PROJECT

Report

Project Title

Build Assessment Questions from Shared Stimuli

Abstract

In courseware business, creating Assessment Questions automatically from a given Shared Stimulus is in high demand. This will reduce the overall manual cost & effort significantly. Question answering is a very popular task in Natural language processing. Question generation has a lot of use cases with the most prominent one being the ability to generate quick assessments from any given content. We built an AI tool based on Text Mining that will have the capability to understand the shared stimulus and then generate different types of assessment content as per requirement.

Contents

1. Introduction

Question answering is a very popular task in Natural language processing.
Question generation has a lot of use cases with the most prominent one being the ability to generate quick assessments from any given content. Nowadays, teachers/professors/tutors (academicians) spend a lot of time generating test papers and quizzes manually.

The pandemic has made everything go digital and hence the need to generate digital questionnaires has become the need of the hour too. It will not only help teachers to generate questions for assignments or exams but will also reduce their burden by automating the task. By relying on the use of this technology, teachers can focus more on the teaching aspects and leave a part of their work for the algorithms to do.

So here in this project we built a model to automatically generate questions from shared stimuli. You can just simply pass in the content from which you would like to create the questions and select the type of questions you would like to generate, and the output will be generated for you. In our model we have created mainly four types of questions that you can generate. They include- Multiple Choice Questions, Fill in the blanks, short answer Questions and One word Answer questions. You can select all or any among those to get the desired results.

2. Background

Generating assessment questions is the one repetitive task that educators must do manually which takes up a lot of time. Apart from this the pandemic has brought about a drastic shift in the teaching methodology with everything shifting to online mode which creates more workload on the educators.

Apart from this with everything going digital and more and more content being generated every day the need to automate this task has become of utmost importance. The model built in this project will allow you to reduce the manpower, time and efforts spent on generating the questions manually for different types of content.

3. Key Benefits

- Helps in reducing time and efforts spent on generating questions manually
- The time and effort thus saved could be utilized on teaching the subject material properly
- With growing usage in online education, it becomes nearly impossible to manually frame questions for every text and hence the need for automation
- The question generator could be used to generate multiple type of questions depending on the user's choice
- Students can also use the model to generate questions to self-assess themselves

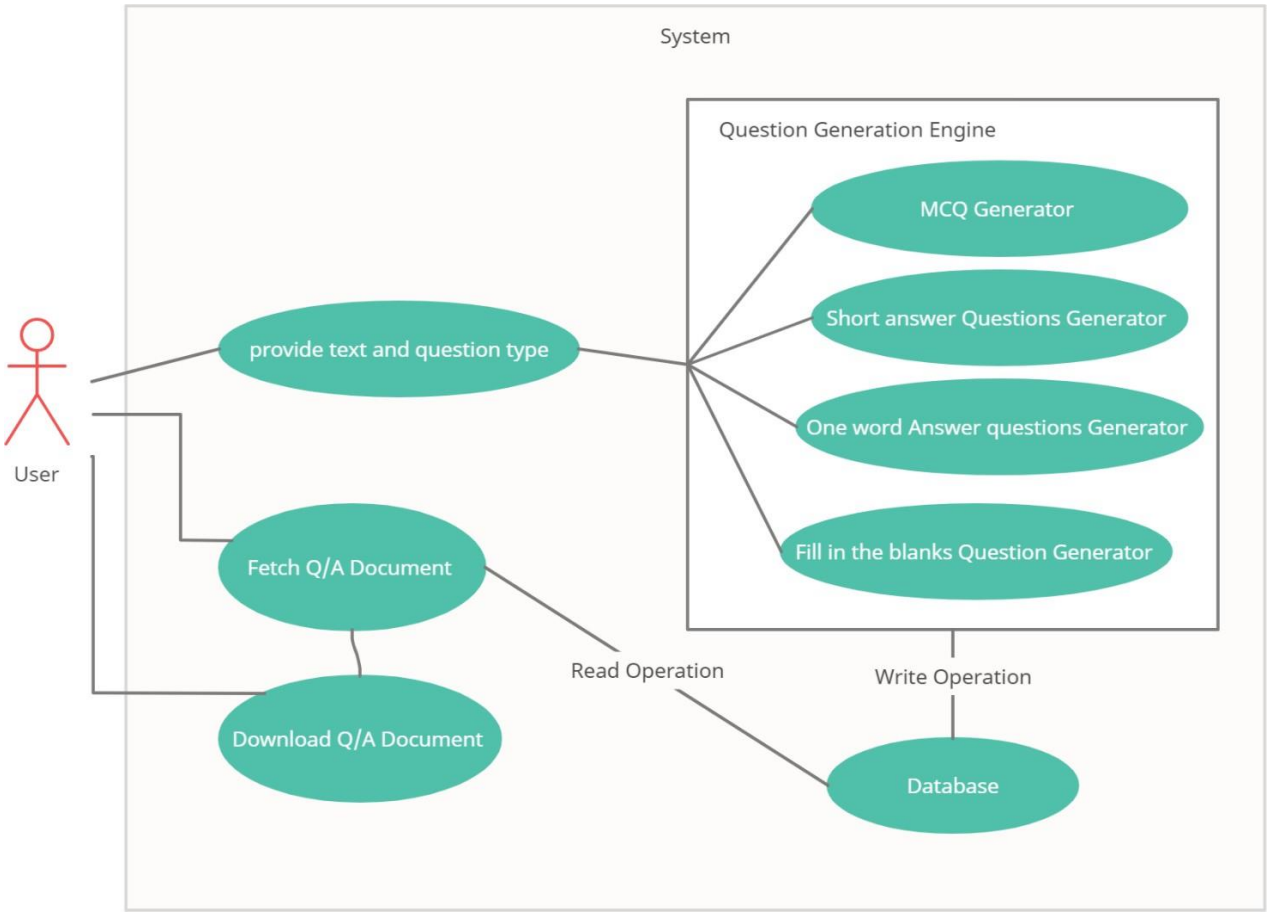
4. Design

We decided to build a Web Application, with a Server that manages API calls from the Web Application and the Question Generation Engine.

