TECHLENT

# 高阶大语言模型课程

Huajun Zeng

12/13/2024 - 1/31/2025
（12月27日和1月3日放假，共计6次课）
每周五 5pm-7pm PT / 8pm-10pm ET

# 课程安排

TECHLENT

| Week | Date | Content | Week | Date | Content |
|---|---|---|---|---|---|
| 1 | 2024-12-13 | Retrieval Augmented Generation (RAG) for LLM<br>● Why augmenting LLMs?<br>● Methods for LLM augmentation<br>● Augmenting LLMs with retrieval<br>● Augmenting LLMs with fine tuning | 4 | 2025-01-17 | Pipeline for LLM Applications: From Code to Products<br>● Full stack LLM: tools needed for an LLM application<br>● Case study: build an LLM app from ground |
| 2 | 2024-12-20 | Chatbot Building with LLM APIs<br>● Environment setup<br>● Introduction to LLM APIs<br>● Using chat completion APIs<br>● Using fine tuning APIs | 5 | 2025-01-24 | More LLM Applications and Course Project<br>● Showcase of potential LLM applications for productivity, creativity and more<br>● More advanced LLM applications: AI Agent, Multi-modality, etc.<br>● Introduction to the course project: requirement and discussion |
| 3 | 2025-01-10 | Chatbot Building with LLM Frameworks and Vector Database<br>● Introduction to Langchain and LlamaIndex<br>● Case study: a chatbot from Langchain and vector database | 6 | 2025-01-31 | Project Presentation<br>● Student presentation on the course project |

