Week 1
Internal Tools & Financial Cost

The better the question. The better the answer. The better the world works.

Shape the future with confidence

Contents

1. Internal Tools
   1. *Project Architecture*
      1. Front-end
      2. Back-end
   2. *Legal Copilot ecosystem*
      1. Approaches
      2. Upload
      3. App
   3. *API (Azure & LLM)*
2. Financial Costs
   1. *Costs of Azure services*
   2. *Costs of LLM*
   3. *Estimated Monthly Costs*
   4. *Cost Analysis*
3. Future Directions
   1. *FrugalGPT*
   2. *Analysis of GPT Models*

1. Internal tools

1. Project Architecture

Clear separation between the front-end and the back end in the app/ directory.

1. Front-end
    1. Manages user interaction (UI).
    2. Includes files and components for display, navigation and user experience (UX).

2. Back-end
    1. Responsible for business logic and interaction with the language model (LLM).
    2. Structured into three main sub-modules and one app python file :
        1. *Approaches*
        2. *Core*
        3. *Upload*
        4. *App.py*

1. Project Architecture

1. Approaches :

Strategies for interacting with the LLM and Azure AI Search.

Key files :

approach.py / chatapproach.py / chatreadretrieveread.py / retrievethenread.py

2. Core :

Essential components for the back-end.

Key files :

authentication.py / message builder.py / modelhelper.py

3. Upload :

Manages and processes files for indexing

Key files :

blobmanager.py / embeddings.py / fileprocessor.py / parser.py /
jsonparser.py / pdfparser.py / textsplitter.py /searchmanager.py

4. app.py :

Links all the files in Approaches, Core and Upload to interact with the demand of the client with Http requests.

2. Legal Copilot ecosystem

1. Approaches :
    1. approach.py :
        - Interacts with search services and language models
        - Dataclasses : Document and ThoughtStep
        - Utility Methods :
            - *build_filter* : Constructs a search filter.
            - *search* : Performs a search using Azure.
            - *get_sources_content* : Extracts the content of sources.
            - *get_citation* : Generates citations.
            - *compute_text_embedding* : Creates text embeddings for queries.

2. Legal Copilot ecosystem

1. Approaches :
    2. chatapproach.py :
        - Manages chat interactions via OpenAI API
        - Roles : System, User, Assistant.
        - <u>Utility Methods :</u>
            - *get_system_prompt* : Generates the system message.
            - *get_search_query* : Extracts the search query.
            - *extract_followup_questions* : Extracts follow-up questions.
            - *get_messages_from_history* : Constructs messages from the conversation history.
        - Main Methods : run with or without streaming.

2. Legal Copilot ecosystem

1. Approaches :
    3. chatreadretrieveread.py :
        - Multistep approach to answer questions.
        - Process :
            - Converts the question into a search query.
            - Retrieves relevant documents.
            - Generates a contextual response.
        - Main method : run until final call

2. Legal Copilot ecosystem

1. Approaches :
    4. retrievethenread.py :
        - Simpler retrieve then read method
        - Process :
            - Retrieves top documents.
            - Constructs a prompt with the documents.
            - Generates a response.
        - Main method : run

2. Legal Copilot ecosystem

2. Upload :

The upload section includes a folder with necessary Python files and prepdocs.py for document preparation.

Key Points :

- File Strategy :
  - *setup_file_strategy :* Handles PDF, JSON, DOCX, PPTX, XLSX
  - *setup_intevectorizer_strategy :* integrated vectorization strategy.
- File Processing : Extracts content and splits documents. Supports local and Azure Data Lake Gen2.
- Embedding Generation : Uses OpenAI for text and image embeddings.
- Uploading and Indexing : Uploads to Azure Blob Storage and indexes in Azure AI Search.
- Key Management : Uses Azure Key Vault for keys and secret.

2. Legal Copilot ecosystem

3. App.py:

The app.py file acts as an orchestrator for the application.

Key points :

- Orchestration : Connects methods and approachs with clients requests.
- Client Configuration : Sets up clients for interacting with Azurer services.
- HTPP Routes : Defines routes to handle HTTP requests.
- Approach Selection : Determines which approach to used based on request type and parameters.
- Efficient Interaction : Ensures smooth interaction between users and backend services.

3. API

The project leverages several Azure services for robust functionality and seamless integration.

Key Points :

- Azure App Service : Hosts the backend app.
- Azure Application Insights : Monitor performance and usage patterns.
- Azure AI Services :
  - *OpenAI :* Enables advanced NLP and conversational AI.
  - *Document Intelligence :* Extracts information from documents.
  - *Computer Vision :* Analyzes and interprets visual Data.
- Azure Key Vault : Securely Stores secrets and keys.
- Azure Storage : Reliable and scalable data storage.

2. Financial Cost

1. Costs of Azure services

Azure Service	Description	Estimated Cost
Azure AI Search	Document search	3$ / 1000 queries (basic level)
Azure Storage (Blob)	Data storage	0.02$ / GB (standard storage)
Azure Functions	Serverless execution	0.2$ / million executions
Azure Key Vault	Secure management of API keys	0.03$ / 10,000 operations (read/write)
Azure App Service	Web application hosting (API)	0.02$ / hour (basic level)
Azure Monitor	Application monitoring	2.3$ / GB of data collected
Azure Application Insights	Performance monitoring	0.125$ / GB of data processed
Azure AI Services - OpenAI	GPT models and ChatGPT deployments	see the detailed costs of LLM below
Azure AI Services - Document Intelligence	Form recognition	1.5$ / 1000 pages
Azure AI Services - Computer Vision	Computer vision functionalities	1.5$ / 1000 transactions

2. Costs of LLM services

LLM Service	Description	Estimated Cost
GPT-3.5 (curie)	Fast and economical GPT-3.5 model	1.5$ / 1Mio tokens input
GPT-3.5 (curie)	Fast and economical GPT-3.5 model	2$ / 1Mio tokens output
GPT-4	Advanced GPT-4 model	30$ / 1Mio tokens input
GPT-4	Advanced GPT-4 model	60$ / 1Mio tokens output
Embeddings (per text)	Generation of embeddings for search	0.1$ / 1Mio tokens

3. Estimated Monthly Costs

For a hypothetical use of the Azure and LLM services, here is an estimate of the monthly costs :

- Azure AI Search: 10,000 queries per month = 30$
- Azure Storage (Blob): 500 GB of storage per month = 10$
- Azure Functions: 1 million executions per month = 0.2$
- Azure Key Vault: 100,000 operations per month = 0.3$
- Azure App Service: 100 hours of usage per month = 2$
- Azure Monitor: 50 GB of data collected per month = 115$
- Azure Application Insights: 20 GB of data processed per month = 2.5$
- GPT-3.5 (input): 5 million tokens of input per month = 7.5$
- GPT-3.5 (output): 5 million tokens of output per month = 10$
- Document Intelligence: 10,000 pages processed per month = 15$
- Computer Vision: 10,000 transactions per month = 15$

Estimated total monthly cost : 207.5$

4. Cost Analysis

The estimated monthly cost of 207.5$. Key points to consider:

1. GPT-3.5 vs GPT-4:
    1. *GPT-3.5:* $1.5 per 1M input tokens, $2 per 1M output tokens.
    2. *GPT-4:* $30 per 1M input tokens, $60 per 1M output tokens.

    Recommendation: Use GPT-3.5 for most tasks to save costs, reserve GPT-4 for complex needs.

2. Azure Monitor:
    1. *$115 per month for 50 GB of data.*

    Recommendation: Reduce data collection or limit monitoring to essential metrics to lower costs.

3. Azure AI Search:
    1. *$30 for 10,000 queries.*

    Recommendation: Optimize query usage, such as through caching, to reduce costs.

4. Storage and Application Insights:
    1. *Costs for 500 GB of storage and 20 GB of data processed.*

    Recommendation: Manage data retention and archive old data to reduce costs.

3. Future Directions

Future Directions

GOAL : this research is to find techniques to optimize the costs associated with using an information retrieval system with language models (RAG), by examining the impact of cloud services like Azure and the choice of LLM model used. The objective is to reduce expenses while maintaining the performance and quality of the system.

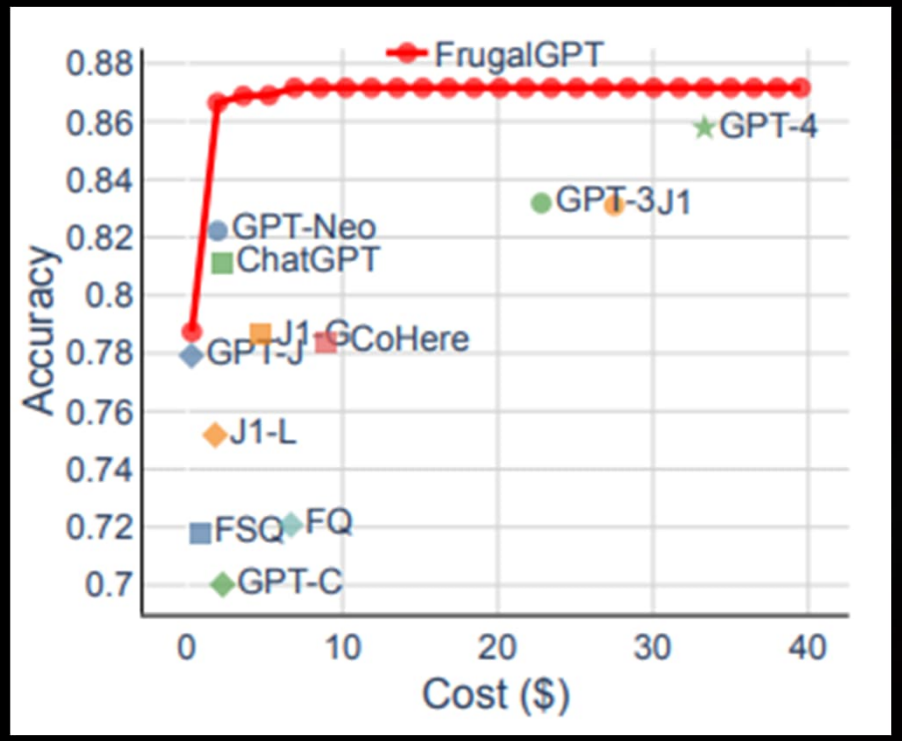
1. FrugalGPT

Key takeaways from the FrugalGPT article:

Key Points:

- Prompt Adaptation: Reduce prompt size to lower costs by:
  - Selecting a subset of relevant examples.
  - Concatenating multiple queries into a single prompt to avoid redundant processing.
- LLM Approximation: Use more affordable models or infrastructures to approximate an expensive LLM by:
  - Using a completion cache to store and reuse responses for similar queries.
  - Fine-tuning a smaller, less expensive model with responses from a powerful LLM.
- LLM Cascade: Adaptively select LLMs based on queries to optimize costs and performance by:
  - Sending a query to a list of LLMs sequentially.
  - Using the response from a less expensive LLM if reliable and querying more expensive LLMs only if necessary.

2. Analysis of GPT Models



- **Low-cost FrugalGPT:** Using FrugalGPT with a limited budget involves using less powerful models, which keeps the accuracy average.
- **Moderate budget increase:** By slightly increasing the budget for FrugalGPT, we achieve higher accuracy than GPT-4, while still being much cheaper than GPT-4.
- **High budget increase:** However, further increasing the budget for FrugalGPT does not significantly improve accuracy, which converges towards 0.87.

Next Week

Goal : Identify alternatives to traditional models to minimize RAG(Retrieval-Augmented Generation) costs.

Context : Reduce the high costs associated with current models.

Approach : Explore new cost-effective solutions.

Expected Outcome : List of viable alternatives with a theoretical evaluation of cost impacts.

Week 2 Possible Approaches & Theoretical Cost Assessment

The better the question. The better the answer. The better the world works.

