Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Design and Implementation of an Automated MCQ Generator using NLP

Himalakshi Das

Assistant Professor , Department of Computer Science and Technology,
Assam Women's University, Jorhat

Abstract
Multiple Choice Questions are widely used for the assessment of candidates in many competitive examinations. In educational institutions , course instructors conduct class tests by providing question papers with MCQs to examine student's understanding of the course. Preparing MCQs manually across different fields or domains is a challenging and time-consuming task that requires significant brainstorming.The automatic generation of multiple choice questions using Natural Language Processing(NLP) has emerged as a prominent research area among computing scientists. Automatic MCQ generation is a challenging task, as it requires generating relevant, semantically valid and contextually correct questions from a given text. Accurate context understanding is crucial for generating relevant and meaningful questions. The proposed system is designed to automatically generate MCQs from a given text. State-of-the-art Natural Language Processing(NLP) techniques are used to design the system. The proposed system accepts a textual passage as input and automatically generates a set of MCQs based on the content. The proposed system analyzes the document to obtain a summarized version of the passage and extracts keywords for generating relevant questions. In the proposed system, the identified keywords function as answer candidates for the MCQs.The generated MCQs follow a fill-in-the-blank format.The proposed system also generates distractors to provide additional answer options alongside the correct response.The implementation of the automated MCQ generator is carried out in Python, utilizing NLP libraries for text processing, keyword extraction and question generation.

Keywords: Natural Language Processing, NLP, Multiple Choice Questions, MCQ , Bidirectional Encoder Representation from Transformer, BERT, Python Keyword Extractor,PKE

1. Introduction

Examinations are conducted to access students in an educational institutions and also to evaluate candidates during job selection process. Questions used for evaluation may be of subjective or objective type. With the advancement of computer networking technology, examination systems are increasingly shifting from offline to online modes. Multiple Choice Questions are widely used in question papers, as they can be evaluated using Optical Mark Recognition sheets or in online examination systems. Generating MCQs manually is a time-consuming task.A system is designed and implemented to generate MCQs automatically from text with the help of NLP (Natural Language Processing) technology.

In the proposed system, any passage can be provided as input. The input passage is summarized using the BERT (Bidirectional Encoder Representation from Transformer) model developed by Google. BERT is

an open source pre-trained model based on deep learning. It can summarize large text and takes less amount of time compared to other legacy method. From the summarized text, keywords are extracted using PKE (Python Keyword extractor) tool.The keyword serves as the answer to the MCQs. Then these keywords are mapped to the sentences. Finally, distractors are generated using Wordnet. Wordnet is an English Lexical Database. Wordnet provides an Application Programme Interface(API) to understand sense of a word.

2. Literature Review

Prtiam Kumar Mehta et al. have proposed a system in their paper titled 'Automatic MCQ generator using natural language processing'[1]. In their proposed system,they are using BERT model developed by Google to summarize the text and then use RAKE python library to extract keywords from the summarized text.

Chidinma A. Nwafor et al. proposed a system in their paper titled 'An automated multiple choice question generation using Natural Language Processing Techniques'[2]. The system is used to process lesson material/document fed by the teacher to generate MCQs along with answers to questions. An application interface is also provided to present the MCQs to the teacher for easy acceptance or rejection. The document is split into sentences. The splitted sentences are tokenized, from which the corpus is built in TF-IDF and N-gram mode.

Bidyut Das et al. proposed a system in their paper titled 'Multiple-choice question generation with auto-generated distractors for computer-assisted educational assessment'[3]. The proposed system generates simple sentences from the text material. Keyword are extracted from the sentences and appropriate sentences are selected for MCQ generation. A feature based clustering approach is proposed for distractor generation.

3. Methodology

The overall process of automatic MCQ generation using NLP can be divided into four sub-processes. The four sub-processes include text summarization, keyword extraction, distractor generation and finally, question generation. For implementing the proposed system, the following sequence of steps are adopted.

Input: any text file.
Output: Fill_in_the_blank type MCQs
Steps:
1. Summarize the text file
2. Generate keywords
3. Do sentence to keyword mapping
4. Remove keywords from mapped sentences and generate fill_in_the_blank type questions by replacing each keyword with a blank.
5. Generate distractors for each MCQ.