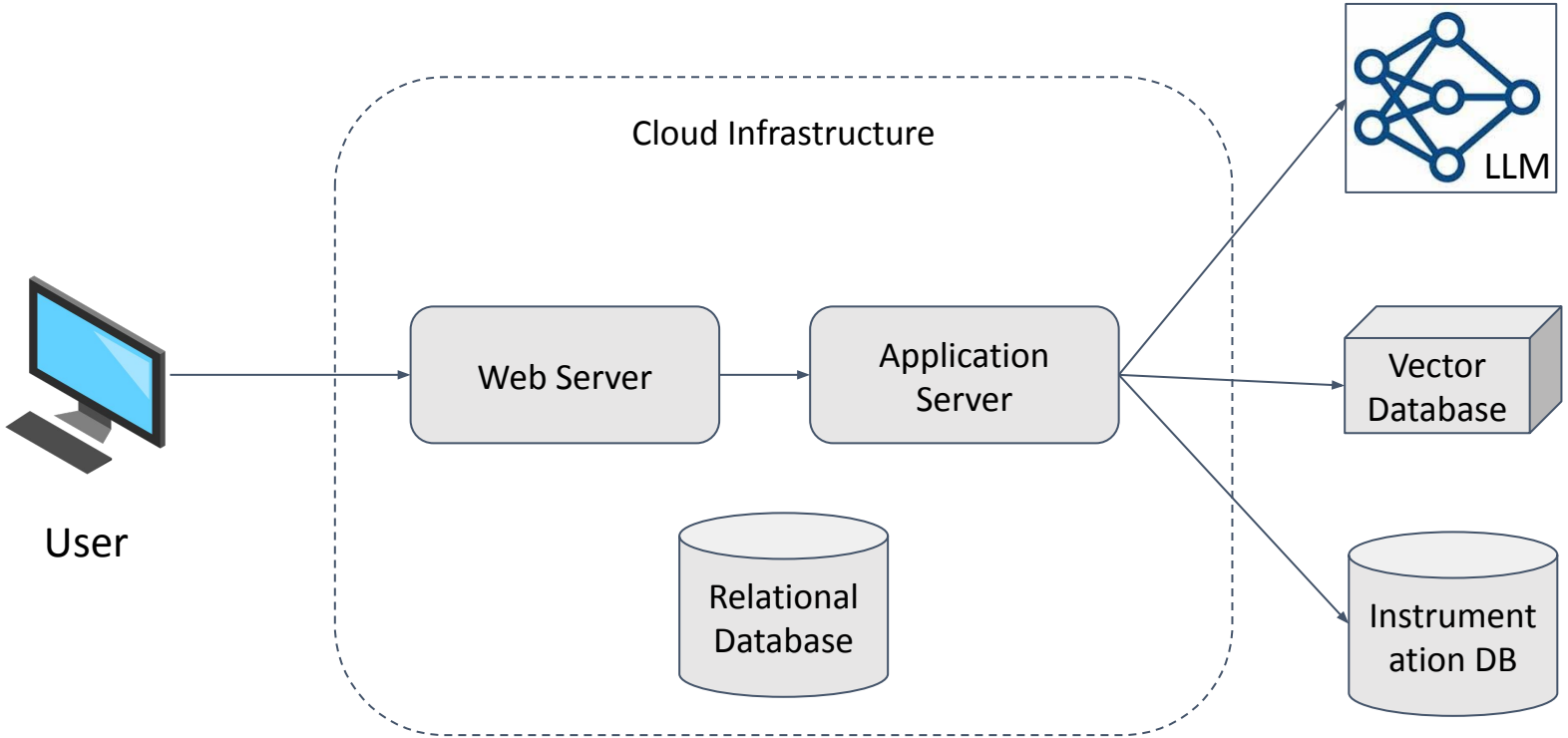
# 家庭作业回顾

- Implement a RAG-based Chatbot using Langchain
  - A console program to continuously receive user input and respond
  - Try smaller dataset first and then larger dataset
  - Refer to the examples in https://github.com/hzeng-otterai/chatbot-example/tree/main/backend_langchain
- Common Issues
  - Understand what’s going on in Langchain
  - Please remove API keys from github code

# 第四课: 全栈开发LLM/RAG应用

- 基于Flask框架的Web应用
- RAG系统的优化

# 应用架构


Cloud Infrastructure
User
Web Server
Application Server
Relational Database
LLM
Vector Database
Instrument ation DB

# 技术栈

- Web Server
  - Flask + HTML + Javascript
- Relational Database
  - SQLite
- Orchestration
  - LangChain
- Vector Database
  - Pinecone
- Deployment
  - AWS EC2

- Those are not the only choices
- Considerations
  - rapid prototyping
  - future extensibility

# Environment setup

- Version management: git
- Python environment:
  - conda create -n chat_env python=3.11
  - pip install -r requirements.txt
  - .env for the keys (should not commit to git)
- IDE
  - Visual Studio Code

# 看代码

# Typical Structures of a Flask App

- app.py
- views.py
- models.py
- <templates> directory

Refer to fullstack_flask_minimal directory for a minimal example

# Good Practices for Flask Apps

- Using a WSGI server for better concurrency
  - gunicorn src.app:app
  - gunicorn.conf.py
- Use Object Relational Mapper (ORM)
  - SQLAlchemy
  - Database sync with the model.py
- Stateless service
  - Avoid keep state in code
  - Manage state outside the application e.g. a DB

