A project report on

AUTOMATED MCQ GENERATOR

Submitted in partial fulfillment for the award of the degree of

MSc. Data Science

by

VIVEK SHARMA (23MSD7013)

BISWAJIT KAR (23MSD7017)

BISHAL SARMAH (23MSD7048)

VIT-AP UNIVERSITY

SCHOOL OF ADAVANCE SCIENCE (SAS)

May,2024

DECLARATION

We here by declare that the thesis entitled "AUTOMATED MCQ GENERATOR " submitted by us, for the award of the degree of MSc. Data Science VIT is a record of bonafide work carried out by us under the supervision of Ms. V LAKSHMI CHETANA.

We further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Vivek Sharma

Biswajit Kar

Bishal Sarma

Place: Amaravati

Date: 29.05.2024 **Signature of the Candidates**

ABSTRACT

Development of an Automated Multiple Choice Question (MCQ) Generator using NLP

Introduction:

In the modern educational landscape, the demand for efficient assessment tools is ever-increasing. Traditional methods of creating multiple choice questions (MCQs) are often labor-intensive and susceptible to biases. To address this challenge, our project focuses on developing an automated MCQ generator leveraging Natural Language Processing (NLP) techniques. This innovation aims to streamline the process for educators, ensuring a more efficient and unbiased approach to creating assessments.

Objectives:

The primary objective of this project is to harness machine learning and NLP to facilitate the automatic generation of MCQs. By automating this process, we aim to save educators significant time and effort, allowing them to focus more on teaching and less on administrative tasks. Additionally, the automation seeks to reduce human biases inherent in traditional MCQ creation, leading to more balanced and objective assessments.

Methodology:

The methodology of our project involves several key NLP techniques. Initially, tokenization is used to break down educational texts into manageable units. Following this, keyphrase extraction is performed to identify the main concepts and terms within the text. This step is crucial as it lays the foundation for generating relevant and accurate questions. Word sense disambiguation is then applied to ensure that the questions and distractors (incorrect options) are contextually appropriate and unambiguous.

Process:

1. **Tokenization:** This process splits the text into words or phrases, facilitating easier analysis.

2. **Keyphrase Extraction:** Important concepts and terms are extracted from the text, which form the basis for questions.

3. **Question Generation:** Based on the key phrases, questions are formulated.

4. **Distractor Creation:** Contextually relevant distractors are generated to accompany the correct answer, enhancing the quality of the MCQ.

Results and Implications:

The system we've developed successfully generates high-quality MCQs. By automating this process, the efficiency of creating educational assessments is significantly enhanced. This innovation not only benefits educators by saving time but also contributes to a more standardized and objective assessment process. Furthermore, our work opens up new avenues for research in the field of automated educational tools, highlighting the potential of NLP in transforming educational practices.

ACKNOWLEDGEMENT

It is our pleasure to express with deep sense of gratitude to Ms V LAKSHMI CHETANA, MSc. Data Science, School of Advance Science (SAS), VIT-AP, for her constant guidance, continual encouragement, understanding; more than all, she taught us patience in our endeavor.Our association with her is not confined to academics only, but it is a great opportunity on our part of work with an intellectual and expert in the field of Artificial Intelligence.

We would like to express our gratitude to Chancellor - Dr. G. Viswanathan , VC - Dr. S. V. Kota Reddy, and Dean – prof. S. Srinivas , School of Advance Science(SAS), for providing with an environment to work in and for their inspiration during the tenure of the course.

In jubilant mood we express ingeniously our whole-hearted thanks to Program char- Dr. Faizan Danish . MSc. Data Science, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on us with zeal, which prompted the acquirement of the requisite knowledge to finalize our course study successfully. We would like to thank our parents for their support.

It is indeed a pleasure to thank our friends who persuaded and encouraged us to take up and complete this task. At last but not least, We express our gratitude and appreciation to all those who have helped us directly or indirectly toward the successful completion of this project.

Vivek Sharma

Biswajit Kar

Place: Amaravati Bishal Sarmah

Date:29.05.2024 **Name of the students**

CONTENTS

INTRODUCTION

The rapid evolution of technology in education has revolutionized the creation and delivery of learning materials and assessments. Among the various assessment tools available, Multiple Choice Questions (MCQs) are particularly favored due to their straightforward grading process and their ability to encompass a wide range of topics efficiently. Despite these advantages, the manual creation of high-quality MCQs is a time-consuming and labor-intensive task for educators. Crafting questions that are clear, unbiased, and aligned with learning objectives requires significant effort and expertise.

This project addresses this challenge by developing an automated MCQ generator using Natural Language Processing (NLP). NLP is a branch of artificial intelligence that focuses on the interaction between computers and human language. By harnessing the power of NLP, this project aims to automate the extraction of key concepts from educational texts and the generation of meaningful questions and plausible answer options. The automation of this process is expected to significantly reduce the workload on educators, allowing them to focus more on teaching and less on administrative tasks.

Furthermore, the use of NLP ensures that the generated questions maintain a level of consistency and fairness, which is often difficult to achieve manually. This consistency is crucial in educational assessments to ensure that all students are evaluated on the same standards. By integrating advanced NLP techniques, the project seeks to produce high-quality MCQs that are both reliable and effective in measuring student understanding and learning outcomes.

In summary, this project leverages the advancements in NLP to create an automated MCQ generator that streamlines the assessment creation process. By doing so, it not only alleviates the burden on educators but also enhances the quality and fairness of educational assessments. This innovation represents a significant step forward in the application of technology in education, promising to improve the overall efficiency and effectiveness of the learning process.

MOTIVATION OF OUR PROJECT

The current educational landscape is experiencing a significant transformation driven by technological advancements. As the demand for personalized and scalable learning solutions continues to grow, the need for efficient educational tools becomes increasingly critical. This project is motivated by the necessity to develop innovative solutions that can meet these evolving educational needs. Specifically, it focuses on the automation of multiple-choice question (MCQ) generation, a task that is traditionally time-consuming and prone to human error and bias.

The Challenge of Manual MCQ Generation:

One of the fundamental components of educational assessment is the creation of multiple-choice questions. These questions are essential for evaluating students' understanding of various subjects, but the manual process of generating them poses several challenges. Educators must ensure that the questions are comprehensive, diverse, and free from bias, which demands a significant investment of time and effort. The manual approach also increases the likelihood of inconsistencies and errors, which can compromise the quality of assessments. In an era where educators are already burdened with numerous responsibilities, the inefficiencies associated with manual MCQ generation can hinder their ability to deliver high-quality education.

The Need for Automation:

Automation offers a promising solution to the challenges of manual MCQ generation. By leveraging advanced algorithms and machine learning techniques, we can develop a system that automates the creation of multiple-choice questions. This not only reduces the time and effort required but also ensures a higher degree of consistency and objectivity in the questions generated. An automated system can quickly generate a large pool of questions, each tailored to specific learning objectives and difficulty levels. This capability is particularly beneficial in a personalized learning environment, where assessments must be adaptable to the unique needs of each student.

Benefits of Automated MCQ Generation:

1. **Efficiency and Time-Saving:** One of the most immediate benefits of automation is the significant reduction in the time and effort required to generate MCQs. Educators can focus more on teaching and engaging with students, rather than spending hours crafting assessment questions. This increased efficiency translates into more productive use of educational resources.

2. **Consistency and Objectivity:** Automated systems can maintain a high level of consistency in question quality, eliminating human biases and errors. This leads to fairer assessments, where each question is designed to accurately measure student understanding without unintended influences.

3. **Scalability:** Automation allows for the rapid creation of large question banks, which can be used to support a wide range of subjects and difficulty levels. This scalability is essential for institutions that cater to large student populations or offer diverse courses.

4. **Personalized Learning:** With automation, it is possible to generate questions that are tailored to individual student needs. Adaptive assessments can be created, where the difficulty and type of questions adjust based on the student's performance, ensuring a more personalized and effective learning experience.

Bridging the Gap Between Technology and Education:

The integration of automated MCQ generation into the educational process represents a significant step towards bridging the gap between technology and education. By providing educators with advanced tools that enhance the quality and efficiency of assessments, we can support the diverse needs of learners and promote better educational outcomes. This project aims to harness the power of technology to create a more effective and inclusive educational environment.