Shape the future with confidence

Contents

1. Possible Approaches & Theoretical Cost Assessment
   1. *Rerank Cohere*
      1. Description of the Method
      2. Theoretical Evaluation of Cost
   2. *FrugalGPT*
      1. Description of the Method
      2. Theoretical Evaluation of Cost
      3. Personnal Thought
   3. *Cache-Augmented Generation*
      1. Description of the Method
      2. Theoretical Evaluation of Cost

2. Other Approaches to Consider
   1. *Knowledge Augmented Generation*
   2. *Document pruning*

1. Possible Approaches & Theoretical Cost Assessment

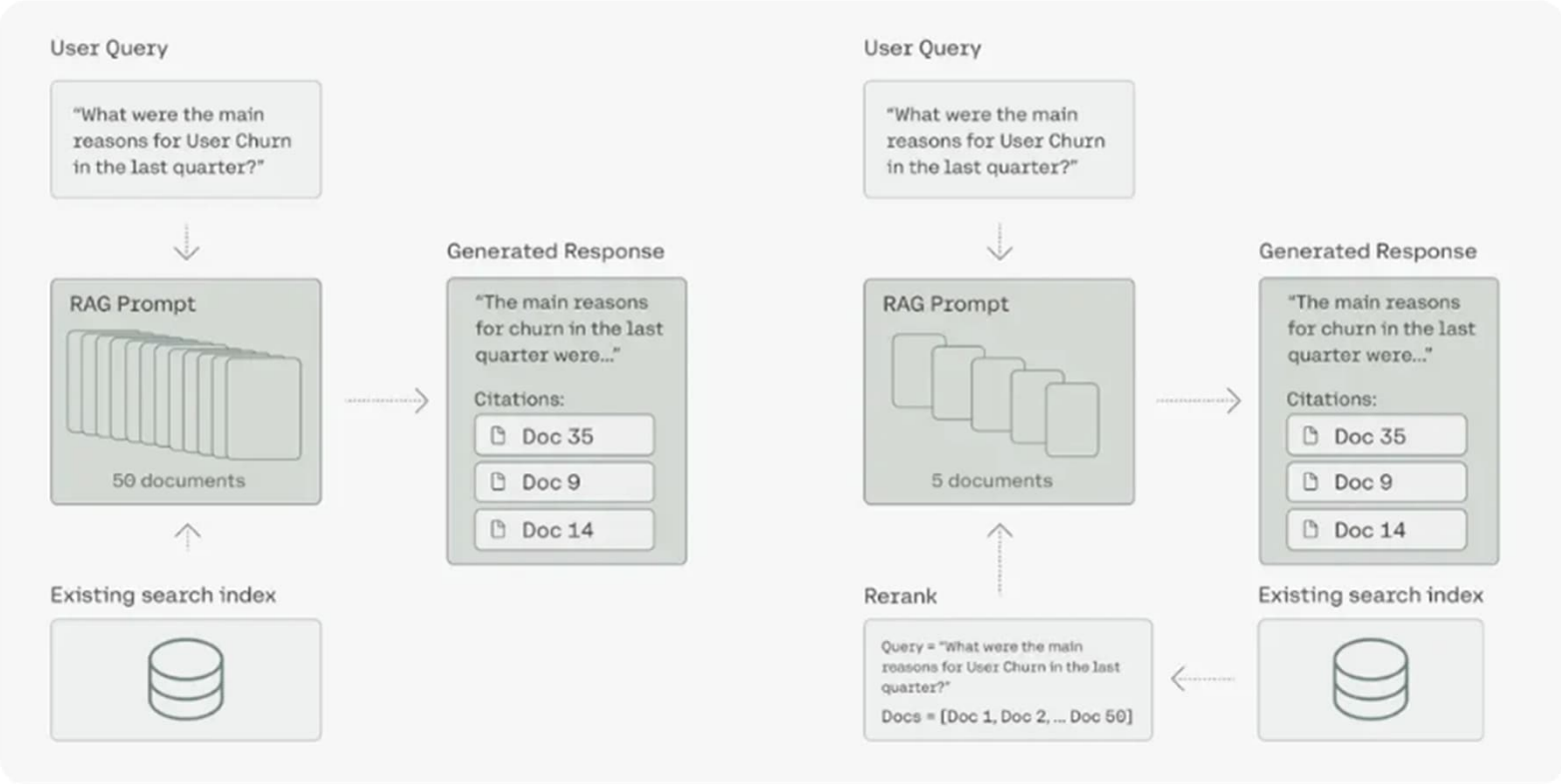
1. Rerank Cohere

Description of the Method :

- Rerank Cohere is a powerful tool designed to enhance the quality of search results by reordering documents based on their semantic relevance to a query. This is particularly useful in Retrieval-Augmented Generation (RAG) systems.

- How it works:
  - Initial Retrieval: Documents are first retrieved using standard search methods (e.g., BM25, BERT) and ranked based on their similarity to the query.
  - Reranking by Cohere: The retrieved documents are then re-evaluated by Cohere's Rerank model, which uses advanced transformer-based architectures to assign a relevance score.
  - Output: A list of documents ranked from most to least relevant is returned, improving the overall quality of search results.

1. Rerank Cohere



(left) Retrieval augmented generation workflow without Rerank. Documents are retrieved from an existing search system and are passed directly to the LLM for grounded generation. (right) Retrieval augmented generation workflow with Rerank. Documents are retrieved from an existing search system and are passed directly to Rerank. High precision semantic reranking allows fewer, higher quality documents to be passed to the LLM for grounded generation.

1. Rerank Cohere

Theoretical Evaluation of Cost :

- Initial Cost: Additional cost for using Rerank Cohere
- Long-term Savings:
  - Improves search result quality (minimizes irrelevant results).
  - Reduces need for multiple additional queries.
  - Optimizes computational resource usage.
  - Decreases need for further document retrieval or searches.

1. Rerank Cohere

Benchmarks

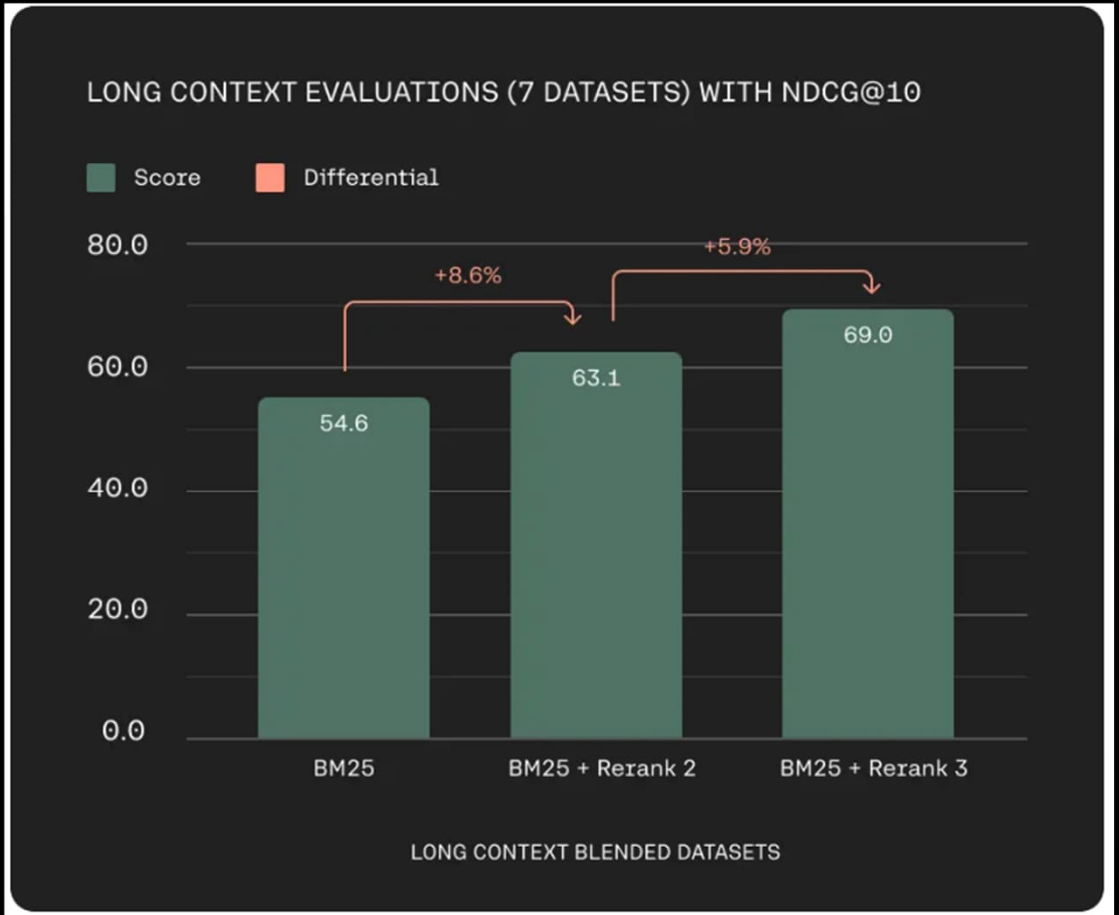
Based on the Cohere Blog : https://cohere.com/blog/rerank-3

Processing Capacity : up to 4,000 tokens

Increased accuracy on benchmarks like TREC 2020

Better semantic ranking of documents compared to traditional methods

1. Rerank Cohere



Long context accuracy based on nDCG@10 on TREC 2019-2022, Conditional QA , NQ-Hard, Qasper, Genomics, QMSum, FinanceBench (higher is better).

- A major Challenge in building semantic search and RAG systems is determining how to best chunk data (split into smaller parts).
- The rerank 3 model now has a 4k context length (enabling the processing of larger documents for ranking).
- This allows the model to consider more context form the document when calculating relevance score.
- It also reduces the need for chunking documents to fit within the model's context window.

1. Rerank Cohere

Cost Analysis

Cohere Reranker 3 :

$2 per 1,000 searches.

GPT-4 Advanced:

$30 per 1 million input tokens.

$60 per 1 million output tokens.

Accuracy Increase: 10% (10% reduction in queries to ChatGPT).

Scenario:

Number of searches: 1,000

Input tokens per search: 1,000

Output tokens per search: 1,000

Calculations:

Cost with Reranker:

Reranker cost: (1,000 / 1,000) * $2 = *$2*

Reduced number of searches: 1,000 * (1 - 0.1) = *900*

Input tokens cost: (900 * 1,000 / 1,000,000) * $30 = *$27*

Output tokens cost: (900 * 1,000 / 1,000,000) * $60 = *$54*

Total cost with Reranker: $2 + $27 + $54 = *$83*

Cost without Reranker:

Input tokens cost: (1,000 * 1,000 / 1,000,000) * $30 = *$30*

Output tokens cost: (1,000 * 1,000 / 1,000,000) * $60 = *$60*

Total cost without Reranker: $30 + $60 = *$90*

Conclusion:

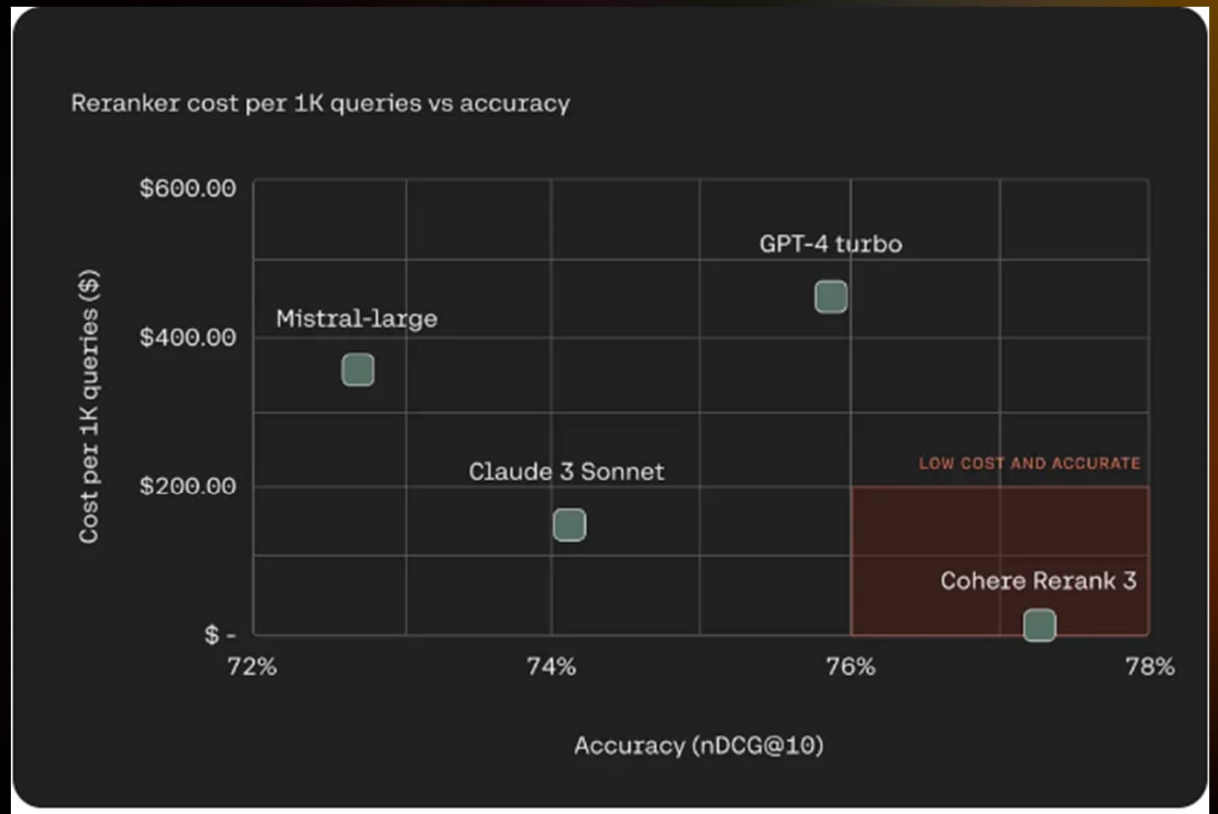
Cost with Reranker: $83

Cost without Reranker: $90

1. Rerank Cohere

Compared to (GTP4, Mistral & Claude)

- Common approach for RAG systems : Using LLM as “rerankers” for document retrieval.
- Rerank 3 outperforms industry-leading LLMs in ranking accuracy.
- Rerank 3 is 90% to 98% cheaper than traditional LLMs



Accuracy based on nDCG@10 using a subset of randomly sampled queries from the test split of the Scifact dataset (higher is better). LLMs are evaluated in a list-wise fashion following the approach used in RankGPT (Sun et al. 2023). Both Rerank and LLMs were given the top 100 documents initially retrieved from BM25.

