Summarize any text based document with 8 lines of Python and LlamaIndex

Thomas Jaensch

4 min read · Sep 30, 2023

> *Use the power of LLMs to summarize or ask questions about any custom document without having to write model plumbing code, LangChain or vector database stuff whatsoever.*

Art my daughter produced in Kindergarten

The need arose at my work to come up with a solution to summarize custom documents using something Llama 2 LLMs just because everybody is crazy about those right now. So I poked around and came up with something that hosts the 7B and 13B parameter Llama 2 models from Hugging Face on Azure Data Lake, and then hooks up the model code via a Databricks notebook powered by a large-ish ML GPU compute cluster with a whole bunch of Hugging Face transformers and pipeline code, LangChain plumbing code to load, chunk, embed, and finally run RetrievalQA chains in combination with a Chroma vector store. The thing worked reasonably well and satisfied the client to give them enough resources and ideas to start experimenting with these things on their own in the given Azure/Databricks environment.

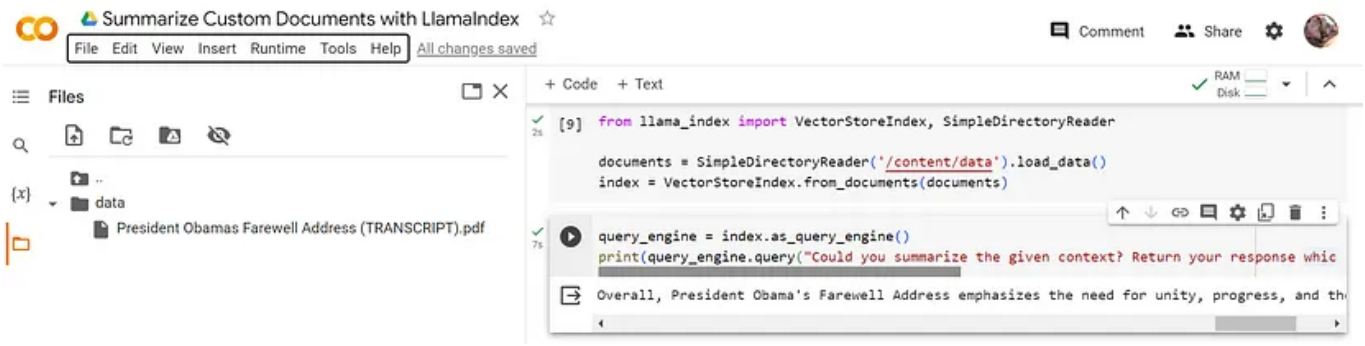
Then a client coworker asked me if we could use LlamaIndex as well...?

To be honest, I didn't really know anything about LlamaIndex at that point at all, started looking at the documentation, and literally less than 30 minutes later had another solution at hand to summarize and ask questions about custom documents that worked just as well as the other thing, but with less than 10 lines of simplistic Python.

This is the whole shebang:

```
pip install llama-index docx2txt pypdf

import os

os.environ["OPENAI_API_KEY"] = "YOUR_OPENAI_API_KEY"

from llama_index import VectorStoreIndex, SimpleDirectoryReader

documents = SimpleDirectoryReader('folder_with_text_based_documents').load_data(
index = VectorStoreIndex.from_documents(documents)

query_engine = index.as_query_engine()
print(query_engine.query("Could you summarize the given context? Return your res
```

Here it is running on Google Colab. No GPU required, just an OpenAI API key:



The PDF to analyze is a 10 page transcript of Barack Obama's Farewell address, and this is what LlamaIndex came up with when asked the following query *"Could you summarize the given context? Return your response which covers the key points of the text and does not miss anything important, please."*:

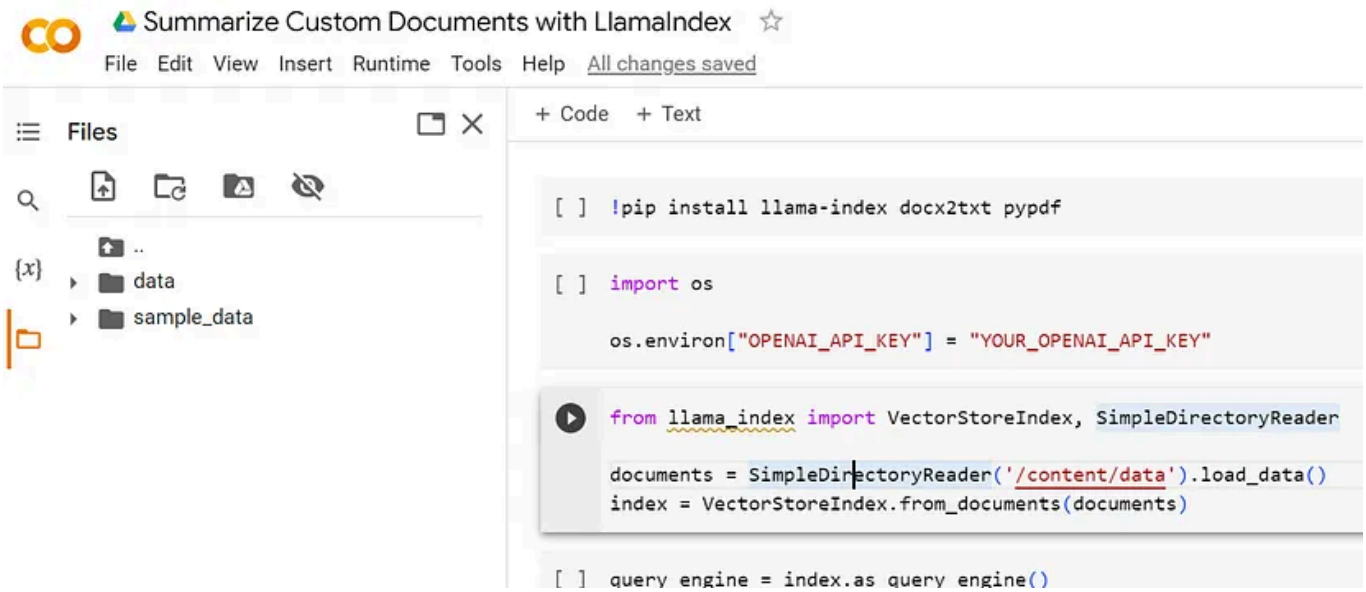
> *The context provided is a transcript of President Obama's Farewell Address. In the address, President Obama discusses the importance of acknowledging and addressing the challenges facing the country. He emphasizes the need to confront problems such as climate change and terrorism, while also protecting the values that define the nation. President Obama highlights the progress made in race relations but acknowledges that more work needs to be done. He calls for upholding laws against discrimination and for hearts to change, urging empathy and understanding among all Americans. The address also touches on economic issues and the importance of creating opportunities for all people. Overall, President Obama's Farewell Address emphasizes the need for unity, progress, and the preservation of democratic values.*

Weirdly, LlamaIndex uses OpenAI's gpt-3.5-turbo model by default according to the documentation and only fallbacks to using a Llama 2 model if no OpenAI API key is provided. 🤪 I did try that on Google Colab but even with choosing a GPU runtime it doesn't seem to work. It does work on Databricks with an Azure Standard_NV12s_v3 compute instance (224 GB Memory, 2

GPUs), albeit it is very slow and takes minutes to download the model to a temp folder on the server and then more minutes to compute the results versus seconds when using the OpenAI API.

Steps to try this yourself

- Open the Google Colab notebook by clicking here.
- Insert your own OpenAI API key in 2nd cell of the notebook.
- Create a folder called “data” in the Files tab of the Colab UI:



- Upload the sample PDF file I used above (get it from here) into the “data” folder.
- Click on Runtime tab and then “Run all”.

This works with text-based files of all kinds (Word documents, PDFs, text files, CSVs), and it can also ingest several documents at once from the directory specified that can then be queried. As with all LLMs, the results are

basically always written in very nice English but can also be misleading, or even completely wrong, and miss important information.

Still, regarding the amount of effort put into this, the outcome is quite impressive.

GitHub repo:
https://github.com/tjaensch/llamaindex_summarize_custom_docs

Llamaindex Llm NLP Machine Learning Python

Written by Thomas Jaensch

Edit profile

21 Followers

WFH, staring at code

More from Thomas Jaensch