The motivation for this project is rooted in the growing need for efficient and scalable educational tools. The challenges associated with manual MCQ generation highlight the importance of developing automated solutions that can enhance the quality and consistency of assessments. By reducing the time and effort required for question creation, educators can focus more on their primary role of teaching and supporting students. Ultimately, this project seeks to provide a tool that not only improves assessment processes but also contributes to the broader goal of advancing educational practices through technology.

OBJECTIVES

The core objective of this project is to utilize advanced Natural Language Processing (NLP) techniques to automate the generation of Multiple Choice Questions (MCQs). This automation aims to significantly streamline the creation of educational assessments, addressing several key challenges faced by educators.

Traditionally, the process of creating MCQs is manual and labor-intensive. Educators must carefully design questions, craft plausible distractors, and ensure that the questions accurately assess the intended knowledge or skills. This method is not only time-consuming but also prone to biases, as it heavily relies on the individual teacher's expertise and perspective. By automating this process, we aim to alleviate these burdens, providing a tool that can generate high-quality MCQs quickly and consistently.

Our specific objectives include:

1. **Efficiency Improvement:** By automating the MCQ generation process, we aim to save educators significant time and effort. The automated system can generate questions at a much faster rate compared to manual creation, allowing educators to focus more on teaching and other important tasks.

2. **Reduction of Bias:** Human-generated questions can inadvertently reflect the biases of the question creator. An automated system can produce more balanced and objective assessments by following standardized algorithms, ensuring a fairer evaluation of students' knowledge.

3. **Quality and Relevance:** The project seeks to ensure that the generated MCQs are of high quality and contextually relevant. Using NLP techniques like keyphrase extraction and word sense disambiguation, the system generates questions that accurately reflect the educational content, providing meaningful and effective assessments.

4. **Accessibility:** By providing an easy-to-use tool for MCQ generation, we aim to make high-quality assessment creation accessible to a broader range of educators, including those with limited resources or expertise in test design.

Through these objectives, our project endeavors to transform the way educational assessments are created, leveraging the power of NLP to enhance the efficiency, fairness, and quality of MCQs, thereby benefiting both educators and students alike.

PROBLEM DEFINITION

Enhancing the Efficiency and Quality of MCQ Generation in Education

The landscape of education is continually evolving, and assessments remain a cornerstone in evaluating student comprehension and learning outcomes. Among various assessment types, Multiple Choice Questions (MCQs) are widely used due to their ability to test a broad range of knowledge and skills efficiently. However, the creation of effective and fair MCQs is a complex and time-consuming task, posing significant challenges for educators across all levels of education. This project aims to address the inefficiencies and inconsistencies inherent in the manual generation of MCQs, proposing a solution that leverages automation to enhance the quality and reliability of educational assessments.

The Challenges of Manual MCQ Generation:

The traditional process of creating MCQs involves several meticulous steps, from drafting questions that align with the curriculum to ensuring that distractors (incorrect options) are plausible yet clearly distinguishable from the correct answers. Educators must also consider the difficulty level of each question, aiming for a balanced assessment that accurately measures student understanding without being overly challenging or too simplistic. This manual process is inherently labor-intensive, often requiring significant time and effort that could be better spent on direct instructional activities or student engagement.

Moreover, the manual creation of MCQs is susceptible to human error and bias. Even experienced educators can unintentionally introduce errors in questions or answer choices, leading to confusion and potential misassessment of student knowledge. Biases, whether conscious or unconscious, can also affect the fairness of the questions, potentially disadvantaging certain groups of students. These issues undermine the validity and reliability of the assessments, highlighting the need for a more efficient and consistent approach to MCQ generation.

Scope and Impact on Educational Institutions:

The problem of inefficient and inconsistent MCQ generation extends across all levels of education, from primary schools to higher education institutions. Teachers, professors, and instructional designers in various educational settings face the challenge of creating assessments that are not only aligned with learning objectives but also capable of providing meaningful insights into student performance. In large classes, the demand for diverse and numerous MCQs is particularly high, further exacerbating the burden on educators.

Automating the MCQ generation process can significantly alleviate these challenges. By utilizing advanced technologies such as natural language processing and machine learning, automated systems can generate a vast array of questions that cover the required content comprehensively. These systems can also ensure that the questions vary in difficulty and are free from common biases and errors, leading to more reliable and equitable assessments.

Benefits of Automated MCQ Generation:

The automation of MCQ generation offers several compelling benefits. First and foremost, it saves educators considerable time and effort, allowing them to focus more on teaching and interacting with students. Automated systems can quickly produce large pools of questions, which can be used to create varied and comprehensive assessments. This not only enhances the learning experience for students but also provides educators with a valuable tool for continuous assessment and feedback.

Furthermore, automated MCQ generation can improve the quality of assessments. By leveraging algorithms that ensure the clarity and accuracy of questions and answer choices, these systems can reduce the incidence of errors and ambiguities. They can also incorporate techniques to minimize bias, such as randomizing distractors and balancing question difficulty. This leads to fairer and more valid assessments, providing a more accurate measure of student learning.

The inefficiency and inconsistency of manual MCQ generation represent a significant challenge in the educational sector, impacting the quality and fairness of assessments. By automating this process, educational institutions can benefit from high-quality, reliable, and equitable MCQs that enhance the overall assessment strategy. This project seeks to develop a robust automated MCQ generation system, addressing the critical need for efficient and effective assessment tools in education. Through this innovative approach, educators can ensure that their assessments are both comprehensive and reflective of true student understanding, ultimately contributing to better learning outcomes.

PROPOSED SYSTEM

Automated MCQ Generator Leveraging NLP Techniques

The education sector continually seeks innovative methods to enhance learning experiences and assessment quality. One such innovation is the automated generation of multiple-choice questions (MCQs) from educational texts. The proposed system is an automated MCQ generator that leverages Natural Language Processing (NLP) techniques to streamline the creation of high-quality MCQs. This system aims to reduce the manual effort involved in question generation and ensure the questions are contextually relevant and varied.

System Overview:

The proposed system is designed to take an input text, such as an educational article or a textbook chapter, and generate relevant MCQs. The process is divided into several key stages: text preprocessing, keyphrase extraction, sentence mapping, sense generation, and distractor generation. Each stage employs specific NLP techniques and algorithms to ensure the accurate and efficient generation of questions.

System Functionalities:

The system works through a series of steps, each involving sophisticated NLP techniques to transform an input text into a set of well-structured MCQs. The primary functionalities include:

1. User Input:

- Users provide a text passage or learning material as input to the system. This can be any educational content from which MCQs need to be generated.

2. Text Preprocessing:

- The input text undergoes preprocessing, where it is cleaned and tokenized into sentences. This involves removing any extraneous information, such as punctuation and stop words, and splitting the text into manageable chunks.

3. Keyword Extraction:

- The system employs the RAKE (Rapid Automatic Keyword Extraction) algorithm to identify the top 25 keyphrases from the text. These keyphrases are crucial as they represent the core concepts and terms around which the MCQs will be formulated.

4. Sentence Tokenization and Filtering:

- The text is split into individual sentences. Sentences that are too short and unlikely to contain substantial information are filtered out, ensuring that only meaningful sentences are considered for question generation.

5. Sentence Mapping:

- Each sentence is analyzed and mapped to the extracted keyphrases. This mapping helps in identifying which sentences contain significant information about each keyphrase.

6. Sense Generation:

- Using the pywsd (Python Word Sense Disambiguation) library, the system determines the sense of each keyword in its context. This step ensures that the keyphrases are accurately understood in terms of their meaning and usage within the text.

7. Distractor Generation:

- Distractors, or incorrect answer options, are essential for creating challenging MCQs. The system generates plausible distractors using WordNet, a lexical database of English. If WordNet fails to provide suitable distractors, the system resorts to using ConceptNet, a semantic network that provides additional contextual knowledge.

8. Distractor Mapping:

- The generated distractors are mapped to the corresponding keyphrases, ensuring that each MCQ has one correct answer and three contextually similar distractors. This mapping is crucial to maintain the coherence and challenge level of the questions.

9. User Interface:

- The system features a user-friendly interface powered by Streamlit, a Python library for building interactive web applications. This interface enables users to effortlessly interact with the system, providing input texts and receiving the corresponding generated MCQs in a seamless manner.