1. Rerank Cohere

Benchmarks

Based on the LanceDB : https://blog.lancedb.com/benchmarking-cohere-reranker-with-lancedb/

- Reranking: The process of reorganizing search results based on additional criteria to enhance relevance.
- Integrated support: LanceDB supports reranking for all types of searches - vector, full-text, and hybrid.
- Benchmark results: Tests show that the Cohere Reranker significantly improves search results compared to vector search alone.

This aligns with what Cohere mentions in their blog

2. *FrugalGPT*

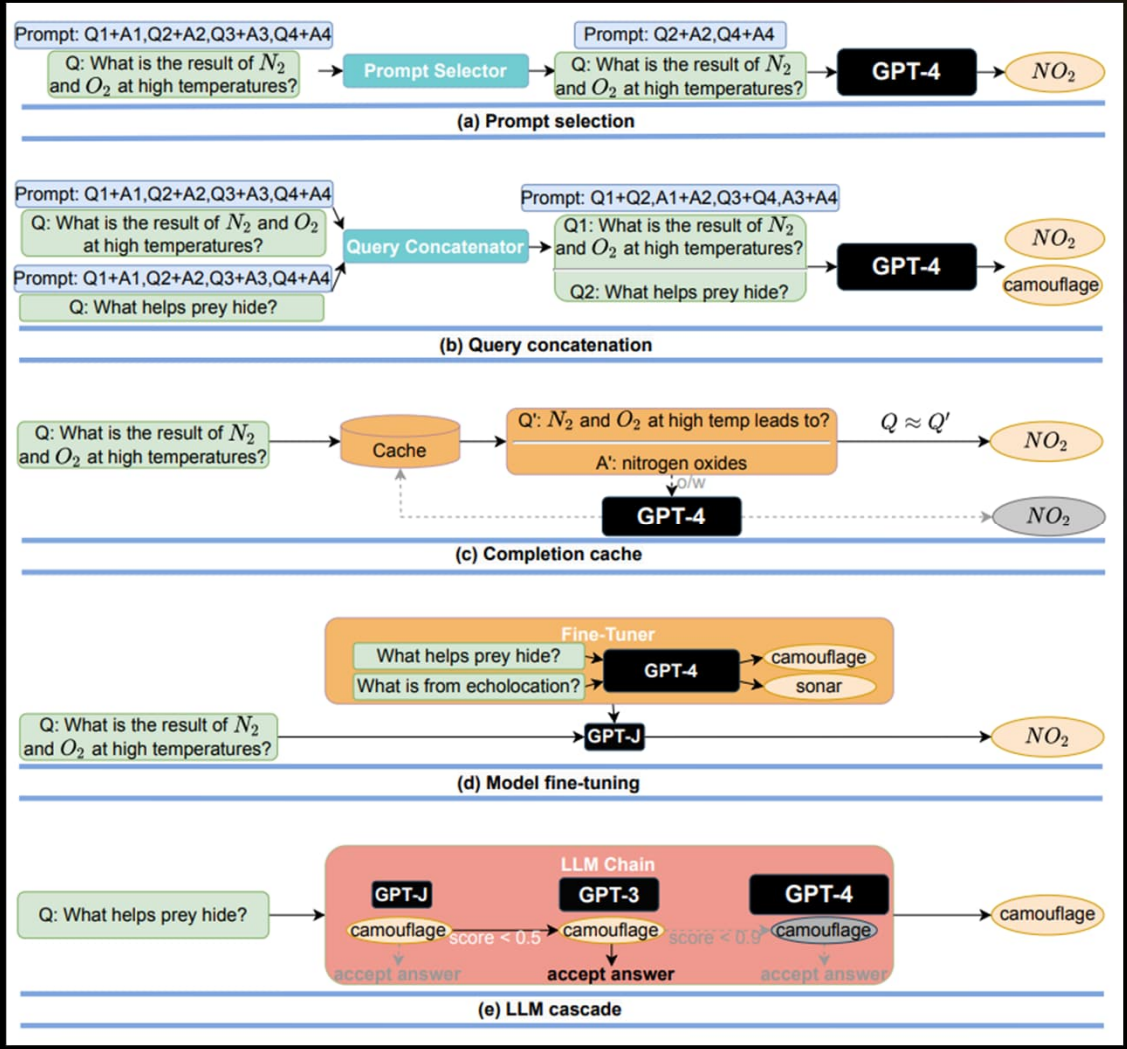
Description of the Method :

FrugalGPT Reduces costs (prompt cost, generation cost, fixed cost per query) of using large language models (LLMs) while maintaining high performance.

Key Takeaways to reduce costs :

- Prompt Adaptation:
  - Reduces prompt size (prompt selection)
  - Selects relevant examples or concatenates multiple queries to avoid redundant processing (query concatenation).
- LLM Approximation:
  - Completion cache to store and reuse responses (completion cache)
  - Fine-tuning smaller models with responses from powerful LLMs (model fine-tuning).
- LLM Cascade:
  - Adaptively selects LLMs based on queries to optimize costs and performance.
  - Queries less expensive LLMs first, escalating to more expensive ones only if necessary.

2. FrugalGPT


Prompt: Q1+A1,Q2+A2,Q3+A3,Q4+A4
Q: What is the result of N_2 and O_2 at high temperatures?
Prompt Selector
Prompt: Q2+A2,Q4+A4
Q: What is the result of N_2 and O_2 at high temperatures?
GPT-4
NO_2
(a) Prompt selection
Prompt: Q1+A1,Q2+A2,Q3+A3,Q4+A4
Q: What is the result of N_2 and O_2 at high temperatures?
Prompt: Q1+A1,Q2+A2,Q3+A3,Q4+A4
Q: What helps prey hide?
Query Concatenator
Prompt: Q1+Q2,A1+A2,Q3+Q4,A3+A4
Q1: What is the result of N_2 and O_2 at high temperatures?
Q2: What helps prey hide?
GPT-4
NO_2
camouflage
(b) Query concatenation
Q: What is the result of N_2 and O_2 at high temperatures?
Cache
Q': N_2 and O_2 at high temp leads to?
A': nitrogen oxides
$Q \approx Q'$
NO_2
o/w
GPT-4
NO_2
(c) Completion cache
Fine-Tuner
What helps prey hide?
What is from echolocation?
GPT-4
camouflage
sonar
Q: What is the result of N_2 and O_2 at high temperatures?
GPT-J
NO_2
(d) Model fine-tuning
LLM Chain
Q: What helps prey hide?
GPT-J
camouflage
score < 0.5
accept answer
GPT-3
camouflage
accept answer
GPT-4
camouflage
accept answer
camouflage
(e) LLM cascade

2. *FrugalGPT*

How is the LLM cascade working ? :

LLM Cascade leverages adaptive selection of language models based on cost and performance. It consists of two major components:

Generation Scoring Function :

- Function g(q,a) : Q x A => [0,1] evaluates the reliability of a response generated by an LLM
- It takes a query q and an answer a as an input and assigns a score indicating the response's quality or relevance.
- If the score exceeds a predefined threshold, the response is accepted and return. Otherwise, the next LLM in the sequence is queried.

LLM Router :

- The router selects a subset m (m = 3) of LLMs from K available models based on factors such as cost and performance on similar queries.
- When a query is submitted, the router sequentially sends it to the selected LLMs
- At each step, the scoring function evaluates the LLM's response. If the response is deemed satisfactory, it is returned immediately, skipping the remaining models to reduce costs.

2. *FrugalGPT*

How to select what model? (without knowing the ground truth) :

Use a Cascade of LLMs (LLMChain)

- Models are ranked in a **sequential order** from cheapest to most expensive.
- The system starts with a lower-cost model and escalates only if necessary.

Score Predictions Using a Pre-trained Scorer

- Each model's response is evaluated using a **scorer model** trained on past data.
- The scorer predicts a **quality score** for the response.
- If the score is above a threshold, the response is accepted. Otherwise, the next model is queried.

Optimize Decision Thresholds (Optimizer)

- Thresholds are determined **dynamically** based on cost-quality trade-offs.
- The optimizer adjusts these thresholds to minimize cost while ensuring that the first selected model meets the required accuracy.

Adaptive Model Selection Based on Budget and Constraints

- The **LLMChain strategy** ensures the system respects a predefined budget.
- If cost constraints are reached, the best-available model is used.

2. *FrugalGPT*

Theoretical Evaluation of Cost :

- Cost Reduction Strategies:
  - Prompt Adaptation: Reduces data processed, lowering computational costs.
  - LLM Approximation: Uses cheaper models to approximate expensive ones, reducing overall expenses.
  - LLM Cascade: Prioritizes cost-effective models, escalating to more expensive ones only when necessary.
- Cost Reduction linked to the strategies :
  - API Cost Reduction: FrugalGPT minimizes expenses by using cheaper models first and calling expensive ones only when necessary.
  - Budget Control: Optimizes responses while adhering to a fixed budget constraint.
  - Resilience to Model Downtime: Automatically switches to an alternative model if one becomes unavailable.

3. Cache Augmented Generation

Description of the Method

- Cache-Augmented Generation (CAG): Addresses limitations of Retrieval-Augmented Generation (RAG) by enabling faster access to data.
- External Knowledge Preloading:
  - Curated documents are preprocessed and formatted to fit the model's context window.
  - LLM processes these documents into a precomputed KV cache.
- Inference:
  - Precomputed KV cache is loaded with the user's query.
  - LLM generates responses using the cached context without additional retrieval steps.
- Cache Reset:
  - KV cache can be reset efficiently by truncating new tokens.
  - Allows rapid reinitialization without reloading the entire cache.

3. Cache Augmented Generation

Cache solution :

Cache-Augmented Generation (CAG) enhances language model efficiency by reusing previously computed representations instead of recalculating them for each new generation. This approach relies on storing and retrieving intermediate representations, specifically Key-Value (KV) caches from the attention layers of Transformer models.

Key-Value Cache (Intermediate Representation Storage)

Stores attention keys and values generated during token processing.

Avoids redundant computations by retrieving precomputed representations.

Reduces inference latency by leveraging stored attention states.

Persistent Storage (File-Based KV Cache)

- Allows storing intermediate representations on disk for future reuse.
- Reduces model warm-up time by reloading precomputed caches.
- Optimizes resource management by limiting reliance on in-memory storage.

