Natural Language Processing

Objective: The goal of this internship assignment is to test your proficiency in natural language processing. You will be tasked with developing a solution that can automatically generate objective questions with multiple correct answers based on a given chapter from a subject. The generated questions should test the reader's understanding of the chapter and have more than one possible correct answer to increase the complexity and challenge of the questions. The generated questions should not only test the reader's comprehension of the chapter but also encourage them to think beyond the surface level and explore different perspectives and possibilities. Ultimately, the objective of this project is to develop a robust and accurate solution that can aid educators in creating engaging and challenging assessments for their students.

Libraries and Tools:

The code utilizes several libraries and tools to achieve the objective:

- Transformers: A library for natural language processing tasks.
- PyPDF2: A library for processing PDF files.
- T5 Model: A pre-trained language model used for text generation.
- PyTorch: Required as a dependency for the 'transformers' library.
- SentencePiece: A tokenization library.

Steps:

1. Installation: The code begins with installing the necessary libraries using pip, which includes Transformers, PyPDF2, PyTorch, and SentencePiece.
2. Import Libraries: The required libraries are imported, including Transformers, PyPDF2, and PyTorch.
3. Model and Tokenizer Initialization: It initializes the T5 model and tokenizer with the "t5-small" pre-trained model.
4. PDF File Path: It specifies the relative path to the PDF file to be processed and converts it into an absolute path. The absolute path is printed for reference.
5. Text Extraction from PDF:
   - It defines a function extract_text_from_pdf that takes the absolute path of the PDF file as input.
   - The function extracts text from the PDF file by reading its pages.
   - It returns the extracted text.
6. Generating Questions:
   - It defines a function generate_question that generates a question from a given prompt using the T5 model and tokenizer.
   - The input text is formatted as a question, encoded, and then passed to the model for question generation.
   - The generated question is returned.
7. Generating Questions and Answers:
   - It extracts text from the PDF file using extract_text_from_pdf.
   - The text is split into smaller sections, which could be paragraphs, headings, or other logical divisions.
   - It generates questions for each section and stores them in a list, including the passage and generated question.
8. Multiple-Choice Questions:
   - It defines a function generate_multiple_choice_question that generates a multiple-choice question with options.
   - The correct answer and multiple choice options are defined.
   - A random incorrect option is selected from the list of incorrect options to create the multiple-choice question.
9. Multiple-Choice Questions Generation:
   - It generates multiple-choice questions for each section and stores them in a list.
   - Questions include the section and the generated multiple-choice question.

10. Saving Questions and Answers:
    - It displays the generated questions and answers in the console for reference.
    - It opens a file named "generated_multiple_choice_questions_and_answers.txt" for writing.
    - It iterates through the list of questions and answers, writing them to the file with labels and separators to distinguish between entries.
11. Objective Statement:
    - The code concludes with an objective statement that describes the purpose of the assignment and the importance of the generated questions in educational assessments.

Summary:

- The code provides a framework for generating multiple-choice questions with multiple correct answers based on the input content.
- Remember that the code aims to generate diverse and challenging questions to test the reader's understanding and encourage critical thinking.
- Please make the necessary modifications to adapt the code to the specific requirements.

Program and Results:

```
# natural language processing tasks
!pip install transformers # For PDF processing
!pip install PyPDF2
# PyTorch as a dependency for the 'transformers' library
!pip install torch
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.34.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.17.3) Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.2)Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.15,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.14.1)Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->transformers) Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0. Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3 Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7) Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2023.7. Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.10/dist-packages (3.0.1)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.1.0+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.12.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages
```

(from torch) (4.5.0) Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.3) Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)

```
# Import T5 model and tokenizer
from transformers import T5ForConditionalGeneration, T5Tokenizer # working with deep learning models
import torch
# working with PDF files import PyPDF2


# tokenization library
!pip install sentencepiece
# specifies the pre-trained T5 model to be used model_name_or_id = "t5-small"
# Initialize the T5 model by loading the pre-trained model using the specified model_name_or_id. model = T5ForConditionalGeneration.from_pretrained(model_name_or_id)
# Initialize the T5 tokenizer by loading the pre-trained tokenizer using the same model_name_or_id. tokenizer = T5Tokenizer.from_pretrained(model_name_or_id)
model_name_or_id = "t5-small"
```

Requirement already satisfied: sentencepiece in /usr/local/lib/python3.10/dist-packages (0.1.99)

