In [1]:
```
#installing pdfquery, library that helps extract data from PDF files
!pip install pdfquery
```

```
Collecting pdfquery
  Downloading pdfquery-0.4.3.tar.gz (17 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: cssselect>=0.7.1 in c:\users\adity\anaconda3\lib\site-packages (from pdfquery) (1.1.0)
Requirement already satisfied: chardet in c:\users\adity\anaconda3\lib\site-packages (from pdfquery) (4.0.0)
Requirement already satisfied: lxml>=3.0 in c:\users\adity\anaconda3\lib\site-packages (from pdfquery) (4.9.1)
Requirement already satisfied: pdfminer.six in c:\users\adity\anaconda3\lib\site-packages (from pdfquery) (20240706)
Collecting pyquery>=1.2.2
  Downloading pyquery-2.0.1-py3-none-any.whl (22 kB)
Collecting roman>=1.4.0
  Downloading roman-5.0-py3-none-any.whl (5.5 kB)
Collecting cssselect>=0.7.1
  Downloading cssselect-1.2.0-py2.py3-none-any.whl (18 kB)
Requirement already satisfied: charset-normalizer>=2.0.0 in c:\users\adity\anaconda3\lib\site-packages (from pdfmine
r.six->pdfquery) (2.0.4)
Requirement already satisfied: cryptography>=36.0.0 in c:\users\adity\anaconda3\lib\site-packages (from pdfminer.six-
>pdfquery) (39.0.1)
Requirement already satisfied: cffi>=1.12 in c:\users\adity\anaconda3\lib\site-packages (from cryptography>=36.0.0->p
dfminer.six->pdfquery) (1.15.1)
Requirement already satisfied: pycparser in c:\users\adity\anaconda3\lib\site-packages (from cffi>=1.12->cryptography
>=36.0.0->pdfminer.six->pdfquery) (2.21)
Building wheels for collected packages: pdfquery
  Building wheel for pdfquery (setup.py): started
  Building wheel for pdfquery (setup.py): finished with status 'done'
  Created wheel for pdfquery: filename=pdfquery-0.4.3-py3-none-any.whl size=16844 sha256=b3c3f4a0687d39978547bd886765
284202c6a36785c6b05cadf9b1f5a46a443a
  Stored in directory: c:\users\adity\appdata\local\pip\cache\wheels\cd\a4\8e\0a60850f5ecf4ebd3d78f902b7ee8a4c714b4d4
bebefcdb859
Successfully built pdfquery
Installing collected packages: roman, cssselect, pyquery, pdfquery
  Attempting uninstall: cssselect
    Found existing installation: cssselect 1.1.0
    Uninstalling cssselect-1.1.0:
      Successfully uninstalled cssselect-1.1.0
Successfully installed cssselect-1.2.0 pdfquery-0.4.3 pyquery-2.0.1 roman-5.0
```

In [11]:
```
#importing all the neccessary libraries
import pdfquery
from collections import Counter
import math
import requests
import json
import re
```

In [12]:
```
#defining a funciton to calculate the cosine similarity between the query entered by used and relevant document from t
def cosine_similarity(query, document):
    #tokenization
    query_tokens = query.lower().split(" ")
    document_tokens = document.lower().split(" ")

    #counter for query and document
    query_counter = Counter(query_tokens)
    document_counter = Counter(document_tokens)

    #calculating dot prod
    dot_product = sum(query_counter[token] * document_counter[token] for token in query_counter.keys())

    #calculating magnitudes
    query_magnitude = math.sqrt(sum(query_counter[token]**2 for token in query_counter))
    document_magnitude = math.sqrt(sum(document_counter[token]**2 for token in document_counter))

    #calculating similarity
    #incase (query_magnitude*document_magnitude)=0, the value of similarity becomes 0 else as calculated
    similarity = dot_product/(query_magnitude*document_magnitude) if query_magnitude*document_magnitude!=0 else 0

    return similarity
```

