Wilbert Misingo
Posted on Feb 12 • Updated on Feb 26

💖 2

How you can create your own custom chatbot with your own custom data using Google Gemini API all for free

#codenewbie #tutorial #beginners #machinelearning

INTRODUCTION

Up until now, the conventional method for building multimodal models involves learning independent parts for various modalities and then piecing them together to approximate some of this functionality. Certain activities, like describing visuals, may be areas in which these models excel, but they have trouble with more conceptual and sophisticated reasoning.

As to this moment, Google's most adaptable model to date, Gemini can operate well on a wide range of platforms, including mobile phones and data centers. Its cutting-edge features will greatly improve how developers and business clients use AI to create and grow.

Google have optimized Gemini 1.0, their first version, for three different sizes:

1. Gemini Ultra, the largest and most capable model for highly complex tasks.
2. Gemini Pro, the best model for scaling across a wide range of tasks.
3. Gemini Nano, the most efficient model for on-device tasks.

This makes Gemini one of the most capable models in the world and thus creating an opportunity for people to explore it in many ways. With respect to that google have also released a huge free tier that could really be helpful to help people to create some cool stuff.

In this article, you are going to learn how to create your own custom chatbot using your own data using the Google Gemini API free tier.

IMPLEMENTATION

Step 01: Getting your API Key

To get started you first need to create an API key that would be used to reference the model, to create your API key you need to signup and create a new key at the Google AI Studio at Google Maker suite platform.

Step 02: Installing Libraries

To build a chatbot, we need to use some Python libraries that are specifically designed for natural language processing and machine learning. These would help to facilitate the implementation of the chatbot.

```
pip install -q llama_index google-generativeai chromadb pypdf transfor
```

Step 03: Importing Libraries

To begin, we need to import the necessary libraries and modules that will be used throughout the chatbot creation process. The code snippet below demonstrates the required imports.

```
from llama_index import SimpleDirectoryReader, VectorStoreIndex
from llama_index.llms import Gemini
from IPython.display import Markdown, display
from llama_index import ServiceContext
from llama_index.vector_stores import ChromaVectorStore
from llama_index.storage.storage_context import StorageContext
from llama_index.prompts import PromptTemplate
import chromadb
import os
```

Step 04: Loading data from the knowledge base

To make the chatbot knowledgeable about your anything, you need to load your desired documents into the chatbot's index. The documents can be in different formats, in this demo all the documents were in pdf. The code snippet below demonstrates loading the data from a specified directory.

```
documents = SimpleDirectoryReader("./data").load_data()
```

Step 05: Innitializing the data embedding database

ChromaDB, a versatile tool for storing vector representations and embeddings, serves as the backbone of our system. Initializing ChromaDB involves creating a client and establishing a collection for storing document embeddings. This sets the stage for efficient storage and retrieval of vector representations.

```
db = chromadb.PersistentClient(path="./embeddings/chroma_db")
chroma_collection = db.get_or_create_collection("quickstart")
```

Step 06: Innitializing the model

Initializing Gemini and creating a service context involves setting up the necessary environment and defining how the model is going to interact and process both user inputs and data source.

```
os.environ['GOOGLE_API_KEY'] = 'PUT YOUR GOOGLE API KEY HERE'
llm = Gemini()
service_context = ServiceContext.from_defaults(llm=llm, chunk_size=80
```

Step 7: Creating Vector Store Index

With the foundational components in place, the next step is to create the vector store index from the loaded documents. This process involves indexing the documents using the specified vector store and service context.

```
index = VectorStoreIndex.from_documents(documents, storage_context=sto
)
```

Step 08: Defining Prompt Template and Configuring Query Engine

To facilitate question-answering, a prompt template is defined. This helps the bot to understand how it should interact with the user according to how you set, giving it tone, role etc. .The query engine is then configured to leverage this template. This step lays the groundwork for engaging in meaningful interactions with the indexed data.

```
template = (
    "We have provided context information below. \n"
    "---------------------\n"
    "{context_str}"
    "\n---------------------\n"
    "Given this information, please answer the question: {query_str}\
)

qa_template = PromptTemplate(template)

query_engine = index.as_query_engine(text_qa_template=qa_template)
```

Step 09: Performing a query

The culmination of our journey involves performing a sample query and displaying the result.

```
response = query_engine.query("What its the shape of the earth?")
print(responce)
```

CONCLUSION

The implementation of a text-to-vector indexing system using ChromaDB, Gemini, and VectorStore opens up a realm of possibilities for advanced NLP applications. This comprehensive guide serves as a foundation for building sophisticated text-based applications. As you continue your exploration, feel free to experiment with different document sets, query templates, and parameters to tailor the system to your specific requirements.

Happy coding!

Do you have a project 🚀 that you want me to assist you 🤝😊: email me 📧

Wanna be the first to know about my posts:-

Follow ✅ me on Twitter/X 𝕏

Follow ✅ me on LinkedIn 💼

Top comments (0)

Code of Conduct • Report abuse

arm Arm PROMOTED •••

[Learn How to Develop a Creative Mindset for Software Development](#)

🧒🏽 Tap into your child-inspired creativity.
🤝 Embrace fearless problem-solving.
💡 Create ground-breaking solutions.
🔑 Level up your design and development skills.
🏆 Learn from an award-winning engineer and innovator.

Register Now

Wilbert Misingo

Hello!! I am a Machine Learning Engineer and Software Developer whom is always interested in creating custom tech solution with the aid of Computer Vision and Chatbots

LOCATION
Dar es salaam, Tanzania

PRONOUNS
He/Him

WORK
Machine Learning Engineer

JOINED
Aug 26, 2022

More from Wilbert Misingo

12 Super Cool Machine Learning Projects that you didn't know are Open Source and made of JavaScript
#machinelearning #javascript #webdev #codenewbie

🚀 Boost Your Career as a Developer: Mastering the Power of Effective Communication! 🗣️✨
#beginners #tutorial #career #codenewbie

ChatGPT knowledge is out of date all this time, this is how to keep it to date
#beginners #tutorial #codenewbie #machinelearning