Akaike Technologies Assignment

NLP Assignment

MCQ Question Generator using Spacy Library

This code generates multiple-choice questions (MCQs) based on a given context paragraph using the Spacy library. The MCQs are designed to have multiple correct answer choices for added variety. The generated MCQs are then displayed to the user.

Import libraries

Imports the required libraries, spacy for natural language processing and random for generating random choices.

```
import spacy
import random
```

Load English language model

Loads the English language model "en_core_web_sm" from Spacy for processing text

```
nlp = spacy.load("en_core_web_sm")
```

Get the input of paragraph

```
paragraph = input("Enter the Content of Paragraph: \n")
```

Function call

Call the get_questions(paragraph) function with paragraph content parameter

```
mca_questions = get_questions(paragraph)
```

Function

Defines a function get_questions that takes a context paragraph and the number of questions to generate.

```
def get_questions(paragraph: str):
        doc = nlp(paragraph)
```

MCQ Generation Function:

Defines a function generate_mcq_questions to create MCQs with multiple correct answers.

```
def generate_mcq_questions(question, correct_answers, other_options, num_options=3):
    options = correct_answers + other_options
    random.shuffle(options)
    mcq = {"question": question, "options": options, "correct_answers": correct_answers}
    return mcq
```

Generate the random Questions function:

Defines a function generate_random_question that randomly selects a sentence and a word from the context to create a ¦ll-in-the-blank question.

```
def generate_random_question():
    sentence = random.choice(list(doc.sents))
    options = random.choice([token for token in sentence if not token.is_punct])
    question_text = sentence.text.replace(options.text, "*_____*")
    correct_answers = [options.text]
    other_options = [token.text for token in doc if token.is_alpha and token.text ! = correct_answers[0]]
    num_correct_options = random.randint(1, 2)
    correct_answers.extend(random.sample(other_options, num_correct_options))
    num_other_options = min(3 - num_correct_options, len(other_options))
    other_options = random.sample(other_options, num_other_options)
    mcq = generate_mcq_questions(question_text, correct_answers, other_options)
    return mcq
```

Generate questions & Process and format questions:

Generates a list of questions using the generate_variety_question function based on the context and speci¦ed number of questions. Formats the generated questions, options, and correct answers into strings.

```
mca_questions = []
  for i, question in enumerate(questions, start=1):
    question_str = f"Q{i}: {question['question']}\n"
    options_str = ""
    for j, option in enumerate(question['options']):
      options_str += f"{chr(96+j+1)}. {option}\n"
    correct_options_formatted = " & ".join([f"({chr(97+question['options'].index(ans))})" for ans in
question['correct_answers']])
    correct_options_str = f"Correct Options: {correct_options_formatted}"
    mca_question = f"{question_str}{options_str}{correct_options_str}\n"
    mca_questions.append(mca_question)
  return mca_questions
```

Print Questions:

Takes user input for the context and number of questions, generates MCQs, and prints each question along with options and correct answers.

```
paragraph = input("Enter the Content of Paragraph: \n")
mca_questions = get_questions(paragraph)
print("\n")
for question in mca_questions:
  print(question)
```

Final Code :

```
import spacy
import random

def get_questions(paragraph: str):
```

```
    def generate_mcq_questions(question, correct_answers, other_options, num_options=3):
        options = correct_answers + other_options
        random.shuffle(options)
        mcq = {"question": question, "options": options, "correct_answers": correct_answers}
        return mcq

    def generate_random_question():
        sentence = random.choice(list(doc.sents))
        options = random.choice([token for token in sentence if not token.is_punct])
        question_text = sentence.text.replace(options.text, "*_____*")
        correct_answers = [options.text]
        other_options = [token.text for token in doc if token.is_alpha and token.text !=
correct_answers[0]]
        num_correct_options = random.randint(1, 2)
        correct_answers.extend(random.sample(other_options, num_correct_options))
        num_other_options = min(3 - num_correct_options, len(other_options))
        other_options = random.sample(other_options, num_other_options)
        mcq = generate_mcq_questions(question_text, correct_answers, other_options)
        return mcq

    doc = nlp(paragraph)
    questions = [generate_random_question() for _ in range(5)]
    mca_questions = []
    for i, question in enumerate(questions, start=1):
        question_str = f"Q{i}: {question['question']}\n"
        options_str = ""
        for j, option in enumerate(question['options']):
            options_str += f"{chr(96+j+1)}. {option}\n"
        correct_options_formatted = " & ".join([f"({chr(97+question['options'].index(ans))})" for ans in
question['correct_answers']])
        correct_options_str = f"Correct Options: {correct_options_formatted}"
        mca_question = f"{question_str}{options_str}{correct_options_str}\n"
        mca_questions.append(mca_question)
    return mca_questions

nlp = spacy.load("en_core_web_sm")
paragraph = input("Enter the Content of Paragraph: \n")
mca_questions = get_questions(paragraph)
print("\n")
for question in mca_questions:
    print(question)
```

Output:

Enter the Content of Paragraph:

The Company had become the Diwan, but it still saw itself primarily as a trader. It wanted a large revenue income but was unwilling to set up any regular system of assessment and collection. The effort was to increase the revenue as much as it could and buy fine cotton and silk cloth as cheaply as possible. Within five years, the value of goods bought by the Company in Bengal doubled. Before

1765, the Company had purchased goods in India by importing gold and silver from Britain. Now the revenue collected in Bengal could finance the purchase of goods for export. Soon it was clear that the Bengal economy was facing a deep crisis. Artisans were deserting villages since they were being forced to sell their goods to the Company at low prices. Peasants were unable to pay the dues that were being demanded from them. Artisanal production was in decline, and agricultural cultivation showed signs of collapse. Then in 1770, a terrible famine killed ten million people in Bengal. About one-third of the population was wiped out.

Q1: Now the revenue collected in Bengal could finance the purchase of goods ____ export.

a. economy

b. a

c. trader

d. for

Correct Options: (d) & (b) & (a)

Q2: Peasants were unable to pay the dues that were ____ demanded from them.

a. and

b. being

c. was

d. a

Correct Options: (b) & (c)

Q3: Peasants were unable to pay the dues that were ____ demanded from them.

a. effort

b. being

c. them

d. from

Correct Options: (b) & (d) & (a)

Q4: Before 1765, the ____ had purchased goods in India by importing gold and silver from Britain.

a. showed

b. were

c. bought

d. Company

Correct Options: (d) & (a) & (c)

Q5: Artisanal production was in decline, and agricultural cultivation ____ signs of collapse.

a. the

b. purchase

c. since

d. showed

Correct Options: (d) & (a) & (c)

Conclusion:

The program takes a paragraph from the user and creates multiple-choice questions using Spacy. It generates different question formats, like fill-in-the-blank, with various correct options. The user specifies the number of questions, and the program displays these questions along with their answer choices.