Dynamic In-Memory Cache

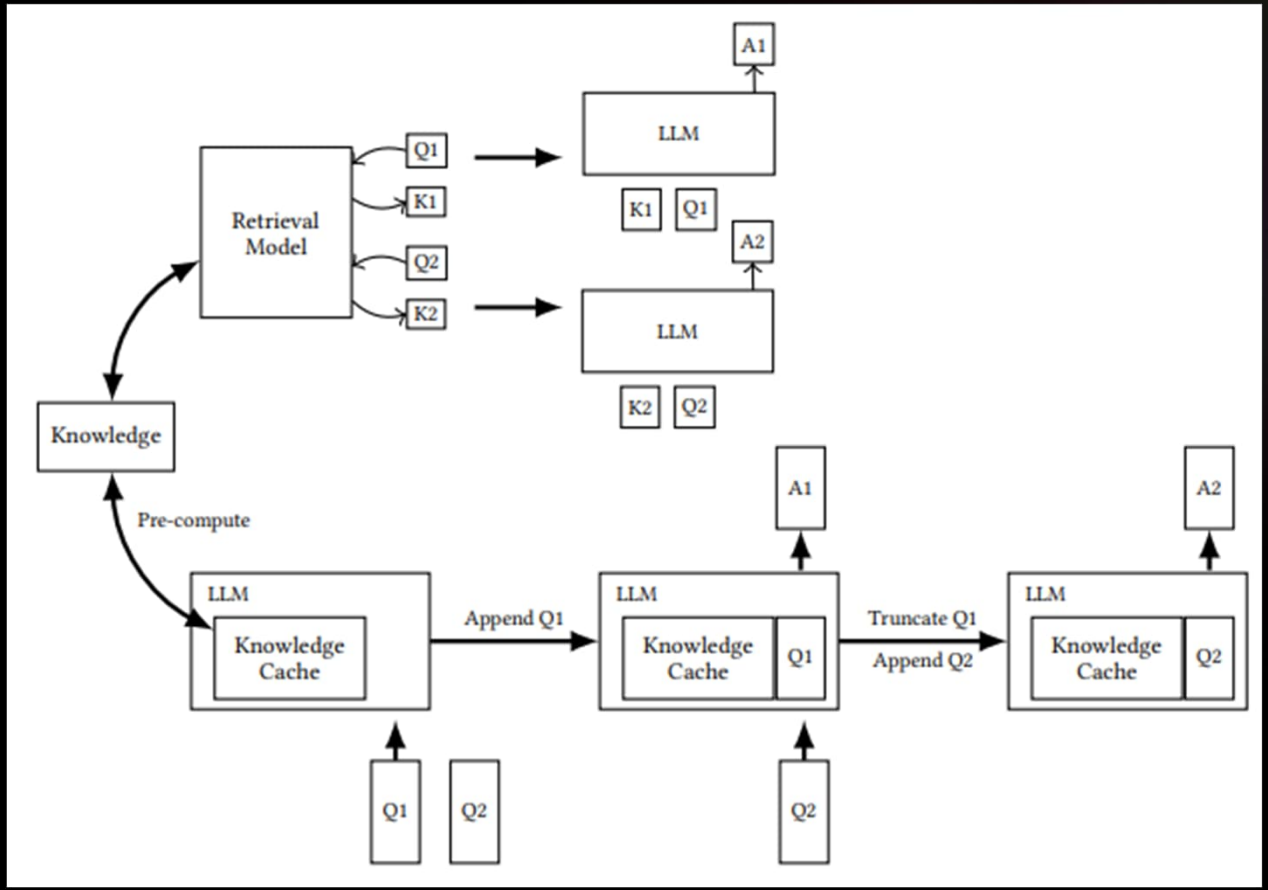
- Maintains representations in memory for fast retrieval.
- Useful for multi-turn conversations by preserving context across interactions.
- Optimizes GPU memory usage by truncating unnecessary cache entries.

3. Cache Augmented Generation

Comparison with Other Caching Methods

- Query-based Cache → Caches full query-response pairs for direct reuse.
- Chunk-based Cache → Stores document fragments for retrieval-augmented generation.
- KV Cache (Used in CAG) → Stores lower-level representations, allowing dynamic and adaptive reuse.

3. Cache Augmented Generation



3. Cache Augmented Generation

Theoretical Evaluation of Cost :

- Lower Latency: Eliminates real-time retrieval delays
- Fewer Retrieval Errors: Reduces issues related to incorrect document selection
- Simplified Architecture: No need for external retrieval mechanisms
- Optimized for Large Context Windows: Leverages LLMs with extended memory capacities

2. Other Approaches to Consider

1. Knowledge Augmented Generation

Integrates structured knowledge graphs or external knowledge bases.

- Key Features:
  - Preloads Knowledge: Creates a precomputed knowledge cache.
  - Mutual Indexing: Efficient navigation and retrieval.
  - Structured Reasoning: Uses logical reasoning with knowledge graphs.
  - Response Generation: Improves accuracy and coherence.
  - Cache Reset: Rapid reinitialization without full reload.

2. Document pruning

Reduces size and complexity of document sets by removing non-essential or redundant parts.

How Document Pruning Works :

- Relevance Analysis: Analyze documents to identify relevant parts.
- Removal of Non-relevant Parts: Remove sections that don't contribute to the task.
- Continuous Optimization: Iterative process with adjustments based on performance.

Benefits of Document Pruning :

- Reduced Document Size: Smaller, easier to manage documents.
- Improved Processing Speed: Faster response times with less data.
- Lower Storage and Computation Costs: Reduced operational costs due to fewer data.

Week 3
Define the chosen technical approach & initial prototyping

Contents

1. Cost of building blocks
   1. *Azure App Service*
   2. *Azure AI Search*
   3. *Azure OpenAI Service*
   4. *Azure Blob Storage*

   Bonus : Approximation cost of January to February 2024

2. Cost comparison
   1. *Rerank (Cohere)*
   2. *FrugalGPT*
   3. *Cache-Augmented Generation*
3. KPI
4. Future Directions

1. Cost of building blocks

Retrieval Augmented Generation

- **Storage (Blob Storage)** → Stores source data.
- **Retrieval (AI Search)** → Fetches relevant documents.
- **Orchestration (App Service)** → Manages application flow.
- **Generation (OpenAI GPT-4)** → Produces the enriched response.

1. Azure App Service

Azure App Service is used to host applications

Cost Factors :

- Plan selection : Costs vary based on plan (Free, Shared, Basic, Standard, Premium)
- Number of instances : More instances for scalability increase the cost
- Instance Size (CPU/RAM) : higher resources lead to higher pricing.
- Enabled Features : Auto-scaling, deployment slots, and caching can raise costs.

Cost Estimate :

- Basic B1 Plan : 1.75 GB RAM, 10GB storage -> $ 54.75/month (Pay-As-You-Go).

2. Azure AI Search

Azure AI Search is used for indexing and searching documents in RAG applications.

Cost Factors:

- Service Tier: Different tiers (Basic, Standard, High Scale) have varying costs. Example: Basic plan costs ~$73.73/month with limited features.
- Index Size & Data Volume: Larger indexes lead to higher costs.
- Search Queries: Monthly query volume impacts pricing.
- Replicas & Partitions: More replicas/partitions improve scalability but increase costs.
- Additional Features: Services like Custom Entity Lookup Skill, Document Cracking, Image Extraction, and Semantic Ranker may incur extra charges based on data volume or operations.

Cost Estimate:

- Basic Plan: 15 GB index storage, ~15 indexes, 1 replica → ~$80–100/month.
- Additional Costs: Using advanced AI features can further increase expenses.

3. Azure OpenAI Service

Azure OpenAI Service provides access to models like GPT-4 and GPT-3.5 for text generation.

Cost Factors:

- Model Used: GPT-4 is more expensive than GPT-3.5. GPT-4 Turbo offers better performance at a lower cost.
- Token Consumption: Costs depend on the number of tokens processed (input + output). Example: GPT-4 Turbo costs ~$93.
- Embeddings Usage: OpenAI embeddings for semantic search or document analysis have separate pricing ($0.13 per million input tokens).
- Response Length: Longer responses or complex queries consume more tokens, increasing costs.
- Request Frequency & Active Users: More queries and higher usage lead to increased expenses.

Cost Estimate:

- GPT-4 Turbo: 1M tokens input + 1M tokens output processed → ~$90/month.
- Embeddings : +$0.13 per million input tokens.
- Total costs depend on token usage and additional AI features.

4. Azure Blob Storage

Azure Blob Storage is used to store documents and generated data for the RAG model.

Cost Factors:

- Total Data Stored: More storage leads to higher costs.
- Storage Tier: Options include Hot, Cool, and Archive. Hot storage is the most expensive but allows faster access.
- Read/Write Operations: $0.065 per 1,000 writes and $0.005 per 1,000 reads. High operation volume increases costs.
- Region & Data Redundancy: Storage replication (LRS, ZRS, GRS, RA-GRS) affects pricing.

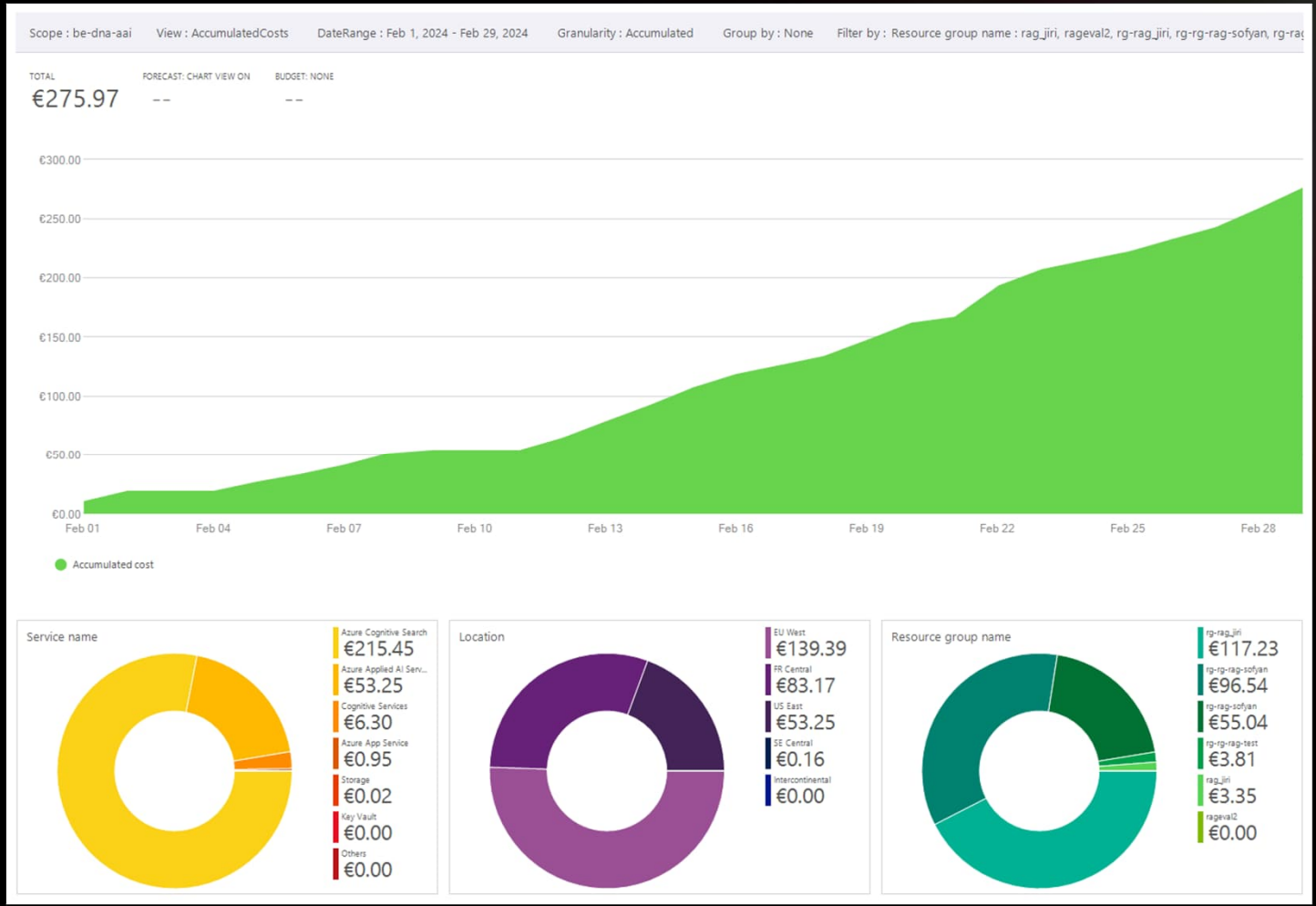
Cost Estimate (Hot Storage, 1TB, 100K read/write ops):

- Write Operations: $6.50 (100K writes × $0.065/1K).
- Read Operations: $0.50 (100K reads × $0.005/1K).
- Storage Cost: $18 (1TB × $0.018/GB).
- Total Estimated Cost: ~$25/month.