# Adding LLM Related Logics

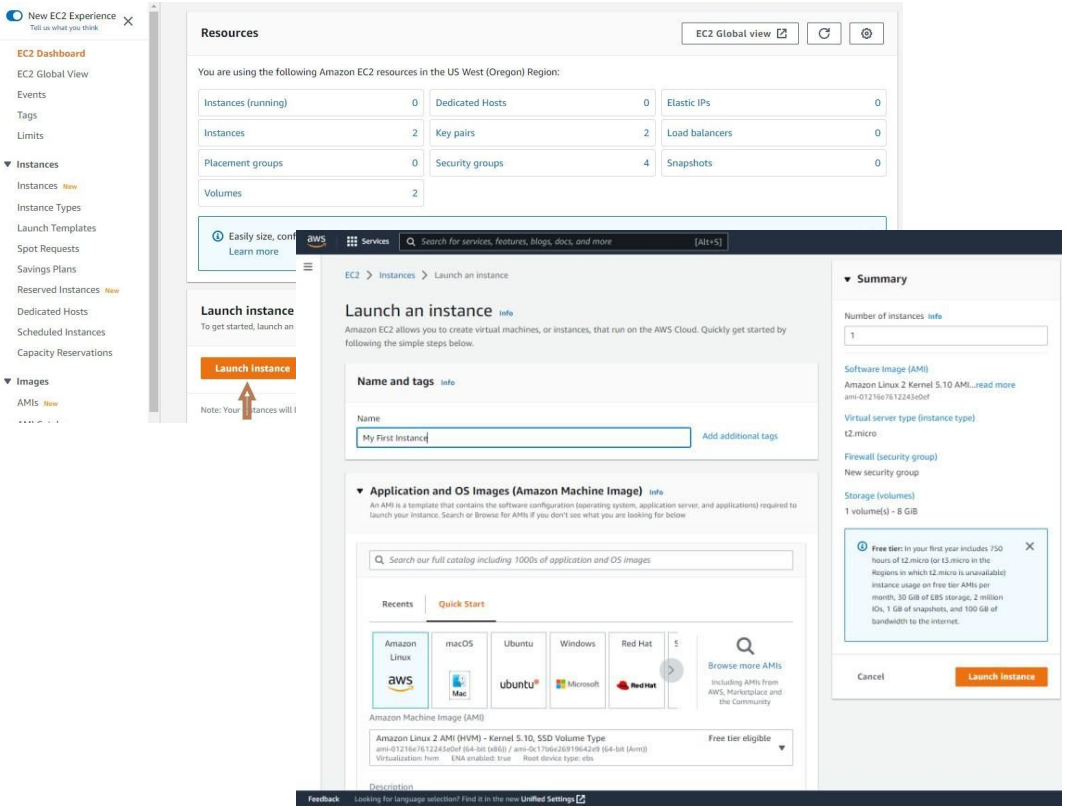
- Adding code using API: calling duckduckgo search and put them into context
- Adding code using LangChain: calling pinecone (a wikipedia dataset) and put the result into context
- Streaming support
  - Utilizing stream_with_context decorator
  - Sending back text line by line
- Async function
  - Not natively supported

# Frontend

- HTML & CSS
  - Bootstrap: For styling and UI components like forms, buttons, etc. The CSS files are loaded from a CDN.
  - Jinja2 template
- Javascript for dynamic effect
  - showdown.js: A Javascript Markdown to HTML converter library. Used to convert the Markdown response from the API to HTML to display.
  - ndjson-readablestream: A library to read NDJSON response streams. Used to read the response from the /chat API endpoint which returns NDJSON.

# Deployment

- AWS
- Create EC2 instance
  - Choose an image
  - Setup the machine
  - Elastic IP



# 产品级App的考虑

- Security
  - https
  - Authentication
- Microservices Architecture
  - Decouple the difference services
  - Docker
- Scalability
  - Flask app scale up
  - Database sharding

# LLM应用优化

- Why
  - Diversity of user’s requests
  - Models in the system can make mistake
- Key is to build the feedback loop
  - Data collection
  - Data labeling
  - Evaluation
  - Algorithm improvement

# Example: LangSmith

pr-ajar-clay-98 - LangSmith

smith.langchain.com/o/5f4d56dc-dfa0-40ad-9ea7-6773562edacd/projects/p/57c2696c-b154-41db-a381-28a4d624eb34?timeModel=%7B"duration"%3A"7d"%7D&runtab=0

Personal > Tracing projects > pr-ajar-clay-98

Add resource tags | DEVELOPER

## pr-ajar-clay-98

ID | Data Retention 14d | Add Rule

Runs | Threads | Monitor | Setup

1 filter | Last 7 days | Root Runs | LLM Calls | All Runs | Columns

| Name | Input | Output | Error | Start Time |
|---|---|---|---|---|
| RunnableSequence | What is attention? | Attention is a fun... | | 1/16/2025, 10:21: |
| RunnableSequence | where did Harrison w... | Harrison worked ... | | 1/16/2025, 10:20 |
| ChatOpenAI | human: Hello, world! | ai: Hello! How ca... | | 1/16/2025, 10:19: |

