RAG_and_LangChain_loading_documents_round1

November 28, 2023

1 RAG and LangChain

There are more than 80 different loaders in LangChain.

1.1 Loading documents

[ ]:
```
!pip install langchain
!pip install openai
```

[8]:
```
from google.colab import userdata
openai_api_key = userdata.get('OPENAI_API_KEY')
```

[9]:
```
from openai import OpenAI
client = OpenAI(api_key=openai_api_key)
```

1.2 PDF

1.2.1 LangChain PDF Extractor:

[2]:
```
!pip install pypdf
```

```
Collecting pypdf
  Downloading pypdf-3.17.1-py3-none-any.whl (277 kB)
                            277.6/277.6
kB 5.7 MB/s eta 0:00:00
Installing collected packages: pypdf
Successfully installed pypdf-3.17.1
```

[3]:
```
pdf_url = "https://arxiv.org/pdf/2005.11401.pdf"
from langchain.document_loaders import PyPDFLoader
loader = PyPDFLoader(pdf_url)
pages = loader.load()

print(f"Number of pages: {len(pages)}")

# each page is a document having two elements: page_content and metadata
page = pages[0]
```

```
print("######## PAGE CONTENT ########")
print(page.page_content[:400])

print("\n######## PAGE METDATA ########")
print(page.metadata)
```

```
Number of pages: 19
######## PAGE CONTENT ########
Retrieval-Augmented Generation for
Knowledge-Intensive NLP Tasks
Patrick Lewis†‡, Ethan Perez ,
Aleksandra Piktus†, Fabio Petroni†, Vladimir Karpukhin†, Naman Goyal†, Heinrich
Küttler†,
Mike Lewis†, Wen-tau Yih†, Tim Rocktäschel†‡, Sebastian Riedel†‡, Douwe Kiela†
†Facebook AI Research;‡University College London; New York University;
plewis@fb.com
Abstract
Large pre-trained language models have be

######## PAGE METDATA ########
{'source': 'https://arxiv.org/pdf/2005.11401.pdf', 'page': 0}
```

1.2.2 PdfReader

[4]:
```
#Need to mount my drive first
from google.colab import drive
drive.mount('/content/drive')
path='/content/drive/MyDrive/02-Articles_ChatGPT/03_notebooks/data/'
path_pdf=path+"2005.11401.pdf"
```

```
Mounted at /content/drive
```

[5]:
```
from pypdf import PdfReader

reader = PdfReader(path_pdf) # In pypdf directly you can't use the pdf url␣
 ↪directly, you need to load it locally beore using it
pages = reader.pages

print(f"Number of pages: {len(pages)}")

# each page is a document having two elements: page_content and metadata
page = pages[0]

print("######## PAGE CONTENT ########")
page = pages[0]
print(page.extract_text()[:400])
```

```
Number of pages: 19
######## PAGE CONTENT ########
Retrieval-Augmented Generation for
Knowledge-Intensive NLP Tasks
Patrick Lewis†‡, Ethan Perez ,
Aleksandra Piktus†, Fabio Petroni†, Vladimir Karpukhin†, Naman Goyal†, Heinrich
Küttler†,
Mike Lewis†, Wen-tau Yih†, Tim Rocktäschel†‡, Sebastian Riedel†‡, Douwe Kiela†
†Facebook AI Research;‡University College London; New York University;
plewis@fb.com
Abstract
Large pre-trained language models have be
```

With LangChain, the results are structured: for each page you have the content and the metadata, which will be useful when retrieving contextual data. Also, you can access the web, while with the standard PdfReader in my pypdf, you need to use a local file.

1.3 YouTube Videos

1.3.1 LangChain

LangChain integrates OpenAI client in its library, you need first to get an OpenAI API Key

In this part, we'll be using Whisper model from OpenAI to convert video/audio to text

[ ]:
```
# ! pip install yt_dlp
# ! pip install pydub
# !pip install librosa
```

```
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub
Successfully installed pydub-0.25.1
```

[7]:
```
from langchain.document_loaders.generic import GenericLoader
from langchain.document_loaders.parsers import OpenAIWhisperParser
from langchain.document_loaders.blob_loaders.youtube_audio import␣
↪YoutubeAudioLoader
```

[ ]:
```
url="https://www.youtube.com/shorts/PdSLpPf9R9U"
save_dir="data/youtube/"
loader = GenericLoader(
    YoutubeAudioLoader([url],save_dir),
    OpenAIWhisperParser()
)
docs = loader.load()
```

```
[youtube] Extracting URL: https://www.youtube.com/shorts/PdSLpPf9R9U
[youtube] PdSLpPf9R9U: Downloading webpage
[youtube] PdSLpPf9R9U: Downloading ios player API JSON
```

```
[youtube] PdSLpPf9R9U: Downloading android player API JSON
[youtube] PdSLpPf9R9U: Downloading m3u8 information
[info] PdSLpPf9R9U: Downloading 1 format(s): 140
[download] data/youtube//The Top Interview Questions to Ask.m4a has already been
downloaded
[download] 100% of  643.84KiB
[ExtractAudio] Not converting audio data/youtube//The Top Interview Questions to
Ask.m4a; file is already in target format m4a
Transcribing part 1!
```

