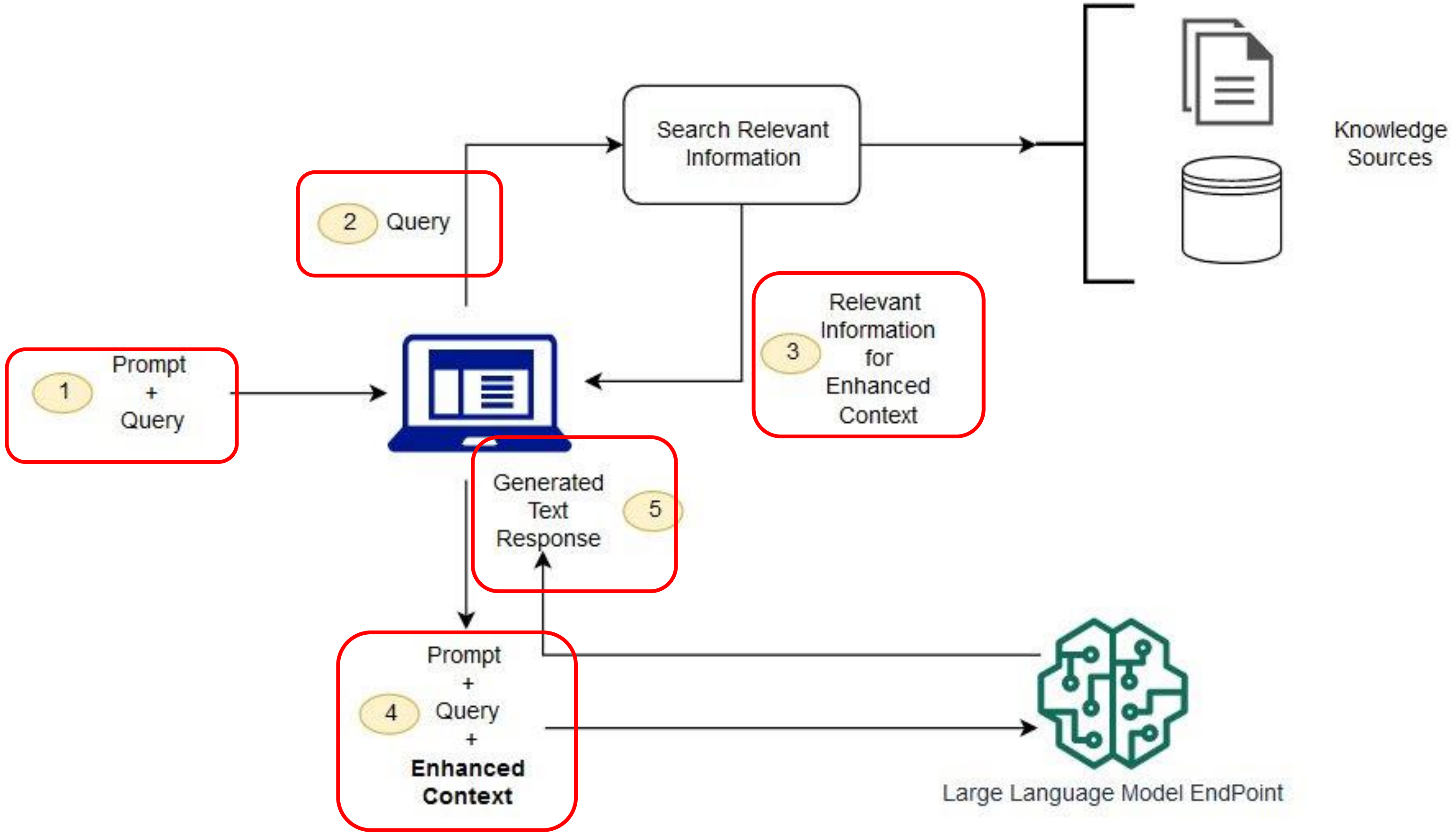
RAG-LLM

Cheng Li
lichengxlxl@gmail.com

Flow of RAG LLM



Source: https://aws.amazon.com/what-is/retrieval-augmented-generation/

My proposal

Core Components of RAG

- **Retriever**: Fetch relevant documents or data.
- **Generator**: An LLM that generates responses using the retrieved context.
- **Indexer**: Indexing mechanism to store and organize data for efficient retrieval.
- **Orchestrator**: Framework or code to combine retrieval, ranking, and generation workflows.
- **Evaluation**: Monitor, evaluate, and debug the system's performance.