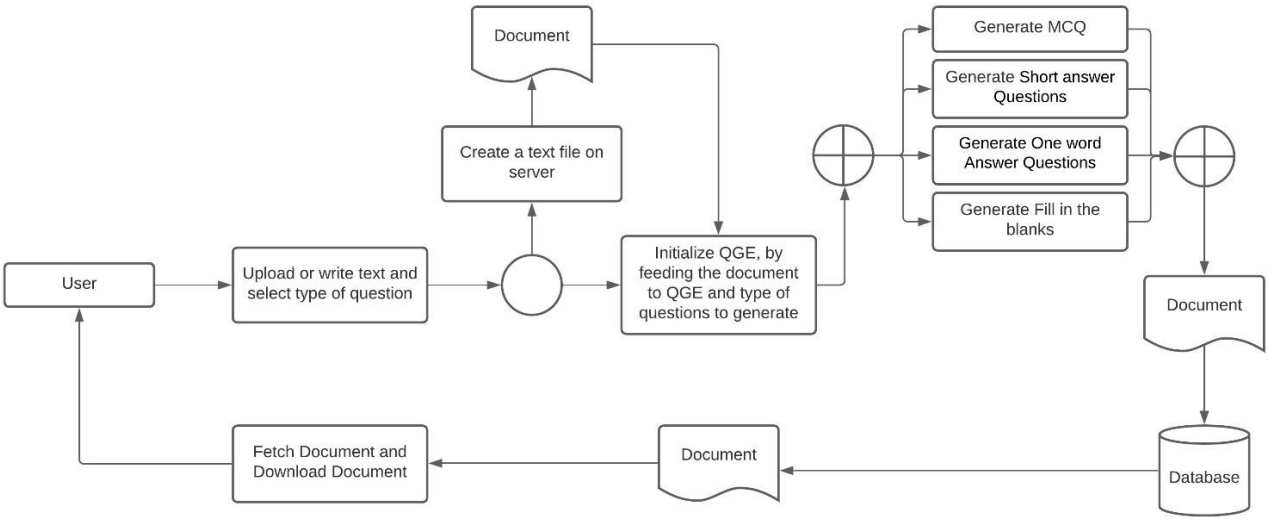
High Level Description:

1. User must write / upload a text document on the Web Application and choose the type of question that has to be generated. After clicking on the “Generate” button, an API call is made to the server. The API call prompts two things:
    a. If a file has been uploaded, it makes the sever save the text file in its file storage
    b. If text has been provided, it makes the server create a text file, write the content in it and name the file with cryptographical strong pseudorandom string of length 10.
2. The API call returns the file name to the Web App in response. The Web app shows the file name in pending section.
3. Then the server initializes the Question Generation Engine with 2 important parameters, the text file and the type of question that has to be generated. Then the model runs.
4. It starts predicting respective questions and answers of the type mentioned by the user and writes them in the database in form of a document.
5. As soon as a document is inserted in the database, server is informed about the change, it gets the document that has been inserted and sends it to the Web App.
6. The Web App then shows the status of the pending file as completed file.
7. User can now use the file name to fetch the document of Questions and Answers and download it on their local file system as a text file.