[ ]:
```python
print(f"Type of response is a {type(docs)}")
print(f"Type of the first element is a document {type(docs[0])}")


print("\n#### OVERVIEW OF THE DOC ####")
print(docs[0])

print("\n")
print("The document has two elements: page_content and metadata")

print("\n######## PAGE CONTENT ########")
print(docs[0].page_content[:400])

print("\n######## PAGE METDATA ########")
print(docs[0].metadata)
```

```
Type of response is a <class 'list'>
Type of the first element is a document <class
'langchain.schema.document.Document'>

#### OVERVIEW OF THE DOC ####
page_content="What are the interview questions that you ask? I think what people
don't understand is simply pattern recognition through volume. People that tend
to answer questions like this tend to not work out if I hire them. People that
tend to answer questions like this do tend to work out when I hire them. It's a
lot more about the responses than it is the questions. I am just going with what
I'm seeing and I'm asking questions based on the feedback I'm getting from that
person and the vibe I'm getting from them. When I say vibe, it's the
associations that I'm getting from that person based on past experiences of
people that have similar associations. And that's why I encourage people so
much. I'm like, take 20 interviews for a position because what is so invaluable
is the pattern recognition that you gain of what good looks like and what bad
looks like." metadata={'source': 'data/youtube/The Top Interview Questions to
Ask.m4a', 'chunk': 0}


The document has two elements: page_content and metadata
```

```
######## PAGE CONTENT ########
What are the interview questions that you ask? I think what people don't
understand is simply pattern recognition through volume. People that tend to
answer questions like this tend to not work out if I hire them. People that tend
to answer questions like this do tend to work out when I hire them. It's a lot
more about the responses than it is the questions. I am just going with what I'm
seeing an

######## PAGE METDATA ########
{'source': 'data/youtube/The Top Interview Questions to Ask.m4a', 'chunk': 0}
```

[ ]:
```
docs[0]
```

[ ]:
```
Document(page_content="What are the interview questions that you ask? I think
what people don't understand is simply pattern recognition through volume.
People that tend to answer questions like this tend to not work out if I hire
them. People that tend to answer questions like this do tend to work out when I
hire them. It's a lot more about the responses than it is the questions. I am
just going with what I'm seeing and I'm asking questions based on the feedback
I'm getting from that person and the vibe I'm getting from them. When I say
vibe, it's the associations that I'm getting from that person based on past
experiences of people that have similar associations. And that's why I encourage
people so much. I'm like, take 20 interviews for a position because what is so
invaluable is the pattern recognition that you gain of what good looks like and
what bad looks like.", metadata={'source': 'data/youtube/The Top Interview
Questions to Ask.m4a', 'chunk': 0})
```

1.4 Using pytube lib and then calling whisper

[10]:
```
!pip install pytube
```

```
Collecting pytube
  Downloading pytube-15.0.0-py3-none-any.whl (57 kB)
                             57.6/57.6 kB
1.7 MB/s eta 0:00:00
Installing collected packages: pytube
Successfully installed pytube-15.0.0
```

[11]:
```
from pytube import YouTube
from moviepy.video.io.ffmpeg_tools import ffmpeg_extract_audio

# Leila Hormozi
video_url="https://www.youtube.com/shorts/PdSLpPf9R9U"

yt = YouTube(video_url)

video_stream = yt.streams.get_highest_resolution()
```

```
video_stream.download()
print(f"Video download: '{yt.title}'")

# Extract audio from the downloaded video:
# The audio file will be smaller than the video one, to avoid exceeding the␣
 ↪OpenAI limit size 25MB
# (in some cases, even with the audio, you need to split)
output_audio_file = yt.title + ".mp3"
input_video_file = yt.title + ".mp4"
ffmpeg_extract_audio(input_video_file, output_audio_file)

print("Audio extraction completed.")
```

```
Video download: 'The Top Interview Questions to Ask'
Moviepy - Running:
>>> "+ " ".join(cmd)
Moviepy - Command successful
Audio extraction completed.
```

