Generating Q&A from PDFs Using Groq’s API

> Hamza Fatnaoui

> *Meriem EL Harkaoui*

Introduction

In this guide, you will learn how to use the llama-3.1-70b-versatile model via Groq's API to generate questions and responses from multiple PDFs. We will demonstrate how to extract valuable insights from your PDFs using Groq's API, providing a comprehensive overview of the process and its potential applications. By the end of this guide, you'll have a solid understanding of how to leverage the llama-3.1-70b-versatile model, powered by Groq's API, to unlock the hidden knowledge within your PDFs.

1. Groq

1.1 What is Groq ?

Groq is a high-performance computing platform designed to **accelerate** AI workloads. By optimizing hardware and software for the unique demands of artificial intelligence, Groq provides a powerful API that allows developers to rapidly deploy and scale AI models. This platform is engineered to deliver exceptional speed, efficiency, and accuracy in **AI inference**

1.2 LPU ?

LPU stands for **Language Processing Unit**. It's a specialized processor designed by Groq to accelerate AI workloads, particularly those related to natural language processing. Think of it as the brain of the Groq platform.

1.3 Available Models

→Here is how you can list the available models while using Groq'API

```
# Listing out the available models
def list_groq_models():
    try:
        models = client.models.list()
        for model in models:
            print(model)
    except Exception as e:
        print(f"An error occurred: {e}")

list_groq_models()
```

```
('data', [Model(id='gemma2-9b-it', created=1693721698, object='model', owned_by=
Model(id='gemma-7b-it', created=1693721698, object='model', owned_by='Google', a
='llama-3.1-70b-versatile', created=1693721698, object='model', owned_by='Meta'
(id='llama-3.1-8b-instant', created=1693721698, object='model', owned_by='Meta'
(id='llama3-70b-8192', created=1693721698, object='model', owned_by='Meta', acti
ma3-8b-8192', created=1693721698, object='model', owned_by='Meta', active=True,
q-70b-8192-tool-use-preview', created=1693721698, object='model', owned_by='Groq
(id='llama3-groq-8b-8192-tool-use-preview', created=1693721698, object='model',
=None), Model(id='mixtral-8x7b-32768', created=1693721698, object='model', owned
ps=None), Model(id='whisper-large-v3', created=1693721698, object='model', owned
ne)])
('object', 'list')
```

2. Why llama-3.1-70b-versatile ?

→ It is a good practice to generate Q&A from a PDF by sending parts of the PDF in separate requests to the model via Groq's API, but in this approach can lose the contextual connection between parts, leading to less accurate Q&A. Instead, we

need to select a model based on its capacity for input, output and summarization tokens.

ID	REQUESTS PER MINUTE	REQUESTS PER DAY	TOKENS PER MINUTE	TOKENS PER DAY
gemma-7b-it	30	14,400	15,000	(No limit)
gemma2-9b-it	30	14,400	15,000	(No limit)
llama-3.1-405b-reasoning	30	14,400	131,072	131,072
llama-3.1-70b-versatile	100	14,400	131,072	1,000,000
llama-3.1-8b-instant	30	14,400	131,072	1,000,000
llama-guard-3-8b	30	14,400	15,000	(No limit)
llama3-70b-8192	30	14,400	6,000	(No limit)
llama3-8b-8192	30	14,400	30,000	(No limit)
llama3-groq-70b-8192-tool-use-preview	30	14,400	15,000	(No limit)
llama3-groq-8b-8192-tool-use-preview	30	14,400	15,000	(No limit)
mixtral-8×7b-32768	30	14,400	5,000	(No limit)

→As you can see, the LLaMA models are the best choice for our requirements. The differences between the LLaMA models lie primarily in their mathematical configurations, so we will proceed with using the llama-3.1-70b-versatile.

3. Byte Pair Encoding (BPE)

Byte Pair Encoding (BPE) is a data compression technique that has been adapted for use in natural language processing (NLP)

Using the `tiktoken` library, we can estimate the number of tokens in each PDF.

```
import tiktoken
```

```
# Counting the number of tokens
def number_of_tokens(text):
  tokenizer = tiktoken.get_encoding("gpt2")
  ids = tokenizer.encode(text)
  return len(ids)
```

4. Text Summary

→We are going to send the extracted text of each page along with a summary of the previous text so that the model can generate questions and answers related to different parts.

```
from transformers import pipeline
```

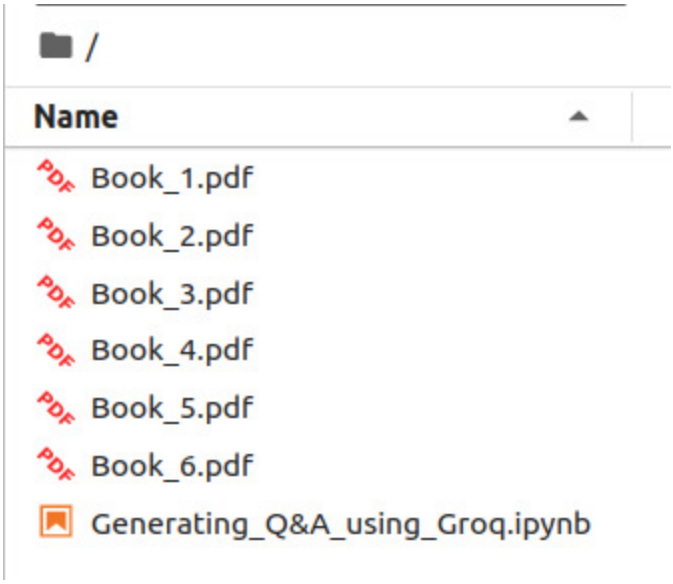
```python
# Function to summarize text
def summarize_text(text, chunk_size=500,device=0):
    summarizer = pipeline("summarization", model="sshleifer/distilbart-cnn-12-6",device=device)

    chunks = [text[i:i+chunk_size] for i in range(0, len(text), chunk_size)]
    summaries = []
    for chunk in chunks:
        max_length = min(150,max(20,number_of_tokens(chunk) // 2))
        min_length = min(50,max(10,number_of_tokens(chunk) // 2))
        summary = summarizer(chunk, max_length=max_length, min_length=min_length, do_sample=False)[0]['summary_text']
        summaries.append(summary)

    combined_summary = " ".join(summaries)
    return combined_summary
```

