



# **GENERATING MCQs FROM TEXT USING NLP**



A Project Report in partial fulfillment of the degree

**Bachelor of Technology**

in

**Computer Science & Engineering/Electrical & Electronics Engineering**

**By**

19K41A05G2  
19K41A05G1  
20K45A0229

M.YASHASHREE  
B. SHRITHAJANI  
A.MAHESH BABU

Under the Guidance of

**D. RAMESH**

Submitted to

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
S R ENGINEERING COLLEGE(A), ANANTHASAGAR, WARANGAL  
(Affiliated to JNTUH, Accredited by NBA)**

**NOV -2022**



**DEPARTMENT OF COMPUTER SCIENCE / ELECTRICAL  
& ELECTRONICS ENGINEERING**

**CERTIFICATE**

This is to certify that the Project Report entitled “Generating MCQs from text Using NLP ” is a record of bonafide work carried out by the student(s) M.Yashasree, M.Shrithajani, A.Mahesh babu bearing Roll No(s) 19K41A05G2, 19K41A05G1, 20K45A0229 during the academic year 2022-2023 in partial fulfillment of the award of the degree of Bachelor of Technology in Electrical & Electronics /Computer Science Engineering by theJawaharlal Nehru Technological University, Hyderabad.

**Supervisor**

**Head of the Department**

**External Examiner**

## **ABSTRACT**

Automatic multiple-choice question generation (MCQG) is a useful yet challenging task in Natural Language Processing (NLP). Examinations and Assessments are undergoing a tremendous revolution. Universities, colleges, and other educational institutes are increasingly shifting towards online examinations. The pattern of assessment is majorly shifting towards the objective assessment i.e. MCQ based, it is very hard to construct and requires a considerable amount of time for setting numerous questions. There's a growing need for a cost-effective and time-efficient automated MCQ generation system. In this paper, the text is first summarized using the BERT algorithm, and accordingly sentence mapping is done for generating MCQs. In order to generate choices for the questions, distractors are generated using wordnet (A lexical database for English). As the BERT algorithm has much better performance over other legacy methods as well as it can process a large amount of data in less time, it will enhance the speed of generating mcq's from given text.

## Table of Contents

<b>S.NO</b>	<b>Content</b>	<b>Page No</b>
1	Introduction	1
2	Literature Review	3
3	Design	5
4	Methodology	7
5	Results	10
6	Output Prediction	12
7	Conclusion	13
8	References	13

## 1. INTRODUCTION

All institutes, colleges, and schools have been switched to online learning. Assessment is an essential tool to test the knowledge of the students. And the pattern of the assessment has changed from subjective based to objective based i.e., Multiple Choice Questions (MCQs).

So, the problem is, it is very difficult for the teachers to set the questions as well as for the students who are preparing for competitive exams. The current method involves the setting of questions manually which requires a lot of human intervention and time. So, there is a growing need for a system that can create questions with ease and less amount of time and requires less human effort.

Automatic multiple-choice question generation (MCQG) is a useful yet challenging task in Natural Language Processing (NLP). It is the task of automatic generation of correct and relevant questions from textual data. Despite its usefulness, manually creating sizeable, meaningful and relevant questions is a time-consuming and challenging task for teachers. In this project, we present an NLP-based system for automatic MCQ generation for computer-based testing examination (CBTE). We used NLP technique to extract keywords that are important words in a given lesson material.

This project tells about a system that generates questions automatically. In Automated MCQ Generator, questions are generated automatically with the help of NLP. The text of any domain is provided as input to the system which is then summarized using the BERT algorithm. BERT (Bidirectional Encoder Representation from Transformers) is a deep learning-based technique for natural language processing, a pre-trained model from Google. Now the keywords are selected from the summarized text using the python keyword extractor (PKE) and accordingly mapping of a keyword is done with a sentence. This keyword will be one of the options of MCQ. Now the main task is generating relevant distractors. Distractors are generated using the wordnet approach. Wordnet is an API used to get the correct sense of the word. So the good and relatable distractors are generated. This system solves the problem of manual creation of questions and reduces time consumption and cost.