Key Technologies and Methodologies:

The proposed system integrates several advanced NLP techniques and resources to achieve its objectives:

1. RAKE Algorithm:

- RAKE is used for keyword extraction. It effectively identifies keyphrases by analyzing the frequency and co-occurrence of words, ensuring that the most relevant terms are selected.

2. Pywsd Library:

- The pywsd library aids in word sense disambiguation. By understanding the precise meaning of each keyword in context, the system can generate more accurate and relevant questions.

3. WordNet:

- WordNet is utilized for distractor generation. It provides synonyms and related terms, which are essential for creating plausible but incorrect answer options.

4. ConceptNet:

- ConceptNet serves as a fallback for distractor generation. It enhances the system's ability to generate contextually similar distractors when WordNet's resources are insufficient.

Benefits and Applications:

The automated MCQ generator offers several significant benefits:

1. Efficiency:

- The system drastically reduces the time and effort required to create MCQs, allowing educators to focus more on teaching and less on assessment preparation.

2. Consistency:

- By using predefined algorithms and databases, the system ensures consistency in the quality and difficulty of the generated questions.

3. Scalability:

- The system can handle large volumes of text, making it suitable for generating MCQs from extensive educational materials.

4. Customization:

- Educators can tailor the input text and keyphrase extraction process to generate questions that align closely with specific learning objectives.

In conclusion, the proposed automated MCQ generator leverages the power of NLP and machine learning to transform educational texts into high-quality assessments efficiently. By automating the tedious process of question generation, this system has the potential to enhance educational practices, making assessments more robust, consistent, and scalable.

KEY TECHNOLOGIES AND METHODOLOGIES

The proposed system leverages several advanced Natural Language Processing (NLP) techniques and resources to achieve its objectives. By integrating these methodologies, the system enhances its ability to extract keywords, disambiguate word meanings, and generate both accurate questions and plausible distractors. This integration ensures the system's outputs are not only relevant but also contextually appropriate. The proposed system integrates several advanced NLP techniques and resources to achieve its objectives:

1. RAKE (Rapid Automatic Keyword Extraction) Algorithm
2. Pywsd Library
3. WordNet
4. ConceptNet

1.RAKE Algorithm:

Rapid Automatic Keyword Extraction (RAKE) is a fundamental component of the system, employed primarily for keyword extraction. RAKE is designed to identify keyphrases in a body of text by analyzing the frequency and co-occurrence of words. This method is particularly effective because it does not rely on a pre-defined dictionary or corpus, making it versatile for various text types and domains.

How RAKE Works

RAKE works by splitting the text into phrases based on the presence of stop words (commonly occurring but semantically empty words like "and," "the," etc.). It then calculates the frequency of each word and the degree of each word (the number of phrases the word appears in). The keyphrases are scored based on the word frequency

and their degree, allowing the algorithm to highlight terms that are both frequent and significant in their contexts.

Advantages of RAKE:

Domain Independence: RAKE does not depend on external knowledge bases or corpora, making it highly adaptable.

Efficiency: It can quickly process large texts, providing real-time keyword extraction.

Simplicity: The algorithm is straightforward to implement and tune, requiring minimal preprocessing.

By employing RAKE, the system ensures that the most relevant and contextually important terms are identified, which is crucial for subsequent processes like question generation and distractor creation.

2. Pywsd Library:

The **pywsd (Python Word Sense Disambiguation)** library plays a critical role in understanding the precise meanings of keywords within their specific contexts. Word sense disambiguation (WSD) is the process of determining which sense of a word is used in a given context when the word has multiple meanings.

Functionality of pywsd :

Pywsd leverages several algorithms and resources to perform WSD, including Lesk, Path similarity, and information content measures. These methods analyze the surrounding text to infer the most likely meaning of a word, ensuring that keywords are interpreted correctly.

Importance of WSD :

Accuracy in Context: By disambiguating words, the system can ensure that questions are generated based on the intended meaning of each keyword.

Enhanced Relevance: Accurate WSD leads to more relevant and meaningful questions and answers, improving the overall quality of the system's output.

By incorporating pywsd, the system enhances its ability to generate contextually accurate questions, as it can correctly interpret the meanings of keywords even in complex or ambiguous texts.

3. WordNet:

WordNet is a lexical database of English that groups words into sets of synonyms called synsets. It also provides short definitions and usage examples and records various semantic relations between these synonym sets.

Usage of WordNet in Distractor Generation:

In the proposed system, WordNet is used to generate distractors—plausible but incorrect answers. By providing synonyms and semantically related terms, WordNet helps create distractors that are contextually relevant and challenging, which is essential for maintaining the difficulty and educational value of the generated questions.

Key Features of WordNet

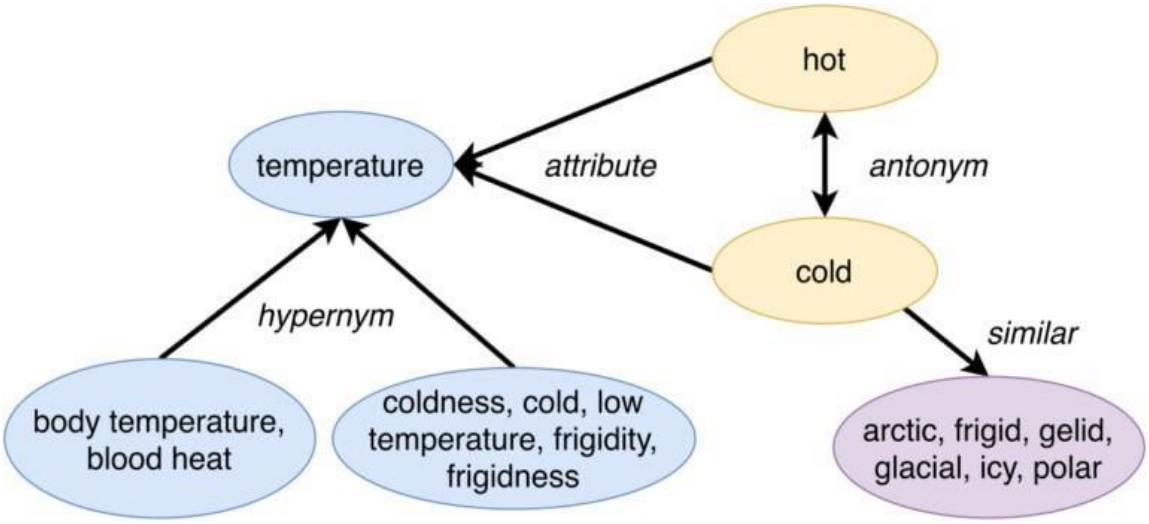
Rich Semantic Relationships: WordNet includes hypernyms (general terms), hyponyms (specific terms), meronyms (part-whole relationships), and antonyms, providing a comprehensive set of related terms.

Synonym Sets: Synsets allow the system to find words with similar meanings, aiding in the creation of plausible distractors.

Sense Definitions: The definitions and examples help ensure that the selected synonyms are contextually appropriate.

Using WordNet, the system can produce distractors that are not only linguistically similar to the correct answers but also contextually appropriate, thereby enhancing the challenge and educational value of the generated questions.

Simple WordNet Diagram:



4. ConceptNet:

ConceptNet is a semantic network that connects words and phrases with labeled, weighted edges, capturing various kinds of relationships between concepts. It is used in the system as a fallback mechanism for distractor generation when WordNet's resources are insufficient.

Role of ConceptNet in the System

ConceptNet enhances the system's ability to generate contextually similar distractors by providing a broader range of related terms and concepts. It captures more diverse relationships than WordNet, including commonsense knowledge and less direct associations.

