AUTOMATIC QUIZ GENERATOR

P Ganesh Patro¹, Aquib Baig¹, Ankit Jena¹, Jyotirmayee Rautray²

Abstract- Automatic quiz generation tackles the problem of generating questions from free-form texts. In this project, we seek to tackle the complex and interesting problem of question generation for the purpose of enhancing the educational experience of students. Learning through evaluation has been widely known to be a very effective method of assessing a student’s knowledge. Nowadays, academicians spend a lot of time generating test papers and quizzes manually and students spend a lot of time for feedback. We wanted to build a system that can help you assess yourself by giving the input as a text of whatever material, and on this basis, they get a set of questions with answers. A similar approach can be used by mentors for creating test papers and quizzes. By making this in a computerized application, we can reduce the task of an educator. Much time can be saved if we can know what appropriate questions can be asked for the given input of text. Hence, we want to develop a system that can generate various logical questions from the given text input. We deal with this problem using three key steps starting from sentence selection, gap selection, and question formation. Finally given an answer, we generate a list of certain discriminators using word2vec to confuse the candidate regarding similarity space of the answer.

Key Words: NLP, word2vec, sentence selection, gap selection, question generation.

I.INTRODUCTION

One of the most important uses of questions is reflection, improving our understanding of things we have found out. Wikipedia, blogs, etc have become a growing trend in recent years, however, the resources to test mastery of learned concepts are severely lacking. People often spend hours by themselves contemplating ideas and working through issues raised by what they have read. These ideas and issues are often articulated in the form of questions. Online examinations have become very popular, including examinations, such as GATE, CAT, and NET. Multiple Choice Questions (MCQ) is very easy for evaluations, and its evaluation is implemented through computerized applications so that results can be declared within a few hours, and the evaluation process is 100% pure. Questions are used from the most elementary stage of learning to original research. In the scientific method, a question often forms the basis of the investigation and can be considered a transition between the observation and hypothesis stages. Students of all ages use questions in their learning of topics, and the skill of having learners creating “investigatable” questions is a central part of inquiry education. Question Generation (QG) is that the task of mechanically generating queries from numerous inputs like raw text, database, or linguistics illustration. though automatic QG may be approached with numerous techniques, QG is largely thought to be a discourse task involving the subsequent four steps: (1) When to raise the question, (2) What the question is concerning, i.e. content choice, (3) Question kind identification, and (4) Question construction.

Automatic question generation is part of Natural Language Processing (NLP). It is an area of research where many researchers have presented their work and is still an area under research to achieve higher accuracy. Many researchers have worked in the area of automatic question generation through NLP, and numerous techniques and models have been developed to generate the different types of question automatically. Work has been done in many languages.

Question Generation (QG) aims to create natural questions from a given sentence or paragraph or paragraphs. NLP (Natural Language Processing) is an area of research and application that explores how

¹ Student, 8th Semester, Dept. of CSE, College of Engineering and Technology, Bhubaneswar, Odisha, India
² Lecturer, Dept. of CSE, College of Engineering and Technology, Bhubaneswar, Odisha, India
computers can be used to understand and manipulate natural language text or speech to do useful things. Our aim is to create a technique which can generate various logical questions from the given text input. Right now, only humans are capable of accomplishing this with higher accuracy.

The goal of the question generation is to generate questions according to some given information (e.g., a sentence or a paragraph). It has been applied in many scenarios, e.g., generating questions for reading comprehension [(1) Duan et al., 2017] and generating data for large-scale question-answering training [(1) Duan et al., 2017]. Since questioning is an important communication skill, question generation plays an important role in both general-purpose Chatbot systems and goal-oriented dialogue systems. In the context of dialogue, many researchers have studied the problem [(2) Mostafazadeh et al., 2016; (3) Bordes & Weston, 2017]. The generated questions are mainly used to start a conversation or to obtain some specific information.

These generated questions should preserve the context of the input. Generating questions this way reduces the work of a question setter who has to go through entire paragraphs to get a meaning out of it and set questions based upon that. Automatic question generation is smart enough to automatically generate a bunch of dissimilar yet confusing options to the examiner. It can also be used to generate MCQ’s which are one of the most abundantly used question formats in tests.

