Financial Document Analysis with LlamaIndex

10-K document question answering using LlamaIndex and Azure OpenAI

Xin Cheng · Follow
5 min read · Mar 11, 2024

This post is based on LlamaIndex sample, but adapated to Azure OpenAI. 10-K document is an annual report required by the U.S. Securities and Exchange Commission (SEC), that gives a comprehensive summary of a company's financial performance. These documents usually have hundred of pages and are time-consuming for people to digest. A 10-K document can have following sections:

- **Business.** This provides an overview of the company's main operations, including its products and services (i.e., how it makes money).
- **Risk factors.** These outline any and all risks the company faces or may face in the future. The risks are typically listed in order of importance.
- **Selected financial data.** This section details specific financial information about the company over the last five years. This section

presents more of a near-term view of the company's recent performance.

- **Management's discussion and analysis** of financial condition and results of operations.
- **Financial statements and supplementary data.**

Large language model is powerful but it does not have access to up-to-date information, the popular way is to build retrieval-augmented generation (RAG) system (retrieve information from external data, and pass to LLM to synthesize the information). The most popular framework to build RAG is LlamaIndex and Langchain.

In this post, we will see how to build RAG to do 10-K QA in minutes. On high-level, basic RAG has following components:

1. Ingest
2. Retrieval
3. Synthesis

Ingest

Download 10-K documents from Lyft, Uber for year 2020, 2021. Store under data/10k folder.

Python Packages: install pypdf to load PDF files.

```
!pip install llama-index pypdf python-dotenv
```

Create .env file to store Azure OpenAI-related environment variables

```
OPENAI_API_TYPE=azure
AZURE_OPENAI_ENDPOINT=
OPENAI_API_VERSION=2023-09-01-preview
AZURE_OPENAI_API_VERSION=2023-09-01-preview
AZURE_OPENAI_API_KEY=
AZURE_OPENAI_MODEL_NAME=gpt-35-turbo
AZURE_OPENAI_DEPLOYMENT_NAME=gpt-35-turbo
AZURE_OPENAI_CHATMODEL_NAME=gpt-35-turbo
AZURE_OPENAI_EMBEDDINGS_MODEL_NAME=
AZURE_OPENAI_EMBEDDINGS_DEPLOYMENT_NAME=
```



```
from pyexpat.errors import messages
import urllib.request
import json
import os
import ssl


from llama_index import (
    StorageContext,
    VectorStoreIndex,
    SimpleDirectoryReader,
    ServiceContext,
    load_index_from_storage,
    set_global_service_context
)
from llama_index.response.pprint_utils import pprint_response
from llama_index.tools import QueryEngineTool, ToolMetadata
from llama_index.query_engine import SubQuestionQueryEngine
```

Define LLM and Embedding model creation utlity methods

```
from llama_index.llms import AzureOpenAI
from llama_index.embeddings import AzureOpenAIEmbedding

import os
import openai

from dotenv import load_dotenv

def load_config():
    load_dotenv()

    api_key = os.getenv("AZURE_OPENAI_API_KEY")
    openai.api_key = api_key

# create LLM and Embedding Model
def create_embedding_model():
    azure_endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")
    api_version = os.getenv("AZURE_OPENAI_API_VERSION")
    api_key = os.getenv("AZURE_OPENAI_API_KEY")

    embed_model = AzureOpenAIEmbedding(
        model=os.getenv("AZURE_OPENAI_EMBEDDINGS_MODEL_NAME"),
        deployment_name=os.getenv("AZURE_OPENAI_EMBEDDINGS_DEPLOYMENT_NAME"),
        api_key=api_key,
        azure_endpoint=azure_endpoint,
        api_version=api_version,
    )
    return embed_model

def create_llm():
    azure_endpoint = os.getenv("AZURE_OPENAI_ENDPOINT")
    api_version = os.getenv("AZURE_OPENAI_API_VERSION")
    api_key = os.getenv("AZURE_OPENAI_API_KEY")

    llm = AzureOpenAI(
        model=os.getenv("AZURE_OPENAI_MODEL_NAME"),
        deployment_name=os.getenv("AZURE_OPENAI_DEPLOYMENT_NAME"),
        temperature=0,
        api_key=api_key,
        azure_endpoint=azure_endpoint,
        api_version=api_version,
    )
    return llm
```

Create LLM and embedding models and set as LlamaIndex default

```
load_config()
embed_model = create_embedding_model()
llm = create_llm()
service_context = ServiceContext.from_defaults(
    embed_model=embed_model, llm=llm
)
set_global_service_context(service_context)
```