Advantages of ConceptNet

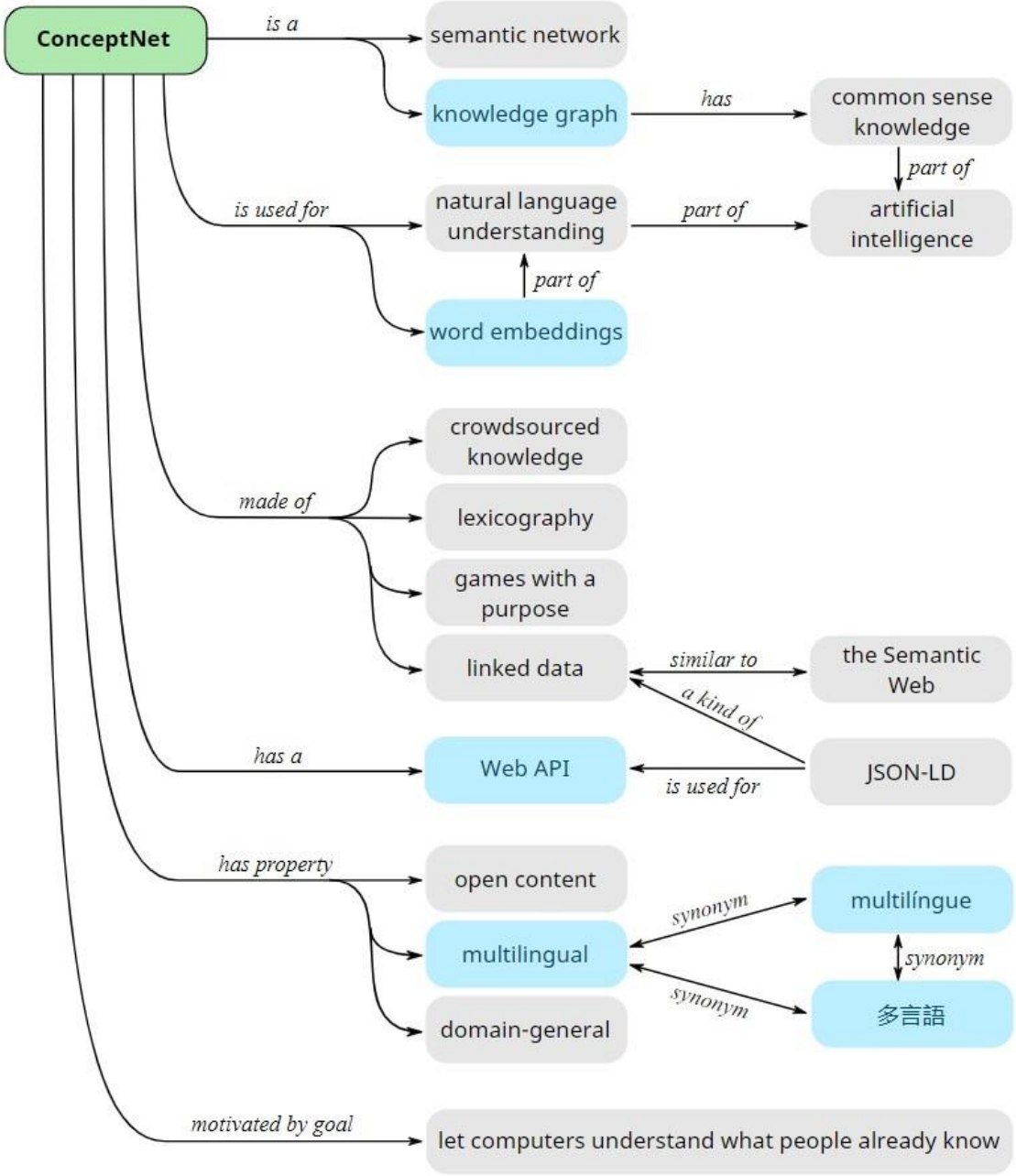
Commonsense Knowledge: ConceptNet includes everyday knowledge and relationships that are not captured by traditional lexical databases, making the distractors more natural and contextually varied.

Diverse Associations: It provides connections between concepts that might not be immediately apparent but are contextually relevant.

Fallback Flexibility: When WordNet does not provide sufficient or suitable distractors, ConceptNet ensures that the system can still generate high-quality distractors.

By integrating ConceptNet, the system gains the ability to produce a wider variety of plausible distractors, enhancing the robustness and adaptability of the question-generation process.

Simple ConceptNet Diagram:



Integration and Workflow:

Workflow Overview

Keyword Extraction: The text is first processed using the RAKE algorithm to extract the most relevant keywords.

Word Sense Disambiguation: The pywsd library is then used to disambiguate the senses of these keywords, ensuring their meanings are accurately understood in context.

Question Generation: Based on the disambiguated keywords, the system generates questions.

Distractor Generation: Using WordNet, the system generates distractors. If WordNet's resources are insufficient, ConceptNet is used as a fallback to ensure the distractors are contextually appropriate.

Benefits of the Integrated Approach

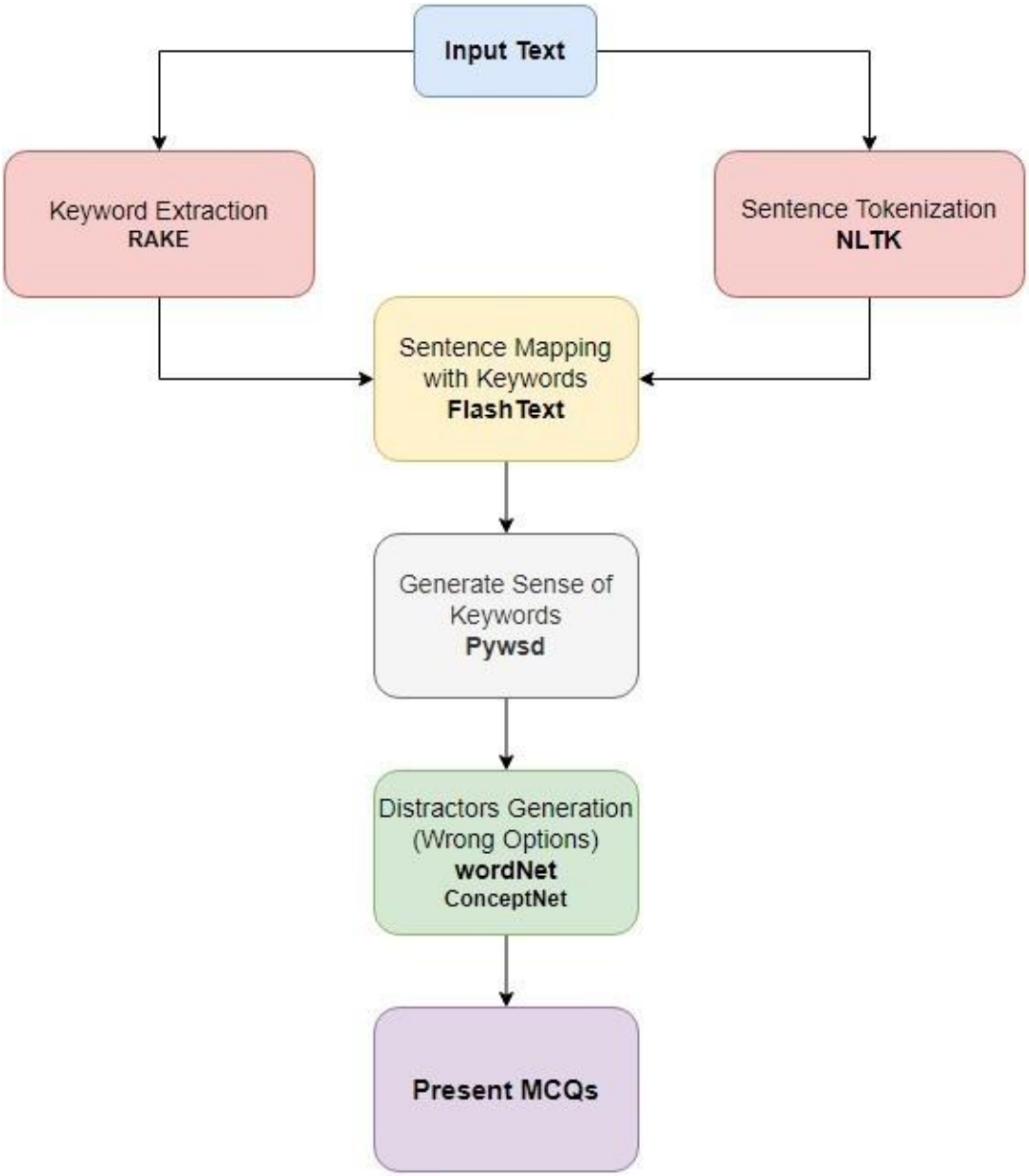
Enhanced Relevance and Accuracy: By combining RAKE, pywsd, WordNet, and ConceptNet, the system can generate questions and distractors that are highly relevant and contextually accurate.