4.1 Use Case Diagram



4.2 Data Flow

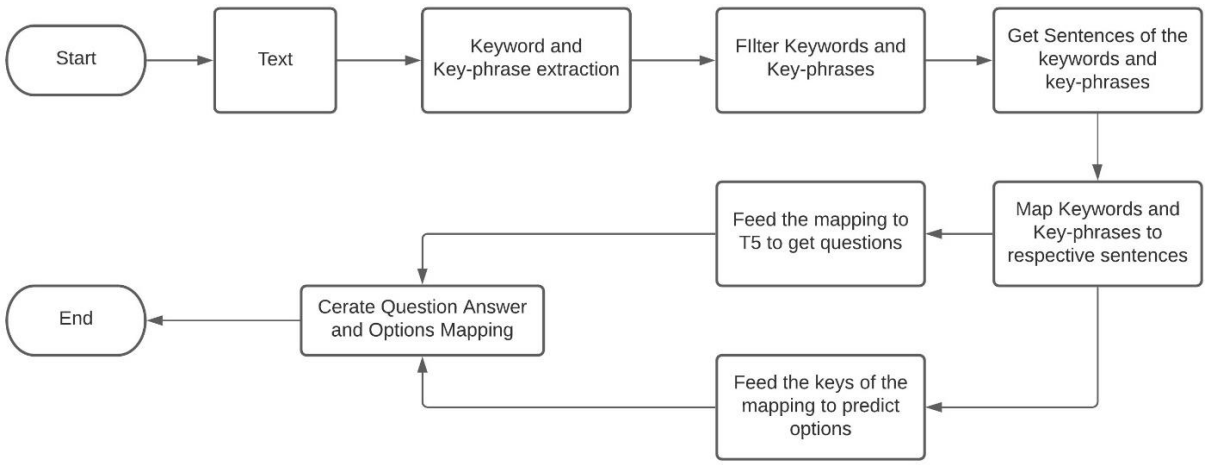

Document
Generate MCQ
Generate Short answer Questions
Generate One word Answer Questions
Generate Fill in the blanks
Create a text file on server
User
Upload or write text and select type of question
Initialize QGE, by feeding the document to QGE and type of questions to generate
Document
Fetch Document and Download Document
Document
Database

4.3 Question Generation Engine

The Question Generation Engine is made up of one main file, which controls the type of questions that have to be generated and 4 modules that generate 4 types of questions those are:

a. MCQ Generation Module
b. Paraphrase Questions Generation Module
c. Single Answer Questions Generation module
d. Fill in the blanks Questions Generation module

4.3.1 MCQ Generation Module



MCQ Generation is devised in 6 main steps:

a. Extract Keywords and Key-phrase from the Text:
   - The task of key-phrase extraction is to automatically identify a set of terms that best describe the document.

- We have used Graph Based Unsupervised Approach to identify the list of nouns and noun phrases and ranked them using Random Walk algorithm based on their bigram frequency (co-occurrence frequency).
- Thus, we get a list of keywords and key-phrases.
- After getting the list of keywords and key-phrases we filter some phrases unnecessary phrases based on normalized levenshtein. So, keywords and key-phrases that are closely related to each other remain in the list.
- The keyword and a key-phrase we get would be a correct answer.

b. Getting the Sentences of those Keywords and Key-phrases:
   - Here simply we get the sentences in which those keywords are present.
   - After this step we get a list of Sentences of respective keywords and key-phrases is returned.

c. Keyword and Key-phrase to Sentence Mapping:
   - Sentences are mapped to respective Keyword and Key-phrases, they together are mapped as a key-value pair (a python dictionary).
   - So now we have a Sentence-Answer Mapping.
   - We use this mapping to get Question-Answer-Option Mapping for MCQ Module.

d. Get Questions:
   - For each key-value pair of question-key, feed the Sentence-Answer Mapping to T5 transformer.
   - We have used T5 Transformer that has been trained on SQuAD2.0 Dataset (The Stanford Question Answering Dataset).
   - Upon feeding it with Sentence and Correct Answer, it formulated a question for the answer provided.
   - Thus, now we have a Question-Answer Pair.

e. Generate Options for the Keywords and Key-phrases:

- Now to generate options we used Sense2Vec.
- Sense2vec is a neural network model that generates vector space representations of words from large corpora.
- Here in our case this large corpus is the set of vectors of Reddit comments 2015.
- For example, for a keyword “Apple”, sense2vec produces 2 vectors in the following format:

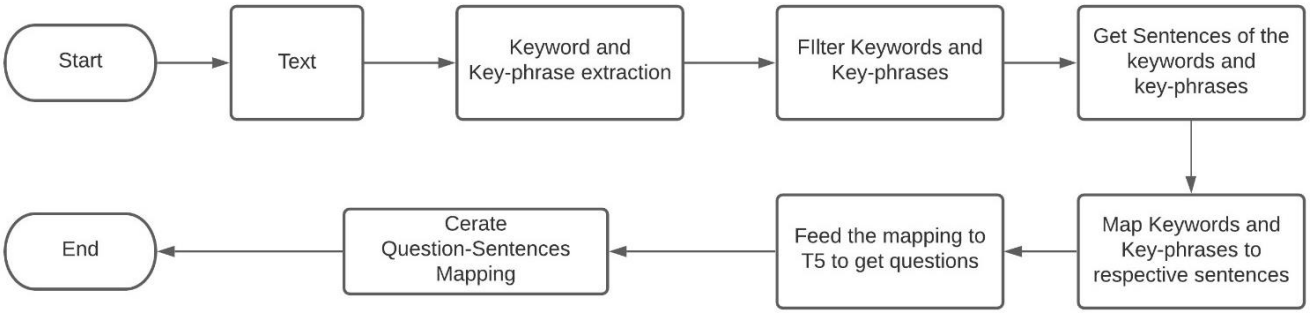
    a. Apple | PROPER NOUN When It is used as the name of the company
    b. Apple | NOUN When it is used as the name of a fruit.

- First, we detect the ‘sense’ (meaning) of the word or a phrase, for example for “Apple” we have 2 senses. We choose one of them and get words that make the same sense.
- If we choose, the sense of PROPER NOUN, we can generate most similar PROPER NOUNS using sense2vec’s in built functions. For “Apple” PROPER NOUN we get “Samsung”,” Google” and” Microsoft”.
- If we choose, the sense of NOUN, we can generate most similar NOUNS using sense2vec’s in built functions. For “Apple” NOUN we get “Orange”,” Banana” and” Mango”.
- So, like this we can generate options by feeding keywords and key-phrases to sense2vec.

f. Generate Questions-Answers-Options Mapping:

- Lastly create a Question-Answer-Options mapping (a python dictionary with 3 keys (Question, Answer, Options) and 3 respective values).