[12]:
```
%%time
output_audio = yt.title + ".mp3"
audio_file = open(output_audio, "rb")
## A release was done on 06 November 2023, and tue audio endpoint has changed:
## instead of :
# transcript = openai.Audio.transcribe("whisper-1", audio_file)
## You need to put:
transcript = client.audio.transcriptions.create(model="whisper-1",␣
 ↪file=audio_file)

## This also can work, but you need to specify your api key in the old way:␣
 ↪openai.api_key = "YOUR_OPENAI_KEY"
# transcript = open.audio.transcriptions.create(model="whisper-1",␣
 ↪file=audio_file)

print(transcript.text[:100])
```

```
What are the interview questions that you ask? I think what people don't
understand is simply patter
CPU times: user 49.5 ms, sys: 5.8 ms, total: 55.3 ms
Wall time: 3.42 s
```

1.5 URL

1.5.1 LangChain

[13]:
```
from langchain.document_loaders import WebBaseLoader
```

[14]:
```
url ="https://realpython.github.io/fake-jobs/"
loader = WebBaseLoader(url) #It's using BeautifulSoup as parser by default
docs = loader.load()
print(len(docs[0].page_content))
print(docs[0].page_content[:100])
```

```
11633



Fake Python
```

```
        Fake Python


        Fake Jobs for Your Web Scraping Journey
```

Each time we ask WebBaseLoader to load an url, it loads only the html text contained in the given page (len(docs) always =1). If you want to go deeper and get all links inludued in the page, you need to go with standard librairies.

When asking to get data contained in this url "https://realpython.github.io/fake-jobs/" ==> it gets only the data in the first page. It did not parse all links (href) included in the page).

> ==> So one needs to scrap first the website with standard solutions to get all inlcuded links and use WebBaseLoader for each one of them

1.5.2 Alternative

[15]:
```
import requests
from bs4 import BeautifulSoup
```

[16]:
```
URL = "https://realpython.github.io/fake-jobs/"
page = requests.get(URL)

soup = BeautifulSoup(page.content, "html.parser")
# soup
```

1.5.3 List of links

[17]:
```
links = soup.find_all("a")
list_href = [link['href'] for link in links if link['href']!='https://www.
 ↪realpython.com']
list_href[:3]
```

[17]:
```
['https://realpython.github.io/fake-jobs/jobs/senior-python-developer-0.html',
 'https://realpython.github.io/fake-jobs/jobs/energy-engineer-1.html',
 'https://realpython.github.io/fake-jobs/jobs/legal-executive-2.html']
```

[18]:
```
list_text=[]
for href in list_href[:3]:
  print(href)
  #You can then call WebBaseLoader
  page1 = requests.get(href)
  soup1 = BeautifulSoup(page1.content, "html.parser")
  list_text.append(soup1.text)

list_text[:200]
```

```
https://realpython.github.io/fake-jobs/jobs/senior-python-developer-0.html
https://realpython.github.io/fake-jobs/jobs/energy-engineer-1.html
https://realpython.github.io/fake-jobs/jobs/legal-executive-2.html
```

[18]:
```
['\n\n\n\n\nFake Python\n\n\n\n\n\n\n        Fake Python\n      \n\n        Fake
Jobs for Your Web Scraping Journey\n      \n\n\n\n\nSenior Python
Developer\nPayne, Roberts and Davis\n\nProfessional asset web application
environmentally friendly detail-oriented asset. Coordinate educational dashboard
agile employ growth opportunity. Company programs CSS explore role. Html
educational grit web application. Oversea SCRUM talented support. Web
Application fast-growing communities inclusive programs job CSS. Css discussions
growth opportunity explore open-minded oversee. Css Python environmentally
friendly collaborate inclusive role. Django no experience oversee dashboard
environmentally friendly willing to learn programs. Programs open-minded
programs asset.\nLocation: Stewartbury, AA\nPosted: 2021-04-08\n\n\n\n\n\n\n\n\n',
 '\n\n\n\n\nFake Python\n\n\n\n\n\n\n        Fake Python\n      \n\n        Fake
Jobs for Your Web Scraping Journey\n      \n\n\n\n\nEnergy engineer\nVasquez-
Davidson\n\nParty prevent live. Quickly candidate change although. Together type
music hospital. Every speech support time operation wear often.\nLocation:
Christopherville, AA\nPosted: 2021-04-08\n\n\n\n\n\n\n\n\n',
 '\n\n\n\n\nFake Python\n\n\n\n\n\n\n        Fake Python\n      \n\n        Fake
Jobs for Your Web Scraping Journey\n      \n\n\n\n\nLegal executive\nJackson,
Chambers and Levy\n\nAdministration even relate head color. Staff beyond chair
recently and off. Own available buy country store build before. Already against
which continue. Look road article quickly. International big employee determine
positive go Congress. Level others record hospital employee toward
like.\nLocation: Port Ericaburgh, AA\nPosted: 2021-04-08\n\n\n\n\n\n\n\n\n']
```