### Stats

Last 7 days

RUN COUNT
3

TOTAL TOKENS
2,742 / $0.00

MEDIAN TOKENS
176

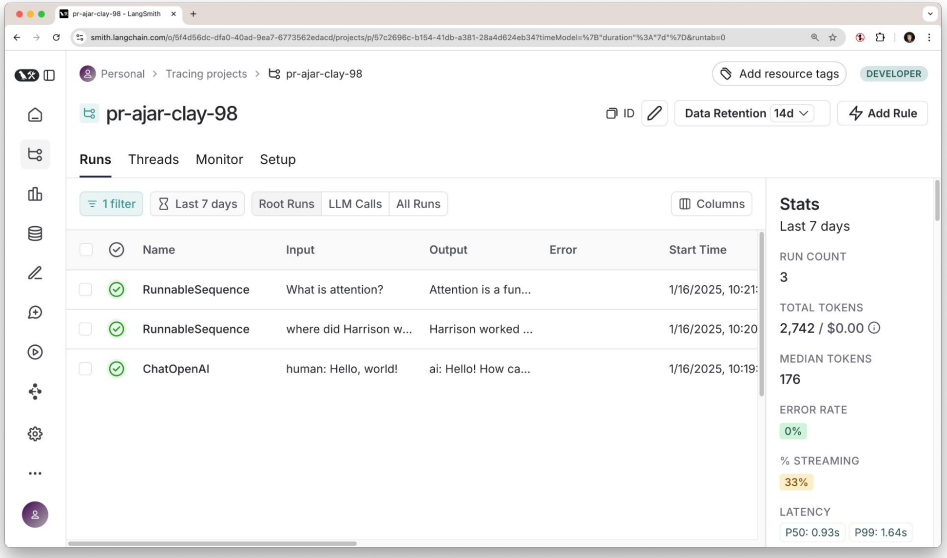
ERROR RATE
0%

% STREAMING
33%

LATENCY
P50: 0.93s | P99: 1.64s

# LangSmith

- A platform tailored for monitoring and evaluating LLM-powered applications, offering features for monitoring intelligent agents and chains in LLM applications



# LangSmith Setup

Python TypeScript

```
export LANGCHAIN_TRACING_V2=true
export LANGCHAIN_API_KEY=<your-api-key>
# The below examples use the OpenAI API, though it's not necessary in general
export OPENAI_API_KEY=<your-openai-api-key>
```

# 优化RAG: 基本优化技巧

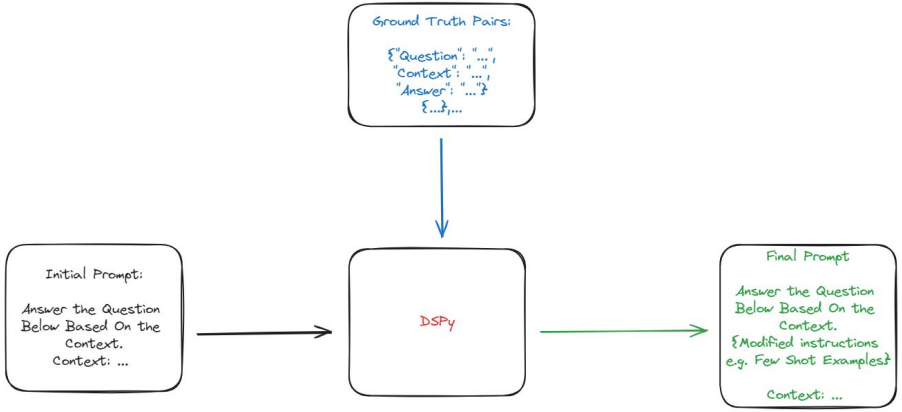
- Prompt Engineering
- LLM model choice
- Embeddings
  - MTEB leaderboard
- Chunk Sizes
- Hybrid Search
  - Embedding search plus keyword search
- Metadata Filters
  - Filtering based on metadata of documents

# Prompt Management Systems

- Using git to manage prompts
- Using specialized tools to manage prompts and experiments
  - Offline tools - good for trying out new ideas
    - Easy to reproduce
    - Running experiments in parallel
    - Visualization of results
  - Online tools - close match to online performance
    - Deal with real data
    - Experiment and deployment
    - A/B testing

# Prompt Optimization Tool

- DSPy
  - A framework for algorithmically optimizing LM prompts and weights
  - Especially when LMs are used one or more times within a pipeline.
- The way it works:
  - First, it separates the flow of your program (modules) from the parameters (LM prompts and weights) of each step
  - Second, DSPy introduces new optimizers, which are LM-driven algorithms that can tune the prompts and/or the weights of your LM calls, given a metric you want to maximize.