Overall Monthly Cost Estimation

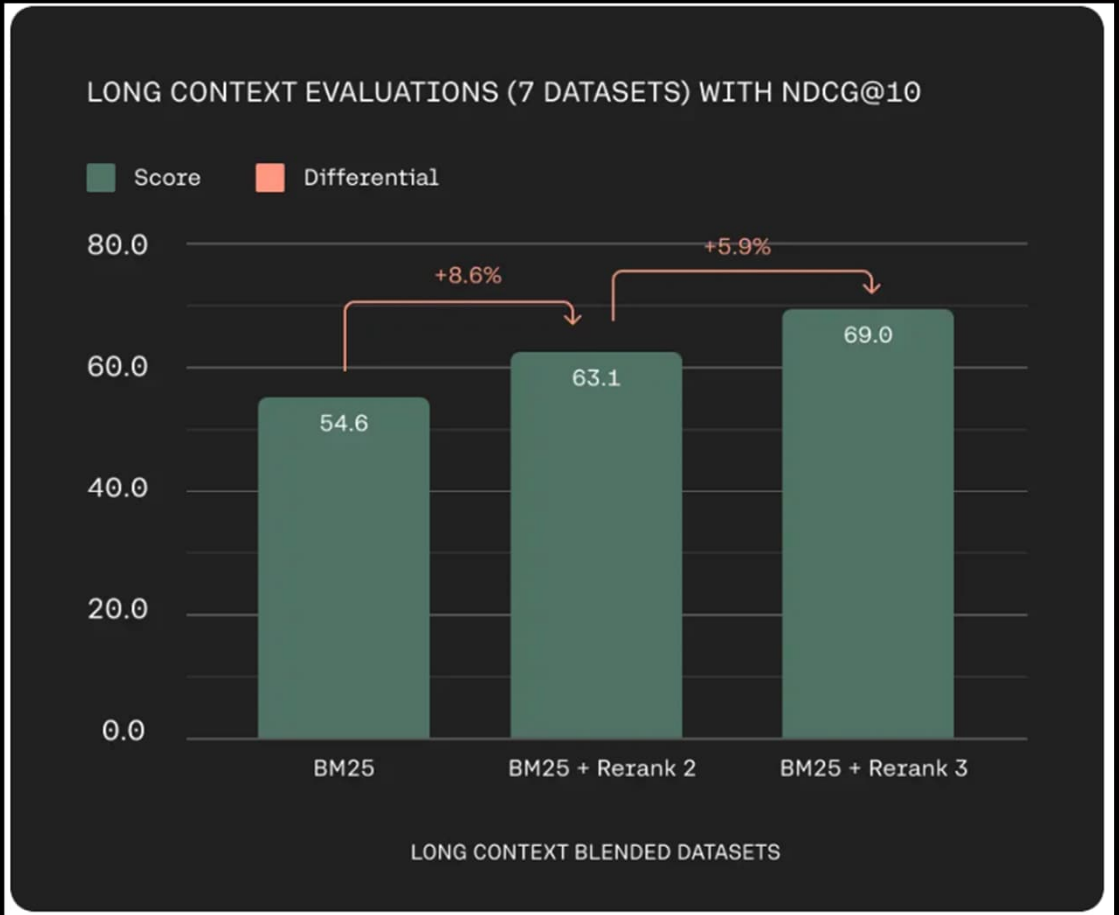
Service	Technical Specifications	Estimated Monthly Cost ($)
Azure App Service (Basic B1)	1.75 GB RAM, 10 GB storage	54.75
Azure AI Search (Basic Plan)	15 GB index, 1 replica	80-100
Azure OpenAI Service (GPT-4 turbo)	1 million tokens processed per month	90
Azure Blob Storage (1 TB in Hot mode)	1 TB storage, 100,000 read/write operations	25
Total Estimated Monthly Cost		249.75 - 269.75

Bonus : Approximation cost of January to February 2024



2. Costs Comparison

1. Rerank Cohere



Long context accuracy based on nDCG@10 on TREC 2019-2022, Conditional QA , NQ-Hard, Qasper, Genomics, QMSum, FinanceBench (higher is better).

- A major Challenge in building semantic search and RAG systems is determining how to best chunk data (split into smaller parts).
- The rerank 3 model now has a 4k context length (enabling the processing of larger documents for ranking).
- This allows the model to consider more context form the document when calculating relevance score.
- It also reduces the need for chunking documents to fit within the model's context window.

1. Rerank Cohere

Cost Analysis

Cohere Reranker 3 :

$2 per 1,000 searches.

GPT-4 Advanced:

$30 per 1 million input tokens.

$60 per 1 million output tokens.

Accuracy Increase: 10% (10% reduction in queries to ChatGPT).

Scenario:

Number of searches: 1,000

Input tokens per search: 1,000

Output tokens per search: 1,000

Calculations:

Cost with Reranker:

Reranker cost: (1,000 / 1,000) * $2 = *$2*

Reduced number of searches: 1,000 * (1 - 0.1) = *900*

Input tokens cost: (900 * 1,000 / 1,000,000) * $30 = *$27*

Output tokens cost: (900 * 1,000 / 1,000,000) * $60 = *$54*

Total cost with Reranker: $2 + $27 + $54 = *$83*

Cost without Reranker:

Input tokens cost: (1,000 * 1,000 / 1,000,000) * $30 = *$30*

Output tokens cost: (1,000 * 1,000 / 1,000,000) * $60 = *$60*

Total cost without Reranker: $30 + $60 = *$90*

Conclusion:

Cost with Reranker: $83

Cost without Reranker: $90

Cost Savings : 8% of cost savings

2. FrugalGPT

Dataset	Best invidual LLM	Cost to reach the same accuracy		Cost Savings
		Best individual LLM	FrugalGPT	
HEADLINES	GPT-4	33.1	0.6	98.3%
OVERULLING	GPT-4	9.7	2.6	73.3%
COQA	GPT-3	72.5	29.6	59.2%

Parameters :

To process 10M input tokens, GPT-J from Textsynth costs only $0.2, but OpenAI's GPT-4 needs $30.

Cascade of 3 LLMs /12 based on an optimization function under constraints.

FrugalGPT is based on a train set and test set.

3. Cache Augmented Generation

Table 2: Experimental Results

Size	System	Top-k	HotPotQA BERT-Score	SQuAD BERT-Score
Small	Sparse RAG	1	0.0673	0.7469
		3	0.0673	0.7999
		5	0.7549	0.8022
		10	0.7461	0.8191
	Dense RAG	1	0.7079	0.6445
		3	0.7509	0.7304
		5	0.7414	0.7583
		10	0.7516	0.8035
	CAG (Ours)		**0.7759**	**0.8265**
Medium	Sparse RAG	1	0.6652	0.7036
		3	0.7619	0.7471
		5	0.7616	0.7467
		10	0.7238	0.7420
	Dense RAG	1	0.7135	0.6188
		3	0.7464	0.6869
		5	0.7278	0.7047
		10	0.7451	0.7350
	CAG (Ours)		**0.7696**	**0.7512**
Large	Sparse RAG	1	0.6567	0.7135
		3	0.7424	0.7510
		5	0.7495	0.7543
		10	0.7358	0.7548
	Dense RAG	1	0.6969	0.6057
		3	0.7426	0.6908
		5	0.7300	0.7169
		10	0.7398	0.7499
	CAG (Ours)		**0.7527**	**0.7640**

Table 3: Comparison of Generation Time

Dataset	Size	System	Generation Time (s)
HotpotQA	Small	CAG	0.85292
		w/o CAG	9.24734
	Medium	CAG	1.66132
		w/o CAG	28.81642
	Large	CAG	2.32667
		w/o CAG	94.34917
SQuAD	Small	CAG	1.06509
		w/o CAG	10.29533
	Medium	CAG	1.73114
		w/o CAG	13.35784
	Large	CAG	2.40577
		w/o CAG	31.08368

3. Cache Augmented Generation

Assumptions:

- ChatGPT Generation Cost: $0.03 per 1000 tokens (input) and $0.06 per 1000 tokens (output)

Cache Augmented Generation (CAG):

- First request: Generation cost $0.09 per request
- Repeated requests (15%): $0 due to local cache (no generation required)

Retrieval-Augmented Generation (RAG):

- Every request requires both retrieval and generation: $0.09 per request

15% Repeated Queries:

CAG:

- First request: $0.09
- Repeated queries (15%): $0 (response from cache)
- Total cost for 1000 queries = (0 * 150) + ($0.09 * 850) = $76.5

RAG:

- Every request: $0.09
- Total cost for 1000 queries = $0.09 * 1000 = $90

Savings with CAG: 15% savings compared to RAG

Overall Monthly Cost Estimation

Service	Technical Specifications	Estimated Monthly Cost ($)
Azure App Service (Basic B1)	1.75 GB RAM, 10 GB storage	54.75
Azure AI Search (Basic Plan)	15 GB index, 1 replica	80-100
Azure OpenAI Service (GPT-4 turbo)	1 million tokens processed per month	90
Azure Blob Storage (1 TB in Hot mode)	1 TB storage, 100,000 read/write operations	25
Total Estimated Monthly Cost		249.75 - 269.75

Rerank => 8% cost saving on the OpenAI service : 242$ to 262 $ (pessimistic case)

FrugalGPT => 50% to 90% cost saving on the OpenAI Service : 204$ to 224$ (50%) – 168$ to 188$ (90%)

CAG => 15% cost saving on the OpenAI service : 235$ to 255$ (pessimistic case)

3. KPI

KPI :

Total Operating Cost (Azure + LLM)

- Calculates the overall cost of infrastructure (Azure) and language model usage, helping to assess the financial impact of the optimized approach.

Accuracy

- Assesses the relevance and precision of the responses generated by the system, ensuring that legal content is accurate and meets user needs.

Latency

- the average time required to process a query and return a response, ensuring that system performance meets user expectations for speed.

System Reliability

- Monitors system stability, tracking failures or downtime, and ensuring that the solution remains dependable under various conditions.

User Feedback

- Collects feedback from internal users and stakeholders to gauge satisfaction, identify pain points, and validate the success of the optimization efforts.

4. Future Directions

RAG vs KAG vs CAG

Aspect	RAG	KAG	CAG
Knowledge Type	Dynamic, unstructured text	Structured knowledge graphs	Preloaded static data
Speed	Slower (retrieval latency)	Moderate (KG traversal)	Fastest (no retrieval)
Best For	Real-time updates, open-domain QA	Factual precision, multi-hop tasks	Stable data, low-latency needs
Complexity	High (retrieval pipelines)	Very high (KG construction)	Low (preloaded cache)
Scalability	Excellent (handles large corpora)	Limited by KG size	Limited by context window

RAG vs KAG vs CAG

Future Trends: Hybrid Systems

Emerging solutions combine these frameworks:

- CAG + RAG: Preload common queries in CAG, use RAG for dynamic data.
- KAG + RAG: Merge KGs for reasoning and real-time retrieval for updates (e.g., legal case laws).
- Agentic AI: Autonomous systems like IM-RAG switch strategies based on query complexity.
- Modular RAG Frameworks: Recent studies have introduced modular RAG systems that decompose complex architectures into independent modules and specialized operators, enhancing reconfigurability and adaptability.
- Fine-Tuning with User Feedback: Integrating user feedback directly into the training process of RAG systems has been shown to improve performance and applicability.

Week 4
Discovery of the internal tool & Technical Plan

Contents

Discovery of the internal tool

1. *Legal Copilot*

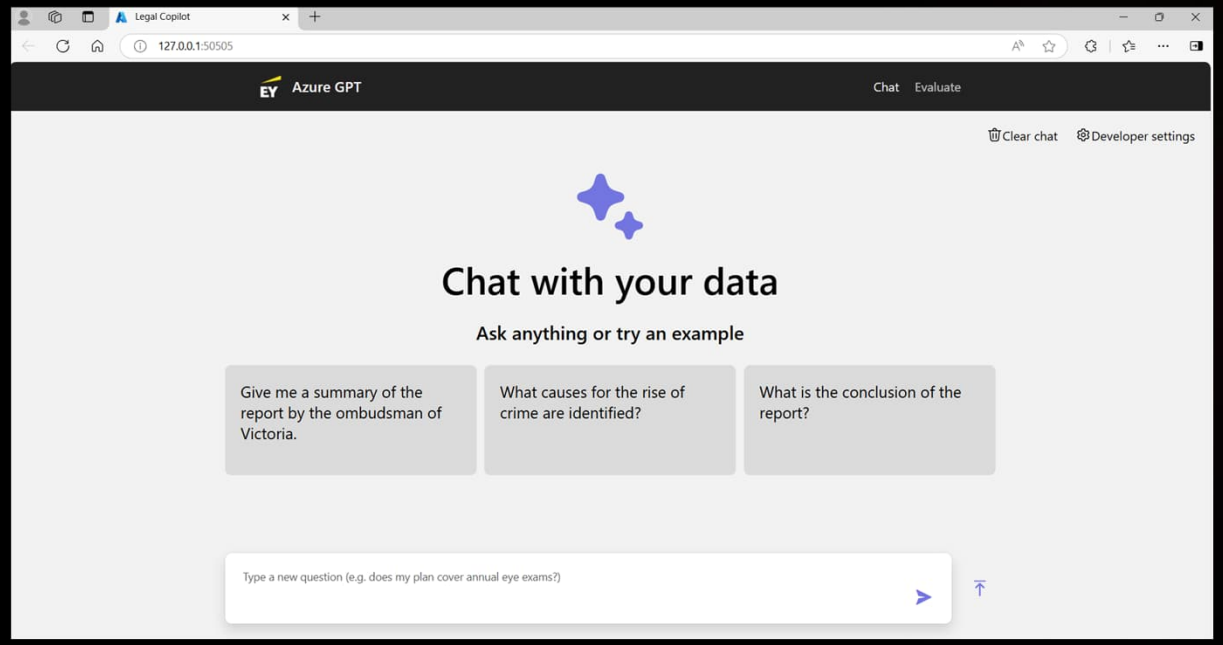
Implementation of Frugal GPT

1. Redefining the internal infrastructure
2. Train, Test & Evaluate
3. Save the strategy

New Paper 09/02/2025

Discovery of the internal tool

1. Legal Copilot



- Access setup 🔑
- Configuration of necessary resources on Azure ⚙️
- Deployment of the local environment 🖥️
- Verification of application functionality ✅

Implementation of FrugalGPT

New Paper of FrugalGPT

Updates

- Update of Evaluated LLMs 📌
  - The new study includes more recent large language models (LLMs), such as Gemini 1.5 and Claude 3.5, reflecting rapid advancements in the LLM field.
- In-Depth Cost Structure Analysis 💰
  - The authors examine the heterogeneous pricing structures of popular LLM APIs.
- Improved FrugalGPT Algorithm ⚙️
  - The framework has been optimized to adaptively select LLMs based on specific queries, reducing costs while improving response accuracy.
- New Use Cases and Evaluated Tasks 🔍
  - The experiments now cover a broader range of NLP tasks, including news classification, reading comprehension, and scientific Q&A, demonstrating FrugalGPT's versatility and efficiency.

