What are Langchain Document Loaders?

Dsanr
15 Jul, 2024

7 min read

Introduction

LLMs (large language models) are becoming increasingly relevant in various businesses and organizations. Their ability to understand and analyze data and make sense of complex information can drive innovation, improve operational efficiency, and deliver personalized experiences across various industries. Integrating with various tools allows us to build LLM applications that can automate tasks, provide insights, and support decision-making processes.

However, building these applications can be complex and time-consuming, requiring a framework to streamline development and ensure scalability. A framework provides standardized tools and processes, making developing, deploying, and maintaining effective LLM applications easier. So, let's learn about LangChain, the most popular framework for developing LLM applications.

Overview

- LangChain Document Loaders convert data from various formats (e.g., CSV, PDF, HTML) into standardized Document objects for LLM applications.
- They facilitate the seamless integration and processing of diverse data sources, such as YouTube, Wikipedia, and GitHub, into Document objects.

What are Langchain Document Loaders?

- They support a wide range of data formats and sources, enhancing the versatility and scalability of LLM-powered applications.
- LangChain's document loaders streamline the conversion of raw data into structured formats, which is essential for building and maintaining effective LLM applications.

Table of contents

LangChain Overview

LangChain has functionalities ranging from loading, splitting, embedding, and retrieving the data for the LLM to parsing the output of the LLM. It includes adding tools and agentic capabilities to the LLM and hundreds of third-party integrations. LangChain ecosystem also includes LangGraph to build stateful agents and LangSmith to productionize LLM applications. We can learn more about LangChain here at Building LLM-Powered Applications with LangChain.

In a series of articles, we will learn about different components of the Langchain. As it all starts with data, we will start by loading data from various file types and data sources with document loaders from Langchain.

What are Document Loaders?

Document loaders convert data from diverse data formats to standardized Document objects. The Document object consists of page_content, which has the data as a string, optionally an ID for the Document, and metadata that provides information on the data.

Let's create a document object to learn how it works:

To get started, install the LangChain framework using 'pip install langchain'

What are Langchain Document Loaders?

```
>>> Document(id='1', metadata={'source': 'AV'}, page_content='T

data.page_content
>>> This is the article about document loaders of Langchain'

data.id = 2 # this changes the id of the Document object
```

As we can see, we can create a Document object with page_content, id, and metadata and access and modify its contents.

Types of Document Loaders

There are more than two hundred document loaders in LangChain. They can be categorized as follows

- **Based on file type:** These document loaders parse and load the documents based on the file type. Example file types include CSV, PDF, HTML, Markdown, etc.
- **Based on data source:** They get the data from different data sources and load it into Document objects. Examples of data sources include YouTube, Wikipedia, and GitHub.

Data sources can be further classified as public and private. Public data sources like YouTube or Wikipedia don't need access tokens, while private data sources like AWS or Azure do. Let's use a few document loaders to understand how they work. Further we will talk about the – LangChain Document Loaders convert data from various formats (e.g., CSV, PDF, HTML) into standardized Document objects for LLM applications.

CSV(Comma-separated Values)

CSV files can be loaded with `CSVLoader`. It loads each row as a Document.

```
from langchain_community.document_loaders.csv_loader import CSV


loader = CSVLoader(file_path= "./iris.csv", metadata_columns=['
data = loader.load()
```

What are Langchain Document Loaders?

metadata_columns. We can also add the column to the source instead of the file name.

```
data[0].metadata
>>> {'source': './iris.csv', 'row': 0, 'species': 'setosa'}

# we can change the source as 'setosa' with the parameter sourc

for record in data[:1]:
    print(record)
>>> page_content='sepal_length: 5.1
    sepal_width: 3.5
    petal_length: 1.4
    petal_width: 0.2' metadata={'source': './iris.csv', 'row':
```

Langchain dataloaders load the document into Document objects.

