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 necessary 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")
```

Define function

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

```
def get_mca_questions(context: str, num_questions: int):
  doc = nlp(context)
```

Define MCQ generation function

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

```
def generate_mcq_with_multiple_correct(question, correct_answers, other_options, num_options=4):
        options = correct_answers + other_options
        random.shuffle(options)

        mcq = {
            "question": question,
            "options": options,
            "correct_answers": correct_answers
        }
```

Generate a variety question

Defines a function generate_variety_question that randomly selects a sentence and a word from the context to create a fill-in-the-blank question

```
def generate_variety_question():
        # randomly select the sentence from content
        sentence = random.choice(list(doc.sents))

        # randomly choose non- pronounciation words from sentence as blank word
        blank_word = random.choice([token for token in sentence if not token.is_punct])

        # create a question text with blank word ----
        question_text = sentence.text.replace(blank_word.text, "______")

        #set correct answers to the blank word
        correct_answers = [blank_word.text]

        #generating other possible answers
        other_options = [token.text for token in doc if token.is_alpha and token.text != correct_answers[0]]

        #randonly determine how many correct options
        num_correct_options = random.randint(1, 2)

        #randomly select correct options to the list of options
        correct_answers.extend(random.sample(other_options, num_correct_options))

        # no of incorrect answers
        num_other_options = min(4 - num_correct_options, len(other_options))
        other_options = random.sample(other_options, num_other_options)

        #generationg final MCQ
        mcq = generate_mcq_with_multiple_correct(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 specified number of questions. Formats the generated questions, options, and correct answers into strings.

```
#created empty list to store multiple choice questions
mca_questions = []

    # enumerate function is used to iterate over the questions
    for i, question in enumerate(questions, start=1):

        #created a string for question number and question text.
        question_str = f"Q{i}: {question['question']}\n"

        #created empty string to store option for current question
        options_str = ""

        #iterate through options
        for j, option in enumerate(question['options']):
            options_str += f"{j+1}. {option}\n"

        #format the correct answers into human redable format
        correct_options_formatted = " & ".join([f"({chr(97+question['options'].index(ans))})" for ans in question['correct_answe

        #combine the questions and options and format the correct answes
        correct_options_str = f"Correct Options: {correct_options_formatted}"

        #add the questions into formated questions
        mca_question = f"{question_str}{options_str}{correct_options_str}\n"
        mca_questions.append(mca_question)

    #return the MCQ questions
    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.

```
#user input for paragraph
context = input("Enter the paragraph: ")

#no of questions user want to generate
num_questions = int(input("Enter the number of questions: "))

#calls the function and generate MCQ questions
mca_questions = get_mca_questions(context, num_questions)
for question in mca_questions:
    print(question)
```

MCQ Final Code

```
import spacy
import random

# Load English language model
nlp = spacy.load("en_core_web_sm")
```

```python
def get_mca_questions(context: str, num_questions: int):
    doc = nlp(context)

    def generate_mcq_with_multiple_correct(question, correct_answers, other_options, num_options=4):
        options = correct_answers + other_options
        random.shuffle(options)

        mcq = {
            "question": question,
            "options": options,
            "correct_answers": correct_answers
        }

        return mcq

    def generate_variety_question():
        sentence = random.choice(list(doc.sents))
        blank_word = random.choice([token for token in sentence if not token.is_punct])

        question_text = sentence.text.replace(blank_word.text, "______")
        correct_answers = [blank_word.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)  # Generate 1 or 2 correct options
        correct_answers.extend(random.sample(other_options, num_correct_options))

        num_other_options = min(4 - num_correct_options, len(other_options))
        other_options = random.sample(other_options, num_other_options)

        mcq = generate_mcq_with_multiple_correct(question_text, correct_answers, other_options)
        return mcq

    questions = [generate_variety_question() for _ in range(num_questions)]

    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"{j+1}. {option}\n"

        correct_options_formatted = " & ".join([f"({chr(97+question['options'].index(ans))})" for ans in question['correct_answe

        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

context = input("Enter the paragraph: ")
num_questions = int(input("Enter the number of questions: "))
mca_questions = get_mca_questions(context, num_questions)
for question in mca_questions:
    print(question)
```

```
Enter the paragraph: On 12 August 1765, the Mughal emperor appointed the East India Company as the Diwan of Bengal. The actual even
Enter the number of questions: 5
Q1: Now it had ______ think of administering the land and or anising its revenue resources.
1. financial
2. to
3. the
4. an
5. Diwani
Correct Options: (b) & (c) & (e)

Q2: In this ______ we will see how the Company came to colonise the countryside, organise revenue resources, redefine the rights of
1. clearly
2. its
3. this
4. chapter
5. As
Correct Options: (d) & (b) & (e)

Q3: But in ______ painting above, ______ event is shown as a majestic occasion, taking place in a grand setting.
1. became
2. it
3. the
4. in
5. by
Correct Options: (c) & (e)
```

```
Q4: The grant of Diwani clearly was ______ such event in British imagination.
1. some
2. one
3. colonise
4. the
5. grand
Correct Options: (b) & (c)

Q5: This had to be done in a way that could yield enough revenue to meet the ______ expenses of the company.
1. took
2. was
3. of
4. growing
5. and
Correct Options: (d) & (a) & (b)
```

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. This is useful for making interactive quizzes or assessments.

✓ 7s completed at 10:17 PM