Natural Language Processing (NLP) is a branch of artificial intelligence that deals with the interaction between computers and humans using the natural language. The ultimate objective of NLP is to read, decipher, understand, and make sense of the written or textual data in natural

languages of human in a manner that is valuable. In this paper, we focus on the setting of MCQs through NLP to improve the method of setting MCQs and modification, and for creating a viable question bank for subsequent use by the academicians for their learners. This will ensure an MCQ that includes appropriate questions and options based on the learning objectives and importance of the topics discussed in a lesson material. We used NLP methods, Term Frequency-Inverse Document Frequency (TF-IDF) and N-grams, to extract the most significant words present in a lesson material and the selection does not depend on any vocabulary; these words are keywords that capture the main topics discussed in a lesson material, they also serve as an indication of document relevance for users in an information retrieval (IR) environment

## 2. LITERATURE REVIEW

We have studied several research papers on multiple choice question generation using different approaches. Santhanavijayan et al. have proposed a system of “Automatic generation of multiple-choice questions for e-assessment”

[1]. Chidinma A Nwafor and Ikechukwu onyenwe “Automatic multiple-choice question generation (MCQG) is a useful yet challenging task in Natural Language Processing (NLP). It is the task of automatic generation of correct and relevant questions from textual data. Despite its usefulness, manually creating sizeable, meaningful and relevant questions is a time-consuming and challenging task for teachers.”

[2]. In their proposed system, they have used fireflies-based preference learning and ontology-based approach to generate MCQs. They have used a web corpus to make it feasible to create questions. The distractors are generated using similarity metrics such as hypernyms and hyponyms. The system also creates analogy questions to test the verbal ability of the students.

[3]. The Ayako Hoshino and Hiroshi Nakagawa “A real-time multiple-choice question generation for language testing: A preliminary study” [2] is based on machine learning to generate questions automatically. They implement machine learning algorithms, such as Naive Bayes and K-Nearest Neighbors, to create questions on English grammar and vocabulary from online news articles. They designed a system that can receive user input in the form of an HTML file and turns it into a quiz session.

[4]. Deepshree S. Vibhandik et al. have proposed a system, “Automatic / Smart Question Generation System for Academic Purpose” [4] in which the Automatic Question Generation system generates specific trigger questions and multiple-choice questions from student's literature review papers. To facilitate the generation of specific trigger questions, the system extracts key concepts from student's papers using the Lingo algorithm. Also, to bring out the generation of multiple-choice questions, the system pulling out abbreviations from student's review papers using the regular expression pattern matching techniques.

[5]. Ms. Nikeeta Patil, Ms. Kratika Kumari, Mr. Devendra Ingale have proposed a system , ”Efficient multiple-choice questions are produced with quality distractors. The necessity of human intervention for generating question paper and answer is eliminated. The proposed system creates automated questions with the help of NLP that reduces human intervention and it is a cost and time-effective system also, the accuracy of the distractor generated is high.”

[6]. Ebrahim Gabajiwala, Priyav Mehta, Ritik Singh have proposed a system, "The recent advancement in NLP techniques has shown a lot of promise. The proposed solution uses an NLP pipeline involving Bert and T5 transformers to extract keywords and gain insights from the text input. From the extracted keywords, different types of questions are generated such as fill in the blanks, true or false, Wh-type and multiple-choice questions. Latest state-of-the-art models proved to perform better in all stages of our pipeline. The results from these models have shown a lot of promise".

[7]. Sonam Sonia, Praveen Kumarb, Amal Saha have proposed "generation (QG) is a very important yet challenging problem in NLP. It is defined as the task of generating syntactically sound, semantically correct and relevant questions from several input formats like text, a structured database or a knowledge base. Question generation can be naturally applied in many domains such as MOOC, automated help systems, search engines, chatbot systems (e.g. for customer interaction), and healthcare for analyzing mental health.