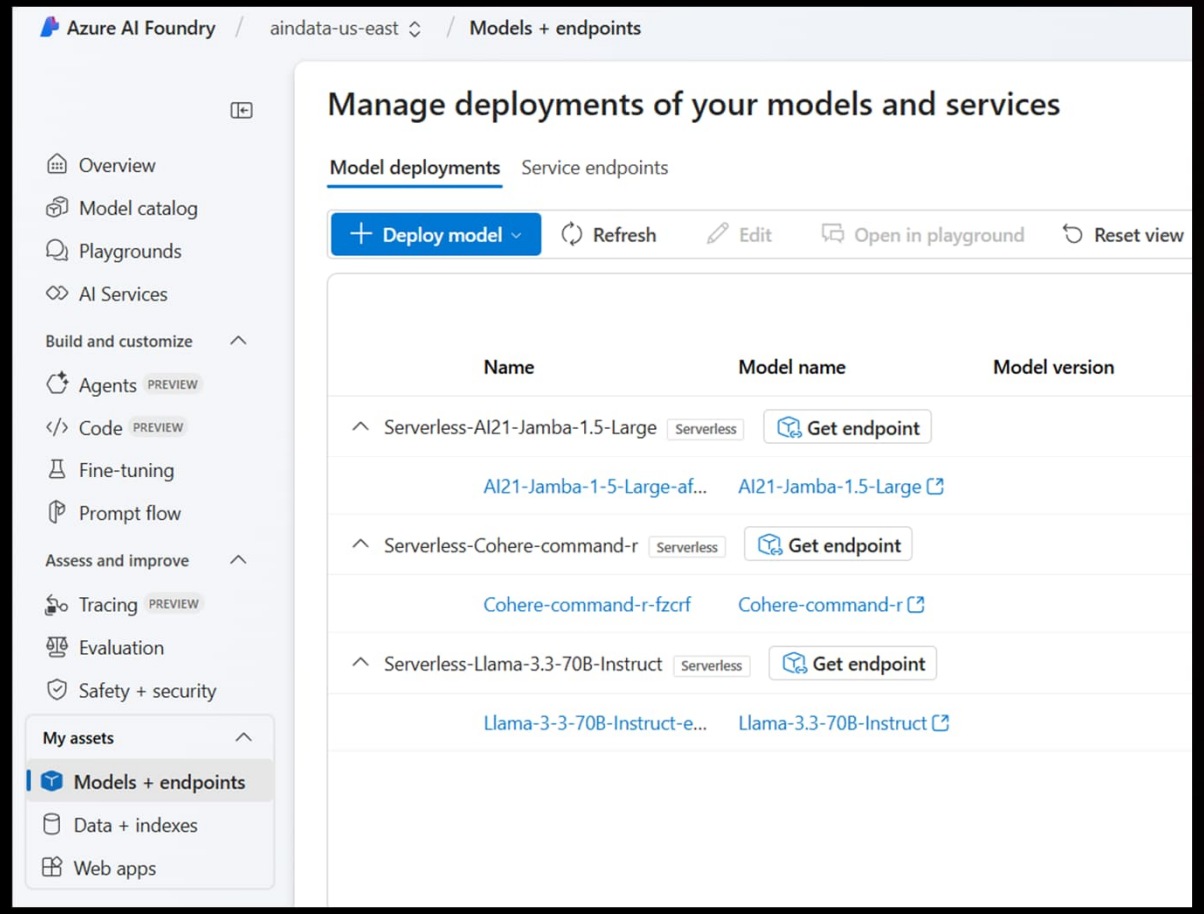
Week 5 Preliminary testing on internal datasets

Contents

Implementation of Frugal GPT

1. Redefining the internal infrastructure
2. Train, Test & Evaluate
3. Save the strategy

1. Redefining the internal infrastructure



- The current infrastructure relies exclusively on OpenAI's LLMs.
- FrugalGPT enables the integration of multiple LLMs via their APIs from Azure AI Foundry to reduce costs in the LLM cascade.
- FrugalGPT has been integrated into Legal Copilot to set up the necessary infrastructure.

2. Train, Test & Evaluate

- Train multiple LLMs using a constrained budget training set.
- Identify which models perform best on specific datasets.
- Deploy the most efficient models accordingly.

3. Save the strategy

By saving the strategy, we can:

- Identify the optimal cascade for a similar theme (e.g., a financial dataset) 📊
- Optimize costs related to LLM usage 💰

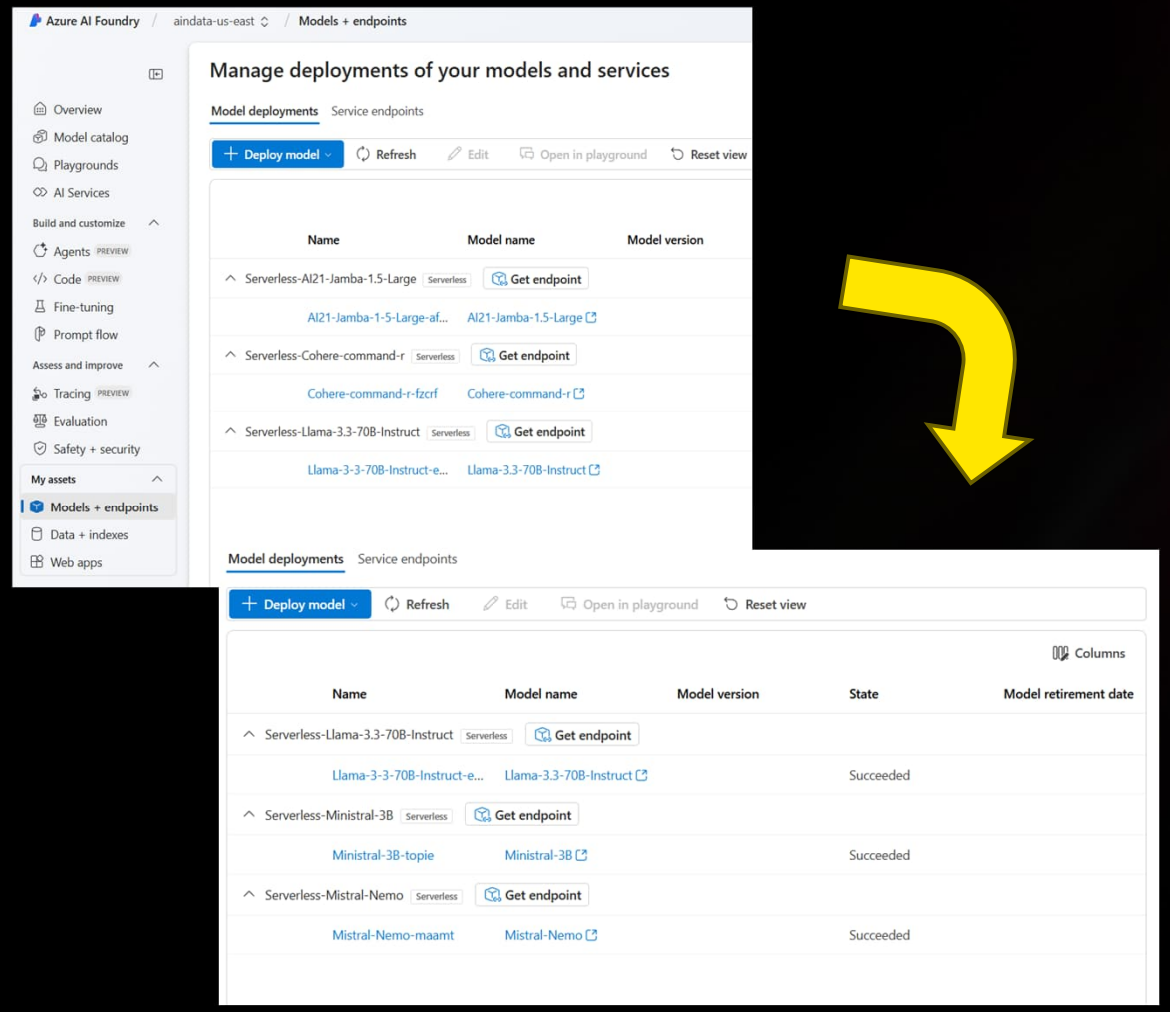
Week 6
Refinement of FrugalGPT

Contents

Update : Frugal GPT

1. Redefining the internal infrastructure
2. Train, Test & Evaluate
3. Save the strategy
4. Analysis

1. Redefining the internal infrastructure



- The current infrastructure relies exclusively on OpenAI's LLMs.
- FrugalGPT enables the integration of multiple LLMs via their APIs from Azure AI Foundry to reduce costs in the LLM cascade.
- FrugalGPT has been integrated into Legal Copilot to set up the necessary infrastructure.
- Update : LLMs have been changed to meet the new requirements in the market.

1. Redefining the internal infrastructure

2. Train, Test & Evaluate

- Train multiple LLMs using a constrained budget training set.
  - GPT4o- mini
  - Llama-3-70b instruct -> GPT4o
  - Ministral
  - Mistal NeMo
  - GPT3.5
- Identify which models perform best on specific datasets.
- Deploy the most efficient models accordingly.
  - Cascade of 3 LLMs

3. Save the strategy

By saving the strategy, we can:

- Identify the optimal cascade for a similar theme (e.g., a financial dataset) 📊
- Optimize costs related to LLM usage 💰

4. Analysis

	Test_acc	Test_cost	Test_size	Train_acc	Train_cost	Train_size	Budget	Method	Provider	Marker
0	0.92	0.000203	100	0.92	0.000203	100	0.000090	FrugalGPT	FrugalGPT	1
1	0.92	0.000203	100	0.92	0.000203	100	0.000744	FrugalGPT	FrugalGPT	1
2	0.92	0.000203	100	0.92	0.000203	100	0.001464	FrugalGPT	FrugalGPT	1
3	0.92	0.000203	100	0.92	0.000203	100	0.002183	FrugalGPT	FrugalGPT	1
4	0.92	0.000203	100	0.92	0.000203	100	0.002903	FrugalGPT	FrugalGPT	1
5	0.92	0.000203	100	0.92	0.000203	100	0.003622	FrugalGPT	FrugalGPT	1
6	0.92	0.000203	100	0.92	0.000203	100	0.004342	FrugalGPT	FrugalGPT	1
7	0.92	0.000203	100	0.92	0.000203	100	0.005061	FrugalGPT	FrugalGPT	1
8	0.92	0.000203	100	0.92	0.000203	100	0.005781	FrugalGPT	FrugalGPT	1
9	0.92	0.000203	100	0.92	0.000203	100	0.006500	FrugalGPT	FrugalGPT	1
0	0.86	0.000295	100	0.90	0.000294	100	10.000000	gpt-3.5-turbo	azure_openai	1
1	0.66	0.000433	100	0.69	0.000431	100	10.000000	llama-3-70b-instruct	azure_llama	1
2	0.92	0.000090	100	0.87	0.000090	100	10.000000	mistral-nemo	azure_mistral_nemo	1
3	0.82	0.000024	100	0.79	0.000024	100	10.000000	ministral	azure_ministral	1
4	0.91	0.000089	100	0.89	0.000089	100	10.000000	gpt-4o-mini	azure_gpt4o_mini	1

```
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\azure-production-rag>
```

Large models like GPT-3.5 and Llama-3-70B come with high costs, while smaller models like GPT-4o-mini and Mistral are much more affordable. FrugalGPT optimizes model usage to reduce costs while maintaining high accuracy.