Data loading

```
lyft_docs = SimpleDirectoryReader(input_files=["./data/10k/lyft_2021.pdf"]).load
uber_docs = SimpleDirectoryReader(input_files=["./data/10k/uber_2021.pdf"]).load
print(f'Loaded lyft 10-K with {len(lyft_docs)} pages')
print(f'Loaded Uber 10-K with {len(uber_docs)} pages')
```

Vector store indexing

```
# vector index in memory, for persistence, use methods below
lyft_index = VectorStoreIndex.from_documents(lyft_docs)
uber_index = VectorStoreIndex.from_documents(uber_docs)

# Below is how you persist vector index and reload index
vector_persist_dir = './vectordb/10k/lyft'
lyft_index.storage_context.persist(vector_persist_dir)
vector_persist_dir = './vectordb/10k/uber'
uber_index.storage_context.persist(vector_persist_dir)
vector_persist_dir='./vectordb/10k/lyft'
storage_context = StorageContext.from_defaults(persist_dir=vector_persist_dir)
lyft_index = load_index_from_storage(storage_context=storage_context)
vector_persist_dir='./vectordb/10k/uber'
```

```
storage_context = StorageContext.from_defaults(persist_dir=vector_persist_dir)
uber_index = load_index_from_storage(storage_context=storage_context)
```

With above steps, Lyft and Uber 10-K documents are in vector store and ready for retrieval.

Retrieval/Synthesis

Simple QA

```
lyft_engine = lyft_index.as_query_engine(similarity_top_k=3)
uber_engine = uber_index.as_query_engine(similarity_top_k=3)
```

1

```
`response = await lyft_engine.aquery('What is the revenue of Lyft in 2021? Answe
```

Result

```
The revenue of Lyft in 2021 was $3,208.3 million. (Page reference: 79)
```

2

```
response = await uber_engine.aquery('What is the revenue of Uber in 2021? Answer
```

Result

```
The revenue of Uber in 2021 is $17,455 million. (Page reference: 129)
```

Advanced QA — Compare and Contrast

We need to use SubQuestionQueryEngine, which breaks down a complex compare-and-contrast query, into simpler sub-questions to execute on respective sub query engine backed by individual indices.

```
query_engine_tools = [
    QueryEngineTool(
        query_engine=lyft_engine,
        metadata=ToolMetadata(name='lyft_10k', description='Provides information
    ),
    QueryEngineTool(
        query_engine=uber_engine,
        metadata=ToolMetadata(name='uber_10k', description='Provides information
    ),
]

s_engine = SubQuestionQueryEngine.from_defaults(query_engine_tools=query_engine_
```

1

```
response = await s_engine.aquery('Compare and contrast the customer segments and
```

Log, we can see it generates 4 sub questions, 1 for customer segments, 1 for geographies * 2 companies

```
Generated 4 sub questions.
[lyft_10k] Q: What were the customer segments that grew the fastest for Lyft in
[uber_10k] Q: What were the customer segments that grew the fastest for Uber in
[lyft_10k] Q: Which geographies experienced the fastest growth for Lyft in 2021?
[uber_10k] Q: Which geographies experienced the fastest growth for Uber in 2021?
[uber_10k] A: Uber's customer segments that grew the fastest in 2021 were the me
[lyft_10k] A: Lyft's 2021 10-K report does not provide specific information abou
[lyft_10k] A: The context information does not provide any specific details abou
[uber_10k] A: Uber experienced the fastest growth in five metropolitan areas in
```

Result

```
The customer segments that grew the fastest for Lyft in 2021 are not specified i

Regarding the geographies that experienced the fastest growth, the context infor
```

2

```
response = await s_engine.aquery('Compare revenue growth of Uber and Lyft from 2
```

Log

```
Generated 4 sub questions.
[uber_10k] Q: What was the revenue of Uber in 2020?
[uber_10k] Q: What was the revenue of Uber in 2021?
[lyft_10k] Q: What was the revenue of Lyft in 2020?
[lyft_10k] Q: What was the revenue of Lyft in 2021?
[uber_10k] A: The revenue of Uber in 2020 was $11,139 million.
[uber_10k] A: The revenue of Uber in 2021 was $17,455 million.
[lyft_10k] A: The revenue of Lyft in 2021 was $3,208,323,000.
[lyft_10k] A: The revenue of Lyft in 2020 was $2,364,681.
```

Result

```
The revenue growth of Uber from 2020 to 2021 was higher compared to Lyft.
```