### 3. DESIGN:

#### 3.1 Requirement Specifications (S/W & H/W)

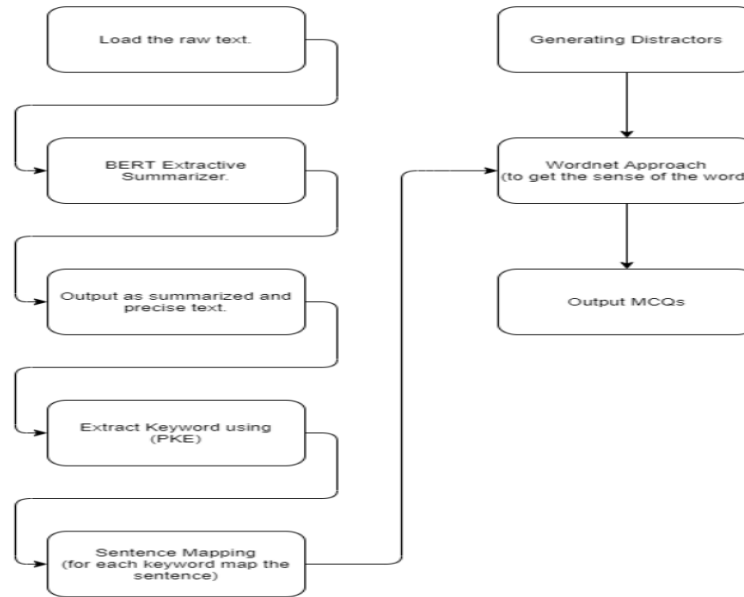
##### Hardware Requirements

- ✓ **System** : Processor Intel(R) Core (TM) i5-8265U CPU @ 1.60GHz, 1800 MHz, 4 Cores, 8 Logical Processors
- ✓ **RAM** : 8 GB
- ✓ **Hard Disk** : 557 GB
- ✓ **Input** : Keyboard and Mouse
- ✓ **Output** : PC

##### Software Requirements

- ✓ **OS** : Windows 10
- ✓ **Platform** : Google Colaboratory
- ✓ **Program Language** : Python

### 3.1. Flowchart



**Fig 1:** Flow chart representation

The text of any domain is provided as input to the system which is then summarized using the BERT algorithm. BERT (Bidirectional Encoder Representation from Transformers) is a deep learning-based technique for natural language processing, a pre-trained model. Now the keywords are selected from the summarized text using the python keyword extractor (PKE) and accordingly mapping of a keyword is done with a sentence. This keyword will be one of the options of MCQ. Distractors are generated using the wordnet approach. Wordnet is an API used to get the correct sense of the word. So, the good and reliable distractors are generated.

## 4. METHODOLOGY:

This section talks about the whole procedure and methods used for this project.

### **LOAD RAW DATA**

The first step is to load raw text i.e. input text of any domain for which the questions to be generated.

### **SUMMARIZER**

Each sentence is not capable of generating questions. Only the sentences that contain a questionable fact can act as a candidate for creating MCQs. Therefore, sentence selection plays a crucial role in the automatic MCQ generation task. Hence for summarizing the text, BERT Algorithm is used. BERT (Bidirectional Encoder Representations from Transformers) is a neural network-based technique for natural language processing. It is a pre-trained open-sourced model from Google. It helps computers to understand the language a bit more as humans do. The input text is summarized using the BERTSUM model, which is fine-tuned BERT for extractive summarization. The architecture of BERTSUM is shown in Fig-2.

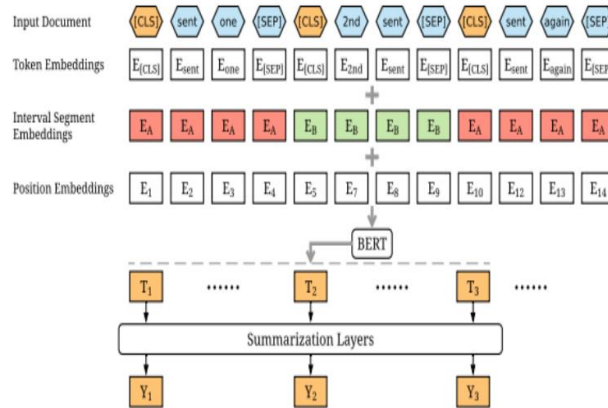


Fig.-2: Overview Architecture of BERTSUM model

### **KEYWORD EXTRACTION**

After summarizing the text, keywords are selected from the sentence. This keyword will be the answer to the question. Since all the words in an informative sentence cannot serve as the key proper

keyword selection is required. The extraction of keyword is done by python keyword extractor.

Python keyword extractor(PKE) provides an end-to-end keyphrase extraction pipeline in which each component can be easily modified or extended to develop new models.

### **GENERATION OF DISTRACTERS**

Generating distractors is the most crucial step in the generation of automated MCQs. The difficulty of MCQs highly relies on the quality of distractors produced. A good distractor is one that is very similar to the key but not the key itself. So, for generating distractors Wordnet approach is used.

WordNet is a lexical database for the English language, which was created by Princeton, and is part of the NLTK corpus. In the WordNet network, the words are connected by linguistic relations. These linguistic relations includes hypernym, hyponym, meronym, holonym, etc. WordNet stores synonyms in the form of synsets (Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms) where each word in the synset shares the same meaning. Basically, each synset is a group of synonyms. Each synset has a definition associated with it. Relations are stored between different synsets. This Lesk algorithm is based on the assumption that words in a given "neighborhood" (section of text) will tend to share a common topic. A simplified version of the Lesk algorithm compares the dictionary definition of an ambiguous word with the terms contained in its neighbourhood.