Steps To Generate Q&A

1. Data Extraction

1.1 Input Data

→We are going to work with 6 PDFs of varying sizes



→From these PDFs, we are going to extract the text using **PyPDF2.**

```python
from PyPDF2 import PdfReader
```

```python
# Extracting Text From pages
def read_pdf(file_path):
    reader = PdfReader(file_path)
    for page in reader.pages:
        yield page.extract_text()
```

→**Pdfminer library** is more powerful than **PDF Reader** due to its advanced features and capabilities. Pdfminer excels in extracting text, metadata, and other information from PDF files with a high degree of accuracy. Unlike PDF Reader, which primarily focuses on rendering PDF documents for viewing

→In our task, we are going to send the extracted text from each page. So the **PDF Reader** would be a good choice for this task

2. Setting Up the Environment

2.1. Install Required Libraries:

- Ensure you have the necessary libraries installed,

```
!pip install -q PyPDF2
!pip install -q tiktoken   # BPE
!pip install -q groq
!pip install -q -U transformers
```

```python
from PyPDF2 import PdfReader
import tiktoken
import os
from groq import Groq
import groq
import glob
import time
from transformers import pipeline
```

3. Set up your API Key

→Configure your API key as an environment variable. This approach streamlines your API usage by eliminating the need to include your API key in each request.

```python
import os

# Set the environment variable
os.environ['GROQ_API_KEY'] = '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'
```

```python
# Initialize Groq client
client = Groq(
    api_key=os.environ.get("GROQ_API_KEY"),
)
```

4. Performing a Chat Completion

4.1. Chat Completion

→ We are going to work with llama-3.1-70b-versatile, which can process around 130,000 tokens per minute and 100 requests per minute.

```python
# Function to generate Q&A pairs from text using Groq's API
def generate_questions_and_answers(text,text_summarization, model="llama-3.1-70b-versatile"):
    response = client.chat.completions.create(
        messages=[
            {
                "role": "user",
                "content": f"""Generate questions and answers from the given text.
                The answers should be based on what's in the text, not on your knowledge.
                Follow this model in generating: 'Q: ...\nA: ...'.
                Here is the text: {text}.
                Also, consider the summarization of the previous text to help
                in generating questions and answers: {text_summarization}."""
            }
        ],
        model=model,
    )

    # Extracting the content of the response
    if response and response.choices:
        return response.choices[0].message.content
    else:
        return "No Q&A generated."
```

5. Generation

5.1 Generation

→We are going to store each output (Q&A) in its own text file.

```
pdf_files = glob.glob('Book_?.pdf')

def Generate():
    Number_of_Tokens_per_minute = 0
    Number_of_Request_per_minute = 0
    Number_of_Tokens_per_day = 0
    Number_of_Request_per_day = 0
    for i,pdf in enumerate(pdf_files):
        output_file = f'Book_{i+1}_q&a.txt'
        text_summarization = ""
        print(f"Book {i+1}")
        with open(output_file, 'w', encoding='utf-8') as file:
            for j,page_text in enumerate(read_pdf(pdf)):

                current_page_tokens = number_of_tokens(page_text)
                current_summary_tokens = number_of_tokens(text_summarization)

                Number_of_Tokens_per_minute += (current_page_tokens + current_summary_tokens)
                Number_of_Tokens_per_day += (current_page_tokens + current_summary_tokens)

                if(Number_of_Tokens_per_minute > 130000 or Number_of_Request_per_minute >= 100):
                    Number_of_Tokens_per_minute = 0
                    Number_of_Request_per_minute = 0
                    print("please just wait 1 min ....")
                    time.sleep(65)
```

```
                if(Number_of_Tokens_per_day >= 1000000 or Number_of_Request_per_day >= 14400):
                    Number_of_Tokens_per_day = 0
                    Number_of_Request_per_day = 0
                    Number_of_Tokens_per_minute = 0
                    Number_of_Request_per_minute = 0
                    print("please you have to wait 1 day ....")
                    time.sleep(86100)

                q_and_a = generate_questions_and_answers(page_text,text_summarization)
                current_q_and_a_tokens = number_of_tokens(q_and_a)

                Number_of_Tokens_per_minute += current_q_and_a_tokens
                Number_of_Tokens_per_day += current_q_and_a_tokens

                Number_of_Request_per_minute += 1
                Number_of_Request_per_day += 1
                text_summarization += summarize_text(page_text)

                file.write(f"Page {j+1} Q&A:\n{q_and_a}\n\n")
                if((j+1)%10==0 or j==0):
                    print(f"-> Page {j+1} converted\n-> ...")

            print(f"-> Q&A generation for pdf {i+1} completed.")
            print('\n')
```

→Generation

```
Generate()
```

```
Book 1
-> Page 1 converted
-> ...
-> Page 10 converted
-> ...
-> Page 20 converted
-> ...
```

5.2. The entire code

→Here is the entire code in this file

Generating_Q&A_using_Groq_EfficientWay.ipynb