**RAG Gemma LangChain**

Setup

* Need computational power to run inference on LLM
* 7B model 🡪 GPU with 8 GB of VRAM – load quantized version
* Load model once and run inference separately

Architecture

* Build webserver to load LLM once to web server (Flask)
* Send queries to it and get respond back
* Monitor and test with Postman

LLM Context Length

* Performance is highest when relevant information occurs at start or end of context
* Degrades when models must reason over information in the middle of context

**Gemini Python API**

Introduction

* Multimodality: text, image, video, audio and code
* Ultra outperforms GPT-4 on multiple benchmarks across modalities

Google AI Studio

* Playground to Gemini Pro models and settings
  + Model settings
  + Safety settings: harassment, hate speech, sexually, dangerous
* Setup Google Gemini
  + Need API key to use Python SDK 🡪 embed in script
  + Genai.list.models() 🡪 check which models are accessible

Text Generation

* Genai.GenerativeModel 🡪 download model
* Generate\_content 🡪 generate response by passing question
* Prompt.feedback 🡪 safety ratings of the answer
* Candidates 🡪 model generates multiple outputs and choose best
* No answer will be given if it is a harmful prompt because no candidates
  + Can set safety settings to include threshold for each type
* Configure hyperparameters with GenerationConfig
  + Candidate\_count
  + Stop\_sequences
  + Max\_output\_tokens
  + Temperate

Multimodal Generation

* Gemini pro vision accepts both image and text
* Multimodal Generation (generate.content)
  + Import picture
  + Import model
  + Invoke LLM and pass prompt with picture as input
* Chat Generation (start\_chat)
  + Message 🡪 send\_message()
  + History 🡪 chat conversation with roles

Embeddings

* Use Gemini to embed texts
* Code
  + Embed.content()
  + Specify parameters 🡪 model, content, task\_type, title

**RAG Gemini**

General

* Build RAG application with Gemini and Langchain
* Use Google GenAI for embeddings
* API Key: <https://www.youtube.com/watch?v=RqO4qw0TfNE>

Multimodal Inference using LangChain to access Gemini API

* Access model and generate response via LangChain
* Define prompt with type of input and output (multimodal)
* Invoke LLM with message/prompt

Chat with Document using RAG

* Load PDF with PyPDFLoader and split documents into chunks (pages)
* Split and embed chunks with Google GenAI model
* Store chunks in a vector store (knowledge base)
* Embed the question and search semantically the most appropriate chunks with (cosine) similarity to generate final answer
* Coding 🡪 setup a chain (RetrievalQA) with model, retriever and return source
* Define a prompt template go get more detailed results
  + Keep answer concise if you don’t know then say that you do not know
  + Or answer question from a certain role