**Automatic Question Generator with Natural Language Processing**

Submitted in partial fulfillment of the requirements for the degree of

**T.E. Information Technology**

By

**Trishali Rao 191099**

**Lincy Rebello 191103**

**Mohmmad Yasir Khan 191056**

**Wahid Shaikh 201263**

Supervisor

**Dr. Nitika Rai**

Assistant HoD



**Department of Information Technology St. Francis Institute of Technology (Engineering College)**

**University of Mumbai 2019-2020**

**CERTIFICATE**

This is to certify that the project entitled **”Automatic Question Generator with Natural Language Processing”** is a bonafide work of **”Trishali Rao 191099, Lincy Rebello 191103, Mohmmad Yasir Khan ,Wahid Shaikh** submitted to the Univercity of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

**Dr. Nitika Rai**

**(Project Guide)**

**Dr. Prachi Raut Dr. Sincy George**

**(Head Of Department) (Principal)**

**Literature Review Report for B.E.**

This project report entitled ***”*Automatic Question Generator with Natural Language Processing*”*** by **Trishali Rao 191099, Lincy Rebello 191103, Mohmmad Yasir Khan ,Wahid Shaikh**is approved for the degree of ***Bachelors of Engineering in Information Technology***.

1. ——————————–

2. ——————————–

#### Date:

#### Place:

### I declare that this written submission represents my ideas in my own words and where others’ ideas or words have been included, I have adequately cited and ref- erenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsi- fied any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

——————–

(Signature) Student1 (PID)

——————–

(Signature) Student2 (PID)

——————–

(Signature) Student3 (PID)

——————–

(Signature) Student4 (PID)

#### Date:

In Natural Language Processing (NLP), Question Generation (QG) is a well-known task that involves generating understandable questions from an input text. There are many useful applications in AQG, especially in educational settings, to create quizzes or reading comprehension papers.

**Contents**

1. **Introduction 1**
   1. [Introduction to the domain of project 1](#_heading=h.gjdgxs)
   2. [Major Challenges in said domain 1](#_heading=h.30j0zll)
   3. [Motivation 1](#_heading=h.1fob9te)
   4. [Problem Statement 1](#_heading=h.3znysh7)
2. **Literature Review 2**
   1. [Existing Work 3](#_heading=h.2et92p0)
      1. [Literature review related to existing system/methodology 3](#_heading=h.tyjcwt)
      2. [Literature review related to existing algorithms 4](#_heading=h.3dy6vkm)
   2. [Gap identified 5](#_heading=h.4d34og8)
3. **Proposed Methodology 6**
   1. [Problem Formulation 6](#_heading=h.2s8eyo1)
   2. [Problem Definition 6](#_heading=h.17dp8vu)
   3. [Scope 6](#_heading=h.3rdcrjn)
   4. [Proposed Methodology 6](#_heading=h.26in1rg)
   5. [Proposed Algorithm 6](#_heading=h.lnxbz9)
   6. [Features of proposed System 6](#_heading=h.35nkun2)
4. **System Analysis 7**
   1. [Functional Requirements 7](#_heading=h.1ksv4uv)
   2. [Non-Functional Requirements 7](#_heading=h.44sinio)
   3. [Specific Requirements 8](#_heading=h.2jxsxqh)
   4. [Use-Case Diagrams 8](#_heading=h.z337ya)
5. **Analysis Modeling 10**
   1. [Data Modeling : ER Diagram 11](#_heading=h.3j2qqm3)
   2. [Activity Diagrams 11](#_heading=h.1y810tw)
   3. [Functional Modeling 12](#_heading=h.4i7ojhp)
      1. [DFD:Level 0 12](#_heading=h.2xcytpi)
      2. [DFD:Level 1 13](#_heading=h.1ci93xb)
   4. [TimeLine Chart 14](#_heading=h.3whwml4)