# 选择大语言模型

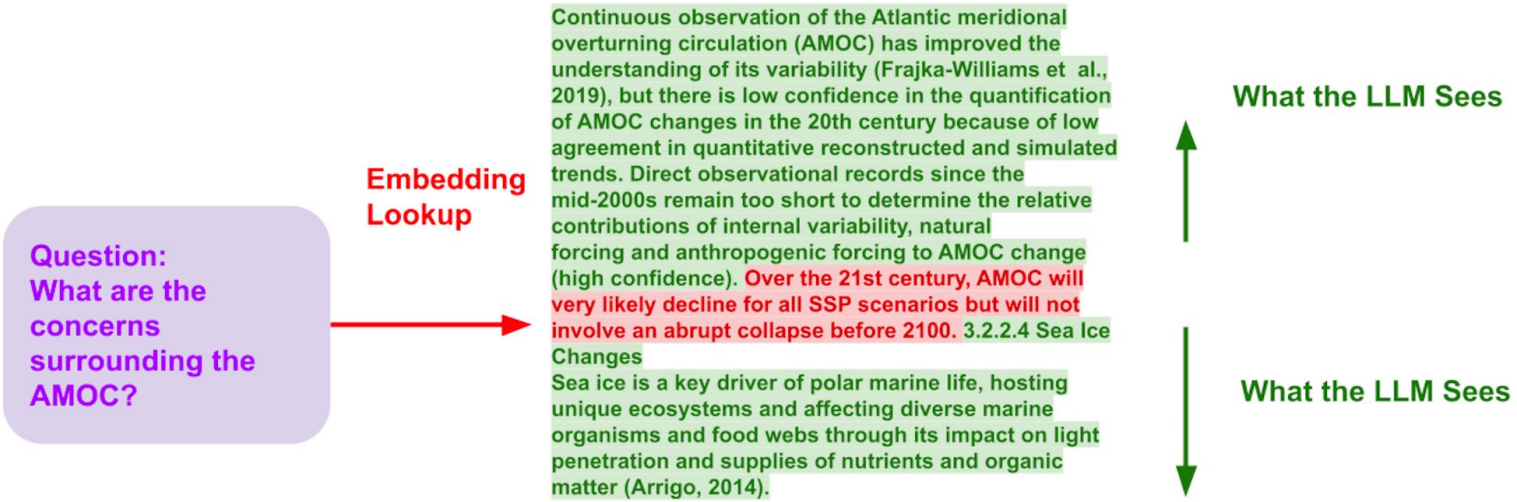
- The best model depends on tradeoffs between:
  - Out of the box quality for your task
  - Team expertise
  - Inference speed /latency
  - Cost
  - Fine-tuneability /extensibility
  - Data security and license permissibility
- Most of the time: start with GPT-4o

# 优化RAG: 高级优化技巧

- Small-to-big retrieval
- Query transformation
- Reranking
- Recursive retrieval
- Embedded tables
- …

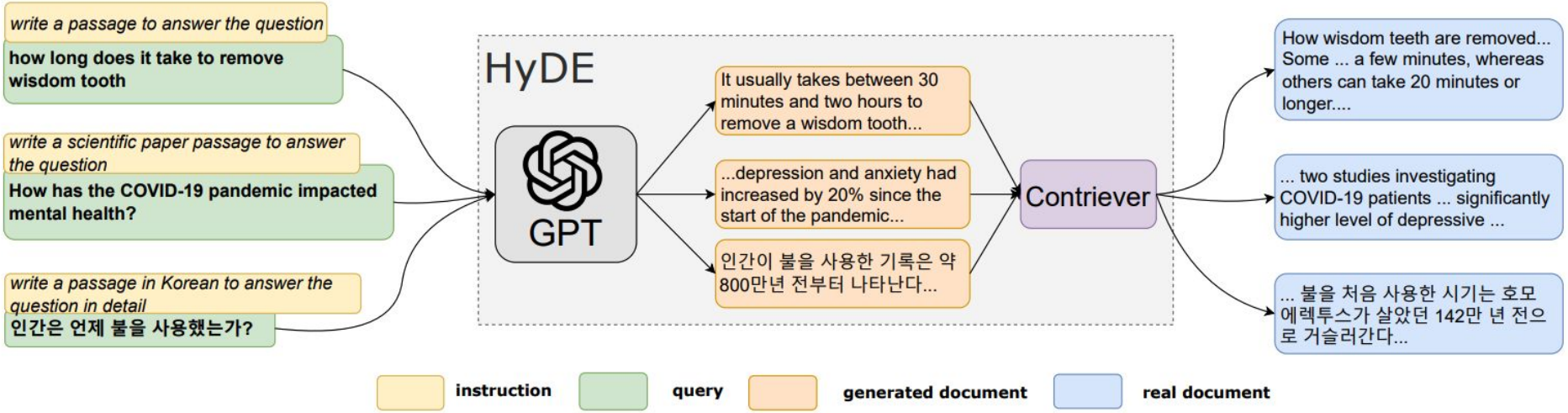
# Small to Big Retrieval

- Using smaller units for embedding but using expanded text for LLM inference
- Refer to test_10_pinecone_with_small_to_big.py