Technology stack selection

- **Retriever**
  - I customized a retriever including a suitable search method and reranker.
- **Generator**
  - openAI API, llama, Hugging Face Models
  - I chose to use llama3.2 because it is good for local running. I recommend to use OpenAI if we have enough credit
- **Indexer**:
  - Various vector DB, such as pipecone, faiss, AstraDB, Qdrant etc.
  - I chose to use FAISS which is great for local, and I do not have other accounts
- **Orchestrator**:
  - langchain, langsmith, llama-index
  - I chose to use llama index because it is flexible enough for tailoring RAG pipelines
- **Evaluation**:
  - LangSmith, Mlflow, Prometheus/Grafana
  - I chose to monitor the retrieval evaluation and the response evaluation
- **Deployment**:
  - FastAPI, Azure ML, docker etc.

Decision Factors for technology selection

1. **Scale**: For enterprise-scale systems, prefer managed solutions (e.g., Pinecone, SageMaker). For smaller use cases, open-source tools (e.g., Faiss, Weaviate) are great.
2. **Cost**: Open-source tools (Faiss, Qdrant) reduce cost but increase management effort. API-based tools (OpenAI, Pinecone) offer simplicity at a cost.
3. **Expertise**: Choose frameworks we are comfortable with (e.g., LangChain for workflows, Hugging Face for custom models).
4. **Latency**: If low latency is critical, optimize with local models, efficient retrievers, and lightweight deployment solutions.

An enhanced RAG-LLM

1. Chunking enhancement: develop a customized section-based chunking strategy

2. Metadata enhancement: extract various metadata, such as doc_title, keywords, section_title, summary etc.

3. Retriever enhancement:

- Select a suitable search method: hybrid search with the **interaction** between vector search and keyword search or the **union** of two
- select a suitable reranker model

4. Prompt enhancement: adjust and optimize prompts for a better response

1. Chunking enhancement

- Common chunking strategies:
  - [Character Splitting]() - Simple static character chunks of data
  - [Recursive Character Text Splitting]() - Recursive chunking based on a list of separators
  - [Document Specific Splitting]() - Various chunking methods for different document types (PDF, Python, Markdown)
  - [Semantic Splitting]() - Embedding based chunking

- I developed a customized section chunking node parser because I found section-based chunks are more meaningful for the given policy files.
  - MarkdownNodeParser cannot work for policy_1.txt because there is '**' instead of '##' and the high-level headers are independent nodes from MarkdownNodeParser, which is not meaningful in our case

Code snippet

```
def get_nodes_from_node(self, node: BaseNode) -> List[TextNode]:
    """Get nodes from document by splitting on headers."""
    text = node.get_content(metadata_mode=MetadataMode.NONE)

    section_pattern =  r'(\d+(\.\d+)*\.\s[^\n]+)'
    # Match sections and subsections using regex
    matches = list(re.finditer(section_pattern, text))

    nodes = []
    # If no matches are found, skip this document
    if not matches:
        print("No sections matched. Please check the document format or regex.")

    # Process matches to split the document into chunks
    for i, match in enumerate(matches):
        # Section title
        section_title = match.group(0).strip()
        # Start of the next section or end of the document
        next_start = matches[i + 1].start() if i + 1 < len(matches) else len(text)
        # Extract content between the current and next section
        section_content = text[match.end():next_start].strip()

        # Combine section title and content into a single node text
        node_text = f"{section_title}\n{section_content}"

        # Create a Node with the text and metadata
        nodes.append(self._build_node_from_split(node_text, node,  metadata={"section_title": section_title}))

    return nodes
```

Section chunking vs sematic chunking

```
print(nodes[9].text)
```

```
1. Introduction
This document articulates [Company Name]'s unwavering commitment to the ethical development, deployment, and management of artificial intelligence (AI). As a leader in AI innovation, we recognize our responsibility to ensure that our technologies enhance societal well-being and are utilized in a manner that respects human dignity and rights. This policy provides the ethical guidelines our employees and partners must follow to uphold integrity and promote the beneficial use of AI.

####
```