4.1 Use Case Diagram 9

* 1. Activity Diagram for User 11
  2. DFD Level 0. 12
  3. DFD Level 1. 13
  4. Time Chart Tasks. 14
  5. Gantt Chart From July 2018 - May 2019. 15
  6. Comparison of methodology 3
  7. Comparison of algorithms 4
  8. Comparison of tools/technology/framework 5

4.1 Use Case Diagram Description

**Chapter 1 Introduction**

# Introduction to the domain of project

Question generation (QG), aiming at generating questions from natural language text, e.g. a sentence or paragraph, is an essential area in natural language processing (NLP). It is received increasing interest in recent years from both industrial and academic communities, due to the booming of Question-and-Answer (QnA) and conversation systems, including Alexa, Cortana, Google Assistant, and Siri, the advancement of QnA or machine comprehension technologies together with the advancement in natural language processing models.

Natural Language Processing (NLP), which falls under machine learning, is the automatic computer manipulation of speech and text language expressions. An algorithm creates a question that is understandable to humans. As it shows a greater learning level for AI machines, from knowledge retention to understanding, AQG is becoming increasingly important in artificial intelligence (AI). Furthermore, AQG is a potential solution to the problems. For instance, AQG can create quiz questions for reading comprehension when the teacher only needs to provide the passage. Rule-based algorithms are used in the early efforts of AQG. The accuracy of these algorithms depends on the person writing them because linguistic and grammatical rules must be manually entered. Recent developments in NLP place a greater emphasis on transformer-based architectures, which can produce questions by "paying attention" to the critical components of a sentence and the language structure without the author entering profound inquiry rules of grammar. Using the answer-aware Recurrent Neural Network (RNN) described by Du et al.transformer design and worldwide recognition for AQG.

# Major Challenges in said domain

The educational systems of today require an effective instrument to assess students' understanding of the core concepts in the learning process. It takes teachers a lot of time to prepare a set of questions for an evaluation.

The selection of suitable target sentences and concepts is a major issue with AQG systems.

# Motivation

Many techniques have been studied for AQG, from rule-based algorithms to complex deep-learning networks. As machine learning advances, transformer-based neural networks are more vigorous and sturdy than rule-based logic in generating questions without prior knowledge of the grammar rules. However, present models still perform worse in paragraph-level texts than in sentence-level inputs.

# Problem Statement

To provide a solution to the problem of first breaking down the context into sub-word token strings. This project aims to automatically generate the question by giving the normal text. Instead of depending on human specialists to manually extract questions from study materials, it takes time and is at times tedious.

**Chapter 2 Literature Review**

# Existing Work

## Literature review related to existing system/methodology

Table 2.1: Comparison of methodology

| **Sr.**  **No.** | **Title of Paper** | **Review** | **Analysis/Limitations** |
| --- | --- | --- | --- |
| [1] | Simplifying Paragraph-level Question Generation via Transformer Language Models | Used a simple single Transformer-based question generation model it is also evaluated on BLEU 4 and ROUGE L metrics. It is trained on the question generation model on version 1.1 of the Stanford Question Answering Dataset (SQuAD) | As a result, they have very low BLEU 4 and ROUGE L scores. |
| [2] | Automatic Fill-the-blank Question Generator for Student Self-assessment | Proposed automatic question generation system for student self-assessment by leveraging. The immediate advantage is to quickly generate and edit questions for pop quizzes and worksheets from their lecture notes.  They used 3 parts to generate questions  1) Sentence Selection: Selecting coherent and important sentences from the text to ask about  2) Gap Selection: Identifying which part of the resulting sentence to choose as the gap. The gap essentially represents the concept being tested  3) Distractor Selection: Crafting effective distractors to be part of the set of options to confuse the learner to ensure that he has a good grasp of the concept being tested | Question Variety, a few students noted that there were repetitive questions tested on the same key concept. |
| [3] | Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks | Use of new sequence to sequence network which contains a gated self-attention encoder and a max out pointer decoder to address the answer-aware QG problem for long text input.  Use of SQuAD is also done. | There is the main limitation in their scores of BLEU 4, METEOR, and ROUGE L. |
| [4] | Neural Question Generation for Reading Comprehension | Incorporated SQuAD dataset wherein the questions are posed by crowd workers and are of relatively high quality. Used the sequence-to-sequence approach.  Then investigated the models using RNN encoder-decoder architecture.  For training and inferencing/predicting the output, they have used beam search.  In implementations, they have blended the model in Torch7 with OpenNMT.  Compared to different Baselines like IR, Seq2seq, H&S, etc. with packages/systems viz. Bleu1-4, Meteor, and Rogue. The resulting model could generate better quality questions than the H & S system. | The flaw in the model is that it only encodes sentence-level information and achieves the best performance across all the metrics and not in paragraph-level information that decreases the performance of QG systems regarding questions of all categories.  Incorporating mechanisms for other language generation tasks (e.g., copy mechanism for dialogue generation) in their model to further improve the quality of generated questions. |