Downloading (…)okenizer_config.json: 100% 2.32k/2.32k [00:00<00:00, 29.3kB/s]

Downloading (…)ve/main/spiece.model: 100% 792k/792k [00:00<00:00, 1.61MB/s]

Downloading (…)/main/tokenizer.json: 100% 1.39M/1.39M [00:00<00:00, 11.0MB/s]

You are using the default legacy behaviour of the <class 'transformers.models.t5.tokenization_t5.T5Tokenizer'>. This is expected, anSpecial tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.

```
import os

# Define the actual relative path to your PDF file relative_path = "sample_data/chapter-2.pdf"

# Get the absolute path to the file
absolute_path = os.path.abspath(relative_path)
```

```
# Print the absolute path
print("Absolute Path:", absolute_path)
```

```
Absolute Path: /content/sample_data/chapter-2.pdf
```

```
# Initialize the path to the PDF file to be processed (provide
the actual path)pdf_path = ""
# Define a function to extract text from a
PDF file# Parameters:

# - absolute_path (str): The absolute path to the PDF file to
be processeddef extract_text_from_pdf(absolute_path):
  # Initialize an empty string to store the
    extracted textpdf_text = ""
    # Open the PDF file in binary
    read mode pdf_file =
    open(absolute_path, "rb")
    # Create a PDF reader object
    pdf_reader = PyPDF2.PdfReader(pdf_file)
    # Get the total number of pages
    in the PDF num_pages =
    len(pdf_reader.pages)
     # Loop through each page
    of the PDFfor page_num in
    range(num_pages):
      # Get the text content of the current page and append it
        to the resultpage = pdf_reader.pages[page_num]
        pdf_text +=
    page.extract_text() # Close
    the PDF file
    pdf_file.close()
    # Return the
    extracted text return
    pdf_text
# Call the function to extract text from a PDF file and store the
result in pdf_text# Provide the actual absolute path to the PDF
pdf_text = extract_text_from_pdf(absolute_path)

# Define a function to generate a question from a given prompt using
a language model.# Parameters:
# - prompt (str): The text prompt for generating
the question.# - model: The pre-trained language
model.
# - tokenizer: The tokenizer used for encoding and
decoding text. def generate_question(prompt,
model, tokenizer):
  # Prepare the input text by formatting it as a
    question input_text = f"question:
    {prompt}"
    # Encode the input text using the provided tokenizer
```

```
    input_ids = tokenizer.encode(input_text,
    return_tensors="pt")# Generate a question from
    the input using the model
    output = model.generate(input_ids, max_length=50,
     num_return_sequences=1, num_beams=4)# Decode and return the
     generated question
    generated_question =
    tokenizer.decode(output[0]) return
    generated_question
# Define the prompt that serves as the input for
generating the question prompt = "Summarize the main
idea of the following passage:"
# Generate a question based on the prompt using the specified
model and tokenizer generated_question =
generate_question(prompt, model, tokenizer)
# Print the generated question to the
consoleprint(generated_question)
```

```
<pad> Summarize the main idea of the following passage:</s>
```

```
# Generate questions and
answers
questions_and_answers =
[]
pdf_text = extract_text_from_pdf(absolute_path)
# Split the PDF text into paragraphs or sections, if needed
# You can split the text based on paragraphs, headings, or any other
logical division# For demonstration, we'll split the text by empty
lines
passages = pdf_text.split('\n\n')

# Generate questions for each passage and store
them in a listfor passage in passages:
    generated_question = generate_question(passage,
    model, tokenizer)questions_and_answers.append({
        "Passage": passage,
        "Question": generated_question
    })

# Display the generated questions
and answers for qa in
questions_and_answers:
    print("Passage:")
    print(qa["Passage"])
    print("\nGenerated
    Question:")
    print(qa["Question"])
    print("-" * 50)

# Open a file named
"generated_questions_and_answers.txt" for writing
```

```
with open("generated_questions_and_answers.txt",
"w") as file:
    # Iterate through a list of questions and
     answers for qa in
     questions_and_answers:
       # Write a label indicating the start of the
        passagefile.write("Passage:\n")
         # Write the passage text to
        the file
        file.write(qa["Passage"] +
        "\n")
          # Write a label indicating the start of the generated question

        file.write("Generated Question:\n")
         # Write the generated question text to
        the file file.write(qa["Question"] +
        "\n")
        # Write a separator line to distinguish
        between entriesfile.write("-" * 50 + "\n")
```