Robustness: The use of multiple resources and fallback mechanisms ensures that the system can handle a wide variety of texts and domains.

Efficiency: Each component is optimized for its specific task, resulting in a streamlined and efficient workflow.

The proposed system's integration of advanced NLP techniques and resources like the RAKE algorithm, pywsd library, WordNet, and ConceptNet represents a sophisticated approach to automated question generation and distractor creation. By leveraging these technologies, the system achieves high relevance, accuracy, and contextual appropriateness in its outputs. This integration not only enhances the educational value of the generated questions but also ensures that the system can adapt to various text types and domains, making it a robust and versatile tool for NLP applications.

PROPOSED ARCHITECTURE DIAGRAM



Desscribe the Architecture step by step:

The automation of multiple-choice question (MCQ) generation from textual content represents a significant advancement in educational technology. This documentation outlines a systematic approach to creating an MCQ generation system, utilizing various natural language processing (NLP) techniques and resources. The process is divided into nine essential steps, each contributing to the efficient production of high-quality MCQs.

Step 1: Importing the Text File

The first step involves importing the text file or article that will serve as the basis for generating the MCQs. This step sets the foundation by reading the content of the file into a manageable format for further processing.

Step 2: Extracting Important Words (Keywords)

Keywords are crucial for formulating questions and options. This step employs RAKE (Rapid Automatic Keyword Extraction). to identify the most significant words in the text. PKE uses a multipartite graph-based unsupervised model to extract keywords, ensuring relevance and efficiency. Typically, the top 25 keyphrases or keywords are selected from the article.

Step 3: Sentence Tokenization

To facilitate easier processing, the article is split into individual sentences through a process known as sentence tokenization. Sentences with fewer than 15 words are excluded to ensure that each sentence provides sufficient context for understanding. This filtering step is vital for maintaining the quality of the generated questions.

Step 4: Mapping Sentences to Keywords

In this step, sentences containing the extracted keywords are mapped to their corresponding keywords. This mapping helps in generating questions where the sentence serves as the context and the keyword is the answer. Sentences are sorted in descending order of length to prioritize those that offer more comprehensive information.

Step 5: Generating the Sense of Keywords

Many English words have multiple meanings or senses. For instance, the word "fly" can refer to both the act of flying and the insect. Disambiguating the sense of each keyword is crucial for generating relevant distractors and accurate questions. Google's sense support tool is used to determine the appropriate sense of each keyword within its context.

Step 6: Generating Distractors from WordNet

Distractors are the incorrect options in an MCQ, designed to challenge the quiz taker. They must share the same sense as the correct answer to maintain the question's difficulty. Using WordNet, a comprehensive lexical database, we identify the hypernym (super-category) of the keyword and find its hyponyms (sub-categories) to generate contextually appropriate distractors.

Step 7: Generating Distractors from ConceptNet

When WordNet fails to provide sufficient distractors, ConceptNet, an online semantic network, is used as a fallback. ConceptNet offers additional distractors through its API, although it may not always guarantee a perfect sense match. This step ensures that we have a robust set of distractors even when WordNet resources are limited.

Step 8: Mapping Distractors to Keywords

The generated distractors are then mapped to their respective keywords. This step ensures that each keyword is paired with a set of plausible distractors, enhancing the quality and challenge of the MCQs. The system decides between using WordNet or ConceptNet based on the availability and relevance of the distractors.

Step 9: Presenting the MCQs

The final step involves presenting the generated MCQs in a clear and readable format. Sentences are converted into questions by replacing the keyword with a blank. The options, including the correct answer and distractors, are listed. This presentation ensures that the MCQs are user-friendly and ready for educational use.

This nine-step approach to automated MCQ generation from textual articles leverages advanced NLP techniques and resources such as RAKE, WordNet, and ConceptNet. By systematically extracting keywords, mapping sentences, disambiguating word senses, and generating distractors, this method ensures the creation of high-quality, contextually relevant multiple-choice questions. This process not only saves time but also enhances the learning and assessment experience in educational settings. Future improvements can further enhance the system's robustness and adaptability, making it an invaluable tool for automated assessments and educational technology.

CODE WALKTHROUGH

In [3]:

```
%%writefile app.py
import streamlit as st
from streamlit_lottie import st_lottie
import nltk
import string
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize
from flashtext import KeywordProcessor
from pywsd.lesk import adapted_lesk
from nltk.corpus import wordnet as wn
import requests
import random
import re
import PyPDF2
from PyPDF2 import PdfReader

from rake_nltk import Rake

r = Rake()



# Function to load Lottie animations
def load_lottie_url(url: str):
    r = requests.get(url)
    if r.status_code != 200:
        return None
    return r.json()

# Download necessary NLTK data
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('popular')

# Define the function to extract important words
def getImportantWords(art):
    r = Rake(stopwords=stopwords.words('english') + list(string.punctuation))
    r.extract_keywords_from_text(art)
    keyphrases = r.get_ranked_phrases_with_scores()
    result = [keyphrase for _, keyphrase in keyphrases[:25]]  # Get the top 25 keyphrases
    return result

# Split the text into sentences
def  splitTextToSents(art):
    s = [sent_tokenize(art)]
    s = [y for x in s for y in x]
    s = [sent.strip() for sent in s if len(sent) > 15]
    return s

# Map sentences to keywords
def mapSents(impWords, sents):
    processor = KeywordProcessor()
    keySents = {}
    for word in impWords:
        keySents[word] = []
        processor.add_keyword(word)
    for sent in sents:
        found = processor.extract_keywords(sent)
        for each in found:
            keySents[each].append(sent)
    for key in keySents.keys():
        temp = keySents[key]
        temp = sorted(temp, key=len, reverse=True)
        keySents[key] = temp
    return keySents

# Get the sense of the word
def getWordSense(sent, word):
    word = word.lower()
    if len(word.split()) > 0:
        word = word.replace(" ", "_")
    synsets = wn.synsets(word, 'n')
    if synsets:
        wup = max_similarity(sent, word, 'wup', pos='n')
        adapted_lesk_output = adapted_lesk(sent, word, pos='n')
        lowest_index = min(synsets.index(wup), synsets.index(adapted_lesk_output))
        return synsets[lowest_index]
    else:
        return None

# Get distractors from WordNet
def getDistractors(syn, word):
    dists = []
    word = word.lower()
    actword = word
    if len(word.split()) > 0:
        word.replace(" ", "_")
```

```
    hypernym = syn.hypernyms()
    if len(hypernym) == 0:
        return dists
    for each in hypernym[0].hyponyms():
        name = each.lemmas()[0].name()
        if name == actword:
            continue
        name = name.replace("_", " ")
        name = " ".join(w.capitalize() for w in name.split())
        if name is not None and name not in dists:
            dists.append(name)
    return dists

# Get distractors from ConceptNet
def getDistractors2(word):
    word = word.lower()
    actword = word
    if len(word.split()) > 0:
        word = word.replace(" ", "_")
    dists = []
    url = f"http://api.conceptnet.io/query?node=/c/en/{word}/n&rel=/r/PartOf&start=/c/en/{word}&limit=5"
    obj = requests.get(url).json()
    for edge  in  obj['edges']:
        link = edge['end']['term']
        url2 = f"http://api.conceptnet.io/query?node={link}&rel=/r/PartOf&end={link}&limit=10"
        obj2 = requests.get(url2).json()
        for edge in obj2['edges']:
            word2 = edge['start']['label']
            if word2 not in dists and actword.lower() not in word2.lower():
                dists.append(word2)
    return dists


# Load Lottie animation from a URL
lottie_animation = load_lottie_url("https://lottie.host/a17cef35-9c09-474f-9c26-225130dab967/dWJO2lpTSY.json")


# Streamlit App
st.markdown("<h1 style='color: skyblue;'>MCQ Generator</h1>", unsafe_allow_html=True)


# Description
st.markdown("""
## Generate Multiple Choice Questions from Text
This app allows you to generate multiple choice questions (MCQs) from any text you provide.
Simply upload a text file, specify the number of questions, and click the "Generate MCQs" button.
""")


# Align the info button to the top right corner
col1, col2 = st.columns([10, 2])
with col1:
    pass  # Empty column to adjust the layout

with col2:
    # Info button
    if st.button(label= "ℹ Info", key='info_button', help="Click for info"):
        st.sidebar.title("Information")
        st.sidebar.markdown("""
        Our project, the MCQ Generator, automatically generates multiple-choice questions (MCQs) from uploaded
        It utilizes natural language processing techniques to extract important words, map them to sentences, a
        distractors for each question, enhancing learning and assessment processes in various educational conte

        **Team Members:**
        -  Biswajit Kar
        -  Vivek Sharma
        -  Bishal Sarmah

        **References:**
        - [MCQ Generator Jupyter Notebook](https://github.com/vaishnaviverma/MCQ-Generator/blob/main/MCQG.ipynb
        - [Lottie Files](https://lottiefiles.com)
        """)



# Display Lottie animation
if lottie_animation:
    st_lottie(lottie_animation, height=300, key="coding")
else:
    st.error("Error loading Lottie animation. Please check the URL or try again later.")



## File upload handler for text files
uploaded_file = st.file_uploader("Choose a text file", type="txt")
if uploaded_file is not None:
    text = uploaded_file.read().decode("utf-8")

    num_mcqs = st.number_input("Enter the number of questions you want:", min_value=1, value=5)
```

```
    if st.button('Generate MCQs', key='generate_button'):
        impWords = getImportantWords(text)
        sents = splitTextToSents(text)
        mappedSents = mapSents(impWords, sents)

        mappedDists = {}
        correctAnswers = {}

        for each in impWords:
            if each not in mappedSents or not mappedSents[each]:
                continue
            distractors = random.sample([k for k in impWords if k != each], 3)
            mappedDists[each] = distractors


        iterator = 1  # To keep the count of the questions
        for each in mappedDists:
            if iterator > num_mcqs:
                break  # exit the loop if the desired number of MCQs has been reached
            if each not in mappedSents or not mappedSents[each]:  # Check if the keyword is not in mappedSents
                continue  # Skip this keyword if it's not found in mappedSents or has no mapped sentences
            sent = mappedSents[each][0]
            p = re.compile(each, re.IGNORECASE) # Converts into regular expression for pattern matching
            op = p.sub("_________", sent)  # Replaces the keyword with underscores(blanks)
            correct_answer = each.capitalize()  # The correct answer
            st.write(f"**Question {iterator}**: {op}")  # Prints the question along with a question number
            options = [each.capitalize()] + mappedDists[each]  # Capitalizes the options
            options = options[:4]  # Selects only 4 options
            opts = ['a', 'b', 'c', 'd']
            random.shuffle(options)  # Shuffle the options so that order is not always same
            for i, ch in enumerate(options):
                st.write(f"\t {opts[i]}) {ch}") # Print the options
            st.write(f"**Correct Answer**: {correct_answer}")  # Print the correct answer
            st.write("\n")
            iterator += 1  # Increase the counter
```

```
Overwriting app.py
```