For example, if there is a sentence “The bat flew into the jungle and landed on a tree” and a keyword is “bat”, we automatically know that here we are talking about the mammal bat that has wings, not a cricket bat or baseball bat. Although we humans are good at it, the algorithms are not very good at distinguishing one from another. This is called word sense disambiguation (WSD). In the wordnet, “bat” may have several senses, one for a cricket bat, one for flying mammal etc. So the function `get_wordsense` tries to get the correct sense of the word given in the sentence. Once the sense of the word is identified the `get_distractors_wordnet` function is called to get the distractors. This function tries to get the distractors with the help of hypernyms and hyponyms of the key.

#### **HYPERNYM:**

Hypernym is a word that names a broad category that includes other words.

**Ex:** animal is hypernym of dog.

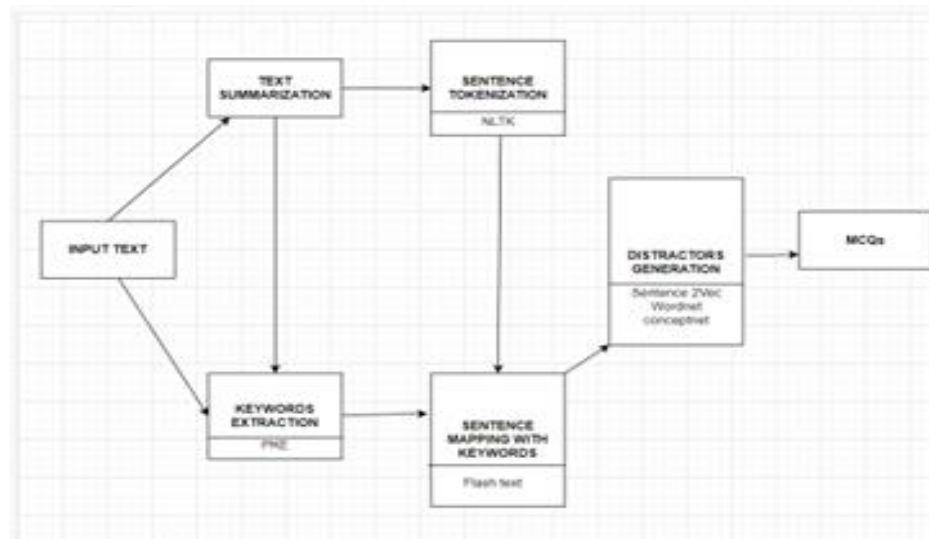
#### **HYPONYM:**

A word of more specific meaning than a superordinate term applicable to it.

**Ex:** car is a hyponym of vehicle.

Now all the possible hypernyms of the key and the corresponding hyponyms for each of the hypernym are found. These hyponyms are considered potential distractors. Then, the potential distractors

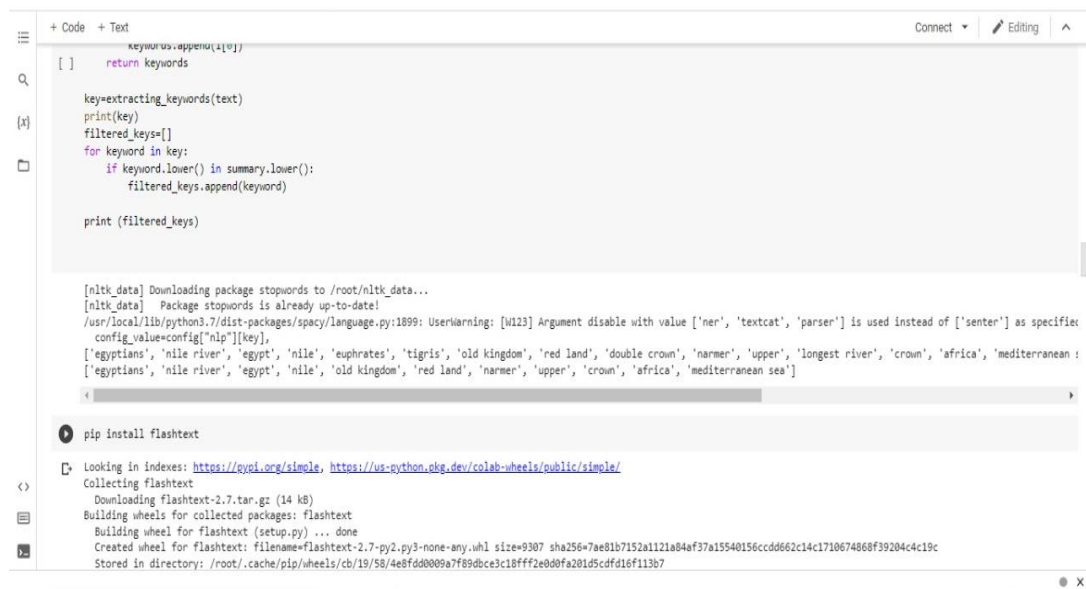
are ranked, and the final distractors are chosen based on their rank. For the word 'bat' as the key, the hypernym for the bat is - an animal of which hyponyms can be an eagle and other birds which can be good distractors for the key. The ranks given to the potential distractors are based on whether it occurs in the text extracted from the summarization. The potential distractors that exist in the extracted text and having the same part of speech structure as the key have a higher rank than those with a different structure. This is because distractors with the same structure as the key tend to confuse the test takers more. Any three potential distractors with higher ranks are chosen at random as the final distractor