II. LITERATURE REVIEW

[2] Thalheimer, W. (2003). The learning benefits of questions :- This report reviews research on the learning benefits of questions from the preeminent refereed journals on learning, memory, and instruction. The research shows that questions can produce significant learning and performance benefits, potentially improving learning by 150% or more. Although traditionally used in quizzes, tests, and exams as mechanisms for assessment, questions make their most profound contributions when they are designed specifically to produce learning.
[3] Chen, CY, Liou, HC, Chang, JS. (2006). Fast: an automatic generation system for grammar tests:- This paper introduces a method for the semi-automatic generation of grammar test items by applying Natural Language Processing (NLP) techniques. Based on manually-designed patterns, sentences gathered from the Web are transformed into tests grammaticality. The method involves representing test writing knowledge as test patterns, acquiring authentic sentences on the Web, and applying generation strategies to transform sentences into items. At runtime, sentences are converted into two types of TOEFL-style questions: multiple choice and error detection.
[4] Kitchenham and Charters (2007); Boland et al. (2013) :- The Systematic Literature Review (SLR) methodology has been used for this study to identify the relevant data sources for the identification of factors influencing knowledge sharing in Multiple Choice Results to minimize the generation of waiting waste. The SLR methodology is a systematized and well-organized approach to attain less impartial results.
[5] Alsubait, Rakangor and Ghodasara (2015):- This paper characterised AOG studies along the following dimensions: 1) purpose of generating questions, 2) domain, 3) knowledge sources, 4) generation method, 5) question type, 6) response format, and 7) evaluation.
[6] Kurdi, G., Leo, J., Parsia, B. et al. A Systematic Review of Automatic Question Generation for Educational Purposes. Int J Artif Intell Educ (2019) :- This paper reviews various approaches made to the journey for Automatic Question Generation (AOG). This review extends a previous review on AOG literature that has been published up to late 2014. It includes 93 papers that were between 2015 and early 2019 and tackle the automatic generation of questions for educational purposes. The aims of this review are to: provide an overview of the AOG community and its activities, summarise the current trends and advances in AOG, highlight the changes that the area has undergone in the recent years, and suggest areas for improvement and future opportunities for AOG.
This paper describes details of the evaluation experiments for questions created by an automatic question generation system. Given a target word and one of its word senses, the system generates a multiple-choice English vocabulary question asking for the closest in meaning to the target word in the reading passage.

III. PROPOSED SYSTEM

Through this paper, we have tried to find a solution to automating question generation for examinations. Automated machine learning, also referred to as automated ML, is the process of automating the time consuming, iterative tasks of machine learning model development. It allows data scientists, analysts, and developers to build ML models with high scale, efficiency, and productivity all while sustaining model quality. In our proposed system, we find the solution to Automatic Question Generation (AQG) from the inside-out. The logic behind generation of a valid question is that the question must have a logically accepted answer and back tracking an answer from a question should be possible, given the required contexts. To find an answer, we parse sentences from the document text first and from that sentence, we find the most weighted words which can also be thought of as words of significance or keywords. The collection of such keywords is taken into a consensus function where an answer is selected. Once an answer has been found, we move on to Question Generation (QG) which has a generation method that transforms the original sentence for example, removing the answer from it. As reviewed by Thalheimer, W. (2003), multiple questions improve the learning capability by 150%, hence to generate them we need to find words whose meanings lie in close proximity to the answer, with the help of a discriminator function. This model includes three simple steps: sentence selection, keyword selection, and question formation. There has been quite a few discussion around how to generate multiple choices around common nouns. The above processes become largely complex if we add this scenario to the picture cause it is really hard to associate similar meanings to common nouns such as places and names. However, we do not need to consider these keywords in our tokenizer, because even if we found out the similar meanings of these words according to the context, deducing the actual answer from the passage won’t be a difficult task as a whole. For multiple questions, it would be a good practice to skip obvious nouns.

A. Sentence Selection

The sentence selection part is the first step of the model. We are using Text summarization as a method to select sentences with more frequent words so that returned sentences have more weight. Note that we are removing all stopwords in the preparation of the text, so occurrences of common English language words are removed by default. The summarization returns the more important sentences out of the text.

