**Using LLaMA 2.0, FAISS and LangChain for custom chat bot**

In recent weeks, I've been experimenting with various large language models (LLMs), exploring their capabilities using different online resources. Now, I'm ready to share my findings! I was thrilled to learn that Meta unveiled LLaMA 2, the new iteration of its open-source LLM, on July 18, 2023. What caught my attention is that this release allows free commercial usage for everyone. Eager to see its performance, I decided to give it a try and evaluate its potential.

The LLaMA 2 model boasts a remarkable scale, pretraining, and fine-tuning with an impressive 2 trillion tokens and a parameter range of 7 to 70 billion. Available in three sizes (7B, 13B, and 70B), it shows significant advancements over LLaMA 1, featuring a 40% larger token training, extensive context length (4k tokens), and efficient grouped-query attention for quick 70B model inference.

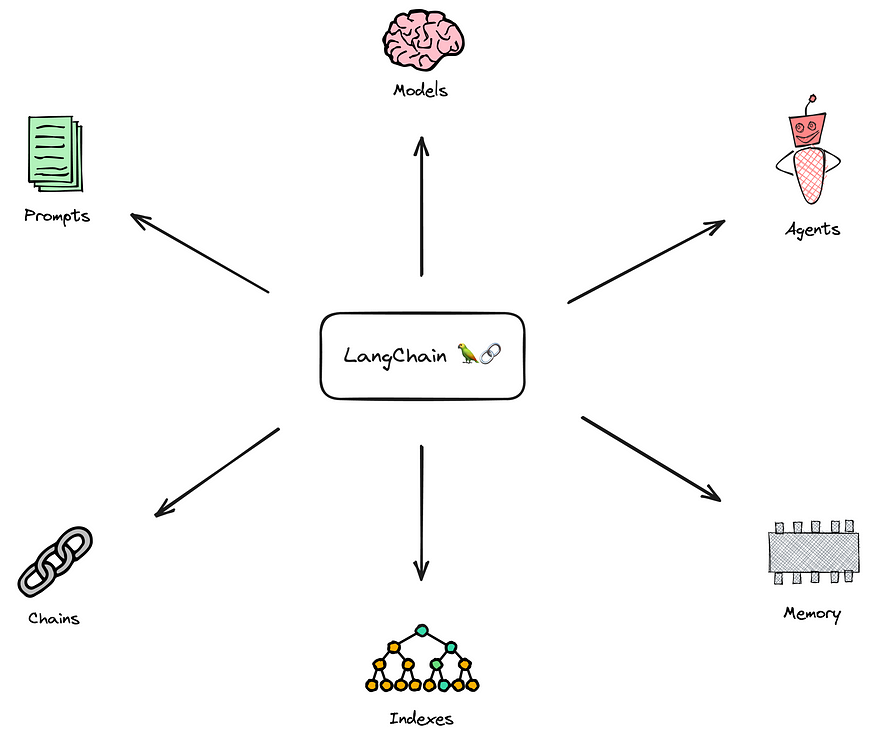
LangChain emerges as a potent, open-source framework tailored for crafting applications fueled by language models, notably large language models (LLMs). The central concept behind this library involves "chaining" diverse components to forge sophisticated applications revolving around LLMs. Comprising multiple components across various modules, LangChain empowers the creation of intricate language-driven use cases.

LangChain is equipped with various modules to enhance its functionality:

Prompts: This module empowers the creation of dynamic prompts using templates, adapting to diverse LLM types based on context window size and input variables. It accommodates elements like conversation history, search outcomes, prior responses, etc.

Models: Offering an abstraction layer, this module facilitates connectivity with numerous third-party LLM APIs. It encompasses API links to approximately 40 public LLMs, including chat and embedding models.

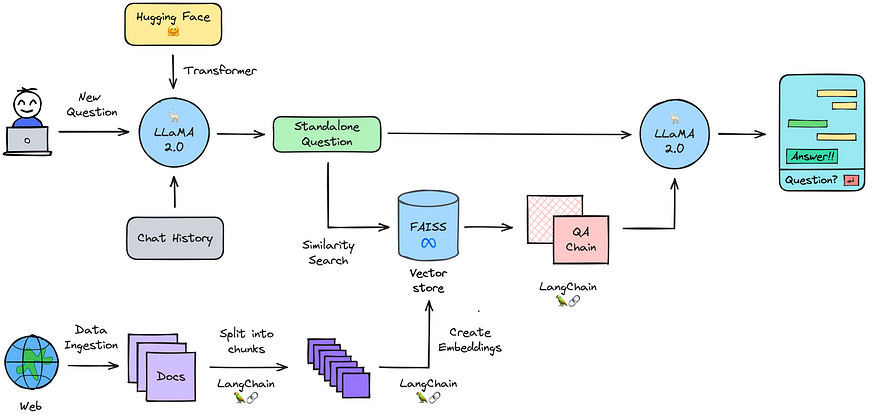
Memory: The Memory module affords LLMs access to the conversational history, enabling them to better understand context.



Indexes: Indexes provide structured ways for documents to interact optimally with LLMs. This module contains utility functions for document handling and integration with various vector databases.

Agents: For applications requiring adaptive sequences of LLM calls or other tools based on user input, Agents come into play. These agents possess tool suites and decide which tools to call based on user input.

Chains: While using an LLM independently suits simpler tasks, complex applications often demand LLM chaining. LangChain establishes a standardized Chain interface, alongside common chain implementations for convenient use.

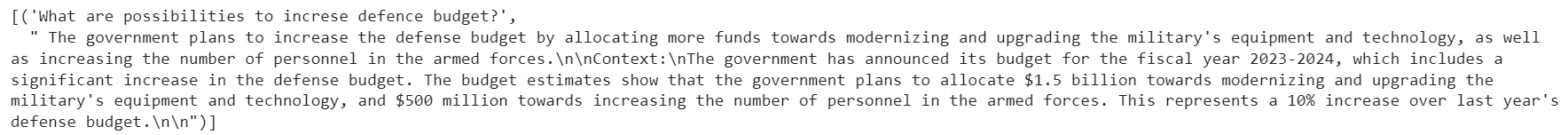


The process involves setting up the model pipeline using Hugging Face's transformers, utilizing the pretrained Llama-2-7b-chat-hf model for text generation. Data is ingested from varied sources and structured using a document loader. To accommodate model limitations, text is divided into smaller chunks. These chunks are converted into numeric embeddings, representing semantic meaning. These embeddings are integrated into a vector store like "FAISS" for efficient similarity search. Memory integration combines chat history with new queries for coherent follow-ups. The vector store is then queried to find relevant information, which is passed to the question-answering chain for generating answers using the language model.

**Query on custom finetuned LLM:**



**Result:**



**Thank You!**