RAG (Retrieval Augmented Generation)

session 1
==========

Prompt engineering -> RAG -> Fine Tuning -> Pre Training

Transformers Architecture - Attention, MLP

Transformers Library - pipeline (Model Inference)

Pre Training - bigram, MLP (forward pass, backward pass to adjust the weights)

Applications of AI (Applied AI)

RAG (the most important thing)

Limitations of LLM (Generation Models)

- the pretrainted model might not be upto date.
- trained on public data (not private)
- I ask what is RAG

We solve the above issues by giving context to the LLM & saying please refer this while answering.

you joined my first big data batch 2018.

Hadoop

2025

I am asking you, tell me what is a dataframe in apache spark

a document on apache spark you feel free to refer this while answering.

you - LLM

document - Context

are we doing any extra training?

Pre trained

are the model weights getting updated?

No

Session 2
===========

Local System

Databricks (community / Azure Databricks)

Google Colab

"What is an ACE?"

"What is a Volley?"

```
from transformers import pipeline

pipe = pipeline("text-generation", model="Qwen/Qwen2.5-1.5B-Instruct")
```

Session 3
===========

Google Colab

Databricks

Local

model="Qwen/Qwen2.5-1.5B-Instruct" (not gated model) meta-llama/Llama-3.2-1B-Instruct (gated model)

if its a gated model

- you request access for it (and wait until its granted)
- create a access token

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

local

databricks

google colab

- gated model (hugging face token)
- not gated model

=========

session 4
==========

Context

a book on apache spark

it has answer to 100 questions

1 page for each chunk and index

How to automate, we can't give the context manually.

Think that we have a document which tells about tennis, this document has a lot of info and with this info we can answer many questions.

document -> parse & cleaning -> chunks -> vector embeddings -> store these embeddings in vector DB.

consider there are 7 chunks

text chunks

vector embeddings - 7

embedding size - 384

each chunk

Basic Rules
- A match can be played as best of three or five sets.
- Each set consists of games, and each game consists of points.
- Points are scored as **0 (Love), 15, 30, 40**, and then **game**.
- A player must win a game by at least **two points**.
- The ball must land within the designated court boundaries.

embedding size - 384

store to vector DB (collection / Index)

7 chunks

384 floating point numbers

embedding model A

Data pipeline is done.

query: what are famous tournaments in tennis?

embedding model A - what are famous tournaments in tennis?

question embedding

context embedding (7 embeddings)

similarity search is applied

context: ## Famous Tournaments
- **Grand Slam Events**:
  - Australian Open
  - French Open
  - Wimbledon
  - US Open

prompt - question
context

2 models
- embedding model (to take text chunks and create vector embeddings)
- Generation model

session 5
==========

/content/sample_data/tennis_details.md

all-MiniLM-L6-v2

384 embeddings for each chunk

in chromadb

similarity_search_with_score function returns cosine distance scores, lower scores represent higher similarity.

- embedding model
- generation model

chromadb

pinecone
weaviate
milvus
pgvector

data pipeline
document -> parse/cleaning -> chunk -> vector embedding -> vector DB

question pipeline
question -> embedding model -> similarity search in vector DB -> context -> prompt engineering (augment the prompt with the context) -> response

RAG

Retrieval
Augmented
Generation

Different things are involved
==============================

- document loading and parsing
- data chunking
- embedding model
- vector DB
- Generation Model
- Chaining tool