Pseudocode/Algorithm

1. Preprocess the text using the re package
   \[
   \text{re.sub}('\\[[0-9]*\\]', '', \text{article\_text})
   \text{re.sub}('\\s+', ' ', \text{article\_text})
   \]
2. Convert text into sentences using nltk.tokenize
   \[
   \text{nltk.sent\_tokenize}(	ext{article\_text})
   \]
3. Find stopwords like articles, prepositions etc. using the nltk package
   \[
   \text{nltk.corpus.stopwords.words('english')}
   \]
4. Build the frequency table whilst not taking into account the stopwords
5. Calculate sentence scores according to the word frequency table.
6. From the sentence scores calculated, build the summary using \( n \) sentences with highest score.
   \[
   \text{heapq.nlargest}(10, \text{sentence\_scores}, \text{key}=\text{sentence\_scores.get})
   \]
IV. TEXT SUMMARISATION

As information explodes on the Internet, it’s hard for users to read through all the published materials potentially interesting. Therefore, it is of great help to present them in a condensed way, i.e. using extracts or abstracts that generalize the content of an article. Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks). The problem with present summarization systems is that they produce one uniform summary for a given document without considering users’ opinions on it, while different users may have different perspectives on the same text, thus need a different summary. A good summary should change corresponding to the interests and preferences of a person. We refer to the adaptation of the summarization process to a particular user as personalized summarization. Text summarisation is a major step in sentence selection because it is not feasible to select every sentence in the paragraph as a candidate to question generation. The paper[1] shows us stages of summarisation as follows:

• **Step 1: Tokenization**: The main task here is to break the text into sentences. Tokens in the input text are also identified.
• **Step 2: Preprocessing**: This typically involves part- of-speech tagging and syntactic parsing. This step is optional; some systems do not perform tagging and parsing at all. Topic segmentation is deployed by some summarization systems, but not many.
• **Step 3: Extraction**: This is the main step in summarizing, in which the automatic summarizer selects key sentences (sometimes paragraphs or phrases) to include in the summary.
• **Step 4: Editing**: Some systems post-edit the extracted sentences to make them more coherent and concise.

*How did we summarise paragraphs?*

The raw paragraph that comes to the application as a data stream is summarised with the help of summarize. There are two types of text-summarisation: abstractive and extractive. We have used the extractive form because the former would be really hard to associate with the input article. In theory, This approach weights the important part of sentences and uses the same to form the summary. Different algorithms and techniques are used to define weights for the sentences and further rank them based on importance and similarity among each other.

*Input document → sentences similarity → weight sentences → select sentences with higher rank.*

Pseudocode

```python
summarized_text, summary_sentences = summarize(article)
```

B. **Keyword Selection**

The Keyword Selection part is the second step of the model. From the summarized sentences we find out the important keywords for each sentence. This is done by extracting information from the text which returns keywords and phrases for each sentence. For an individual sentence more than one keyword might be found. So we are choosing the most prevalent using the previously calculated word frequency table. Next we feed the sentence and the selected keyword to the question generation part.

Pseudocode

1. For each sentence in summarized sentences, do
2. Create a textblob of the sentence using a noun phrase extractor
   
   ```python
   blob=TextBlob(sentence, np_extractor =extractor)
   ```
3. Find out the most important phrase using word frequency table
   
   ```python
   important_word=select(blob.noun_phrases)
   ```

V. NOUN PHRASE EXTRACTION
Noun phrase extraction or Noun Phrase Chunking is the process of selecting or extracting out noun phrases from a given text. From the linguistic aspect, we usually say that the main “building blocks” of a sentence are Noun Phrases (NP) and Verb Phrases (VP). The Noun Phrases are usually the topics or objects in the sentence, or in simple words – this is what the sentence is talking about, while Verb Phrases describe some action between the objects in the sentence. Generally in a given sentence the Noun Phrase (NP) depicts the answer if the sentence is converted into a question.

How is noun phrase extraction done?

We are using TextBlob to create a POS tagger of the input sentences. The POS tagger tags the sentences for parts of speech. Since we already removed articles and prepositions. Then we pass in an extractor model to build phrases and categorize them.

Textblob provides two extractors by default

- FastNPExtractor
- ConllExtractor

We are using the ConllExtractor which is based on the CoNLL 2000 corpus. The CoNLL 2000 corpus includes phrasal chunks which better selects phrases.

C. Question Generation.

Question generation (QG) is the task of mechanically generating queries from numerous inputs like raw text, database, or linguistics illustration. Though automatic QG is may be approached with numerous techniques, QG is largely thought to be a discourse task involving the subsequent steps:

1. Take out important parts for the text input i.e., summarizing (Done in the previous step).
2. What the question is concerning, i.e. content choice.
3. Question Kind identification

Question Construction

Question Generation Application

Question generation – The purposeful asking and response of questions about what's browse – serves the goal of reading comprehension instruction not solely of its own accord, however conjointly in conjunction with multiple reading comprehension methods. QG methods, additionally to being a natural precursor to Question respondent (QA), is a superb compliment to a different and proved strategy - comprehension observance.

