Install Required Packages

```
! pip install -q --upgrade google-generativeai langchain-google-genai python-dotenv
```

```
#///// prompt: create a .env file in the workspace
# https://makersuite.google.com/

!echo -e 'AIzaSyAc3ma3dyWjQzyVygkoMasnLBtcEuz-jlk' > .env
```

```
!ls -a
```

```
.  ..  .config  .env  sample_data
```

```
!pip install python-dotenv
```

```
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1
```

```
from dotenv import load_dotenv
load_dotenv()
```

```
True
```

```
from IPython.display import display
from IPython.display import Markdown
import textwrap


def to_markdown(text):
  text = text.replace('•', '  *')
  return Markdown(textwrap.indent(text, '> ', predicate=lambda _: True))
```

```
import google.generativeai as genai
import os
```

```
import os
genai.configure(api_key='AIzaSyAc3ma3dyWjQzyVygkoMasnLBtcEuz-jlk')
```

Text Generation

```
model = genai.GenerativeModel(model_name = "gemini-pro")
model
```

```
genai.GenerativeModel(
    model_name='models/gemini-pro',
    generation_config={},
    safety_settings={},
    tools=None,
    system_instruction=None,
    cached_content=None
)
```

```
prompt = [
    "What is Mixture of Experts?",
]

response = model.generate_content(prompt)
```

```
to_markdown(response.text)
```

Mixture of Experts (MoE) is an ensemble machine learning model that combines multiple expert models to improve predictive performance. It consists of:

1. Gate Network:

- A neural network that receives the input data and assigns a weight (gate) to each expert model.
- These gates represent the probability of each expert making the best prediction for a given input.

2. Expert Models:

- A set of individual neural networks that are specialized in different aspects of the problem.
- Each expert is trained independently on a subset of the data.

3. Output Layer:

- A layer that combines the predictions of the expert models using the gate values.
- The final output is a weighted average of the expert predictions, with higher weights assigned to more confident experts.

How MoE Works:

1. The input data is passed through the gate network, which assigns gates to each expert.
2. Each expert model makes a prediction for the input data.
3. The predictions of the experts are weighted by their gate values and combined in the output layer.
4. The final output is the weighted combination of expert predictions.

Advantages of MoE:

- **Improved Accuracy:** By leveraging the strengths of multiple experts, MoE can make better predictions than individual models.
- **Efficient Training:** Experts are trained independently, allowing for parallel training and reducing computational cost.
- **Flexibility:** New experts can be easily added or removed to adapt to changing data or tasks.
- **Interpretability:** Gate values provide insights into which experts contribute most to the final prediction.

Applications:

MoE has been successfully used in:

- Natural language processing
- Computer vision
- Speech recognition
- Recommendation systems

```
# response.prompt_feedback
# response.candidates
```

Use LangChain to Access Gemini API

```
!pip install langchain
!pip install google-generativeai
```

```
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from google-generativeai) (4.12.2)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/python3.10/dist-packages (from google-ai-generativelang
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /usr/local/lib/python3.10/dist-packages (from google-api
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in /usr/local/lib/python3.10/dist-packages (from google-api-core->google-
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth>=2.15.0->google-ge
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth>=2.15.0->google-ger
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth>=2.15.0->google-generative
Requirement already satisfied: httplib2<1.dev0,>=0.19.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client->go
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from google-api-python-
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from google-api-python-client->googl
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.10/dist-packages (from pydantic->google-generativeai)
Requirement already satisfied: pydantic-core==2.23.4 in /usr/local/lib/python3.10/dist-packages (from pydantic->google-generativeai)
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core[grpc]!=2.0.*,
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in /usr/local/lib/python3.10/dist-packages (from google-api-core[grpc
Requirement already satisfied: pyparsing!=3.0.0,!=3.0.1,!=3.0.2,!=3.0.3,<4,>=2.4.2 in /usr/local/lib/python3.10/dist-packages (from h
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-au
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->google-api
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->goog
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0.dev0,>=2.18.0->goog
```

```python
from langchain_google_genai import ChatGoogleGenerativeAI
```

```python
llm = ChatGoogleGenerativeAI(model="gemini-pro", google_api_key="AIzaSyDW_n54TMnahHUp9pMmKu0SjvhujCkSB7I")
```

```python
result = llm.invoke("What is the sky blue ?")
```

```python
to_markdown(result.content)
```

