**IISC REPORT**

Learnt about basics of python, machine learning ,its types and SQL

//converting pdf to JSON file

import PyPDF2

import json

from nltk import sent\_tokenize, word\_tokenize

from nltk.sentiment import SentimentIntensityAnalyzer

def extract\_text\_from\_pdf(pdf\_path):

    with open(pdf\_path, 'rb') as file:

        reader = PyPDF2.PdfReader(file)

        text = ''

        NumPages = len(reader.pages)

        for i in range(0, NumPages):

            PageObj = reader.pages[i]

            print("this is page " + str(i))

            text += PageObj.extract\_text()

        return text

def perform\_semantic\_analysis(text):

    sentences = sent\_tokenize(text)

    sid = SentimentIntensityAnalyzer()

    results = []

    for sentence in sentences:

        tokens = word\_tokenize(sentence)

        sentiment\_scores = sid.polarity\_scores(sentence)

        result = {

            'sentence': sentence,

            'tokens': tokens,

            'sentiment': sentiment\_scores['compound']

        }

        results.append(result)

    return results

def generate\_json\_file(data, json\_path):

    with open(json\_path, 'w') as file:

        json.dump(data, file, indent=4)

# Example usage

pdf\_path = 'C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\doctorgpt\\Altok10.pdf'

json\_path = 'C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\doctorgpt\\altok.json'

text = extract\_text\_from\_pdf(pdf\_path)

results = perform\_semantic\_analysis(text)

generate\_json\_file(results, json\_path)

//worked on google collab to extract page number,content ,images, links etc

## <https://colab.research.google.com/drive/1zCXSbD3ezvgqFaNtsT6YWZPdyZSu0OMU?authuser=0#scrollTo=1PkkuhHu2Z_H>

//**extracting the headlines from the docx file**

from docx import Document

document = Document('ALTOK.docx')

for paragraph in document.paragraphs:

    if paragraph.style.name.startswith('Heading'):

        print(paragraph.text)

**//creating dictionary based on content and page number, chapter names**

dict1={

     'FOREWORD':[9],

     'IMPORTANT':[10],

     'WARRANTY POLICY':[11-12],

     'EMISSION WARRANTY POLICY':[13-14],

     'TABLE OF CONTENTS':[15],

     'FUEL RECOMENDATION':[17,18],

     'BEFORE DRIVING':[19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62],

     'OPERATING YOUR VEHICLE':[63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81],

      'DRIVING TIPS':[83,84,85,86,87,88,89,90,91],

      'OTHER CONTROLS AND EQUIPMENT':[93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,123,124,125,126,127,128,129,130,131],

      'VEHICLE LOADING AND TOWNING':[133,134,135,136],

      'INSPECTION AND MAINTENANCE':[137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,154,155,156,157,158,159,160,161,162,163,164,165,167,168],

      'EMERGENCY SERVICE':[169,170,171,172,173,174,175,176,177],

       'APPEARANCE CARE':[179,180,181,182,182,183,184],

       'GENERAL INFORMATION':[185,186],

       'SPECIFICATIONS':[187,188,189,190],

}

dict1={

 'chapter 1' : ['FUEL RECOMENDATION'],

 'chapter 2' : ['BEFORE DRIVING'],

 'chapter 3' : ['OPERATING YOUR VEHICLE'],

 'chapter 4' :['DRIVING TIPS'],

 'chapter 5' :['OTHER CONTROLS AND EQUIPMENT'],

 'chapter 6' :['VEHICLE LOADING AND TOWNING'],

 'chapter 7' :['INSPECTION AND MAINTENANCE'],

 'chapter 8' :['EMERGENCY SERVICE'],

 'chapter 9' :['APPEARANCE CARE'],

 'chapter 10' : ['GENERAL INFORMATION'],

 'chapter 11' : ['SPECIFICATIONS'],

}

**//extracting page content of a docx file**

from docx import Document

def extract\_paragraphs(docx\_path):

