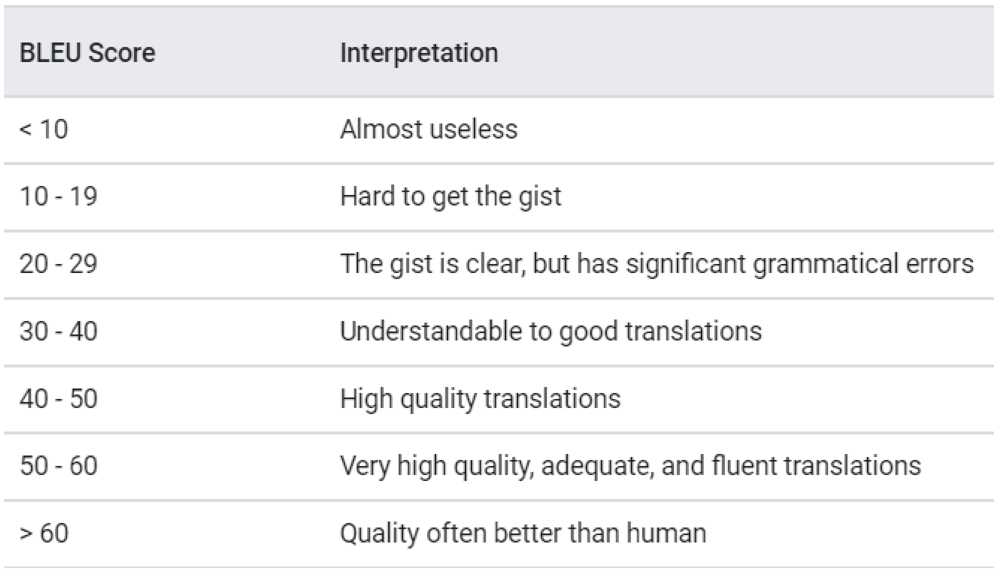
BLEU Score Statistics for Answers generated by the end to end pipeline				
Individual 1 gram	Individual 2 gram	Individual 3 gram	Cumulative 2 gram	Cumulative 3 gram
0.3562	**0.4322**	**0.4479**	**0.3810**	**0.3976**

Table 6 : BLEU Statistics 1

BLEU Score Statistics for Questions generated by the end to end pipeline				
Individual 1 gram	Individual 2 gram	Individual 3 gram	Individual 4 gram	Individual 5 gram
0.6232	**0.5187**	**0.6496**	**0.7762**	**0.8643**
	Cumulative 2 gram	Cumulative 3 gram	Cumulative 4 gram	Cumulative 5 gram

0.5307	**0.5316**	**0.5643**	**0.6046**

Table 7: BLEU Statistics 2



BLEU Score	Interpretation
< 10	Almost useless
10 - 19	Hard to get the gist
20 - 29	The gist is clear, but has significant grammatical errors
30 - 40	Understandable to good translations
40 - 50	High quality translations
50 - 60	Very high quality, adequate, and fluent translations
> 60	Quality often better than human

Figure 24: Interpretation of BLEU Scores

Besides presenting the BLEU Scores, we also tested the coverage through quantifying the number of question answer pairs generated per paragraph by our pipeline with respect to the ground truth question answer pairs. Even in this case, we are able to provide more results as the number of question answer pairs generated by the pipeline is more in comparison to the ground truth results. The results are as shown below.

Average number of question answer pairs generated for a paragraph of text	
Ground Truth	Machine Generated
10.28	**24.58**

Table 8: Average QA Statistics

Average number of question answer pairs generated per sentence	
Ground Truth	Machine Generated
0.875	**2.092**

Table 9: Average QA Statistics 2

CHAPTER 11

CONCLUSION AND FUTURE WORK

The architecture we have developed has shown very promising results. We have taken into account the factor of coherence between sentences and also the fact that additional domain knowledge can be used to generate better results. So we have used the concepts of coherence measurement and using a knowledge graph to gain additional domain knowledge. These have resulted in more meaningful results as demonstrated through the very promising metrics we have obtained during the testing phase. In comparison to the currently existing question answer generation pipelines, we have gone a step forward and have added our own innovations to the various steps in the question answer generation pipeline. That said, there is scope for further improvement to make the solution more robust and open it to more applications in different avenues.

Domain knowledge can be more tightly knit with the entire architecture and more use of this domain knowledge can be made than what we are doing currently. Different representations can be tried to suitably represent additional information rather than just through a standard knowledge graph. Furthermore, one can experiment with fine tuning the entire pipeline on a particular domain so that the question answer pairs generated related to that domain have a great quality and shown immense robustness.

To open up the question answer generation pipeline to more avenues, input formats like videos and audios that contain some speech can be experimented with. For example, given a video as input, the information in the video can be suitably extracted and question answer pairs can be generated for the the content present in the video. The same is applicable for when audio is passed as input as well. Such kinds of different input formats can be trialled and developed further to increase the applications of the automatic question answer generation pipeline. Another aspect that can be looked into, is expanding our pipeline to generate multiple choice questions automatically. This would be really useful in academia and will help in designing various assignments in educational institutes.