Section chunking

```
print(nodes[10].text)
```

```
2. Scope
This policy encompasses all AI-related activities at [Company Name], including the design, development, procurement, deployment, maintenance, and decommissioning of AI systems. It applies universally across our global operations, affecting all employees, contractors, consultants, and business partners involved with AI technologies.
```

```
### 1. Introduction
#### 1.1 Purpose
This document delineates the guiding principles and procedures for the governance of artificial intelligence (AI) model development, deployment, and monitoring within our organization. It aims to establish a robust framework that ensures our AI models operate effectively, ethically, and in compliance with all applicable regulations. The policy serves to safeguard our technological innovations, uphold our ethical commitments, and maintain stakeholder trust.

#### 1.2 Scope
The Model Governance Policy applies to all AI models developed, deployed, or managed by our company, across all departments and subsidiaries. It includes models at various stages of the lifecycle, from initial conception and development through to deployment and continuous monitoring.

#### 1.3 Audience
This policy is intended for a broad range of stakeholders including AI developers, project managers, compliance officers, and executive leadership. Each role has a responsibility to understand and implement the guidelines set forth in this policy to ensure effective and ethical model governance.

### 2. Policy Statement
#### 2.1 Commitment to Ethical AI
Our company is committed to the highest standards of ethical conduct in all our AI endeavors. We pledge to develop and deploy AI technologies that are fair, transparent, accountable, and secure, respecting privacy and ensuring non-discrimination.
```

Semantic chunking

2. Metadata enhancement

- Helping **retrievers and rerankers** filter or prioritize documents during a query
- Enhancing Search and Ranking: providing contextually relevant details that embeddings alone might not capture
- Enabling Advanced Use Cases: Personalization and Multi-modal indexing
- Debugging and Observability: Where results came from? Why certain nodes were retrieved or ranked higher.

An example for our metadata

```
1 nodes[9].metadata
```

```
{'file_path': 'C:\\Users\\liche\\Documents\\RAG_assignment\\RAG_assignment\\Policy_exm\\Policy_2.txt',
 'file_name': 'Policy_2.txt',
 'file_type': 'text/plain',
 'file_size': 5319,
 'creation_date': '2024-05-09',
 'last_modified_date': '2024-12-03',
 'doc_title': 'Comprehensive AI Ethics Policy Document',
 'section_title': '1. Introduction',
 'questions_this_excerpt_can_answer': 'Based on the provided context, here are three questions with specific answers that are unlikely to be found elsewhere:\n\n1. What is [Company Name] and what does it represent in terms of AI development and deployment?\nAnswer: [Company Name] is likely a fictional company or organization, but based on the content of the policy document, it can be inferred that it represents a leader in AI innovation with a commitment to ethical development and deployment.\n\n2. What specific guidelines are outlined in this policy for employees and partners regarding AI utilization?\nAnswer: The policy outlines guidelines for ensuring that AI technologies enhance societal well-being and respect human dignity and rights, as stated in the section "1. Introduction".\n\n3. When was the last time the document was modified or updated?\nAnswer: According to the provided context, the document was last modified on 2024-12-03.\n\nAs for higher-level summaries of surrounding context, here are a few possibilities:\n\n* What is the relationship between [Company Name] and AI innovation? Is it a pioneer, a leader, or a follower in this field?\n* In what ways does this policy document reflect broader societal attitudes towards AI development and deployment?\n* Are there any notable features or considerations that distinguish this company\'s approach to AI ethics from others in similar fields?\n\nThese summaries can provide more context for answering these questions.',
 'prev_section_summary': 'The summary of the key topics and entities of the section is:\n\n**Key Topics:**\n\n1. Data privacy and security\n2. User trust and transparency\n3. Policy review and updates\n\n**Entities:**\n\n1. [Company Name] (the company mentioned in the data privacy policy)\n2. [User] (referenced as "our users" in the policy)\n\nNote that the company name is not explicitly mentioned in the excerpt, but can be inferred from the file metadata or context provided.',
 'section_summary': "Here is a summary of the key topics and entities in the section:\n\n**Key Topics:**\n\n1. Introduction to [Company Name]'s AI ethics policy\n2. Commitment to ethical development, deployment, and management of AI\n3. Responsibility to ensure AI enhances societal well-being and respects human dignity and rights\n\n**Entities:**\n\n1. [Company Name]: a fictional company or organization that represents a leader in AI innovation (likely a real company)\n2. Artificial Intelligence (AI): the focus of the policy document",
 'excerpt_keywords': 'artificial intelligence, ethics, leadership, innovation, responsibility'}
```

3. Retriever Enhancement

- Embedding models:
  - impact embedding vector and retrieval nodes
  - I tested two embedding models: 'BAAI/bge-small-en-v1.5' and sentence-transformers/all-MiniLM-L6-v2'. More options are available.