    doc = Document(docx\_path)

    paragraphs = [paragraph.text for paragraph in doc.paragraphs if paragraph.text.strip()]

    return paragraphs

def get\_content\_by\_page(paragraphs, target\_page):

    paragraphs\_per\_page = len(paragraphs) / total\_pages

    start\_paragraph = int((target\_page - 1) \* paragraphs\_per\_page)

    end\_paragraph = int(min(start\_paragraph + paragraphs\_per\_page, len(paragraphs)))

    content = '\n'.join(paragraphs[start\_paragraph:end\_paragraph])

    return content

docx\_path = 'C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\doctorgpt\\ALTOK.docx'

total\_pages = 189  # Total number of pages in your document (estimated)

target\_page = 26

paragraphs = extract\_paragraphs(docx\_path)

page\_content = get\_content\_by\_page(paragraphs, target\_page)

print(page\_content)

//**extracting page content of a pdf**

import PyPDF2

def extract\_page\_content(pdf\_path, page\_number):

    with open(pdf\_path, "rb") as pdf\_file:

        pdf\_reader = PyPDF2.PdfReader(pdf\_file)

        if page\_number <= len(pdf\_reader.pages):

            page = pdf\_reader.pages[page\_number - 1]  # Adjust for 0-based indexing

            extracted\_text = page.extract\_text()

            return extracted\_text

        else:

            return "Page number is out of range."

pdf\_path = "C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\ALTOK.PDF"

page\_number = 26  # Replace with the desired page number

page\_content = extract\_page\_content(pdf\_path, page\_number)

print(page\_content)

//**extracting keyword from a pdf based on page number**

import PyPDF2

import spacy

def extract\_keywords\_from\_page(pdf\_path, page\_number):

    extracted\_keywords = []

    with open(pdf\_path, "rb") as pdf\_file:

        pdf\_reader = PyPDF2.PdfReader(pdf\_file)

        if page\_number <= len(pdf\_reader.pages):

            page = pdf\_reader.pages[page\_number - 1]  # Adjust for 0-based indexing

            extracted\_text = page.extract\_text()

            # Load spaCy NLP model

            nlp = spacy.load("en\_core\_web\_sm")

            doc = nlp(extracted\_text)

            # Extract keywords (nouns and proper nouns)

            for token in doc:

                if token.pos\_ in ["NOUN", "PROPN"]:

                    extracted\_keywords.append(token.text)

    return extracted\_keywords

pdf\_path = "C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\ALTOK.PDF"

page\_number = 26  # Replace with the desired page number

keywords = extract\_keywords\_from\_page(pdf\_path, page\_number)

print("Extracted keywords:", keywords)

import PyPDF2

import spacy

def extract\_keywords\_from\_page(pdf\_path, page\_number):

    extracted\_keywords = set()  # Using a set to store unique keywords

    with open(pdf\_path, "rb") as pdf\_file:

        pdf\_reader = PyPDF2.PdfReader(pdf\_file)

        if page\_number <= len(pdf\_reader.pages):

            page = pdf\_reader.pages[page\_number - 1]  # Adjust for 0-based indexing

            extracted\_text = page.extract\_text()

            # Load spaCy NLP model

            nlp = spacy.load("en\_core\_web\_sm")

            doc = nlp(extracted\_text)

            # Extract unique keywords (nouns and proper nouns)

            for token in doc:

                if token.pos\_ in ["NOUN", "PROPN"]:

                    extracted\_keywords.add(token.text)

    return extracted\_keywords

def extract\_keywords\_from\_multiple\_pages(pdf\_path, start\_page, end\_page):

    all\_page\_keywords = {}  # Dictionary to store keywords for each page

    for page\_number in range(start\_page, end\_page + 1):

        keywords = extract\_keywords\_from\_page(pdf\_path, page\_number)

        all\_page\_keywords[page\_number] = keywords

    return all\_page\_keywords

pdf\_path = "C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\ALTOK.PDF"

start\_page = 9  # Replace with the starting page number

end\_page = 100  # Replace with the ending page number

page\_keywords = extract\_keywords\_from\_multiple\_pages(pdf\_path, start\_page, end\_page)