In [13]:
```
#function to return a document similar to the query from the corpus
def return_response(query, corpus):
    similarities = []
    for doc in corpus:
        similarity = cosine_similarity(query, doc)      #calls the cosine_similarity function
        similarities.append(similarity)                 #adds the calculated similarity score to the similarities list

    return corpus[similarities.index(max(similarities))]
```

In [14]:
```
# Creating an infinite loop until the user wants to terminate the program
while True:
    print("Choose the RAG model you want to interact with: ")
    print("1. Diseases and its symptoms")
    print("2. Diseases and its treatments")
    print("3. Terminate the program")

    choice = int(input("Enter your choice: "))  # Accepting the model choice of the user from the menu displayed

    pdf = []
    if choice == 1:
        pdf = pdfquery.PDFQuery(r"C:\Users\adity\OneDrive\Desktop\DISEASES AND ITS SYMPTOMS.pdf")
    elif choice == 2:
        pdf = pdfquery.PDFQuery(r"C:\Users\adity\OneDrive\Desktop\DISEASES AND ITS TREATEMENTS.pdf")
    else:
        print("Closing Program.. Thank you!!")
        break

    pdf.load()

    # Extract all text elements
    text_elements = pdf.pq('LTTextLineHorizontal')
    text = " ".join([t.text for t in text_elements])  # Combine all text

    # Split text into sentences using regex
    corpus = re.split(r'(?<=[.!?]) +', text)

    while True:
        print("Enter Quit to exit from the model")
        user_query = input(">>> ")

        relevant_document = return_response(user_query, corpus)

        if user_query.lower() == 'quit':
            print("\n\n")
            print(f"{'=' * 30}")
            break

        full_response = []

        prompt = """
```

```
You are a medical assistant bot that helps patients identify potential diseases based on symptoms and provides
You answer only in 60 words, ensuring you encourage the patient to follow the treatment.
These are the symptoms/disease provided: {user_input}
Provide the most appropriate disease or symptoms while reassuring the patient and encouraging them to seek med
"""

url = "http://localhost:11434/api/generate"

# Now requesting the above URL which is running locally to execute the below command
# We are using JSON format for making the request

data = {
    "model": "llama3",
    "prompt": prompt.format(user_input=user_query, relevant_document=relevant_document)
}

headers = {'Content-Type': 'application/json'}

response = requests.post(url, data=json.dumps(data), headers=headers, stream=True)

try:
    for line in response.iter_lines():
        # Filter out keep-alive new lines
        if line:
            decoded_line = json.loads(line.decode('utf-8'))
            full_response.append(decoded_line['response'])
finally:
    response.close()

print(''.join(full_response))
print(f"{'*' * 30}")
```

```
Choose the RAG model you want to interact with:
1. Diseases and its symptoms
2. Diseases and its treatments
3. Terminate the program
Enter your choice: 1
Enter Quit to exit from the model
>>> what is diabetes
Diabetes is a chronic condition characterized by high blood sugar levels. Common symptoms include increased thirst an
d urination, fatigue, blurred vision, and recurring skin infections. If you're experiencing these symptoms, don't hes
itate! See your doctor for proper diagnosis and treatment. With proper management, you can control your diabetes and
enjoy a healthy life.
******************************
Enter Quit to exit from the model
>>> 2
I'm here to help! Please provide the two symptoms, and I'll do my best to identify a potential disease or provide rel
evant information. Remember, it's always important to consult with a healthcare professional for an accurate diagnosi
s and proper treatment.

Please share the two symptoms you're experiencing.
******************************
Enter Quit to exit from the model
>>>  dizziness, nosebleeds
I'm here to help! Based on your symptoms of dizziness and nosebleeds, I'm concerned that you might be experiencing Va
sovagal Syncope, a common condition where sudden drops in blood pressure cause dizziness and fainting. It's essential
to seek medical attention as soon as possible to rule out any underlying conditions. Please follow treatment instruct
ions and consider seeking emergency care if symptoms persist or worsen.
******************************
Enter Quit to exit from the model
>>> Quit



==============================
Choose the RAG model you want to interact with:
1. Diseases and its symptoms
2. Diseases and its treatments
3. Terminate the program
Enter your choice: 3
Closing Program.. Thank you!!
```

In [ ]: