**LLM RAG:**

**Introduction:**

* Large Language Models (LLMs) are remarkable at compressing knowledge about the world into their billions of parameters. However, LLMs have two major limitations: They only have up-to-date knowledge up to the time of the last training iteration. And they sometimes tend to make up knowledge (hallucinate) when asked specific questions. Using the RAG technique, we can give pre-trained LLMs access to very specific information as additional context when answering our questions. In this article, I will walk through the theory and practice of implementing Google’s LLM Gemma with additional RAG capabilities using the Hugging Face transformers library, LangChain, and the Faiss vector database.
  + An overview of the RAG pipeline is shown in the figure below, which we will implement step by step.
  + **A diagram of a software

    Description automatically generated**

**Retrieval-Augment Generation (RAG):**

* The term “Retrieval Augmented Generation” (RAG) comes from the paper “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks” from the year 2020 by researchers at Facebook AI Research, University College London, and New York University
* A diagram of a computer

  Description automatically generated
* The basic idea is as follows:
  + We start with a knowledge base, such as a bunch of text documents “z\_i” from Wikipedia, which we transform into dense vector representation “d(z)” (also called **embeddings**” using an encoder model.
  + Next, if we have a user question x, we also transform this text into an embedding vector q(x) using an encoder model.
  + Next, if we have a user question x, we also transform this text into an embedding vector q(x) using the same encoder model.
  + Then, we want to find similar vector to q(x) from all available d(z) using a similarity metric.
  + Encoding our question and finding similar documents in the knowledge base is called the retriever component.
  + Given our question and the additional context from the retrieved documents, we can feed this into an LLM called the generator component and get our answer.
  + The generator is usually an encoder-decoder or decoder-only LLM.
  + Let’s implement this RAG pipeline

**Generator Component: LLM Model**

* The generator is an LLM that takes text (a question) as input and produces new text as output. The original RAG paper used BART-large as its generator LLM model. However, nowadays there are many open source LLMs to choose from. For a RAG chatbot,, I want an instruction model that has been fine-tuned on conversational data and that is small enough for my local machine.
  + For this tutorial, I choose Google’s recently released model Gemma-2b-it. However, feel free to try another model.