## Literature review related to existing algorithms

Table 2.2: Comparison of algorithms

| **Sr.**  **No.** | **Methodology Used/ Title of Paper** | **Review** | **Analysis/Limitations** |
| --- | --- | --- | --- |
| [1] | AQG with Rule-based Algorithms | The early efforts on AQG mostly used rule-based algorithms, i.e., a complex set of grammar rules coded into a program. These rules break down a sentence into parts of speech (POS), identifying the subjects, objects, and verbs in the grammatical structure to form questions. Different rules are applied to different question types. For instance, the “Who” question is created by replacing the subject with “Who”, or the “What” question is made by replacing the object with “What”. Both Zerr (2016) [4] and Das et al. (2016) [5] proposed similar work, either detecting POS by their own logic rules or with another machine learning model. | Both studies highlighted that the accuracy depends significantly on the POS tagger, i.e., if the algorithm identifies sentence structures wrongly, the question generated is incomprehensible. |
| [2] | AQG with Transformers | Previous work by Lopez et al. [15] uses GPT for a non-answer aware AQG model.  T5 has different versions with varying sizes, but in this project, the author focuses on the T5-base that has 220 million parameters in total, with 12 layers (“blocks”) within each encoder and decoder, model embedding size of 768, feed-forward layers of 3072 dimensions and key-value vectors for the attention of 64 dimensions [15]. | Compared to GPT, T5 is a smaller model with text-to-text processing, making it more efficient in transfer learning. T5 was also trained on a large dataset (Colossal Cleaned Crawled Corpus), preventing overfitting for word embeddings.  As T5 is previously introduced on question answering (QA), the author can apply transfer learning on a question-answer dataset to train T5 for question generation. |
| [3] | AQG with Neural Networks | Many previous studies have leveraged the Seq2Seq model for AQG, including additional features on the encoder by Zhou et al. (2017) [7], or model AQG with answer-awareness by Zhao et al. (2018) [8] and Dong et al. (2019)[9].  The author[8] proposed a max-out pointer mechanism with a gated self-attention encoder to address the challenges of processing long text inputs for question generation. | A drawback of this Seq2Seq architecture is that when the input is a long sequence, there is a high chance that the initial context is lost along the series of RNNs due to the vanishing gradient problem.  During backpropagation, the chain rule applied to the derivatives of gradients will lead to the gradient decreasing in value and approaching zero. |

# Gap identified

1. Even when very advanced algorithms are used, the accuracy of receiving written questions as output is relatively poor.

2. In the existing system if the algorithm identifies sentence structures wrongly, the question generated is incomprehensible

3.

**Proposed Methodology**

# Problem Formulation

This project aims to comprehensively study different AQG models in long paragraphs to implement the most accurate model for AQG and fine-tune the same that generates decent questions for long texts.

# Problem Definition

To focus more on text2text transformer-based transfer learning models, which can emphasize the critical points of a sentence and the language structure to generate questions without the creator inputting profound grammar rules.

# Scope

The scope of the project is limited to the analysis of

* In our project, we solely place an emphasis on creating questions.
* Using a pre-trained model for Text2textGeneration.