**Fig 3:** Process flow

## 5. RESULTS:

So, from the input text we are summarizing that text. From that summarized text we extracted keywords. After extracting those keywords, we will be mapping those keywords to their sentences and once the keywords are extracted, we tried to generate distractors. As we know in the multiple-choice questions there will be correct answer in the options and also remaining options will be incorrect. Distractors are nothing but the wrong options for that particular multiple-choice question. Then the keywords are replaced with ----- . Hence becoming the question, the keyword as an answer and distractors as the other options. Finally putting all together we have generated multiple choice questions.



```
+ Code + Text
keyworus.append(i[0])
[ ] return keywords

key=extracting_keywords(text)
print(key)
filtered_keys=[]
for keyword in key:
    if keyword.lower() in summary.lower():
        filtered_keys.append(keyword)

print (filtered_keys)

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
/usr/local/lib/python3.7/dist-packages/spacy/language.py:1899: UserWarning: [U123] Argument disable with value ['ner', 'textcat', 'parser'] is used instead of ['sender'] as specified
  config_value=config["nlp"][key],
['egyptians', 'nile river', 'egypt', 'nile', 'euphrates', 'tigris', 'old kingdom', 'red land', 'double crown', 'narmer', 'upper', 'longest river', 'crown', 'africa', 'mediterranean sea']
['egyptians', 'nile river', 'egypt', 'nile', 'old kingdom', 'red land', 'narmer', 'upper', 'crown', 'africa', 'mediterranean sea']

pip install flashtext

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting flashtext
  Downloading flashtext-2.7.tar.gz (14 kB)
Building wheels for collected packages: flashtext
  Building wheel for flashtext (setup.py) ... done
Created wheel for flashtext: filename=flashtext-2.7-py2.py3-none-any.whl size=9307 sha256=7ae81b7152a1121a04af37a15540156cdd662c14c1710674868f39204c4c19c
Stored in directory: /root/.cache/pip/wheels/cb/19/58/4e8fd0009a7f89dbce3c18fff2e0d0fa201d5cdfd16f113b7
```

**Fig 4:** keywords are extracted

```
+ Code + Text
Connect Editing

from nltk.tokenize import sent_tokenize
from flashtext import KeywordProcessor

def tokenize_sentences(text):
    sentences = [sent_tokenize(text)]
    sentences = [y for x in sentences for y in x]
    # Remove any short sentences less than 20 letters.
    sentences = [sentence.strip() for sentence in sentences if len(sentence) > 20]
    return sentences

def get_sentences_for_keyword(key, sentences):
    keyword_processor = KeywordProcessor()
    keyword_sentences = {}
    for word in key:
        keyword_sentences[word] = []
        keyword_processor.add_keyword(word)
    for sentence in sentences:
        keywords_found = keyword_processor.extract_keywords(sentence)
        for key in keywords_found:
            keyword_sentences[key].append(sentence)

    for key in keyword_sentences.keys():
        values = keyword_sentences[key]
        values = sorted(values, key=len, reverse=True)
        keyword_sentences[key] = values
    return keyword_sentences

sentences = tokenize_sentences(summary)
keyword_sentence_mapping = get_sentences_for_keyword(filtered_keys, sentences)

print (keyword_sentence_mapping)

{'egyptians': ['Egyptians cut the stems into strips, pressed them, and dried them into sheets that could be rolled into scrolls.', 'The Nile provided so well for Egyptians that sometimes they had surpluses,
```