3.1 Loading Raw Text

The system accepts any text passage as input.

3.2 Summarizing the raw Text

The input passage can consists of any number of sentences. Not all sentences carry equal importance for generating questions. Some sentences carry more meaning with regard to the context of the passage. Therefore, text summarization is an essential step in the proposed system. After summarizing, the resulting passage contains candidate sentences suitable for question generation. Besides text summarization automatically generates a shorter version of a long passages.

In the proposed system, extractive summarization is used. Extractive summarization is unlikely to change the meaning of the text. In extractive summarization, subsets of sentences are extracted from the document and then assemble to form a summary paragraph. Bidirectional Encoder Representation from Transformer(BERT) is an open-source model developed by Google and can be used for text summarization. BERT is a neural network-based method used in Natural Language Processing. BERTSUM is an extractive summarizer. In the BERTSUM method, each sentence is assigned a score based on its importance to the context of the passage. Sentences with the highest scores are kept in the summarized text.

3.3 Keyword Extraction

Keywords are the answers to the questions. Not every word in the summarized text can be a keyword. Therefore, keyword extraction is a crucial step in the proposed process. For keyword extraction, the RAKE (Rapid Automatic Keyword Extraction) library of Python is used.

3.4 Distractor Generation-

Distractors are the alternative options for the answers to the questions. They are the incorrect answers. Distractor generations is a very important step in the process, as well-designed distractors can challenge students and thus make the MCQs more effective..

For distractor generation, the WordNet approach is used. WordNet is a lexical database for the English language and is integrated into the Python NLTK (Natural Language Tool Kit) library. Wordnet contains synonyms gouped as synsets of words. Context of a word can be determined by examining its neighbouring words.

3.5. MCQ generation

Fill-in-the blank type questions are generated by removing the keywords from each mapped sentences and replacing them with blanks .

4. System Implementation

The system is developed and implemented using Python, leveraging its NLP libraries and tools. Mainly four Python libraries are used for designing the automated MCQ generator.The System's Graphical User Interface(GUI) is implemented using the Tkinter library in Python.The Summarizer library is used for summarizing the raw text.Rake library is used for keyword extraction from summarized text. Nltk library is used for the purpose of distractor generation.

The following sequence of steps are followed during the interaction of a user with the system.

1. The user is promted to select and open a text file that contains the passage for MCQ generation.After the user opens the file, its content is displayed .

2. The ‘View content summary’ button is enabled.The user can view the summary of the passage by clicking the button .Summary of the passage is displayed inside a text box.The system allows the user to edit or modify the summarized text if desired.
3. The ‘Click to generate keyword’is enabled.The user needs to click this button to generate keywords.
4. A list of keywords is displayed to the user. The system allows the user to delete any inappropriate keyword.
5. The ‘Generate MCQ’ button is enabled. The user needs to click this button to generate MCQs.A list of MCQs based on the keywords, along with distractors is shown to the user.
6. User can save the MCQs in a file by clicking the ‘ save’ button.