We are giving our Git hub link where anyone can find the source code and the web app code for deploy the project in streamlit localhost :

Biswajit Kar : https://github.com/1905biswajit/NLP-MCQ-GEN

Bishal Sarmah : https://github.com/Bishalsarmah/MCQ-questions-and-answer-generator

Vivek Sharma : https://github.com/viveksharma777/mcqgen-using-nlp

RESULT AND DISCUSSION

The primary aim of this project is to develop an automated Multiple Choice Questions (MCQ) generator. The generated MCQs are designed to assess understanding of a text by focusing on key concepts extracted from the provided article. The implementation follows a structured nine-step approach to ensure the generation of relevant and challenging MCQs. This section discusses the results obtained from deploying this project and the effectiveness of the methodology employed.

Results

MCQ Generation Process:

The MCQ generation process involves several crucial steps, each contributing to the creation of high-quality questions. Below are the detailed results from each step:

Importing the Text File

The project begins by importing the text file that contains the article to be used for MCQ generation. This is accomplished using Python's open and read functions. The ability to handle various text formats efficiently ensures that the project can work with a wide range of input sources.

Extraction of Important Words (Keywords)

Using the RAKE library, the project extracts the most important keywords from the text. This step employs a multipartite unsupervised model, which ensures efficient extraction by identifying the top 25 keyphrases or keywords. These keywords are pivotal as they form the basis for the questions and options.

Splitting the Article into Sentences

The article is split into individual sentences through sentence tokenization. Sentences with fewer than 15 words are excluded to ensure sufficient context for understanding.

This step creates an array of sentences, making it easier to map keywords to relevant sentences later.

Mapping Sentences to Keywords

Sentences containing the extracted keywords are mapped to those keywords. This mapping is crucial for generating questions where the keyword will be replaced with a blank. Sentences are sorted by length in descending order, ensuring that the most informative sentences are selected for question generation.

Generating Sense of Keywords

To address the issue of polysemy (multiple meanings of a word), the sense of each keyword is determined. This is vital for creating relevant distractors and ensuring that the context of the keyword is maintained in the generated questions.

Generating Distractors from WordNet

Distractors, or incorrect options, are generated using WordNet. By identifying the hypernyms (super-categories) and hyponyms (sub-categories) of the keywords, distractors are created that share the same sense as the keyword. This method ensures that the distractors are plausible and contextually appropriate.

Generating Distractors from ConceptNet

In cases where WordNet fails to generate suitable distractors, ConceptNet is used as an alternative. ConceptNet provides additional distractors through its online API, although they may not always perfectly match the sense of the keyword.

Mapping Distractors to Keywords

The final step in creating distractors involves mapping them to the respective keywords. This step checks the sense of the keyword and decides whether to use WordNet or ConceptNet for distractor generation.

Presenting the MCQs

The sentences mapped to keywords are converted into questions by replacing the keyword with a blank. The questions, along with four options (one correct keyword and three distractors), are presented in a user-friendly format. The questions are numbered and displayed clearly to facilitate easy understanding and usability.

Deployment in Streamlit

The project is deployed using Streamlit, a popular framework for creating interactive web applications. Users can upload a text file and specify the number of MCQs they wish to generate. The system then processes the input and generates the requested number of MCQs, providing users with a convenient and efficient tool for quiz creation.

Discussion

Effectiveness of the MCQ Generation

The project successfully automates the generation of MCQs, making it a valuable tool for educators and learners. The use of PKE for keyword extraction ensures that the most relevant concepts are identified, while the mapping of sentences to keywords guarantees that questions are meaningful and contextually accurate. The dual approach for generating distractors (WordNet and ConceptNet) enhances the robustness of the system, ensuring that distractors are relevant and challenging.

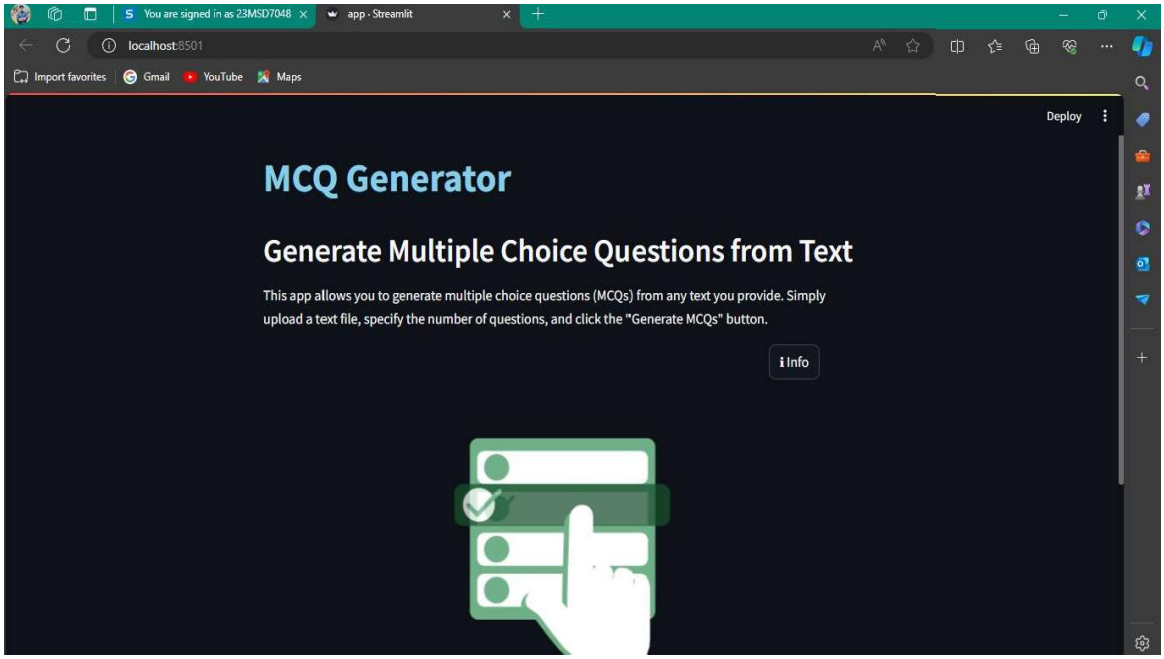
Challenges and Limitations

Sense Disambiguation: While the project employs Google's sense support for disambiguating keyword meanings, the accuracy of sense determination can vary. This might sometimes result in distractors that are not entirely appropriate.