# Proposed Methodology

**Data setup**

The T5 AQG model is trained on the Stanford Question Answering Dataset (SQuAD) version 2.0 [5]. The SQuAD contains numerous paragraph texts (“contexts”). Within each paragraph is a set of questions and answers related to the content of the contexts. There are, in total, 150,000 questions available in SQuAD 2.0, stored in JSON format.

However, the author only selects the first question for each paragraph for training. The context and answer are parsed into model inputs as a continuous body of texts, separated by a special token. Meanwhile, the question is parsed as the targeted value or "labels".

**Model Pipeline**

Figure 2 below shows the training pipeline of the model. In the training loop, the context, for example, a SQuAD paragraph about the singer Beyonce and the answer (“in the late 1990s”) are concatenated into a single text separated by a special “SEP” token. Both context and answer are then passed through a T5 tokenizer, which converts the texts into numerical vectors that the T5 model can understand. Then, T5 will predict the output vectors, which also can be de-tokenized back into text questions. Meanwhile, the given question, “When did Beyonce start becoming popular?” is also passed through a tokenizer.

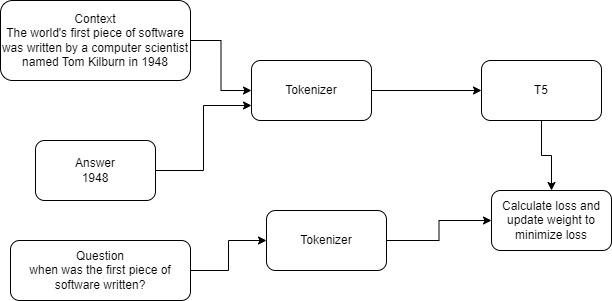


Figure 2

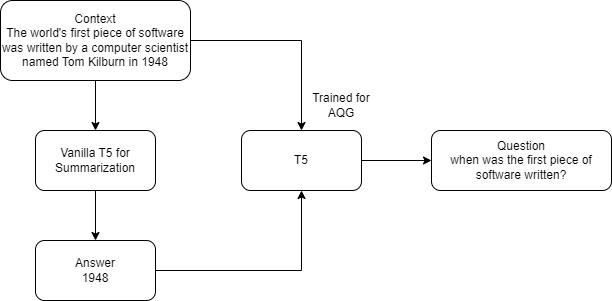


Figure 3

However, during inference (Fig 3), there will be no answer text given. As such, a helper vanilla T5 is used to generate the answer. The figure below shows the inference pipeline, where a pre-trained T5 is used to summarize the context. The summarized phrase (the "answer) and the context are treated as inputs for T5 for AQG. Finally, T5 generates questions based on the inputs. All T5 models used in this project are of version "t5-base" from Hugging Face[6].

# Proposed Algorithm

Early AQG efforts rely on rule-based algorithms[9]. These algorithms require manual input of linguistic and grammatical rules, so their accuracy depends on the person creating them. Current advances in NLP focus more on transformer-based architectures, which can “pay attention” to the critical points of a sentence and the language structure to generate questions without the creator inputting profound grammar rules. Du et al. (2017) [8] proposed an answer-aware Recurrent Neural Network (RNN) that relies on the transformer architecture and global attention for AQG.

# Features of the proposed System

The system accepts text passages as input that is subjected to tokenization, lemmatization, and stemming for pre-processing. Potential sentences are selected from these processed phrases with help of discourse markers which undergo syntactic analysis using POS tagging and semantic analysis using NER. Grammatically sound questions are formed using NER. Questions other than wh-questions (like true/false, MCQs, etc.) can be incorporated. incorporated. An Answer evaluation module can be integrated to evaluate and score the test answers submitted by students by calculating semantic similarity with the correct answer.