> The sky is blue because of the way light interacts with the molecules in the atmosphere. White light from the sun is made up of all the colors of the rainbow. When sunlight passes through the atmosphere, the blue light is scattered more than the other colors because it has a shorter wavelength. This is because the blue light waves are more likely to interact with the molecules in the atmosphere, which are much smaller than the wavelength of light. The other colors of light, such as red and yellow, are not scattered as much, so they travel in a straight line to our eyes. This is why we see the sky as blue.

Chat

```python
model = genai.GenerativeModel("gemini-1.5-flash")
chat = model.start_chat(history=[])
chat
```

```
ChatSession(
    model=genai.GenerativeModel(
        model_name='models/gemini-1.5-flash',
        generation_config={},
        safety_settings={},
        tools=None,
        system_instruction=None,
        cached_content=None
    ),
    history=[]
)
```

```python
response = chat.send_message(
    "In one sentence, explain how a computer works to a young child."
)
to_markdown(response.text)
```

> A computer is like a really fast brain that follows instructions from you to do things like show you pictures, play games, and help you learn!

```python
chat.history
```

```
[parts {
   text: "In one sentence, explain how a computer works to a young child."
 }
 role: "user",
 parts {
```

```
          text: "A computer is like a really fast brain that follows instructions from you to do things like show you pictures, play games,
       and help you learn! \n"
        }
        role: "model"]
```

```
response = chat.send_message(
    "Okay, how about a more detailed explanation to a high schooler?", stream=True
)

for chunk in response:
    print(chunk.text)
    print("_" * 80)
```

```
Computers work by
________________________________________________________________________________
 taking instructions from you, converting them into a series of ones and zeros (binary
________________________________________________________________________________
 code), processing those instructions using a central processing unit (CPU), and then displaying the results
________________________________________________________________________________
 on your screen or through other outputs.
```

```
for message in chat.history:
    display(to_markdown(f"**{message.role}**: {message.parts[0].text}"))
```

user: In one sentence, explain how a computer works to a young child.

model: A computer is like a really fast brain that follows instructions from you to do things like show you pictures, play games, and help you learn!

user: Okay, how about a more detailed explanation to a high schooler?

model: Computers work by taking instructions from you, converting them into a series of ones and zeros (binary code), processing those instructions using a central processing unit (CPU), and then displaying the results on your screen or through other outputs.