Classification of Queries

It is necessary to categorize queries as different classes of queries need different methods for automatic generation of question type.

Gap Questions: Fill-in-the-blank questions, conjointly called diagnostic test questions, could be a sentence with one or a lot of blanks in it with four alternatives to fill those blanks. (For example: _____ is the father of our nation.)

Short Answer Questions: Wh-questions use interrogative words, like why, when, who, where, which, etc., to request data. They can't be answered with an affirmative or no. Non-polar wh- queries are unit in distinction with polar yes-no queries, that don't essentially gift a variety of other answers, or essentially prohibit that vary to 2 alternatives. (For example, What time did you get across last night?)

True/False Question: In linguistics, a question, formally called a polar question, could be a question whose expected answer is either affirmative or no. Formally, they give associate degree exclusive disjunction, a combination of alternatives of that just one is appropriate. In English, such queries may be shaped in each positive and negative forms (For example, can you be here tomorrow? and Won’t you be here tomorrow?).

Gap Question Generation

This section involves the way to generate gap questions/ fill-in-the-blanks questions. After the summarization of an article, noun phrases are using the ConllExtractor from TextBlob. Certain noun phrases are then randomly chosen from the available nouns from each sentence.
Pseudo code for gap question generation:

```python
word = random.choice(blob.noun_phrases)
```

After the word is chosen, it's replaced by blank space from the same sentence.

*Short Question Generation*

After generation of the TextBlob for each and every line of the summarized text, those sentences are selected which are capable of producing interrogative questions.

Before generation of the questions, certain patterns are defined, which are followed to create the short answer type questions.

**POS description:**

- NNS : Noun, plural
- JJ : Adjective
- NNP : Proper noun, singular
- VBG : Verb, gerund or present participle
- VBN : Verb, past participle
- VBZ : Verb, 3rd person singular present
- VBD : Verb, past tense
- IN : Preposition or subordinating conjunction
- PRP : Personal pronoun
- NN : Noun, singular or mass

**POS tags pattern:**

- 11 = ['NNP', 'VBG', 'VBZ', 'IN']
- 12 = ['NNP', 'VBG', 'VBZ']
- 13 = ['PRP', 'VBG', 'VBZ', 'IN']
- 14 = ['PRP', 'VBG', 'VBZ']
- 15 = ['PRP', 'VBG', 'VBD']
- 16 = ['NNP', 'VBG', 'VBD']
- 17 = ['NN', 'VBG', 'VBZ']
- 18 = ['NNP', 'VBZ', 'JJ']
- 19 = ['NNP', 'VBZ', 'NN']
- 110 = ['NNP', 'VBZ']
- 111 = ['PRP', 'VBZ']
- 112 = ['NNP', 'NN', 'IN']
- 113 = ['NN', 'VBZ']

For each and every sentence in the summarized text, the algorithm finds whether any particular line has all the POS parts of any one of the patterns described above. After a particular line is selected, a certain pattern is
followed to generate the question:

Ex: “Interrogative pronoun” + tags of a lx { x belongs to [1, 13] } + “?”

VI. CONCLUSION

The present system proposed by this paper works wonderfully on a good amount of text i.e. sentences. Fairly accurate questions are developed with just the click of a button. The system not only generates factual questions but also some descriptive questions. We also calculate possible multiple choice questions once an answer is found for a specific question. That is the added flavor of the system. This work is a true reflection of what NLP can do. 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.

VII. FUTURE SCOPE

After successful implementation of this work, in future it will lead to Question Generation from entire documents as collections. In that case it would be required to first find out those sentences from a document on whom questions can be formed. After that the current system algorithm with minor improvement can be deployed. The ultimate system will be able to independently handle any pdf or word or any other type of text file, analyze it and find important sentences for QG. From those sentences a variety of questions could be formed with minimum inaccuracy. Also, the accuracy of generating possible multiple choice questions can be enhanced by adding similar meaning words from the context itself rather than the entire vocabulary. This will not only reduce the search space but also reduce the time elapsed in producing desired results. We can even add functionalities to export the question set as some normalised output format such as csv or pdf that could be further used in printing or displaying the generated questions.

REFERENCES