HTML(Hyper Text Markup Language)

we can load an HTML page either directly from a saved HTML page or a URL

```
from langchain_community.document_loaders import UnstructuredHT
from langchain_community.document_loaders import UnstructuredUR

loader = UnstructuredURLLoader(urls=['https://diataxis.fr'], mo
data = loader.load()
len(data)
>>> 61
```

The entire HTML page is loaded as one document if the mode is single. if the mode is ‘elements, ‘ documents are made using the HTML tags.

```
# accessing metadata and content in a documnent

data[28].metadata
>>> {'languages': ['eng'], 'parent_id': '312017038db4f2ad1e9332
'filetype': 'text/html', 'url': 'https://diataxis.fr', 'categor

data[28].page_content
>>> "Diátaxis is a way of thinking about and doing documentatio
```

Markdown

Markdown is a markup language for creating formatted text using a simple text editor.

What are Langchain Document Loaders?

```
len(data)
>>> 1458
```

In addition to single and elements, this also has a 'paged' mode, which partitions the file based on the page numbers.

```
data[700].metadata
>>> {'source': 'README.md', 'last_modified': '2024-07-09T12:52:

data[700].page_content
>>> 'NeuralProphet ( 28 ·  3.7K) - NeuralProphet: A simple
```

JSON

We can copy the JSON content from here – How to load JSON?

```
from langchain_community.document_loaders import JSONLoader

loader = JSONLoader(file_path='chat.json', jq_schema='.', text_
data = loader.load()
len(data)
>>> 1
```

In JSONLoader, we need to mention the schema. If jq_schema = '.' all the content is loaded. Depending on the content we need from the json, we can change the schema. For example, jq_schema='.title' for title, jq_schema='.messages[].content' to get only the content of the messages.

MS Office docs

Let's load an MS Word file as an example.

```
from langchain_community.document_loaders import UnstructuredWo

loader = UnstructuredWordDocumentLoader(file_path='Polars.docx'
                                        max_characters=200, new


data = loader.load()
len(data)
>>> 67
```

As we have seen, Langchain uses the Unstructured library to load files in different formats. As the libraries are

What are Langchain Document Loaders?

'add_chunking_strategy' function in Github.

PDF(Portable Document Format)

Multiple PDF parser integrations are available in Langchain. We can compare various parsers and choose a suitable one. Here is the Benchmark.

Some of the available parsers are PyMuPDF, PyPDF, PDFPlumber, etc.

Let's try with UnstructuredPDFLoader

```
from langchain_community.document_loaders import UnstructuredPD

loader = UnstructuredPDFLoader('how-to-formulate-successful-bus

data = loader.load()
len(data)
>>> 177
```

Here is the code explanation:

- The 'strategy' parameter defines how to process the pdf.
- The 'hi_res' strategy uses the Detectron2 model to identify the document's layout.
- The 'ocr_only' strategy uses Tesseract to extract the text even from the images.
- The 'fast' strategy uses pdfminer to extract the text.
- 'The default 'auto' strategy uses any of the above strategies based on the documents and parameter arguments.

Multiple Files

If we want to load multiple files from a directory, we can use the following

```
from langchain_community.document_loaders import DirectoryLoade

loader = DirectoryLoader(".", glob="**/*.json", loader_cls=JSON
                         show_progress=True, use_multithreading

docs = loader.load()
```

the loader_cls parameter and the loader's arguments using the loader_kwargs parameter.