The implementation of the proposed system is carried out using the Python code presented below.

```
from tkinter import *
from tkinter.ttk import *
from tkinter import filedialog as fd
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
from summarizer import Summarizer
from summarizer import TransformerSummarizer
from rake_nltk import Rake
import nltk
from nltk.corpus import wordnet as wn
import random

def open_file():
    global content
    file_path = fd.askopenfilename(
        title="Select a Text File", filetypes=[("Text files", "*.txt")])
    if file_path:
        with open(file_path, 'r') as file:
            content = file.read()
            text_widget.delete(1.0, tk.END)  # Clear previous content
            text_widget.insert(tk.END, content)

    mcq_buton['state'] = tk.NORMAL

#code for summarizing the content
def sumz():
    global bert_summary
    bert_model=Summarizer()
```

```
    bert_summary=''.join(bert_model(content))
    text_widget1.delete(1.0, tk.END)  # Clear previous content
    text_widget1.insert(tk.END,bert_summary)
    next_button['state'] = tk.NORMAL

#code for keyword generation
def keyword_generate():
    r=Rake(include_repeated_phrases=False,min_length=1, max_length=2)
    l=r.extract_keywords_from_text(bert_summary)
    global p
    p=r.get_ranked_phrases()
    return p

def openNewWindow():
    global master
    master = Tk()
    master.title('Automatic MCQ Generator')
    master.geometry("1000x1000")
    Label(master,  text ="Keywords are the correct options for MCQ.Select
the keyword for removal",font   = ("Times New Roman", 15)).pack()
    ttk.Label(master, text = "Select the keyword :", font = ("Times New
Roman", 15)).pack( padx = 10, pady = 25)
    p=keyword_generate()
    global combo
    combo = ttk.Combobox(master, values=p,width = 27)
    combo.bind("<<ComboboxSelected>>", option_selected)
    combo.pack()
    buton=Button(master,text="Remove keyword",
command=deleteKeyword).pack(padx=10,pady=25)
    fr=Frame(master)
    fr.pack()
    buton=Button(fr,text="Generate
MCQ",command=keywordMapping).pack(padx=10,pady=25)
    global fr2
    fr2=Frame(master)
    fr2.pack()
    scroll_bar = Scrollbar(fr2)

    scroll_bar.pack( side = RIGHT,
                fill = Y )
    global text_wd
    text_wd = tk.Text(fr2, wrap="word",yscrollcommand = scroll_bar.set )
```

```
    text_wd.pack()
    scroll_bar.config( command = text_wd.yview )
    fr3=Frame(master)
    fr3.pack()
    buton_file=Button(fr3,text="save in file
",command=fileSave).pack(padx=10,pady=25)

def fileSave():
    file_path = fd.asksaveasfilename(defaultextension=".txt",
filetypes=[("Text files", "*.txt"), ("All files", "*.*")])
    if file_path:
        try:
            with open(file_path, 'w') as file:
                text_content =   text_wd.get("1.0", "end-1c")
                file.write(text_content)
            messagebox.showinfo("Information","File is saved")
        except Exception as e:
            status_label.config(text=f"Error saving file: {str(e)}")

def deleteKeyword():
    p.remove(selected_option)
    combo['values']=p
    messagebox.showinfo("Information","selected keyword deleted")

def option_selected(event):
    global selected_option
    selected_option = combo.get()

#code for keyword to sentence mapping
def keywordMapping():
    sentence=bert_summary.split('.')
    count=1
    text_wd.delete(1.0, tk.END)
    for i in p:
        for j in sentence:
            if i in j:
                new=j
                new=new.replace(i,'_________ ')
                text="\n"+str(count)+"."+new
                text_wd.insert(tk.END, text)
                count=count+1
                get_distractor(i)
```

```
    value=text_wd.get("1.0","end-1c")

#Code for distractor generation
def get_distractor(word):
    distractor=[]
    word=word.lower()
    if len(word.split())>0:
        word=word.replace(" ","_")
    syn = wn.synsets(word)
    ls=[]
    choice=random.choices(p,k=3)
    if(len(syn)>0):
        syns=wn.synsets(word)[0]
        try:
            ls=syns.hypernyms()[0].hyponyms()
        except:
            ll=0

    distractor.append(word)
    if len(ls)>0:
        distractor.append(ls[0].lemmas()[0].name())
        if len(ls)>1:
            distractor.append(ls[1].lemmas()[0].name())
            if len(ls)>2:
                distractor.append(ls[2].lemmas()[0].name())
            else:
                distractor.append(choice[0])
        else:
            distractor.append(choice[0])
            distractor.append(choice[1])
    else:
        distractor.append(choice[0])
        distractor.append(choice[1])
        distractor.append(choice[2])
    i=1
    for m in distractor:
        text="\n\t"+str(i)+"."+ m
        text_wd.insert(tk.END, text)
        i=i+1

# Code for Main Window
```

```
root=Tk()
root.title('Automatic MCQ Generator')
root.geometry('3000x3000')
labl1=Label(root,text="Welcome to automatic MCQ
generator!",font=('Times',20))
labl2=Label(root, text="You have to choose a text file for question
generation",font=('Times',10))
labl1.grid()
labl2.grid()
fr=Frame(root)
fr.grid()
buton=Button(fr,text="Click here to open file",command=open_file)
buton.pack()
text_widget = tk.Text(root, wrap="word")
text_widget.grid()
mcq_fr=Frame(root)
mcq_fr.grid()
global mcq_buton
mcq_buton=Button(mcq_fr,text="View content summary",command=sumz,state=
DISABLED)
mcq_buton.pack()
text_widget1 = tk.Text(root, wrap="word",width=40, height=10)
text_widget1.grid()
last_fr=Frame(root)
last_fr.grid()
global next_button
next_button=Button(last_fr,text="Click to generate keywords",
command=openNewWindow,state= DISABLED)
next_button.pack(side=LEFT,padx=20,pady=20)
root.mainloop()
```