Fig. 16 – A sawar of Bengal in the service of the Company, painted by an unknown Indian artist, 1780After the battles with the Marathas and the Mysore rulers, the Company realised the importance of strengthening its cavalry force.chap 1-4.indd 23 4/22/2022 2:49:43 PMRationalised 2023-24

Let's recall

1. Match the following:

Diwani Tipu Sultan

"Tiger of Mysore" right to collect land revenuefaujdari adalat Sepoy

Rani Channamma criminal court sipahi led an anti-British movement in Kitoor

2. Fill in the blanks:

(a) The British conquest of Bengal began with theBattle of_.

(b) Haidar Ali and Tipu Sultan were the rulers of .

(c) Dalhousie implemented the Doctrine of .

(d) Maratha kingdoms were located mainly in the part of India.

3. State whether true or false:

(a) The Mughal empire became stronger in the eighteenth century.

(b) The English East India Company was the onlyEuropean company that traded with India.

(c) Maharaja Ranjit Singh was the ruler of Punjab.

(d) The British did not introduce administrative changes in the territories they conquered. Let's imagineYou are living in England in the late eighteenth or early nineteenth century. How would you have reacted to the stories of British conquests? Remem

Let’s discuss

4. What attracted European trading companies toIndia?
5. What were the areas of conflict between the Bengal nawabs and the East India Company?

6. How did the assumption of Diwani benefit theEast India Company?
7. Explain the system of “subsidiary alliance”.
8. In what way was the administration of the Company different from that of Indian rulers?
9. Describe the changes that occurred in the composition of the Company’s army.

Let’s do

10. After the British conquest of Bengal, Calcutta grew from small village to a big city. Find out about the culture, architecture and the life of Europeans and Indians of the city
11. Collect pictures, stories, poems and information about any of the following – the Rani of Jhansi, Mahadji Sindhia, Haidar Ali, Maharaja Ranjit Singh, Lord Dalhousie or any other

Generated Question:
<pad> <extra_id_0> the Company, which was governed by the British, was governed by the British government. the Company was govern

```
from random import sample

# Function to generate multiple-choice questions with options
def generate_multiple_choice_question(prompt, correct_answer,
    options, num_options=6): incorrect_options = [option for option
    in options if option != correct_answer]
    random_incorrect_options = sample(incorrect_options,
    num_options - 1) options = random_incorrect_options
    + [correct_answer]
    question = f"Question: {prompt}\n"
    for i, option in enumerate(options):
        question += f"{chr(65 + i)}.
    {option}\n"return question

# Split the PDF text into smaller sections (e.g., paragraphs)
sections = pdf_text.split("\n\n") # Split based on empty lines; adjust as

neededquestions_and_answers = []

for section in sections:
    generated_question = generate_question(section,
    model, tokenizer)# Replace with the correct answer
    for the section
    correct_answer = "The British government"
```

```
    answer_options = ["French government", "The Spanish monarchy", "The American government", "British rule", "The British"]

    # Generate multiple-choice question
    mcq = generate_multiple_choice_question(generated_question, correct_answer, answer_options)

    questions_and_answers.
        append({   "Section":
        section,
        "Question": mcq
    })


# Display the generated questions
and answers for qa in
questions_and_answers:
    print("Section:")
    print(qa["Section"])
    print("\nGenerated
    Question:")
    print(qa["Question"])
    print("-" * 50)
# Open a file named "generated_questions_and_answers.txt" for writing
with
   open("generated_multiple_choice_questions_and_answers.t
   xt", "w") as file: # Iterate through a list of questions and
   answers
    for qa in questions_and_answers:
      # Write a label indicating the start of the
        passagefile.write("Section:\n")
         # Write the passage text to
        the file
        file.write(qa["Section"] +
        "\n")
          # Write a label indicating the start of the
        generated question file.write("Generated
        Question:\n")
         # Write the generated question text to
        the file file.write(qa["Question"] +
        "\n")
        # Write a separator line to distinguish
        between entriesfile.write("-" * 50 + "\n")
```

Generated Question:
Question: <pad> <extra_id_0> the Company, which was governed by the British, was governed by the British government. the Company was governed by the British government in the 18th century.
A. The American government
B. British rule
C. The British
D. French government
E. The Spanish monarchy
F. The British government