Automatic Question Answer Generation has immense potential and has scope for improvement and expansion for use in newer applications. With the base architecture that we have developed, one can build on it and improvise and find newer applications.

REFERENCES/ BIBLIOGRAPHY

[1] Ko, Wei-Jen & Chen, Te-Yuan & Huang, Yiyan & Durrett, Greg & Li, Junyi. (2020). Inquisitive Question Generation for High Level Text Comprehension. (https://www.aclweb.org/anthology/2020.emnlp-main.530.pdf)

[2] Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. (2020). Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus. In Proceedings of The Web Conference 2020 (WWW '20). Association for Computing Machinery, New York, NY, USA, 2032–2043. DOI (https://doi.org/10.1145/3366423.3380270)

[3] Lovenia, Holy & Limanta, Felix & Gunawan, Agus. (2018). Automatic Question-Answer Pairs Generation from Text.10.13140/RG.2.2.33776.92162. (https://www.researchgate.net/profile/Holy_Lovenia/publication/328916588_Automatic_Question-Answer_Pairs_Generation_from_Text/links/5beb1663a6fdcc3a8dd460d3/Automatic-Question-Answer-Pairs-Generation-from-Text.pdf)

[4] Last, Mark & Danon, Guy. (2018). A Syntactic Approach to Domain-Specific Automatic Question Generation. (https://arxiv.org/abs/1712.09827)

[5] Qi, Peng. (2019). Answering Complex Open-Domain Questions At Scale. (https://nlp.stanford.edu/pubs/qi2019answering.pdf)

[6] Badugu, S., Manivannan, R. A study on different closed domain question answering approaches. Int J Speech Technol 23, 315–325 (2020). (https://doi.org/10.1007/s10772-020-09692-0)

[7] Borut G., Marko F., Milan O., A Question Answering System On Domain Specific Knowledge With Semantic Web Support, International Journal Of Computers, Issue 2, Volume 5 (https://www.naun.org/main/NAUN/computers/19-1156.pdf)

[8] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, Yoav Artzi. (2019). BERTScore: Evaluating Text Generation with BERT. (https://arxiv.org/abs/1904.09675)

[9] Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (arxiv:1810.04805) (https://arxiv.org/abs/1810.04805)

[10] Kumar, Vishwajeet & Muneeswaran, Sivaanandh & Ramakrishnan, Ganesh & Li, Yuan-Fang. (2019). ParaQG: A System for Generating Questions and Answers from Paragraphs. (https://arxiv.org/abs/1909.01642)

[11] Das, B., Majumder, M., Phadikar, S. et al. Automatic question generation and answer assessment: a survey. RPTEL 16, 5 (2021). https://doi.org/10.1186/s41039-021-00151-1 (https://telrp.springeropen.com/track/pdf/10.1186/s41039-021-00151-1.pdf)

[12] Amidei, J., Piwek, P., Willis, A. (2018). Evaluation methodologies in automatic question generation 2013-2018. In Proceedings of The 11th International Natural Language Generation Conference. Association for Computational Linguistics, Tilburg University, (pp. 307–317). (https://www.aclweb.org/anthology/W18-6537.pdf)

[13] Brown, Jonathan & Frishkoff, Gwen & Eskenazi, Maxine. (2005). Automatic Question Generation for Vocabulary Assessment.. 10.3115/1220575.1220678. (https://www.researchgate.net/profile/Gwen-Frishkoff/publication/220816907_Automatic_Question_Generation_for_Vocabulary_Assessment/links/0c9605399d2426306a000000/Automatic-Question-Generation-for-Vocabulary-Assessment.pdf)

[14] Rajpurkar, Pranav & Zhang, Jian & Lopyrev, Konstantin & Liang, Percy. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. 2383-2392. 10.18653/v1/D16-1264.(https://nlp.stanford.edu/pubs/rajpurkar2016squad.pdf)

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010. (https://dl.acm.org/doi/pdf/10.5555/3295222.3295349)

APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS

- FAQs (Frequently Asked Questions) - FAQs are a list of questions and answers related to a particular subject, usually providing basic information.
- GPT2 (Generative Pretrained Transformer 2) - GPT-2 is a large transformer-based language model with 1.5 billion parameters, trained on a large corpus of webpages.
- BERT (Bidirectional Encoder Representations from Transformers) - BERT is a pre-trained unsupervised natural language processing model.
- BLEU (Bilingual Evaluation Understudy Score) - It is a metric for evaluating a generated sentence to a reference sentence.