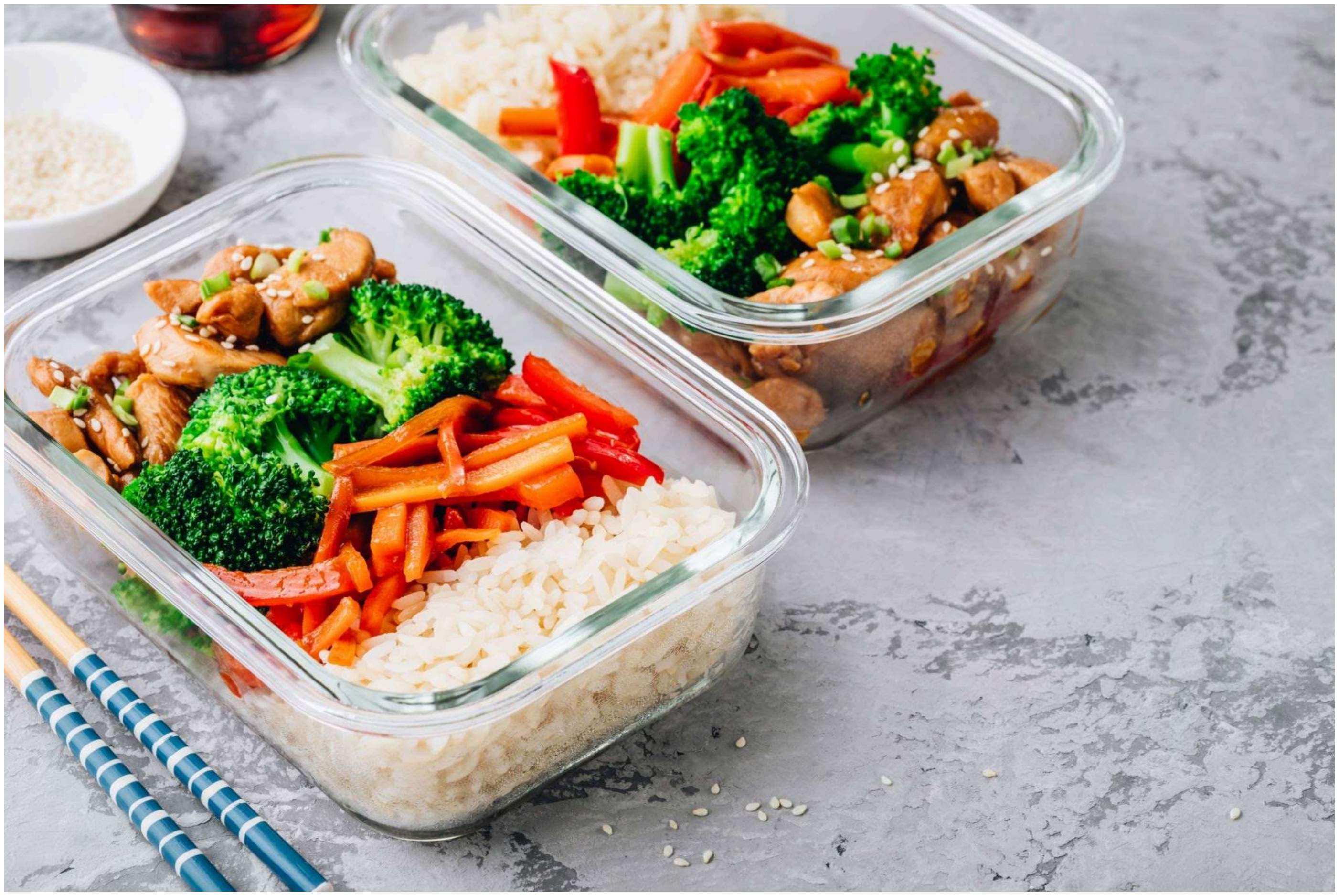
Image & Text Generation

```
!curl -o image.jpg https://t0.gstatic.com/licensed-image?q=tbn:ANd9GcQ_Kevbk21QBRy-PgB4kQpS79brbmmEG7m3VOTShAn4PecDU5H5UxrJxE3Dw1JiaG17V88QI
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  405k  100  405k    0     0  2021k      0 --:--:-- --:--:-- --:--:-- 2025k
```

```
import PIL.Image

img = PIL.Image.open("image.jpg")
img
```



```
model = genai.GenerativeModel("gemini-1.5-flash")
```

```
response = model.generate_content(img)

to_markdown(response.text)
```

The image shows two glass containers filled with rice, vegetables, and chicken. The containers are on a light grey surface, and there are some chopsticks and sesame seeds scattered around them. The vegetables include broccoli, red pepper, and carrots. The chicken is in a teriyaki sauce.