**System Analysis**

# Functional Requirements

* + - The user can give his/her own input paragraph or sentence.
    - There could be a key answer input Textarea where the user must be asked to give an answer
    - There could be an option to set a number of questions to be generated.
    - Clicking the generate button will take the input phrase to generate questions.
    - Generated questions will be displayed below the input Textarea

# Non-Functional Requirements

* + - Accessibility: The application should be accessible as a web application.
    - Usability: The application will be simple and user-friendly.
    - Maintenance: The model can be trained to increase accuracy in the future.
    - Acceptance: the system will be capable of generating state of art questions.
    - Responsive: The function response time should be smooth and must generate questions in minimum time.
    - Modifiable: The user interface should be modifiable.
    - User-Friendly Graphical Interface.

# Specific Requirements

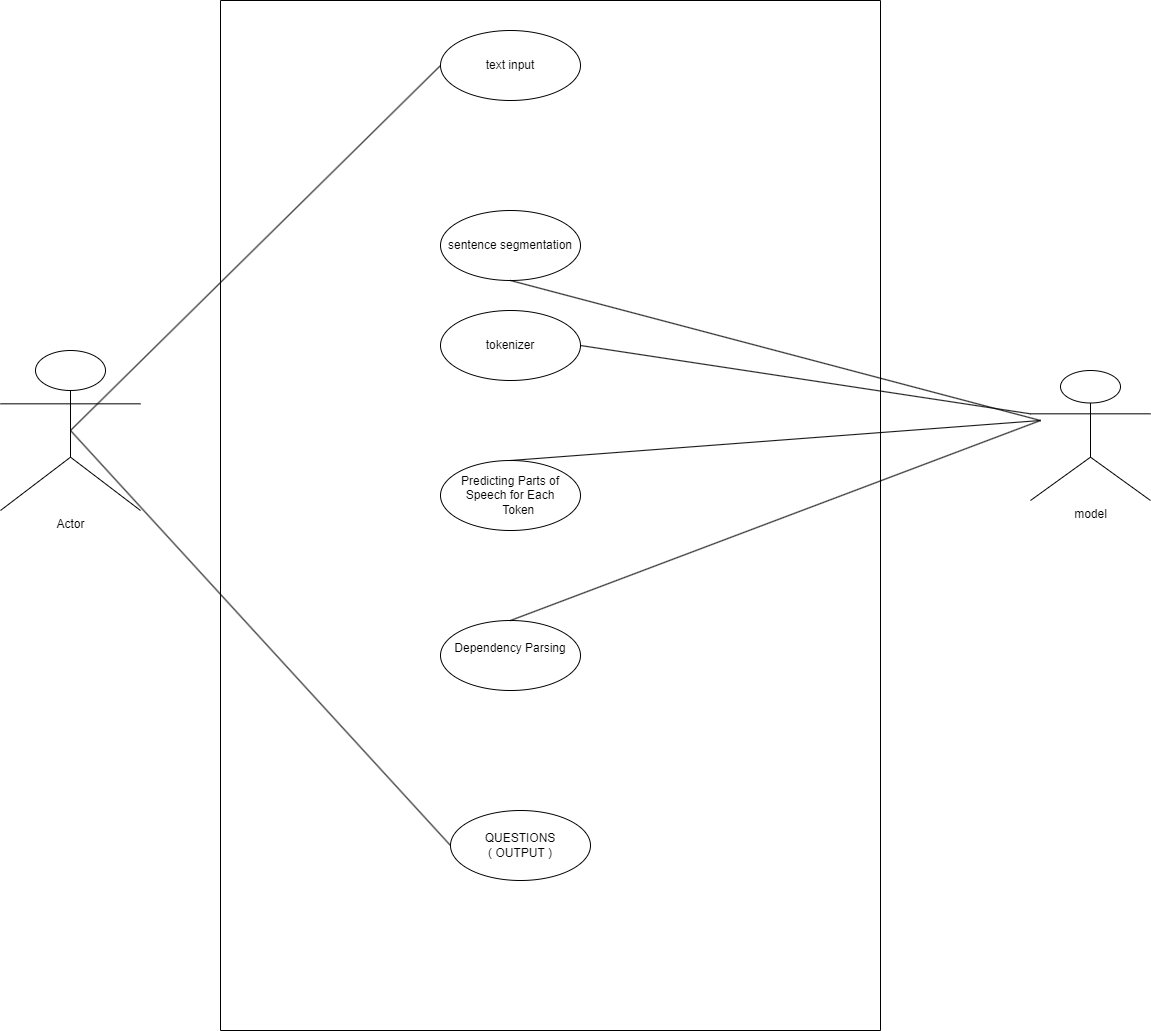
#### Hardware:

* + - CPU: 64 Bit Intel or AMD Processor.
    - GPU: Minimum 2GB Graphics Memory with DX10.
    - RAM: 8GB or above.
    - Memory: Minimum 10 GB for installation and additional project files.
    - Operating System: Windows 10 or above/ Linux.

#### Software:

* + - Visual Studio Code
    - Google Collab

# Use-Case Diagrams

Figure 4.1: Use Case Diagram

In the above diagram we show the overview of our project using a use case diagram where the actor is the user and he can enter the paragraph as input and he gets the question generated by the model as output. Model is the inside function of the project where the important words in the paragraph are selected based on the questions generated.

**Chapter 5**

**Analysis Modeling**

# Activity Diagrams

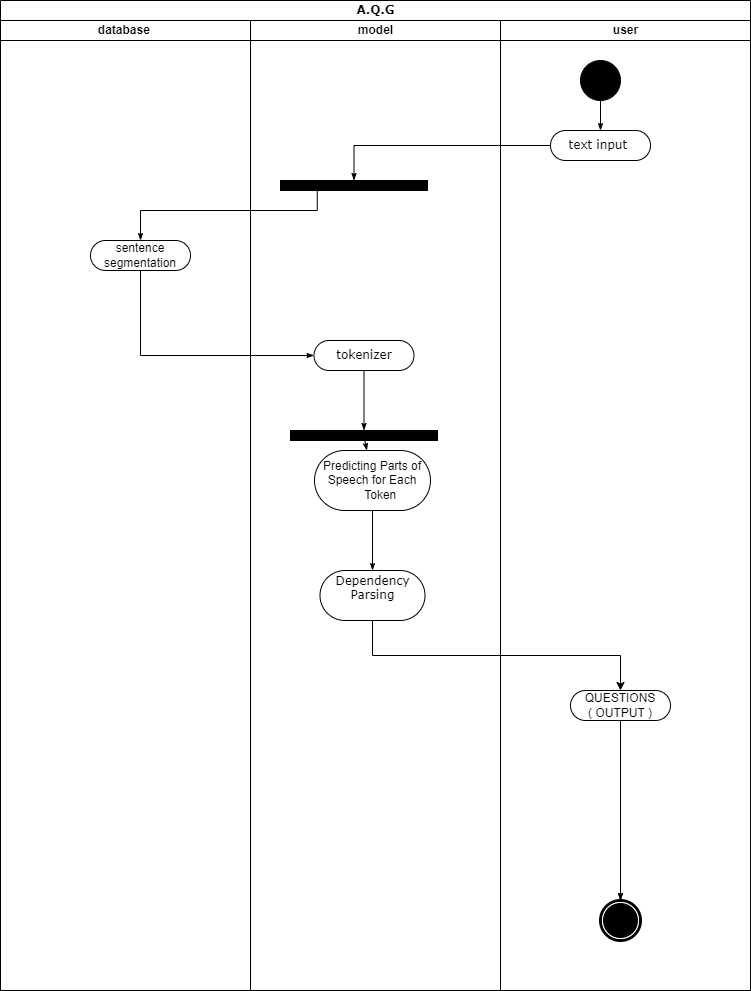
****

Figure 5.1: activity diagram

The above diagram shows the brief working of the project stepwise the input is taken from the user and then the input goes into the database for segmentation of sentence

and then the model selects the important words in the paragraph are selected based on that the questions are generated. And then generated questions are given as output to the user