- Similarity search methods. I compared there methods below:
  - Vector search
  - Hybrid search AND: the intersection of vector search and keyword search
  - Hybrid search OR: the union of vector search and keyword search

- Reranking models:
  - Rerank nodes after similarity search
  - I only test two rerank models over the baseline: 'BAAI/bge-reranker-base' and 'BAAI/bge-reranker-larger' and the baseline 'without reranker'

Code snippets

```
# choose a best combination of embedding model, rerank model and search method

from llama_index.core.postprocessor import SentenceTransformerRerank

EMBEDDINGS = {
    "bge-small": HuggingFaceEmbedding(model_name='BAAI/bge-small-en-v1.5'),
    "MiniLM-L6-v2": HuggingFaceEmbedding(model_name='sentence-transformers/all-MiniLM-L6-v2')
}
RERANKERS = {
    "WithoutReranker": None,
    "bge-reranker-base": SentenceTransformerRerank(model="BAAI/bge-reranker-base", top_n=5),
    "bge-reranker-large": SentenceTransformerRerank(model="BAAI/bge-reranker-large", top_n=5)
}
SEARCHERS = {
    "Vector Search": "Vector", # vector search
    "Hybrid Search AND": "HybridAND", # hybrid search using the interaction between vector and keyword
    "Hybrid Search OR": "HybridOR" # hybrid search using the union of vector and keyword
}
```

How to do

1. Generate question and context pairs using a LLM generator
    - generate_question_context_pairs

2. Take generated questions as queries and retrieve relevant nodes

3. Calculate metrics

Retrievers Embedding/Search/Reranker	hit_rate	mrr	precision	recall	ap	ndcg
bge-small/Hybrid Search OR/bge-reranker-large	0.925	0.719	0.185	0.925	0.719	0.261
bge-small/Hybrid Search OR/bge-reranker-base	0.925	0.731	0.185	0.925	0.731	0.264
bge-small/Hybrid Search OR/Without Reranker	0.987	0.247	0.099	0.987	0.247	0.092
bge-small/Hybrid Search AND/bge-reranker-large	0.762	0.627	0.227	0.762	0.627	0.288
bge-small/Hybrid Search AND/bge-reranker-base	0.762	0.619	0.227	0.762	0.619	0.286
bge-small/Hybrid Search AND/WithoutReranker	0.762	0.395	0.227	0.762	0.395	0.221
bge-small/Vector Search/bge-reranker-large	0.875	0.712	0.175	0.875	0.712	0.255
bge-small/Vector Search/bge-reranker-base	0.875	0.708	0.175	0.875	0.708	0.254
bge-small/Vector Search/WithoutReranker	0.875	0.397	0.175	0.875	0.397	0.174
MiniLM-L6-v2/Hybrid Search OR/bge-reranker-large	0.900	0.699	0.180	0.900	0.699	0.254
MiniLM-L6-v2/Hybrid Search OR/bge-reranker-base	0.912	0.721	0.182	0.912	0.721	0.260
MiniLM-L6-v2/Hybrid Search OR/Without Reranker	0.950	0.266	0.096	0.950	0.266	0.094
MiniLM-L6-v2/Hybrid Search AND/bge-reranker-large	0.737	0.639	0.218	0.737	0.639	0.285
MiniLM-L6-v2/Hybrid Search AND/bge-reranker-base	0.737	0.627	0.218	0.737	0.627	0.280
MiniLM-L6-v2/Hybrid Search AND/Without Reranker	0.737	0.386	0.218	0.737	0.386	0.216
MiniLM-L6-v2/Vector Search/bge-reranker-large	0.812	0.695	0.162	0.812	0.695	0.246
MiniLM-L6-v2/Vector Search/bge-reranker-base	0.812	0.685417	0.162	0.812	0.685	0.243
MiniLM-L6-v2/Vector Search/Without Reranker	0.812	0.362292	0.162	0.812	0.362	0.159