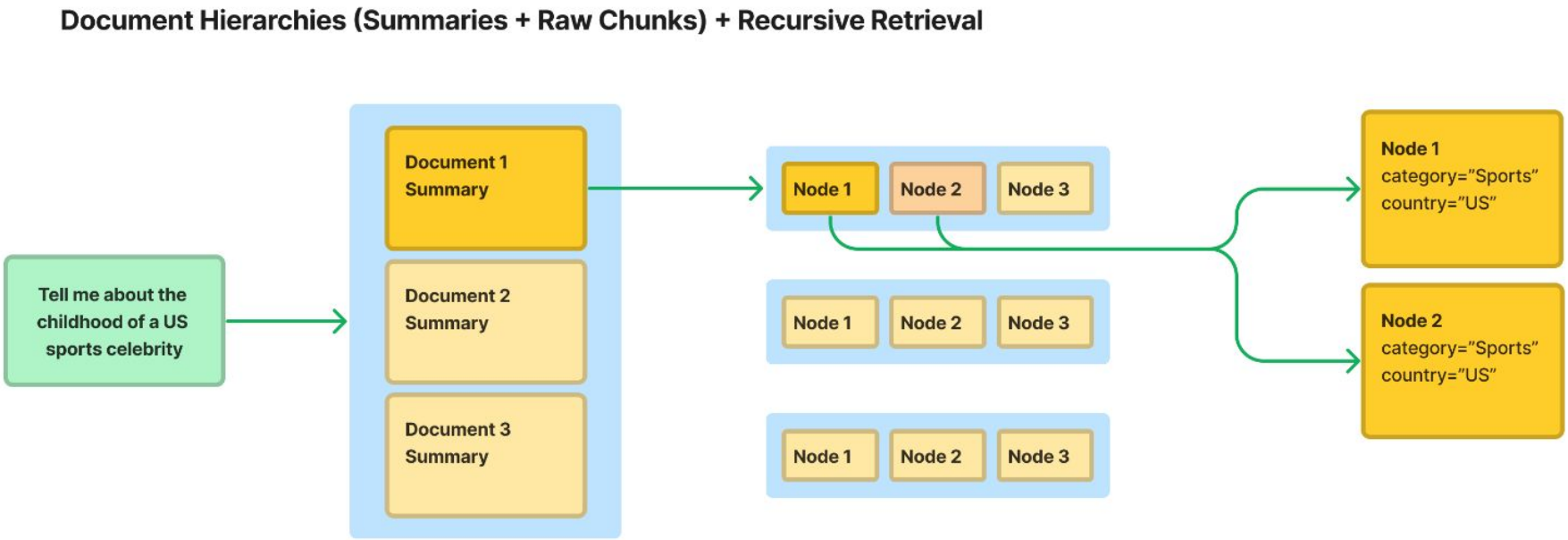


# Query Transformation - HyDE

- HyDE: Hypothetical Document Embeddings
- Refer to test_11_pinecone_with_hyde.py