Artificial Intelligence | Large Language Models | Machine Learning | Microsoft Azure

Written by Xin Cheng

Follow

340 Followers

Multi/Hybrid-cloud, Kubernetes, cloud-native, big data, machine learning, IoT developer/architect, 3x Azure-certified, 3x AWS-certified, 2x GCP-certified

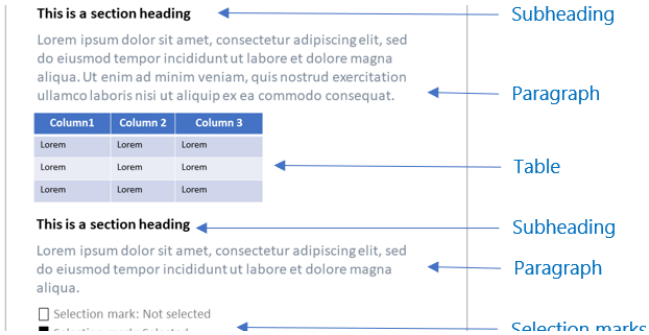
More from Xin Cheng

Xin Cheng

Azure OpenAI GPT-4o with Langchain

Latest model that can reason across audio, vision, and text

May 25 12



Xin Cheng

Document Table Extraction

To Pandas Dataframe with Azure Document Intelligence, Amazon Textract

May 19 2

Xin Cheng

OpenAI Whisper Fine-tuning

With Huggingface transformers and transfer learning

Dec 12, 2023 70 1

Xin Cheng

Data Lakehouse Best Practices and Latest Trends

Articles that are interesting to read

Aug 28

See all from Xin Cheng

Recommended from Medium

Vamshidhar Pandrapagada

Abish P... in Writing in the World of Artificial Intelli...

LLMs with Tabular Data using Langchain and Prompt Engineering

In artificial intelligence and machine learning, the ability to process and analyze tabular dat...

Jul 26 1

Step aside LangChain, building production grade RAG pipelines...

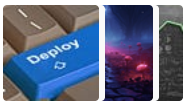
Haystack has emerged as a powerful open-source framework that empowers developer...

Aug 28 1

Lists

Natural Language Processing

1679 stories · 1253 saves



Predictive Modeling w/ Python

20 stories · 1492 saves

AI Regulation

6 stories · 558 saves

Practical Guides to Machine Learning

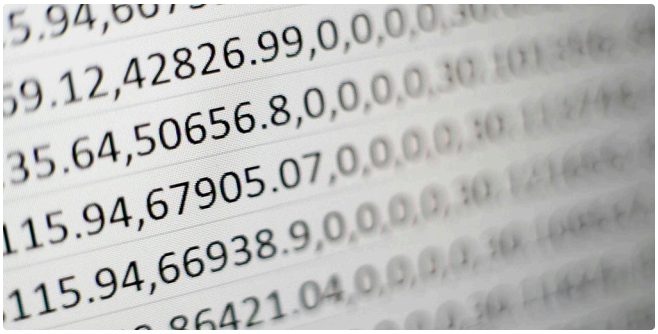
10 stories · 1822 saves

Ming

Comparing LangChain and LlamaIndex with 4 tasks

LangChain v.s. LlamaIndex—How do they compare? Show me the code!

Jan 11 1.2K 9



Intel in Intel Tech

Tabular Data, RAG, & LLMs: Improve Results Through Data...

How to ingest small tabular data when working with LLMs.

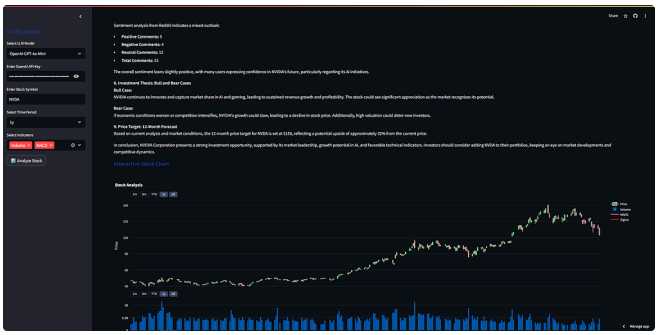
May 14 253 5

Srinath Sridharan

Beyond Tables: The Gateway to Unstructured Data—Web Scrapin...

This article is part of my blog series—Beyond Tables: A Journey into Unstructured Data...

Mar 25 1



Batuhan Sener

AI-Powered Financial Analysis: Multi-Agent Systems Transform...

Click the Link !

Jul 31 180

See more recommendations