5. Result

The proposed system has been tested multiple times.

1. The primary Graphical User Interface(GUI) for the Automatic MCQ generator is shown below.

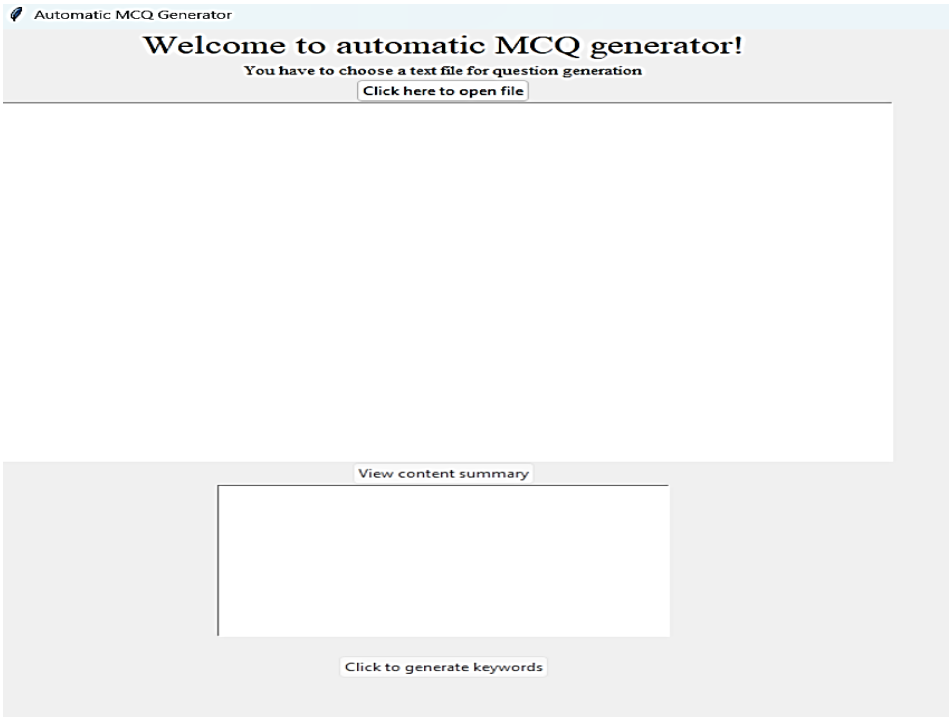

Figure 1: GUI for interacting with the Generator

2. After opening the file, the following result is shown to the user.

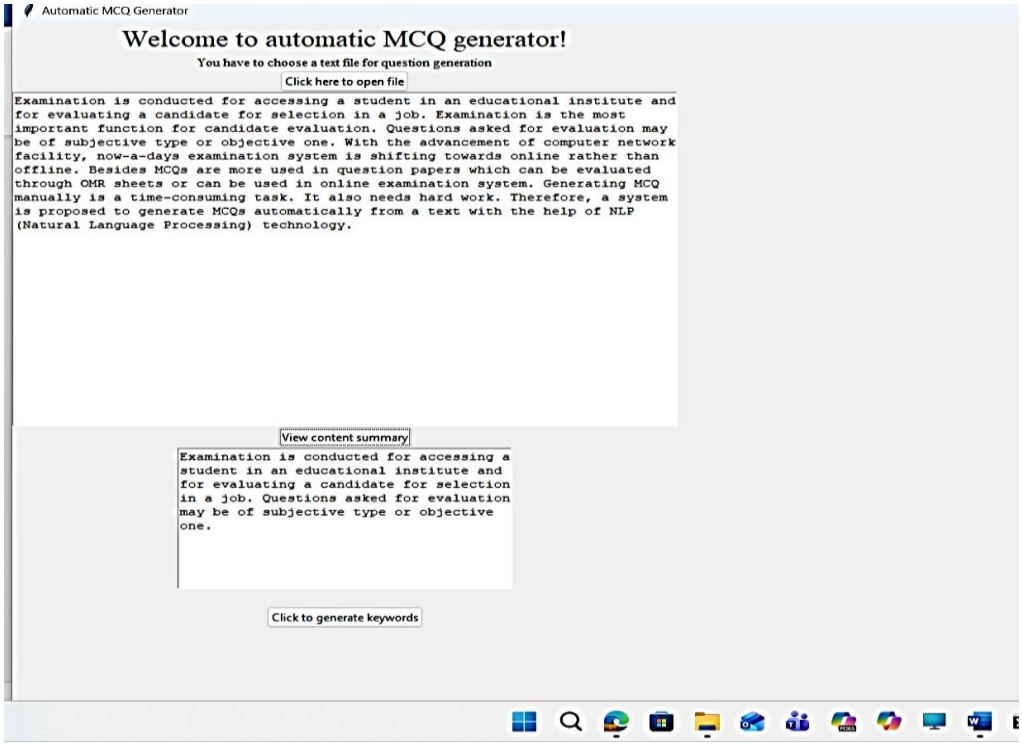

Figure 2:GUI for Passage Summarization

3. The following window is for keyword generation.

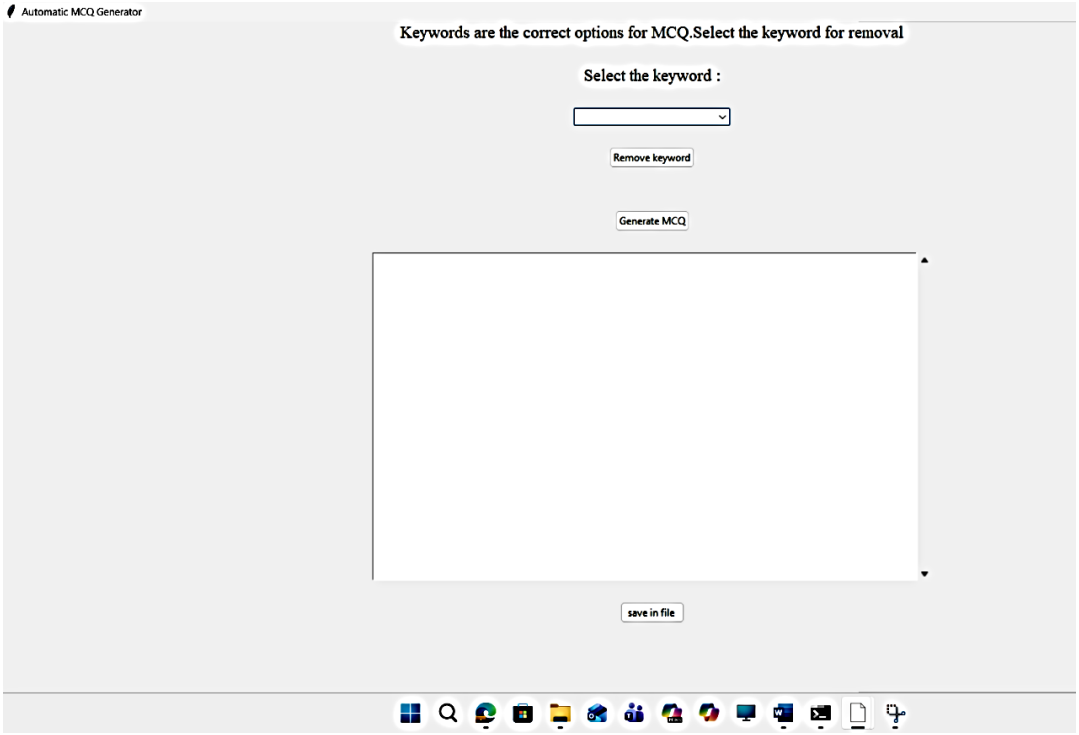

Figure 3: GUI for Keyword Generation

4. The follwing window is for final output generation.

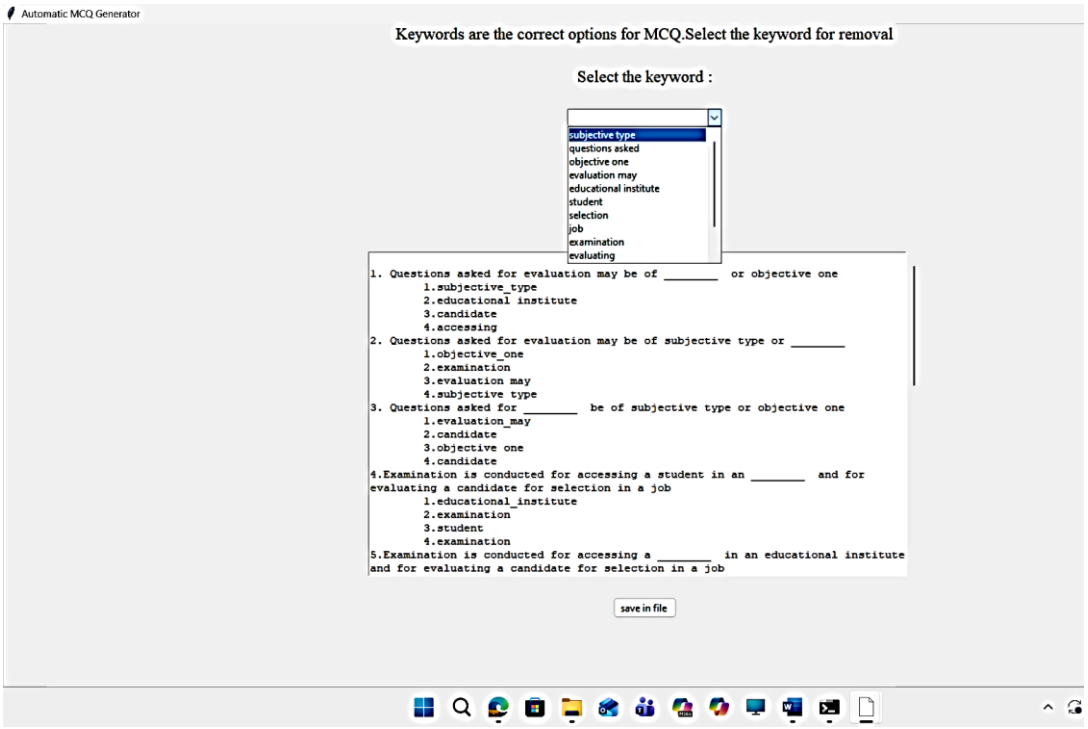

Figure 4: GUI for MCQ Generation

6. Conclusion and Future Work

The automated MCQ generation system successfully produces MCQs with carefully generated distractors, demonstrating its effectiveness. The proposed system uses WordNet, BERT and Rake library. With the further improvement of these libraries, the system will perform better. The proposed system opens opportunities for further development and optimization.This system generates only fill-in-the-blank type of questions. Future enhancement may include generating MCQs based on interogrative words to create more varied and contextually rich questions. The proposed system does not provide any mechanism to measure user satisfaction.

Bibliography References

1. Pritam Kumar Mehta, Prachi Jain, Chetan Makwana, Dr. C M Raut., "Automatic MCQ generator using natural language processing", International Research Journal of Engineering and Technology, Volume 08(Issue 5), May 2021.
2. Chidinma A. Nwafor and Ikechukwu E. Onyenwe., "An Automated Mutliple Choice Question Generation Using Natural Language Processing Techniques", arXiv preprint arXiv:2103.14757, 2021 - arxiv.org.
3. Bidyut Das, Mukta Majumder, Santanu Phadikar, Arif Ahmed Sekh**.,"** Multiple-choice question generation with auto-generated distractors for computer-assisted educational assessment.",Multimedia Tools and Applications (2021), July 2021.