**Fig 5:** mapping the keywords to the sentences

## 6. OUTPUT PREDICTION:

As we mentioned in the results we generated multiple choice questions with appropriate distractors.

```
Warming up PyWSD (takes ~10 secs)... took 6.916054010391235 secs.
1) _____ cut the stems into strips, pressed them, and dried them into sheets that could be rolled into scrolls.
   a ) Angolan
   b ) Egyptians
   c ) Algerian
   d ) Bantu
More options: ['Basotho', 'Beninese', 'Berber', 'Black African', 'Burundian', 'Cameroonian', 'Carthaginian', 'Chadian', 'Chewa', 'Congolese', 'Djiboutian', 'Egyptian', 'Ethiopian',

2) Because the pharaoh was thought to be a god, government and religion were not separate in ancient _____.
   a ) Iraq
   b ) Egypt
   c ) Kuwait
   d ) Saudi Arabia
More options: ['Jordan', 'Israel', 'Fertile Crescent', 'Turkey', 'Iran', 'Lebanon', 'Shari', 'Mauritania', 'Nigeria', 'Somali peninsula', 'Sierra Leone', 'Malawi', 'North Africa',

3) The _____ provided so well for Egyptians that sometimes they had surpluses, or more goods than they needed.
   a ) Buganda
   b ) Entebbe
   c ) Gulu
   d ) Nile
More options: ['Jinja', 'Lake Edward', 'kayunga', 'gulu', 'entebbe', 'Port Sudan', 'Omdurman', 'Darfur', 'Libyan Desert', 'Kordofan', 'Khartoum', 'Nubian Desert', 'Nyala', 'Aswan Hi

4) Nubia was the Egyptian name for the area of the _____ Nile that had the richest gold mines in Africa.
   a ) Upper Berth
   b ) Lower Berth
   c ) Upper
```

Fig 6: Output result



## **7. CONCLUSION:**

Multiple Choice Questions (MCQs) are generated successfully. Efficient questions are produced with good quality distractors. The problem of manually creating questions is solved with the proposed system. The proposed system creates automated questions with the help of NLP that reduces human intervention and it is a cost and timeeffective system. And the accuracy of the distractor generated is reasonably high. This system not only helps teachers with E-assessments but also helps students who are preparing for competitive exams. Students can test their ability to solve the questions and can also check their understanding of the concepts.

## **8. FUTURE SCOPE:**

Since our proposed system is based on Google's BERT Model, the accuracy of the system will increase in the future as the performance of the model is improved and as the research in the field of NLP is trying to reach the human level every day. We can further develop into an android where we can directly access it.

## 9. REFERENCES:

- [1] [https://www.researchgate.net/publication/351665611\\_An\\_Automated\\_Multiple-Choice\\_Question\\_Generation\\_using\\_Natural\\_Language\\_Processing\\_Techniques](https://www.researchgate.net/publication/351665611_An_Automated_Multiple-Choice_Question_Generation_using_Natural_Language_Processing_Techniques)
- [2] [https://www.researchgate.net/publication/317610512\\_Automatic\\_generation\\_of\\_multiple\\_choice\\_questions\\_for\\_e-assessment](https://www.researchgate.net/publication/317610512_Automatic_generation_of_multiple_choice_questions_for_e-assessment)
- [3] <https://aclanthology.org/W05-0203.pdf>
- [4] <https://www.ijettcs.org/Volume4Issue4/IJETTCS-2015-07-13-27.pdf>
- [5] [https://ijsret.com/wp-content/uploads/2021/05/IJSRET\\_V7\\_issue3\\_470.pdf](https://ijsret.com/wp-content/uploads/2021/05/IJSRET_V7_issue3_470.pdf)
- [6] [https://research.spit.ac.in/storage/290/8\\_Quiz-Maker--Automatic-Quiz-Generation-from-Text-using-NLP.pdf](https://research.spit.ac.in/storage/290/8_Quiz-Maker--Automatic-Quiz-Generation-from-Text-using-NLP.pdf)
- [7] <https://deliverypdf.ssrn.com/delivery.php?ID=740102120112125028008094012081087098016073055036039026094121111075069090031124025111057021116004010037021064027026011095099027041086005060042090101116006002122067089062011064008086029106068010127122096094086080121100067007027029071026093016074066069009&EXT=pdf&INDEX=TRUE>
- [8] <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>
- [9] Rahul, S. Adhikari and Monika, "NLP based Machine Learning Approaches for Text Summarization," 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), 2020, pp. 535-538