Week 7
Advanced development of the technical solution

Contents

Update : FrugalGPT

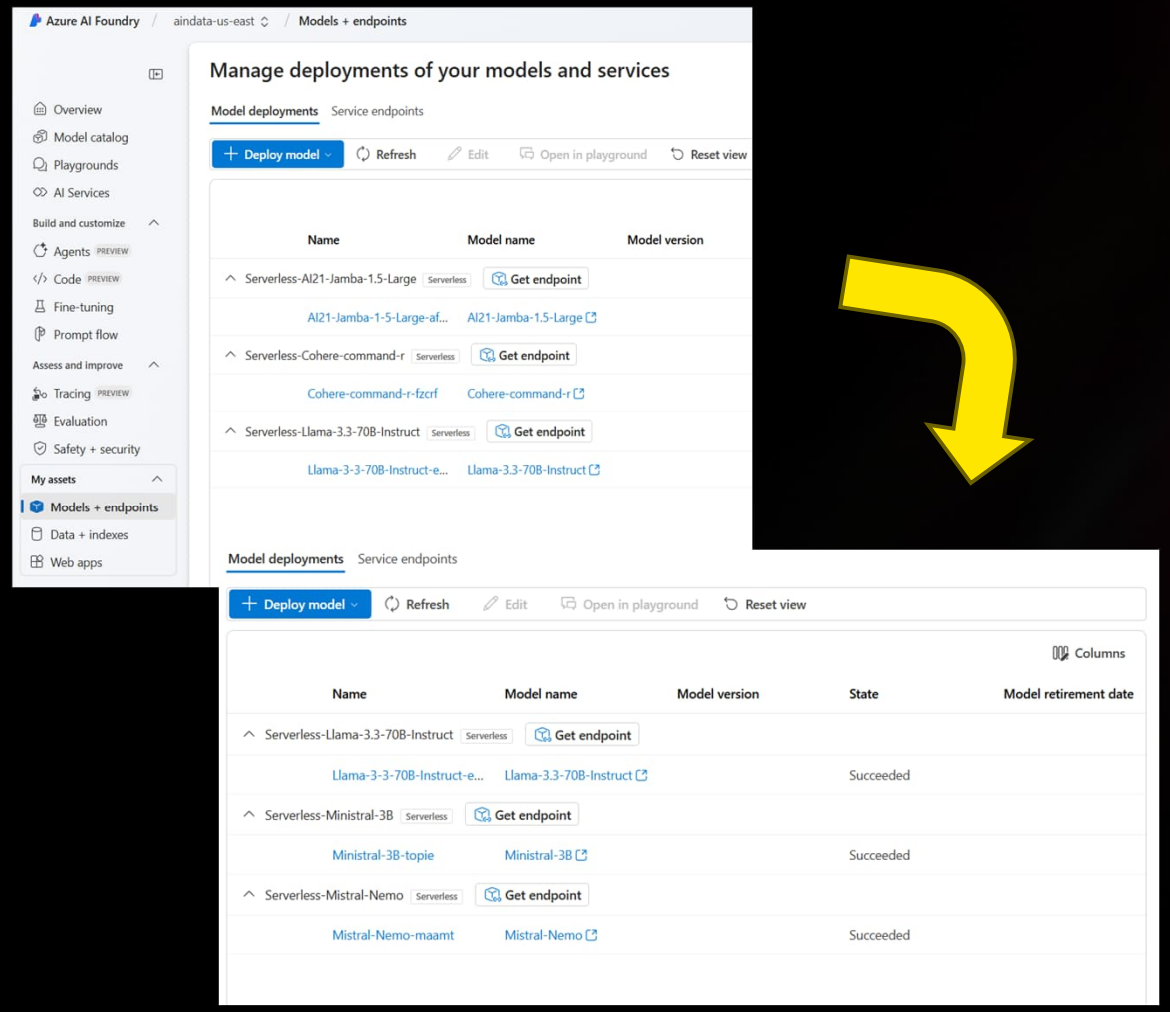
1. Redefining the internal infrastructure
2. Train, Test & Evaluate
3. Save the strategy
4. Analysis

Current work : Cascade of LLMs in Legal Copilot

1. Replacing OpenAI AI with Cascade of LLM
2. Next Steps for Enhancing Legal Copilot

Update : FrugalGPT

1. Redefining the internal infrastructure



- The current infrastructure relies exclusively on OpenAI's LLMs.
- FrugalGPT enables the integration of multiple LLMs via their APIs from Azure AI Foundry to reduce costs in the LLM cascade.
- FrugalGPT has been integrated into Legal Copilot to set up the necessary infrastructure.
- Update : LLMs have been changed to meet the new requirements in the market.

1. Redefining the internal infrastructure

2. Train, Test & Evaluate

- Train multiple LLMs using a constrained budget training set.
  - GPT4o- mini
  - Llama-3-70b instruct -> GPT4o
  - Ministral
  - Mistal NeMo

- Identify which models perform best on specific datasets.

- Deploy the most efficient models accordingly.
  - Cascade of 3 LLMs

3. Save the strategy

By saving the strategy, we can:

- Identify the optimal cascade for a similar theme (e.g., a financial dataset) 📊
- Optimize costs related to LLM usage 💰


```
{
    "budget": {
        "1e-05": {
            "thres_list": [],
            "model_list": [],
            "quantile": []
        },
        "0.0005733333333333334": {
            "thres_list": [
                0.30147149735624973,
                0.35670822208390784,
                1.0
            ],
            "model_list": [
                "azure_mistral_nemo/mistral-nemo",
                "azure_ministral/ministral",
                "azure_gpt4o_mini/gpt-4o-mini"
            ],
            "quantile": [
                0.07693153846153845,
                0.15385307692307693
            ]
```

4. Analysis

	Test_acc	Test_cost	Test_size	Train_acc	Train_cost	Train_size	Budget	Method	Provider	Marker
0	0.93	0.000090	100	0.93	0.000090	100	0.000090	FrugalGPT	FrugalGPT	1
1	0.93	0.000095	100	0.93	0.000095	100	0.000744	FrugalGPT	FrugalGPT	1
2	0.93	0.000095	100	0.93	0.000095	100	0.001464	FrugalGPT	FrugalGPT	1
3	0.93	0.000095	100	0.93	0.000095	100	0.002183	FrugalGPT	FrugalGPT	1
4	0.93	0.000095	100	0.93	0.000095	100	0.002903	FrugalGPT	FrugalGPT	1
5	0.93	0.000095	100	0.93	0.000095	100	0.003622	FrugalGPT	FrugalGPT	1
6	0.93	0.000095	100	0.93	0.000095	100	0.004342	FrugalGPT	FrugalGPT	1
7	0.93	0.000095	100	0.93	0.000095	100	0.005061	FrugalGPT	FrugalGPT	1
8	0.93	0.000095	100	0.93	0.000095	100	0.005781	FrugalGPT	FrugalGPT	1
9	0.93	0.000095	100	0.93	0.000095	100	0.006500	FrugalGPT	FrugalGPT	1
0	0.90	0.001477	100	0.89	0.001474	100	10.000000	gpt-4o	azure_gpt4o	1
1	0.91	0.000091	100	0.87	0.000091	100	10.000000	mistral-nemo	azure_mistral_nemo	1
2	0.77	0.000024	100	0.79	0.000024	100	10.000000	ministral	azure_ministral	1
3	0.92	0.000089	100	0.87	0.000089	100	10.000000	gpt-4o-mini	azure_gpt4o_mini	1

```
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\azure-production-rag>
```

Large models like GPT-4o come with high costs, while smaller models like GPT-4o-mini and Ministral are much more affordable. FrugalGPT optimizes model usage to reduce costs while maintaining high accuracy.

Current work : Cascade of LLMs in Legal Copilot

1. Replacing OpenAI Call with Cascade of LLMs

TERMINAL python - app

```
 * Serving Quart app 'app'
 * Debug mode: False
 * Please use an ASGI server (e.g. Hypercorn) directly in production
 * Running on http://localhost:50505 (CTRL + C to quit)
<azure.search.documents.indexes.aio._search_indexer_client.SearchIndexerClient object at 0x0000021027153A90>
[2025-03-13 10:17:27 - root - CRITICAL] loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\azure-production-rag\app\backend\strategy\AGNEWS_Model20252602\cascade_strategy.json
[2025-03-13 10:17:30 - root - CRITICAL] loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\azure-production-rag\app\backend\strategy\AGNEWS_Model20252602\cascade_strategy.json
[2025-03-13 10:17:31 +0100] [7864] [INFO] Running on http://127.0.0.1:50505 (CTRL + C to quit)
[2025-03-13 10:17:31 - hypercorn.error - INFO] Running on http://127.0.0.1:50505 (CTRL + C to quit)
[2025-03-13 10:17:37 +0100] [7864] [INFO] 127.0.0.1:54150 GET /auth_setup 1.1 200 386 5600
[2025-03-13 10:17:37 +0100] [7864] [INFO] 127.0.0.1:54150 GET /auth_setup 1.1 - - 6730
[2025-03-13 10:17:37 +0100] [7864] [INFO] 127.0.0.1:54150 GET /.auth/refresh 1.1 404 207 3009
[2025-03-13 10:17:37 +0100] [7864] [INFO] 127.0.0.1:54150 GET /.auth/refresh 1.1 - - 4006
[2025-03-13 10:17:37 +0100] [7864] [INFO] 127.0.0.1:54150 GET /config 1.1 200 83 6174
[2025-03-13 10:17:37 +0100] [7864] [INFO] 127.0.0.1:54150 GET /config 1.1 - - 7176
[2025-03-13 10:18:05 - root - CRITICAL] Using service: azure_mistral_nemo/mistral-nemo
[2025-03-13 10:18:06 +0100] [7864] [INFO] 127.0.0.1:54240 POST /chat 1.1 200 - 4879378
[2025-03-13 10:18:06 +0100] [7864] [INFO] 127.0.0.1:54240 POST /chat 1.1 - - 4880378
[2025-03-13 10:18:34 - root - CRITICAL] Using service: azure_mistral_nemo/mistral-nemo
[2025-03-13 10:18:34 - root - CRITICAL] Using service: azure_ministral/ministral
[2025-03-13 10:18:35 +0100] [7864] [INFO] 127.0.0.1:54258 POST /chat 1.1 200 - 1928569
[2025-03-13 10:18:35 +0100] [7864] [INFO] 127.0.0.1:54258 POST /chat 1.1 - - 1929569
```

Clear chat Developer settings

What is the capital of France?

The capital of France is Paris.,azure_mistral_nemo\mistral-nemo

Did you like this response?

What is the capital of Belgium?

The capital of Belgium is Brussels.,azure_ministral\ministral

Did you like this response?

2. Next Steps for Enhancing Legal Copilot

- Model Framework :
  - Define whether each model should be flexible in its responses or strictly based on the database.