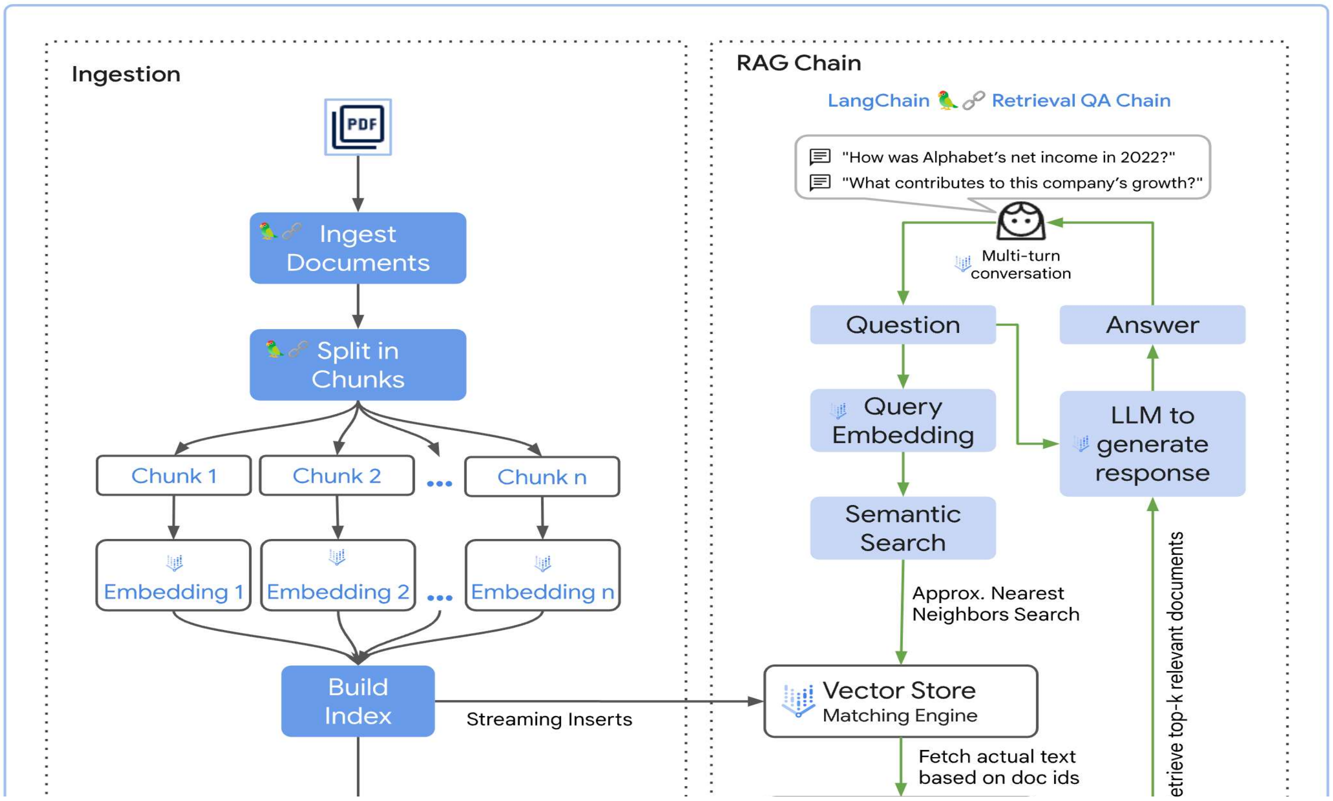
```
response = model.generate_content(
    [
        "Write a short, engaging blog post based on this picture. It should include a description of the meal in the photo and talk about my
        img,
    ],
    stream=True,
)
response.resolve()
```

```
to_markdown(response.text)
```

The Joy of Meal Prepping (and This Delicious Chicken & Veggie Bowl)

Chat with Documents

to_markdown(f"""


Ingestion
Ingest Documents
Split in Chunks
Chunk 1
Chunk 2
Chunk n
Embedding 1
Embedding 2
Embedding n
Build Index
Streaming Inserts
RAG Chain
LangChain 🦜🔗 Retrieval QA Chain
"How was Alphabet's net income in 2022?"
"What contributes to this company's growth?"
Multi-turn conversation
Question
Answer
Query Embedding
LLM to generate response
Semantic Search
Approx. Nearest Neighbors Search
Vector Store
Matching Engine
Fetch actual text based on doc ids
etrieve top-k relevant documents