4.3.2 Short Answer Questions Generation Module



Short Answer Question Generation is devised in 5 main steps:

a. Extract Keywords and Key-phrase from the Text:
   - The task of key-phrase extraction is to automatically identify a set of terms that best describe the document.
   - We have used Graph Based Unsupervised Approach to identify the list of nouns and noun phrases and ranked them using Random Walk algorithm based on their bigram frequency (co-occurrence frequency).
   - Thus, we get a list of keywords and key-phrases.
   - After getting the list of keywords and key-phrases we filter some phrases unnecessary phrases based on normalized levenshtein. So, keywords and key-phrases that are closely related to each other remain in the list.

b. Getting the Sentences of those Keywords and Key-phrases:
   - Here simply we get the sentences in which those keywords are present.
   - After this step we get a list of Sentences of respective keywords and key-phrases is returned.

c. Keyword and Key-phrase to Sentence Mapping:
   - Sentences are mapped to respective Keyword and Key-phrases, they together are mapped as a key-value pair (a python dictionary).

- So now we have a Sentence-Answer Mapping.
- We use this mapping to get Question-Answer-Option Mapping for MCQ Module.

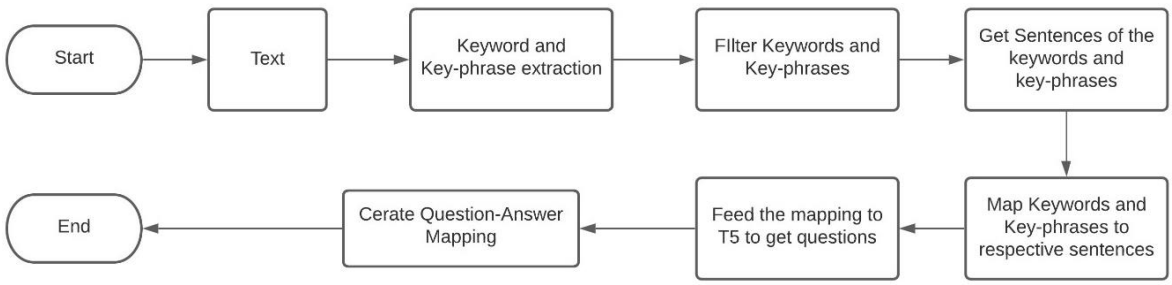
d. Get Questions:

- For each key-value pair of question-key, feed the Sentence-Answer Mapping to T5 transformer.
- We have used T5 Transformer that has been trained on SQuAD2.0 Dataset (The Stanford Question Answering Dataset).
- Upon feeding it with Sentence and Correct Answer, it formulated a question for the answer provided.
- Now we have a Question, Sentence and a keyword or a key-phrase.

e. Generate Questions-Sentences Mapping

- For all Questions and Sentence, make a mapping of Question-Sentence to get short answer questions.

4.3.3 One Word Answer Question Generation Module



One Word Answer Question Generation is devised in 5 main steps:

a. Extract Keywords and Key-phrase from the Text:

- The task of key-phrase extraction is to automatically identify a set of terms that best describe the document.
- We have used Graph Based Unsupervised Approach to identify the list of nouns and noun phrases and ranked them using Random Walk algorithm based on their bigram frequency (co-occurrence frequency).
- Thus, we get a list of keywords and key-phrases.
- After getting the list of keywords and key-phrases we filter some phrases unnecessary phrases based on normalized levenshtein. So, keywords and key-phrases that are closely related to each other remain in the list.
- The keyword and a key-phrase we get would be a correct answer.

b. Getting the Sentences of those Keywords and Key-phrases:
   - Here simply we get the sentences in which those keywords are present.
   - After this step we get a list of Sentences of respective keywords and key-phrases is returned.

c. Keyword and Key-phrase to Sentence Mapping:
   - Sentences are mapped to respective Keyword and Key-phrases, they together are mapped as a key-value pair (a python dictionary).
   - So now we have a Sentence-Answer Mapping.
   - We use this mapping to get Question-Answer-Option Mapping for MCQ Module.

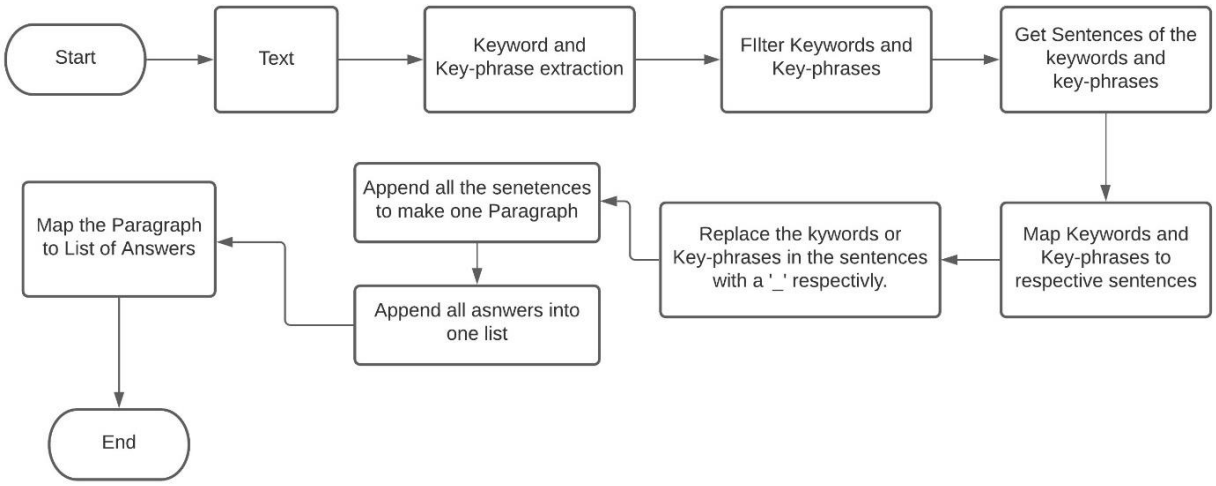
d. Get Questions:
   - For each key-value pair of question-key, feed the Sentence-Answer Mapping to T5 transformer.
   - We have used T5 Transformer that has been trained on SQuAD2.0 Dataset (The Stanford Question Answering Dataset).