# Print keywords for each page

for page\_number, keywords in page\_keywords.items():

    print(f"Page {page\_number} Keywords:",keywords)

//**extracting keywords,chapter name and page number based on given input page**

page\_ranges = {

    (17,18): "chapter 1 : ['FUEL RECOMENDATION']",

    (19,62): "Chapter 2 : ['BEFORE DRIVING']",

    (63,81): "Chapter 3: ['OPERATING YOUR VEHICLE']",

    (83,91): "Chapter 4:['DRIVING TIPS']",

    (93,131):"Chapter 5:['OTHER CONTROLS AND EQUIPMENT']",

    (133,136):"Chapter 6:['VEHICLE LOADING AND TOWNING']",

    (137,168):"Chapter 7:['INSPECTION AND MAINTENANCE']",

    (169,177):"Chapter 8:['EMERGENCY SERVICE']",

    (179,184):"Chapter 9:['APPEARANCE CARE']",

    (185,186):"Chapter 10: ['GENERAL INFORMATION']",

    (187,190):"Chapter 11:['SPECIFICATIONS']",

}

def get\_content\_by\_page(page\_number):

    for page\_range, content in page\_ranges.items():

        start\_page, end\_page = page\_range

        if start\_page <= page\_number <= end\_page:

            return content

    return "Page not found"  # Handle the case where the page number is not within any range

# Example usage

page\_number = 190

content = get\_content\_by\_page(page\_number)

print(f"Content for page {page\_number}: {content}")

import PyPDF2

import spacy

def extract\_keywords\_from\_page(pdf\_path, page\_number):

    extracted\_keywords = set()  # Using a set to store unique keywords

    with open(pdf\_path, "rb") as pdf\_file:

        pdf\_reader = PyPDF2.PdfReader(pdf\_file)

        if page\_number <= len(pdf\_reader.pages):

            page = pdf\_reader.pages[page\_number - 1]  # Adjust for 0-based indexing

            extracted\_text = page.extract\_text()

            # Load spaCy NLP model

            nlp = spacy.load("en\_core\_web\_sm")

            doc = nlp(extracted\_text)

            # Extract unique keywords (nouns and proper nouns)

            for token in doc:

                if token.pos\_ in ["NOUN", "PROPN"]:

                    extracted\_keywords.add(token.text)

    return extracted\_keywords

pdf\_path = "C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\ALTOK.PDF"

page\_number = 190 # Replace with the desired page number

keywords = extract\_keywords\_from\_page(pdf\_path, page\_number)

print("Extracted keywords:", keywords)

**//extracting keyword,chapter name ,page number without giving page number**

import PyPDF2

import spacy

page\_ranges = {

    (17, 18): "chapter 1 : ['FUEL RECOMMENDATION']",

    (19, 62): "Chapter 2 : ['BEFORE DRIVING']",

    (63, 81): "Chapter 3: ['OPERATING YOUR VEHICLE']",

    (83, 91): "Chapter 4:['DRIVING TIPS']",

    (93, 131): "Chapter 5:['OTHER CONTROLS AND EQUIPMENT']",

    (133, 136): "Chapter 6:['VEHICLE LOADING AND TOWING']",

    (137, 168): "Chapter 7:['INSPECTION AND MAINTENANCE']",

    (169, 177): "Chapter 8:['EMERGENCY SERVICE']",

    (179, 184): "Chapter 9:['APPEARANCE CARE']",

    (185, 186): "Chapter 10: ['GENERAL INFORMATION']",

    (187, 190): "Chapter 11:['SPECIFICATIONS']",

}

def extract\_keywords\_from\_page(pdf\_path):

    extracted\_keywords\_per\_page = {}  # Dictionary to store keywords for each page

    with open(pdf\_path, "rb") as pdf\_file:

        pdf\_reader = PyPDF2.PdfReader(pdf\_file)

        for page\_number in range(len(pdf\_reader.pages)):

            page = pdf\_reader.pages[page\_number]

            extracted\_text = page.extract\_text()