# Functional Modeling

## DFD: Level 0

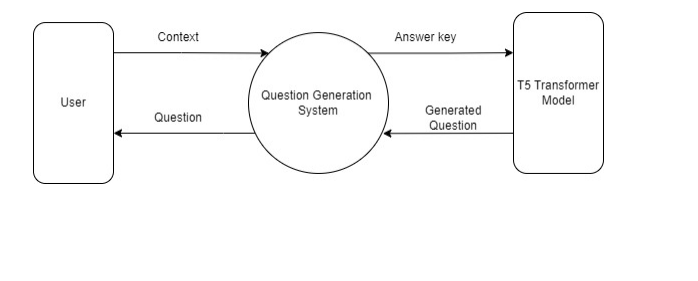
****

Figure 5.2: DFD Level 0.

## DFD: Level 1

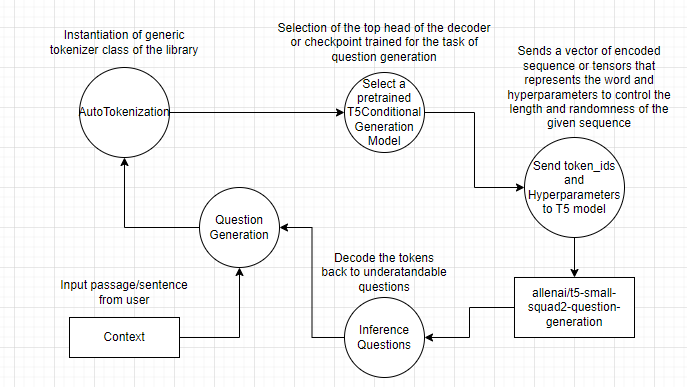
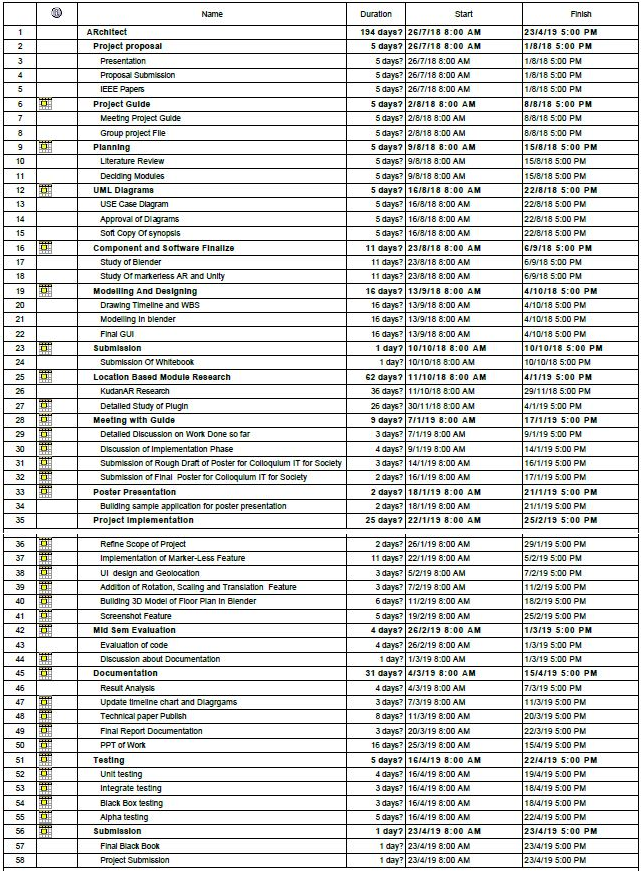
****

Figure 5.3: DFD Level 1.

# TimeLine Chart

Figure 5.4: Time Chart Tasks.

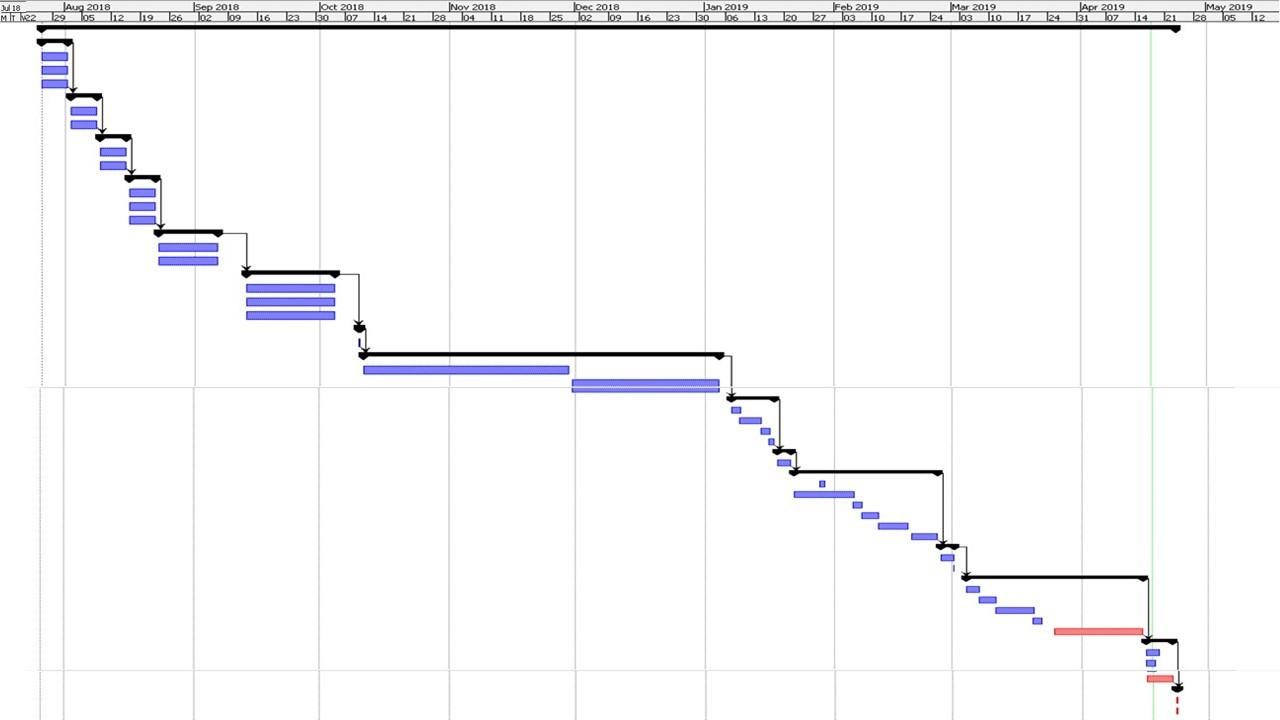


Figure 5.5: Gantt Chart From July 2018 - May 2019.

# Literature Cited

[1]Lopez, Luis Enrico, et al. "Simplifying paragraph-level question generation via transformer language models." *Pacific Rim International Conference on Artificial Intelligence*. Springer, Cham, 2021.

[2] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. [Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks](https://aclanthology.org/D18-1424). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

[3]G. Kumar, R. E. Banchs and L. F. D'Haro, "Automatic fill-the-blank question generator for student self-assessment," *2015 IEEE Frontiers in Education Conference (FIE)*, 2015, pp. 1-3, doi: 10.1109/FIE.2015.7344291.

[4] <https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/>

(accessed on 5/08/2022 2:45pm)

[5] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for SQuAD,” Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2018.

[6] “Crossentropyloss,” CrossEntropyLoss - PyTorch 1.11.0 documentation, 2019. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html. [Accessed: 23-Mar-2022].

[7] “T5-base · hugging face,” t5-base · Hugging Face. [Online]. Available: https://huggingface.co/t5-base. [Accessed: 23-Mar-2022].

[8] X. Du, J. Shao, and C. Cardie, “Learning to ask: Neural question generation for reading comprehension,” Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017.

[9] V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan, “A detailed account of the first question generation shared task evaluation challenge,” Dialogue & Discourse, vol. 3, no. 2, pp. 177–204, 2012.