Tips to read retrieval metrics

* Explain tips when there are no multiple retrieval gt per query for better understanding

Retrieval precision

- Important if the LLM have short context length
- Useful when you want to reduce hallucination

Retrieval Recall

- Most case it is useful
- Easy to understand
  If recall is 0.5, the retrieval couldn't retrieve gt 50% of questions

Retrieval F1

- Don't use this when you don't want to use retrieval precision
- Harmonic mean of retrieval precision and retrieval recall

NDCG

- Rank-aware metric
- Hard to understand, but effective to classify retrieval module performance

mAP

- Rank-aware metric
- Easy to understand
  If mAP is 0.2, the retrieval system gets the answer in top-5 retrieved passages

mRR

- Rank-aware metric
- Useful when you have multiple passages that have same meaning

Source: https://medium.com/@autorag/tips-to-understand-rag-retrieval-metrics-71e9a2bd4b96

Findings

- The ‘bge-small/Hybrid Search OR/Without Reranker’ and ‘MiniLM-L6-v2/Hybrid Search OR/Without Reranker’ have the highest ‘hit rate’
    - because the methods with Hybrid Search OR/without reranker in our implementation return >=5 nodes while others return 5 nodes
- For the retrievers with the same embedding model and search method, with reranker and without reranker in our case have the same hit rate
    - because both returns top 5 nodes. But the order after applying reranker is improved and thus the precision with reranker is higher than without reranker

Search methods comparison

Criteria	Vector Search	Hybrid Search (AND)	Hybrid Search (OR)
Precision	Moderate	High	Low
Recall	High	Low	High
Query Specificity	Broad queries	Narrow, well-defined queries	Broad or exploratory queries
Relevance to Keywords	Weak	Strong	Moderate
Semantic Context	Strong	Strong + Keyword Filtering	Moderate
Use Case Examples	Conversational search	Legal, technical, or medical	Customer support, general FAQs

My retriever

- The **reranker** appears to improve the performance for all the embedding models, aligning with the findings of most other people.
- The reranker ‘bge-reranker-large’ is better than ‘bge-reranker-base’ and ‘without reranker’ at most cases though it may take longer for reranking
- The embedding model ‘bge-model’ is better than ‘all-MiniLM-L6-v2’ in our case
- I chose the Hybrid search OR because we may focus more on hit rate and the precision of the method can be further improved after using a reranker.
- The retriever I chose may take more time to get the top 5 nodes, but it gave a higher hit rate and precision

4. Prompt enhancing

- To **reduce hallucination,** I ask the RAG-LLM not to add additional information on the context and just say no information available if the relevant information is not found

```
from llama_index.core import PromptTemplate

qa_prompt = PromptTemplate(
    "Context information is below.\n"
    "---------------------\n"
    "{context_str}\n"
    "---------------------\n"
    "Given the context information and not prior knowledge, "
    "answer the query.\n"
    "Do not add any information that is not explicitly present in the retrieved content. "
    "If the information is not available, respond with, \"The provided documents do not contain this information.\"\n"
    "Do not try to make up an answer.\n"
    "Query: {query_str}\n"
    "Answer: "
)
```

Prompt optimization vs Query optimization

- **Prompt optimization**:
  - Optimizing how the model interprets and responds to the prompt.
  - Methods: Framing, structure, examples, and controlling the output style.
  - Adjust temperature in LLM. A low temperature makes the response consistent to the original context

- **Query optimization**:
  - Optimizing how the retrieval system processes the user's query.
  - Methods: Query rephrasing, adding synonyms, removing noise, and refining for better match.

5. Evaluation

- **Retrieval Evaluation**: Are the retrieved sources relevant to the query?
  - Hit rate, MRR, Precision, Recall, AP, NDCG

- **Response Evaluation**: Does the response match the retrieved context? Does it also match the query? Does it match the reference answer or guidelines?
  - Faithfulness
  - Answer relevancy
  - Context relevancy
  - GuidelineAdhere

Retrieval evaluation metrics

Retrievers Embedding/Search/Reranker	hit_rate	mrr	precision	recall	ap	ndcg
bge-small/Hybrid Search OR/bge-reranker-large	0.925	0.719	0.185	0.925	0.719	0.261

I explained the retrieval evaluation explicitly before.