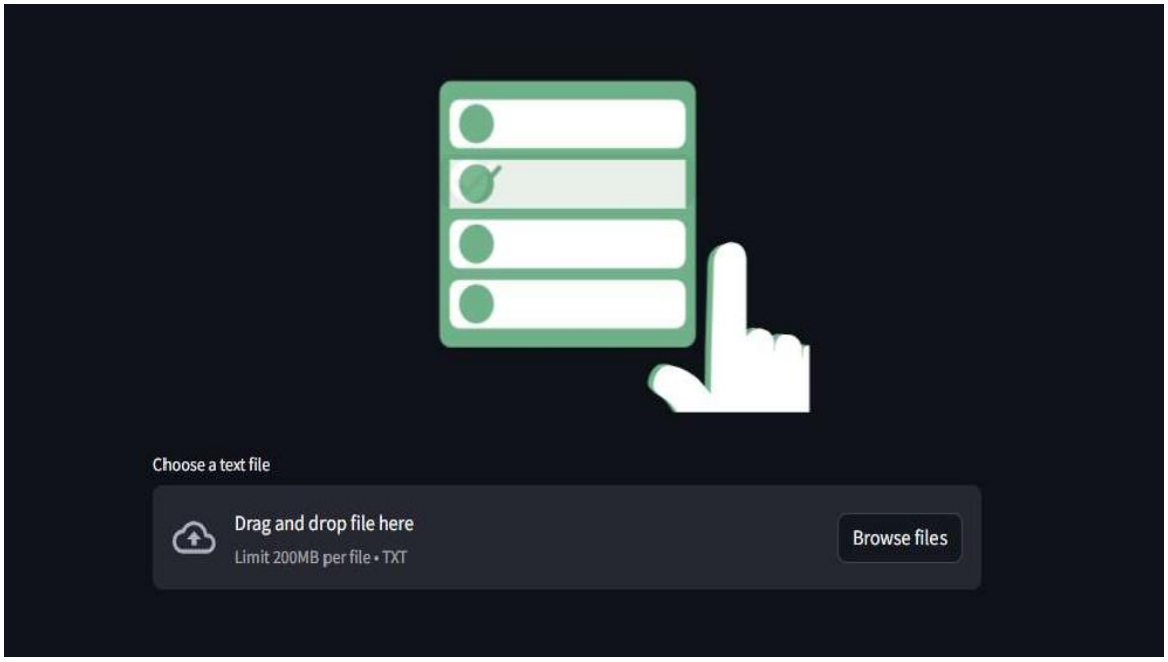
Dependency on External APIs: The use of ConceptNet, an online API, introduces a dependency on internet connectivity and the availability of the API. This could be a limitation in environments with restricted internet access.

Contextual Relevance: Ensuring that the generated distractors maintain the same sense as the keyword is challenging. Although the project uses WordNet and ConceptNet, there might be instances where the distractors do not perfectly fit the context.

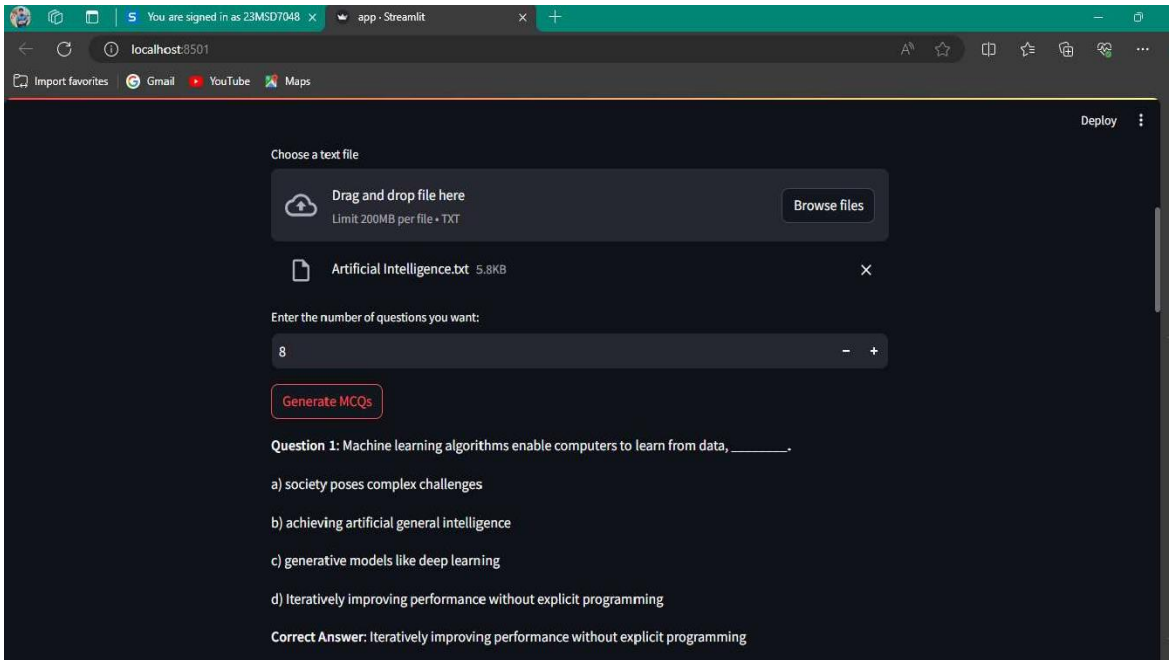
SNAPSHOT:



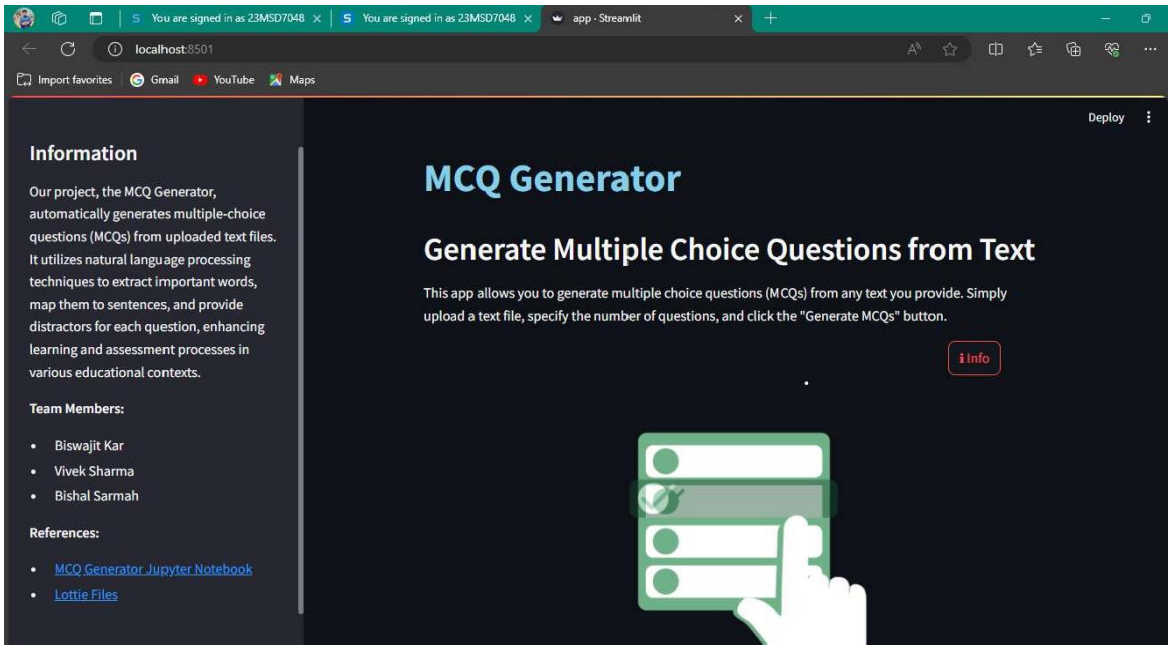
This is the user interface(UI) of our project in streamlit.