            # Load spaCy NLP model

            nlp = spacy.load("en\_core\_web\_sm")

            doc = nlp(extracted\_text)

            # Extract unique keywords (nouns and proper nouns)

            extracted\_keywords = set()

            for token in doc:

                if token.pos\_ in ["NOUN", "PROPN"]:

                    extracted\_keywords.add(token.text)

            # Store keywords for the current page

            extracted\_keywords\_per\_page[page\_number + 1] = extracted\_keywords

    return extracted\_keywords\_per\_page

pdf\_path = "C:\\Users\\Karun\\Desktop\\intership\\doctorgpt\\ALTOK.PDF"

page\_keywords = extract\_keywords\_from\_page(pdf\_path)

for page\_number, keywords in page\_keywords.items():

    content = get\_content\_by\_page(page\_number)

    print(f"Page {page\_number} Content: {content}")

    print(f"Extracted keywords: {keywords}\n")

**//worked on private gpt4all llm model**

!apt-get install poppler-utils

!pip install -Uqqq pip --progress-bar off

!pip install -qqq langchain==0.0.173 --progress-bar off

!pip install -qqq chromadb==0.3.23 --progress-bar off

!pip install -qqq pypdf==3.8.1 --progress-bar off

!pip install -qqq pygpt4all==1.1.0 --progress-bar off

!pip install -qqq pdf2image==1.16.3 --progress-bar off

!gdown 1DpFisoGXsQbpQJvijuvxkLW\_pg-FUUMF

!wget <https://gpt4all.io/models/ggml-gpt4all-j-v1.3-groovy.bin>

from langchain.chains import RetrievalQA

from langchain.document\_loaders import PyPDFLoader

from langchain.embeddings import HuggingFaceEmbeddings

from langchain.llms import GPT4All

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.vectorstores import Chroma

from pdf2image import convert\_from\_path

images = convert\_from\_path("ms-financial-statement.pdf", dpi=88)

len(images)

images[0]

images[1]

## Load Data

loader = PyPDFLoader("ms-financial-statement.pdf")

documents = loader.load\_and\_split()

len(documents)

print(documents[0].page\_content)

text\_splitter = RecursiveCharacterTextSplitter(chunk\_size=1024, chunk\_overlap=64)

texts = text\_splitter.split\_documents(documents)

len(texts)

print(texts[0].page\_content)

## Create Embeddings

embeddings = HuggingFaceEmbeddings(model\_name="sentence-transformers/all-MiniLM-L6-v2")

db = Chroma.from\_documents(texts, embeddings, persist\_directory="db")

## Create Chain

model\_n\_ctx = 1000

model\_path = "./ggml-gpt4all-j-v1.3-groovy.bin"

llm = GPT4All(model=model\_path, n\_ctx=1000, backend="gptj", verbose=False)

qa = RetrievalQA.from\_chain\_type(

llm=llm,

chain\_type="stuff",

retriever=db.as\_retriever(search\_kwargs={"k": 3}),

return\_source\_documents=True,

verbose=False,

)

## Ask Questions

%%time

res = qa(

"How much is the dividend per share during during 2022? Extract it from the text."

)

res

print(res["result"])

%%time

prompt = f"""How much is the investment amount in Microsoft on 6/22? Extract the answer from the text."""

res = qa(prompt.strip())

print(res["result"])

## <https://github.com/curiousily/Get-Things-Done-with-Prompt-Engineering-and-LangChain/blob/master/06.private-gpt4all-qa-pdf.ipynb>