Response evaluation

1. Generate question and context pairs using a LLM generator
    - generate_question_context_pairs()
    - RagDatasetGenerator ()

2. Generate responses for questions above from our LLM (e.g. llama 3.2)

3. Evaluate the responses using another LLM (e.g. gemma2:2b, phi3:mini)

4. Calculate Metrics e.g. Faithfulness and Relevancy

Response evaluation

- Faithfulness measures whether the model's response sticks to the retrieved context without fabricating information
  - faithfulness Score: 0.9382

- Relevance evaluates how well the response addresses the query.
  - relevancy Score: 0.8518

Potential work: Use multiple evaluation LLMs to cross-check results. This reduces dependence on a single model's limitations and provides a more comprehensive assessment.

Results demo of our RAG-LLM

Q: What is the primary principle guiding [Company Name]'s business operations regarding personal data?

A: *At [Company Name], safeguarding the privacy and security of personal data is a foundational principle of our business operations.*

Q: what is AI model?

A: *An AI Model is a software algorithm designed to perform tasks that typically require human intelligence, such as decision-making, prediction, or pattern recognition.*

Q: Does the company allow three years of maternity leave?

A: The provided documents do not contain this information.

Q: "What is AI Ethics Policy? How do I follow the policy?“

A: According to the context information, the AI Ethics Policy is a document that outlines the guiding principles and procedures for the governance of artificial intelligence (AI) model development, deployment, and monitoring within the company.

To follow the policy, you can refer to sections 1.3 Audience, which states that the policy is intended for a broad range of stakeholders including AI developers, project managers, compliance officers, and executive leadership. Each role has a responsibility to understand and implement the guidelines set forth in this policy to ensure effective and ethical model governance.

Additionally, section 5.4 Reporting and Auditing mentions setting up an anonymous ethics violation reporting system, encouraging stakeholders to report any unethical AI practices without fear of retribution, which implies that you can follow the policy by reporting any concerns or issues related to AI ethics.

The answer is not good for the second part because what ‘the policy’ refers to is not clear. The query needs a further clarification or optimization.

Discussions and recommendations

Data and preprocessing

- Some additional data may be useful, such as:
  - Policy Updates and Change Logs, allowing RAG to distinguish between outdated and current policies.
  - Common employee inquiries and their resolutions, Enhancing the model's ability to provide actionable and contextually accurate responses.

- More data preprocessing: Data cleaning, tokenization using SpaCy or NLTK

- Adaptive chunking strategies: we may need mixed chunking strategies if we have additional data or data with different formats

Hyperparameter settings

- Top_k: the number of retrieval nodes before reranking
- Top_n: the number of nodes from a reranker
- Temperature: control the strictness to the original content

Monitoring performance after post deployment

- Key indicators monitoring, such as response latency, user satisfaction from feedbacks.
- Evaluation metrics monitoring , automated evaluation tools such as langsmith, deepEval, openAI evaluation API
- Query and retrieval monitoring. Monitor the quality of retrieved documents by tracking cosine similarity or other distance metrics. Some tools are Pinecone, Weaviate, FAISS etc.
- Real-time logging and error Tracking. Log and analyze errors, including failed queries, retrieval mismatches, and generation errors.
- A/B Testing. Experiment with variations in retrieval techniques or LLM prompt designs or LLM models etc.

Continuous Improvement Techniques

a. Feedback Loop Integration

- Create pipelines to ingest user feedback into model retraining processes (**RLHF**).
- Tools: Label Studio, Amazon SageMaker Ground Truth.

b. Fine-tuning and Updating

- Periodically fine-tune the LLM on updated retrieval results and user queries when they are enough
- Fine-tuning embedding model based on the synthetic question and answer dataset.
- Tools: Hugging Face Transformers, OpenAI Fine-tuning API, llama-index

c. Retraining the Retriever

- Update the retriever with new domain-specific data or embeddings or metafilter.
- **Vector DB Maintenance**:
  - Tools: Weaviate, Pinecone, FAISS.
  - Periodically rebuild or optimize vector indexes to remove stale data.