Here user can browse any text file for generate MCQs form the text file.



User can select how many MCQs they want.And then System generate that numbers of MCQs as output.



Here user can see about our app,team member’s names,and references .

FUTURE ENHANCEMENTS

Future Enhancements for the MCQ Questions and Answers Generator

Advanced NLP Techniques

One of the most promising directions for enhancing the MCQ generator is the incorporation of advanced Natural Language Processing (NLP) techniques, specifically leveraging models like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer). These models, known for their deep understanding of context and language, can significantly improve the accuracy of sense disambiguation and keyword extraction.

BERT and GPT for Improved Accuracy

- **Sense Disambiguation:** BERT's ability to understand context at a nuanced level will help in accurately determining the sense of a keyword within a specific sentence. This is particularly useful for words with multiple meanings, ensuring that the generated questions and distractors are contextually appropriate.

- **Keyword Extraction**: By using BERT or GPT, we can move beyond traditional keyword extraction methods. These models can identify not just key phrases but also understand the importance of concepts in relation to the overall text, leading to more relevant and insightful MCQs.

Contextual Analysis

Enhancing the system's ability to perform deeper contextual analysis will ensure that the distractors generated are closely aligned with the keyword's sense within the given sentence. This involves:

- **Semantic Similarity:** Utilizing semantic similarity measures to ensure that the distractors are not only grammatically correct but also make sense within the context of the question. This will make the questions more challenging and realistic.

- **Sentence Embeddings:** Implementing sentence embeddings to understand the context of entire sentences, which can help in generating more coherent and contextually appropriate distractors.

User Feedback Mechanism

Integrating a user feedback mechanism into the system will be crucial for continuous improvement. This can be achieved by allowing users to rate the quality of the generated MCQs. Feedback can be collected on various parameters such as relevance, difficulty, and clarity.

Benefits of User Feedback

- **Algorithm Refinement:** Feedback from users will help in identifying common issues and areas for improvement, allowing for iterative refinement of the algorithms.

- **Adaptive Learning:** The system can use the feedback to learn and adapt over time, improving its performance based on user inputs.

Multilingual Support

Expanding the system to support multiple languages will make it accessible to a broader audience. This involves:

- **Language Detection:** Automatically detecting the language of the input text to apply appropriate processing techniques.

- **Translation Services:** Integrating with translation services to handle text in various languages, ensuring that the system can generate MCQs in the language of the input text.

Challenges and Solutions

- **Semantic Nuances:** Different languages have unique semantic nuances, which can be challenging to handle. Using multilingual models like mBERT (Multilingual BERT) can help address these challenges.

- **Localization:** Ensuring that the generated questions are culturally and contextually appropriate for different languages and regions.

Expansion to True/False Questions

In addition to MCQs, future iterations of the system can explore generating True/False questions. This will diversify the types of assessment materials the system can produce, catering to a wider range of educational needs and assessment styles.

Implementation Strategy

- **Statement Analysis:** The system will analyze statements within the text to determine their veracity.

- **Fact Verification:** Using external sources or databases to verify the factual accuracy of statements, ensuring that True/False questions are reliable.

Enhanced Input Flexibility

Currently, the system accepts text files as input. To increase its versatility and accessibility, we plan to extend its compatibility to various formats, including:

- **PowerPoint Presentations:** Extracting text from slides to generate MCQs.

- **PDFs:** Parsing PDF documents to retrieve text content for question generation.

- **Images:** Using Optical Character Recognition (OCR) to extract text from images.

- **Webpage Links:** Scraping text content from webpages to generate questions.

Benefits of Expanded Input Options

- **Versatility:** Allowing users to input content from various sources increases the tool's versatility.

- **Accessibility:** Educators can use the tool regardless of the format in which their content is available, making it more accessible.

Revolutionizing Educational Assessment

The MCQ Questions and Answers Generator represents a significant advancement in the field of educational assessment tools. By automating the process of question generation, it not only saves time but also ensures the creation of high-quality assessment materials. Our system leverages natural language processing and machine learning techniques to extract relevant information from text and generate challenging yet plausible multiple-choice questions.

Future Vision

Looking ahead, future iterations of the system will not only focus on generating MCQs but also explore expanding its capabilities to include True/False questions. This expansion will diversify the types of assessment materials it can generate, catering to a wider range of educational needs and assessment styles. By providing educators with even greater flexibility in crafting evaluation tools, we aim to enhance the overall learning experience.

Enhancing Input Flexibility

In our future development plans, enhancing the system's input flexibility is a key focus. While currently designed to accept text files, we aim to extend its compatibility to various formats such as PowerPoint presentations, PDFs, images containing text, and even webpage links. This broader range of input options will make the tool more versatile and accessible to users across different platforms and content sources.

Commitment to Educational Excellence

In essence, the MCQ Questions and Answers Generator is poised to revolutionize the way educators create assessment materials, offering efficiency, consistency, and adaptability. As we continue to refine and expand its capabilities, we remain committed to supporting educators in their pursuit of providing meaningful and effective learning experiences for their students. By leveraging the latest advancements in NLP and machine learning, we aim to create a tool that not only meets the current needs of educators but also anticipates and adapts to future trends in educational assessment.

CONCLUSION

The MCQ Questions and Answers Generator project represents a significant advancement in educational technology, providing a robust tool for automating the creation of high-quality multiple-choice questions. By leveraging state-of-the-art natural language processing (NLP) techniques and machine learning models, the system effectively extracts relevant keywords from text and generates challenging yet plausible questions. This automation not only saves valuable time for educators but also ensures consistency and accuracy in assessment materials.

The implementation of the system follows a comprehensive nine-step process, starting from importing the text file to presenting the final MCQs. Key aspects of this process include keyword extraction using Python RAKE, sentence tokenization, sense disambiguation, and the generation of distractors using resources like WordNet and ConceptNet. Each step is designed to ensure that the generated questions are contextually relevant and pedagogically sound.

Deploying the project through Streamlit enhances its accessibility and usability, allowing users to upload text files and specify the number of MCQs they require. The interactive interface makes it easy for educators to generate customized quizzes tailored to their specific needs, providing a practical and efficient solution for educational assessments.

Looking ahead, there are several promising avenues for enhancing the quality and effectiveness of the MCQ generator. Incorporating advanced NLP models like BERT and GPT can improve the accuracy of keyword extraction and sense disambiguation. Implementing deeper contextual analysis will ensure that distractors are closely aligned with the intended meaning of the keywords. Additionally, integrating a user feedback mechanism will enable continuous improvement based on real-world usage, and expanding the system to support multiple languages will broaden its applicability.

Future iterations of the system could also explore generating True/False questions, further diversifying the types of assessment materials available to educators. Enhancing the

system’s input flexibility by supporting various formats such as PowerPoint presentations, PDFs, images, and webpage links will make it even more versatile and accessible.

In essence, the MCQ Questions and Answers Generator is poised to revolutionize the way educators create assessment materials. By automating the process, it offers efficiency, consistency, and adaptability, making it an invaluable tool for modern education. As we continue to refine and expand its capabilities, we remain committed to supporting educators in their mission to provide meaningful and effective learning experiences for their students. Through ongoing innovation and user-centered development, this project aims to stay at the forefront of educational technology, continuously adapting to meet the evolving needs of the educational community.

REFERENCES

1. https://github.com/vaishnaviverma/MCQ-Generator

2. https://en.wikipedia.org/wiki/WordNet

3. https://conceptnet.io

4. https://streamlit.io

THANK YOU