- Upon feeding it with Sentence and Correct Answer, it formulated a question for the answer provided.
- Now we have a Question, Sentence and a keyword or a key-phrase.

e. Generate Questions-Sentences Mapping

- For all Questions and Answers, make a mapping of Question-Answer to get short answer questions.

4.3.4 Fill in the Blanks Question Generation Module



Fill in the Blanks Question Generation is devised in 5 main steps:

a. Extract Keywords and Key-phrase from the Text:

- The task of key-phrase extraction is to automatically identify a set of terms that best describe the document.
- We have used Graph Based Unsupervised Approach to identify the list of nouns and noun phrases and ranked them using Random Walk algorithm based on their bigram frequency (co-occurrence frequency).
- Thus, we get a list of keywords and key-phrases.

- After getting the list of keywords and key-phrases we filter some phrases unnecessary phrases based on normalized levenshtein. So, keywords and key-phrases that are closely related to each other remain in the list.
- The keyword and a key-phrase we get would be a correct answer.

b. Getting the Sentences of those Keywords and Key-phrases:

- Here simply we get the sentences in which those keywords are present.
- After this step we get a list of Sentences of respective keywords and key-phrases is returned.

c. Keyword and Key-phrase to Sentence Mapping:

- Sentences are mapped to respective Keyword and Key-phrases, they together are mapped as a key-value pair (a python dictionary).
- So now we have a Sentence-Answer Mapping.
- We use this mapping to get Question-Answer-Option Mapping for MCQ Module.

d. Replace Keyword and Key-phrases with "_":

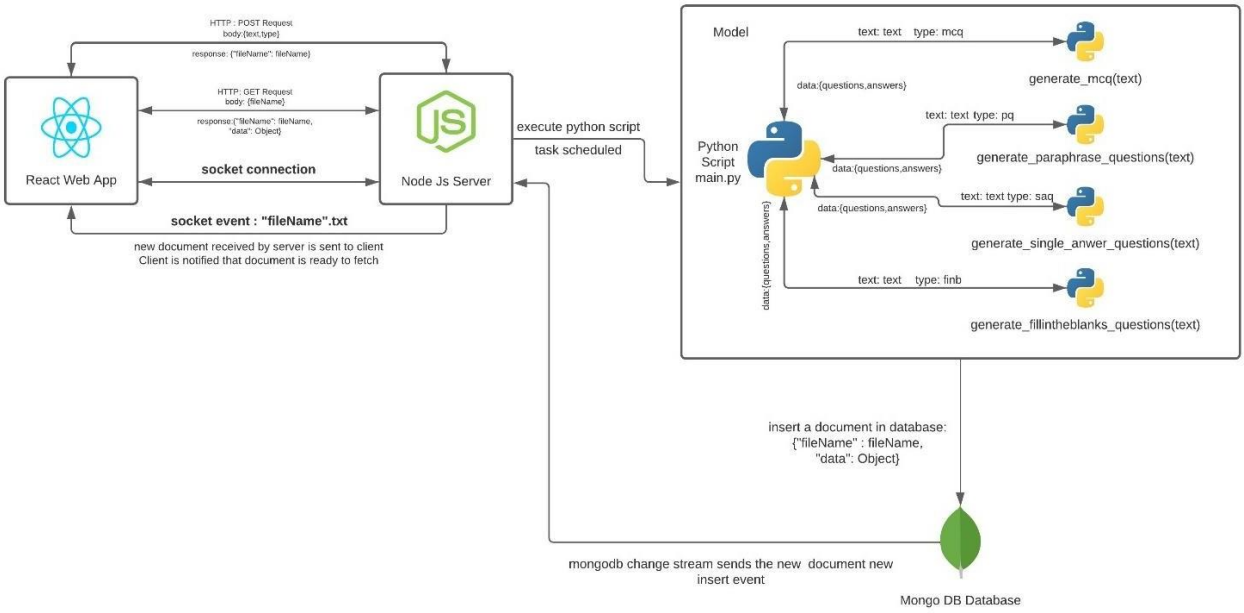
- In each sentence replace keyword or key-phrase with a blank "_".
- This would give us a question of fill in the blank type.
- Join all the sentences to generate a paragraph.
- Make a list of all the Keywords and key-phrases in order of the appended sentences.

e. Map paragraph and list of answers:

- Lastly map the Paragraph and List of answers as key value pairs.

5. Implementation

5.1 System Architecture



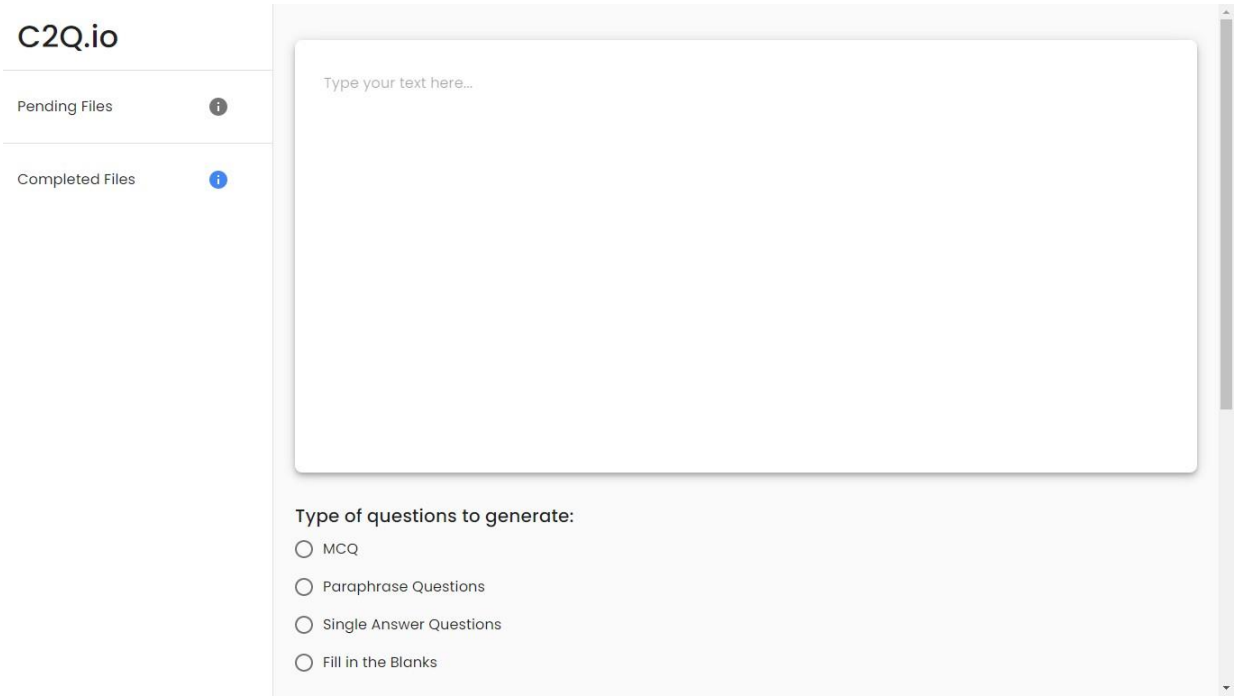
5.1.1 Front-End

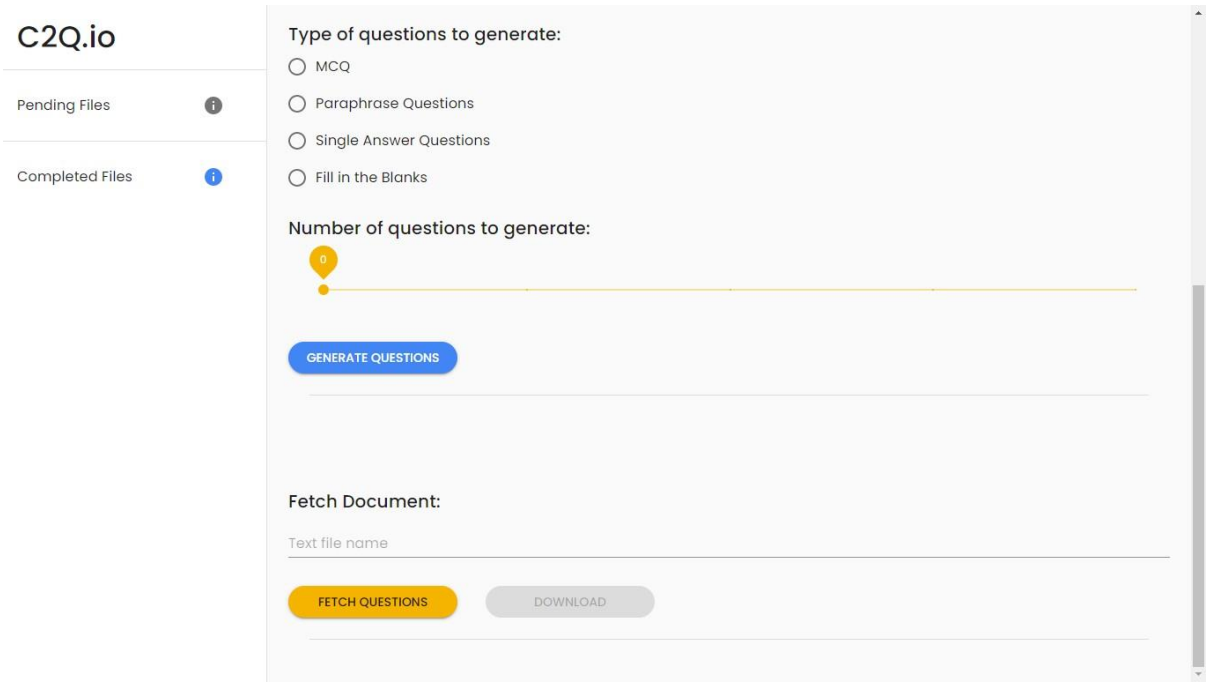
We used React Js to build a Web Application.

- Dependencies:

```
"@material-ui/core": "^4.11.4",
"@material-ui/icons": "^4.11.2",
"@material-ui/lab": "*",
"@testing-library/jest-dom": "^5.14.1",
"@testing-library/react": "^11.2.7",
"@testing-library/user-event": "^12.8.3",
"react": "^17.0.2",
"react-dom": "^17.0.2",
"react-scripts": "^1.1.5",
"socket.io-client": "^4.1.3",
"web-vitals": "^1.1.2"
```

- File Structure:

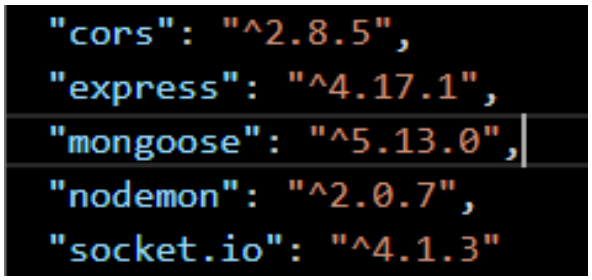

C2Q.io
Pending Files
Completed Files
Type your text here...
Type of questions to generate:
MCQ
Paraphrase Questions
Single Answer Questions
Fill in the Blanks


C2Q.io
Pending Files
Completed Files
Type of questions to generate:
MCQ
Paraphrase Questions
Single Answer Questions
Fill in the Blanks
Number of questions to generate:
0
GENERATE QUESTIONS
Fetch Document:
Text file name
FETCH QUESTIONS
DOWNLOAD

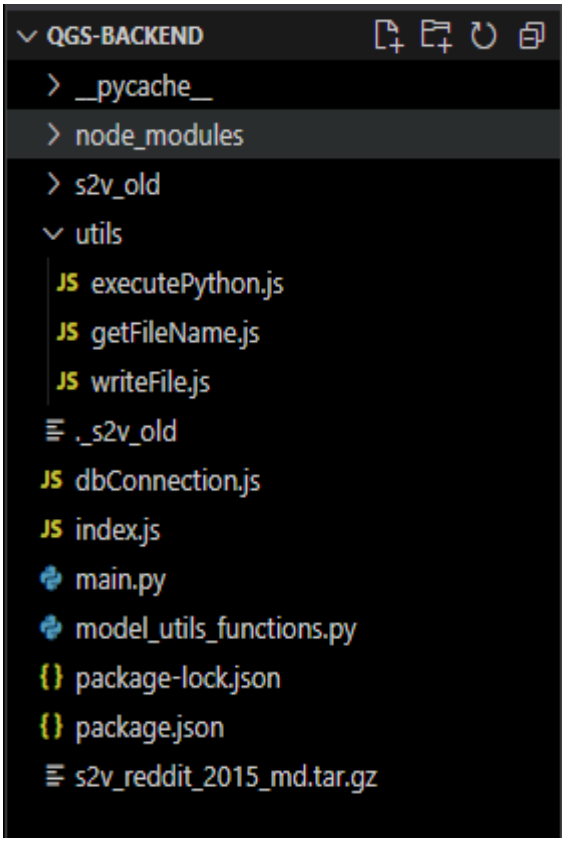
5.1.2 Back-End

For building back-end we used Node Js as a server.

- Dependencies:



- File Structure



5.1.3 Database

MongoDB as No-SQL database to store documents.

- Database Details:

a. Database Name: c2q.io

b. Collection Name: qaDocs

c. Schema:

```
_id: Unique MongoDB Id
filename: String
data: [
        {
                question: String,
                answer: String,
                options: []
        },
        .
        .
        .
        ]
```

5.1.4 Question Generation System

The QGE consists of 4 modules. Each module, before working on question generation, works on preprocessing the text data and identify necessary keywords and key-phrases from the data. After which the flow to generate questions changes depending upon the type of questions that need to be generated.