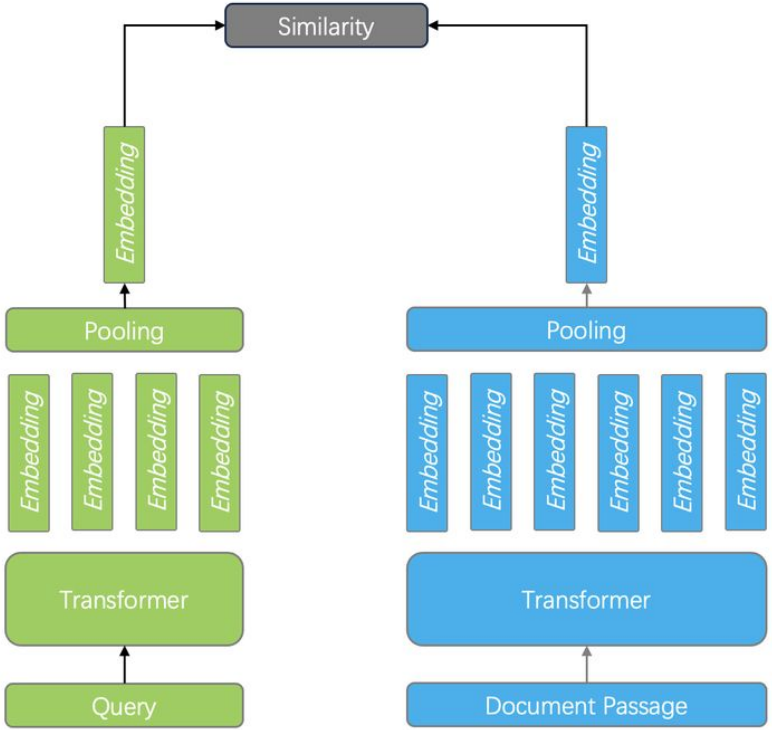
# Recursive Retrieval

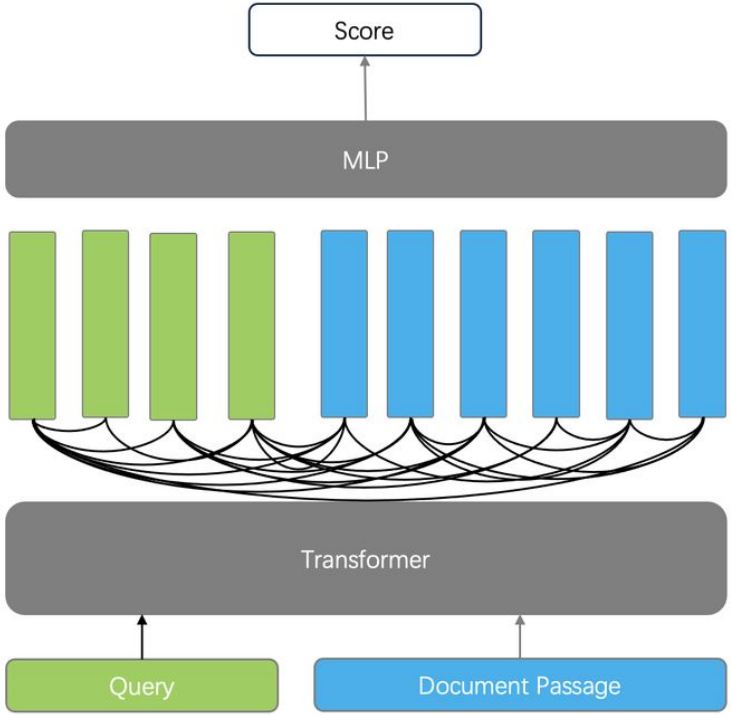


# Reranking

- A step to rerank the top n results returned from the embedding search
- Why reranking
  - Embedding model might be not strong enough
  - To use context information or user-specific information
- Choosing a reranking model
  - LLMRerank
  - Cohere Rerank
  - SentenceTransformerRerank
  - …

# Reranking Model Architecture



# 评测

- Just because your new prompt looks better on a few examples doesn’t mean that it’s better in general
  - Try to collect representative data and queries
  - Start with small amount of examples and add more incrementally
  - Capture “interesting” cases as much as possible
- No quantitative metrics most of the time
  - Convert it to supervised tasks if possible
  - Direct comparison in different dimension: models, prompts, or runs
  - Try some automatic way to evaluate, but not relying on them too much: Rouge scores, LLM scores
- Chain of components make it even more difficult
  - End to end evaluation plus component wise evaluation

# 监控和A/B测试

- User feedbacks
  - Thumbs up/down
  - Is it better
- Performance drops
  - Incorrect answer
  - Hallucination
  - Prompt injection
  - Harmful information
- A/B testing
  - Metrics
  - Experiment groups

# 家庭作业

- Add Flask to the Chatbot you developed
- Successfully run it on your local desktop
- (Optional) Deploy it to AWS and serve it online

# Questions?