Smart FAQ and Response Generation System

ORIGINALITY REPORT

11%	8%	4%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Madanapalle Institute of Technology and Science Student Paper	2%
2	www.pes.edu Internet Source	2%
3	mineshmathew.github.io Internet Source	1%
4	www.engineering-admission.in Internet Source	<1%
5	Submitted to Ain Shams University Student Paper	<1%
6	www.ibm.com Internet Source	<1%
7	arxiv.org Internet Source	<1%
8	www.aclweb.org Internet Source	<1%
9	huggingface.co Internet Source	<1%

10	www.statmt.org Internet Source	<1 %
11	Submitted to University of Southampton Student Paper	<1 %
12	export.arxiv.org Internet Source	<1 %
13	Lecture Notes in Computer Science, 2008. Publication	<1 %
14	cs.uwc.ac.za Internet Source	<1 %
15	aclanthology.org Internet Source	<1 %
16	www.techrepublic.com Internet Source	<1 %
17	mullikine.github.io Internet Source	<1 %
18	www.djj.state.fl.us Internet Source	<1 %
19	Submitted to Indian Institute of Technology, Bombay Student Paper	<1 %
20	"Intelligent Information and Database Systems", Springer Science and Business Media LLC, 2021 Publication	<1 %

21	Shivani G. Aithal, Abishek B. Rao, Sanjay Singh. "Automatic question-answer pairs generation and question similarity mechanism in question answering system", Applied Intelligence, 2021 Publication	<1 %
22	Submitted to Turun yliopisto Student Paper	<1 %
23	Zheng, X., Z. Hu, A. Xu, D. Chen, K. Liu, and B. Li. "Algorithm for recommending answer providers in community-based question answering", Journal of Information Science, 2012. Publication	<1 %
24	www.trump.net.in Internet Source	<1 %
25	wwwmatthes.in.tum.de Internet Source	<1 %
26	repository.tudelft.nl Internet Source	<1 %
27	"Text, Speech, and Dialogue", Springer Science and Business Media LLC, 2021 Publication	<1 %
28	www.qasigma.com Internet Source	<1 %
29	Submitted to Birkbeck College	

Student Paper

<1%

30 seominjoon.github.io
Internet Source
<1%

31 Submitted to Indiana University
Student Paper
<1%

32 Lecture Notes in Computer Science, 2012.
Publication
<1%

33 Submitted to University of Sheffield
Student Paper
<1%

34 www.allied-telesis.co.jp
Internet Source
<1%

35 Submitted to City University
Student Paper
<1%

36 Ghadeer Mobasher, Lukrecia Mertova, Sucheta Ghosh, Olga Krebs, Bettina Heinlein, Wolfgang Mueller. "Combining dictionary- and rule-based approximate entity linking with tuned BioBERT", Cold Spring Harbor Laboratory, 2021
Publication
<1%

37 Submitted to Imperial College of Science, Technology and Medicine
Student Paper
<1%

38 Submitted to University of Sydney
Student Paper

		<1 %
39	scholars.bgu.ac.il Internet Source	<1 %
40	www.diplomarbeiten24.de Internet Source	<1 %
41	www.termpaperwarehouse.com Internet Source	<1 %
42	"Advances in Artificial Intelligence", Springer Science and Business Media LLC, 2019 Publication	<1 %
43	en.wikipedia.org Internet Source	<1 %
44	rajpurkar.github.io Internet Source	<1 %
45	www.coursehero.com Internet Source	<1 %
46	"Chinese Computational Linguistics", Springer Science and Business Media LLC, 2019 Publication	<1 %
47	"Knowledge Graph and Semantic Computing: Knowledge Graph and Cognitive Intelligence", Springer Science and Business Media LLC, 2021 Publication	<1 %

48	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2018 Publication	<1 %
49	Angelica Willis, Glenn Davis, Sherry Ruan, Lakshmi Manoharan, James Landay, Emma Brunskill. "Key Phrase Extraction for Generating Educational Question-Answer Pairs", Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale, 2019 Publication	<1 %
50	Ruqing Zhang, Jiafeng Guo, Lu Chen, Yixing Fan, Xueqi Cheng. "A Review on Question Generation from Natural Language Text", ACM Transactions on Information Systems, 2022 Publication	<1 %
51	baadalsg.inflibnet.ac.in Internet Source	<1 %
52	bspace.buid.ac.ae Internet Source	<1 %
53	ceur-ws.org Internet Source	<1 %
54	dokumen.pub Internet Source	<1 %
55	etheses.whiterose.ac.uk Internet Source	

<1%

56	git.legeox.net Internet Source	<1%
57	krisztianbalog.com Internet Source	<1%
58	link.springer.com Internet Source	<1%
59	tel.archives-ouvertes.fr Internet Source	<1%
60	www.entrance.net.in Internet Source	<1%
61	www.theseus.fi Internet Source	<1%
62	Alok Kumar, Aditi Kharadi, Deepika Singh, Mala Kumari. "Automatic question-answer pair generation using Deep Learning", 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), 2021 Publication	<1%
63	Kalyan Moy Gupta. "Taxonomic Conversational Case-Based Reasoning", Lecture Notes in Computer Science, 2001 Publication	<1%

64	Lecture Notes in Computer Science, 2015. Publication	<1 %
65	Miroslav Blšták, Viera Rozinajová. "Automatic question generation based on sentence structure analysis using machine learning approach", Natural Language Engineering, 2021 Publication	<1 %

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		