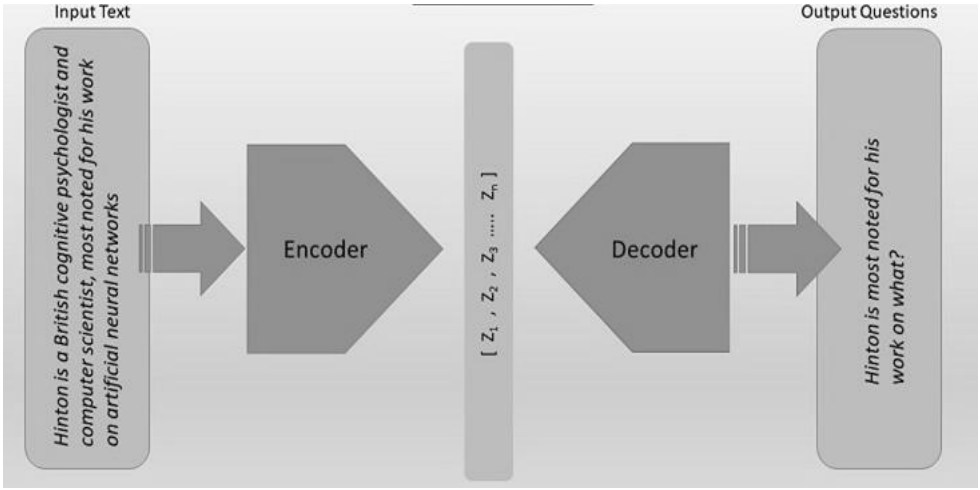
1. Preprocessing the Text:
   - In preprocessing, we tokenize sentences using sentence tokenizer *sent_tokenize () method* of the nltk library.
   - After tokenizing the sentences, we join them to make a new paragraph.
   - This new paragraph is fed to *get_keywords ()* function, to identify keywords and key-phrases in the paragraph. The keywords and key-phrases that we considered to select were NOUNS and NOUN PHRASES.
   - We used *pke. unsupervised.MultipartiteRank()* method from Python Keyword Extraction Library to build a Multpartite Graph to identify the list of nouns and noun phrases and ranked them using Random Walk algorithm based on their bigram frequency (co-occurrence frequency).
   - We chose top 10 best nouns or noun-phrases as our keywords or Key-phrases.
   - These keywords and key-phrases become the "Answer" part of our output.

2. Getting the Sentences:
   - We wrote a simple function *get_sentences_for_keyword (keywords, sentences)* to get all the sentences in which the keywords and key-phrases are present.
   - Map the Keywords and key-phrases to respective sentences.

3. Generate Questions:

   The question generation technique remains same for Multiple Choice Questions, Short answer Questions and One word Answer questions. It slightly changes for Fill in the blanks Question.

   - For MCQ, SAQ, OWAQ:
     a. Sentence-Answer Mapping generated in previous step are fed to T5 transformer to generate Questions.
     b. The T5 transformer model we used *t5_squad_v1* which was trained on SQuAD Dataset, which is a dataset of Pairs of Question and Answer.
     c. The model generates questions for us by providing in the Question-Answer Pair.
     d. Transformers use Encoder-Decoder Nets.
     e. Encoder-decoder nets have been used by Google for its neural machine translation (language translation) and recurrent neural networks. By keeping encoder-decoder at the core, we also take help from Stanford Parser and NLTK for grammar analysis and more basic natural language analysis.
     f. On a high level it works as shown in diagram below,



     g. This image shows how encoder-decoder network works internally.

h. The encoder takes a preprocessed sentence from the input text and converts it according to the weights of the hidden layer. This hidden layer creates an intermediate representation of the input text and passes it to the decoder.
i. The decoder converts the hidden-layer information into question form. Machine translation uses the same concept. Here, we treat questions, essentially, as another language.
j. Now based on Question type make the mapping.

- For Fill in the blanks:
  a. For fill in the blanks, we are not using transformers.
  b. Simple take the Sentence-Answer mapping and replace Answers with a blank ("_ ") in their respective sentences.
  c. Now append all the new sentences with blanks to make a Paragraph.
  d. Append all answer in a list called Answer List.

4. Create Mappings:
   - For MCQ:
     a. Create Questions-Answer-Options mapping.
   - For SAQ:
     a. Create Question-Sentence mapping.
   - For OWAQ:
     a. Create Questions-Answer mapping.
   - For Fill in the blanks:
     a. Create Paragraph-Answer List mapping.

5.2 System Requirements

Minimum system requirements for our model are as follow:

5.2.1 Hardware

- 8 GB Ram
- GPU with CUDA cores (for better model performance)
- i5 intel core CPU

5.2.2 Software

- Python- 3.7.1
- Transformers – 3.9.0
- PyTorch GPU Supported
- Spacy
- Nltk
- Python Keyword Extractor
- Pymongo
- Sense2vec
- Express Js

5.3 Improvements

The scope of further improvement lies in our system.

- The problem of Word Sense Disambiguation can be resolved to find the correct contextual sense of the chosen words and thereby generating effective and more precise questions and distractors for the word.

- An alternate strategy could be to first summaries a long text effectively so as it takes all the important lines from the text to be made into questions and then frame questions out of it.

- The computational power required by the system is also high and it takes around 45 seconds to generate the output. This is due to the underlying neural network architecture of the T5 model used in our system.

6. Conclusions

Our system can be used in multiple self-analysis scenarios. For example, students can use it to make learning easier as well as more interactive and interesting. Teachers and professors can use this system to quickly create a quiz. A central examination board can use this system to generate a unique test that is not known to any professor, eliminating the possibility of cheating, and thereby securing the privacy and integrity of the examination. The question generator system not only saves time and resources but also increases the efficiency of the teacher to focus on other aspects of teaching. In courseware business, creating Assessment Questions automatically from a given Shared Stimulus is in high demand. This will reduce the overall manual cost and efforts significantly.

7. Bibliography

- https://datasciencemilan.medium.com/question-generation-using-nlp-95e21a42b51e
- https://www.slideshare.net/Insitute_of_Contemporary_Sciences/extracting-keywords-from-texts-sanda-martincic-ipsic
- https://spacy.io/models
- https://github.com/boudinfl/pke
- https://www.youtube.com/watch?v=0DzcUXkn9HY&t=1579s
- https://huggingface.co/transformers/
- https://rajpurkar.github.io/SQuAD-explorer/