YouTube

If you want the summary of a YouTube video or want to search through its transcript, this is the loader you need. Make sure you use the video_id not the entire URL, as shown below

```
from langchain_community.document_loaders import YoutubeLoader

video_url = 'https://www.youtube.com/watch?v=LKCVKw9CzFo'
loader = YoutubeLoader(video_id='LKCVKw9CzFo', add_video_info=T
data = loader.load()
len(data)
>>> 1
```

We can get the transcript using data[0].page_content and video information using data[0].metadata

Wikipedia

We get the Wikipedia article content based on a search query. The code below extracts the top five articles based on Wikipedia search results. Make sure you install the Wikipedia package with 'pip install Wikipedia'

```
from langchain_community.document_loaders import WikipediaLoade

loader = WikipediaLoader(query='Generative AI', load_max_docs=5
data = loader.load()
len(data)
>>> 5
```

We can control article content length with doc_content_chars_max. We can also get all the information about the article.

```
data[0].metadata.keys()
>>> dict_keys(['title', 'summary', 'source', 'categories', 'pag

for i in data:
    print(i.metadata['title'])
>>>Generative artificial intelligence
AI boom
Generative pre-trained transformer
```

What are Langchain Document Loaders?

Conclusion

LangChain offers a comprehensive and versatile framework for loading data from various sources, making it an invaluable tool for developing applications powered by Large Language Models (LLMs). By integrating multiple file types and data sources, such as CSV files, MS Office documents, PDF files, YouTube videos, and Wikipedia articles, LangChain allows developers to gather and standardize diverse data into Document objects, facilitating seamless data processing and analysis.

In the next article, we will learn why we need to split the documents and how to do it. Stay tuned to Analytics Vidhya Blogs for the next update!

Artificial Intelligence | Deep Learning | Langchain | Large Language Model | LLM Applications | LLMs | Python

D Dsanr
15 Jul, 2024

Generative AI | Github | Intermediate | Large Language Models | LLMs

What are Langchain Document Loaders?

Frequently Asked Questions

Q1. What is LangChain, and why is it important for developing LLM applications?

Ans. LangChain offers a range of functionalities, including loading, splitting, embedding, and retrieving data. It also supports parsing LLM outputs, adding tools and agentic capabilities to LLMs, and integrating with hundreds of third-party services. Additionally, it includes components like LangGraph for building stateful agents and LangSmith for productionizing LLM applications.

Q2. What functionalities does LangChain offer for working with data?

Q3. What are document loaders in LangChain, and what is their purpose?

Q4. How does LangChain handle different types of files and data sources?

RECOMMENDED ARTICLES

90+ Python Interview Questions and Answers (202...

6 Easy Ways to Access ChatGPT-4 for Free

How to Read and Write With CSV Files in Python?

Best Roadmap to Learn Generative AI in 2024

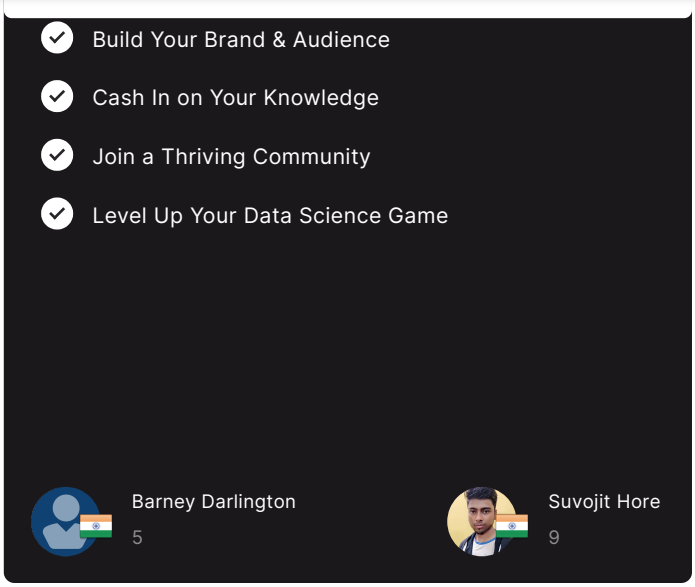
Top 10 Machine Learning Algorithms to Use in 2024

Prompt Engineering: Definition, Examples, Tips ...

What is LangChain?

45 Questions to test a data scientist on basics...

What are Langchain Document Loaders?



What are Langchain Document Loaders?