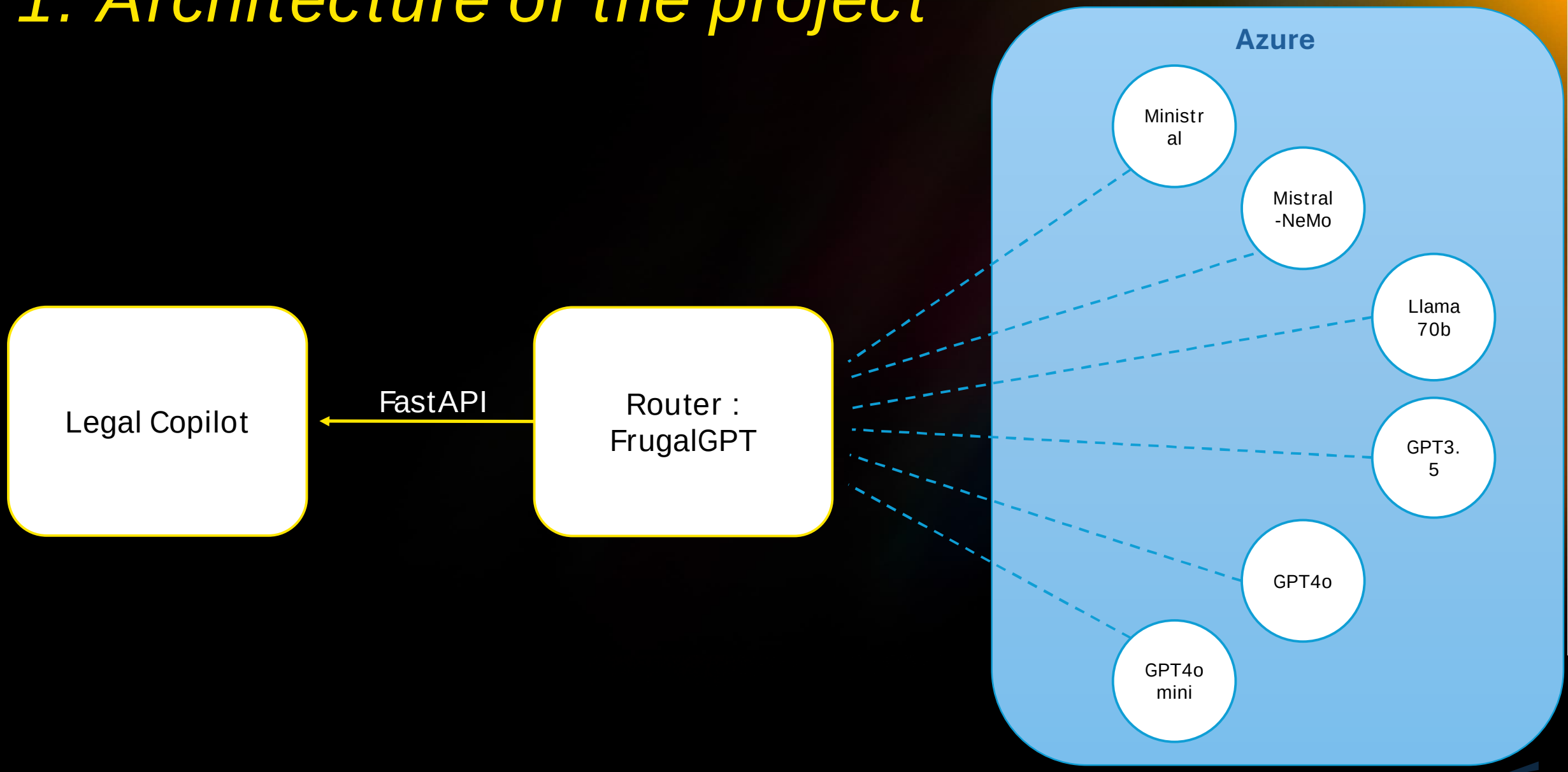
- Database integration :
  - Link the database to the models by creating training data from documents. Currently, some questions on the homepage cannot be answered by any model.

Week 8
Advanced development of the technical solution

Contents

Update : FrugalGPT as a Router
1. Architecture of the project
2. FrugalGPT
2.1 LLM Cascade
2.2 LLM Approximation
2.3 Cost comparison w/ & w/o the cascade

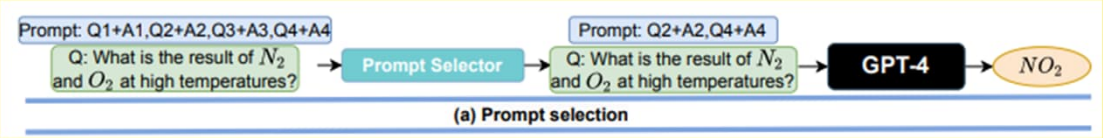
1. Architecture of the project



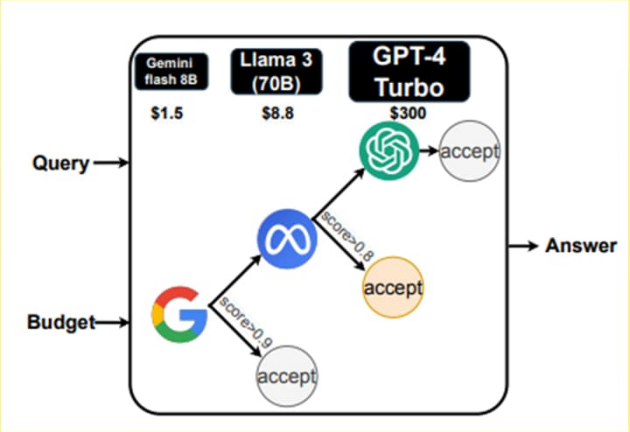
2. FrugalGPT

FrugalGPT

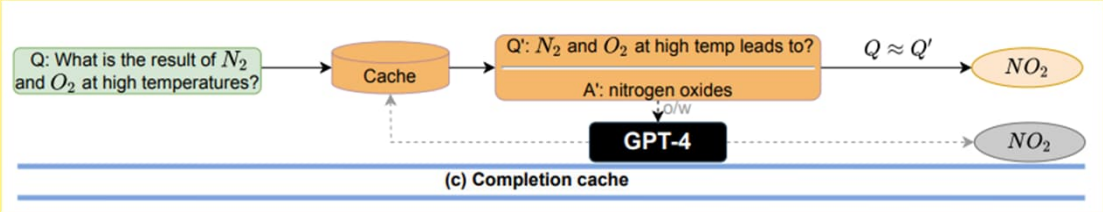
Prompt Adaptation



LLM Cascade



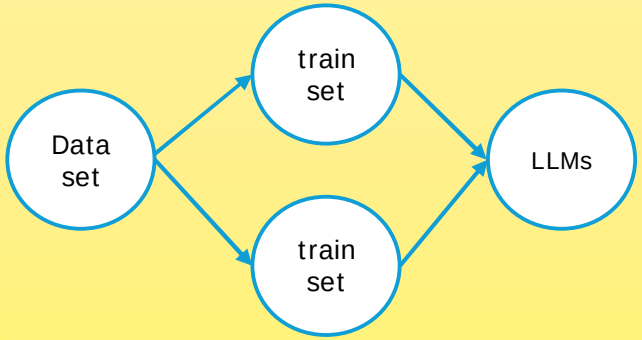
LLM Approximation



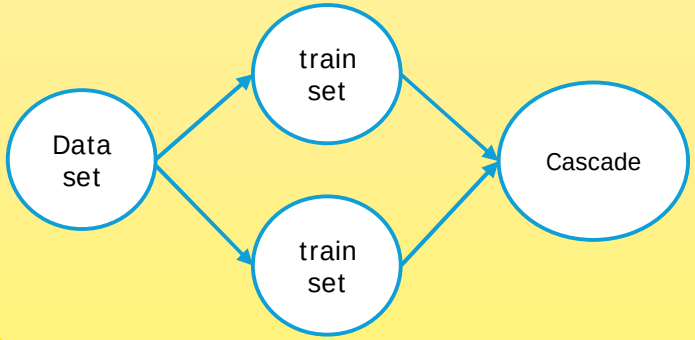
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



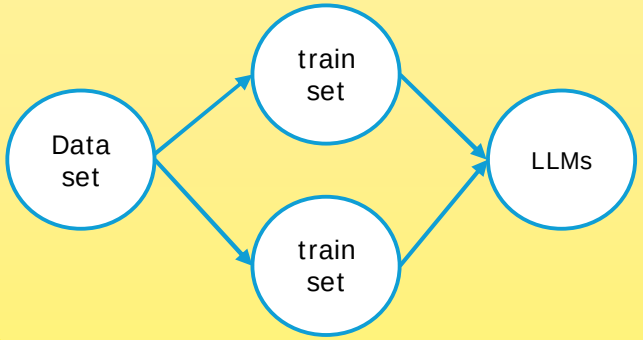
Cascade

1. Optimal strategy by testing LLM permutations that meet budget and threshold constraints
2. Derives a strategy that aligns with the various cost requirements
3. Reading the cascade to respond to a new query
4. BERT used to evaluate the response quality checking if the threshold is exceeded. If so, the response is retained

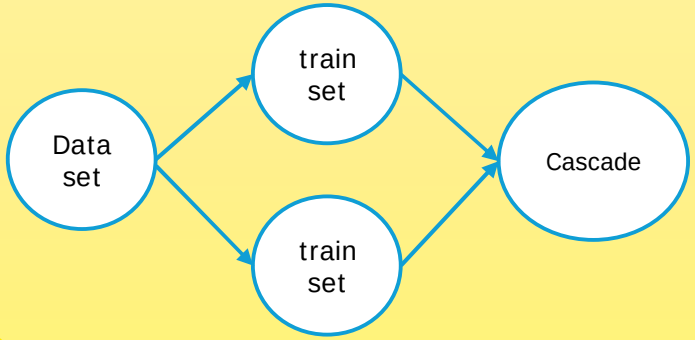
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



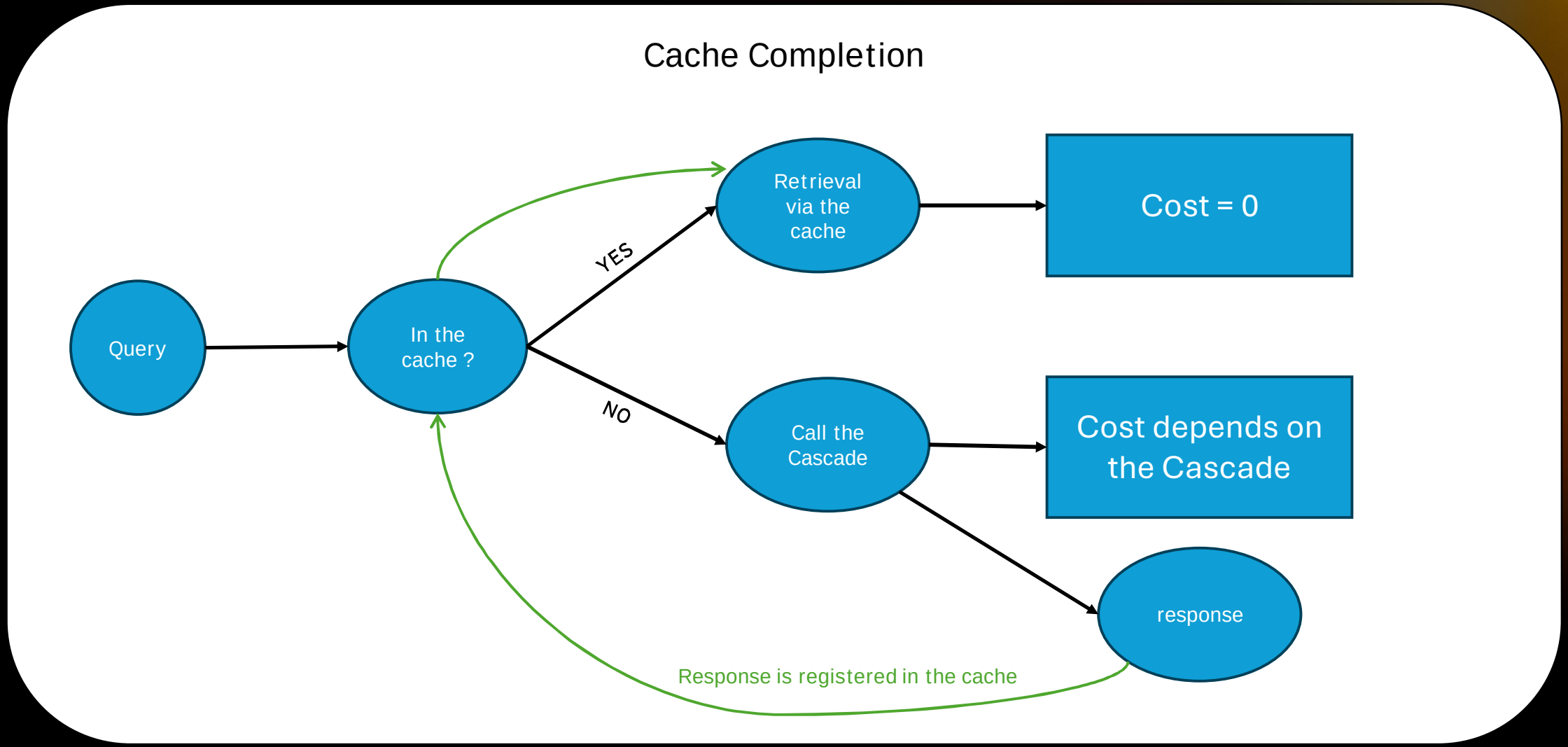
	Test_acc	Test_cost	Test_size	Train_acc	Train_cost	Train_size	Budget	Method	Provider	Marker
0	0.00	0.000000	100	0.00	0.000000	100	0.000010	FrugalGPT	FrugalGPT	1
1	0.94	0.000153	100	0.94	0.000153	100	0.000573	FrugalGPT	FrugalGPT	1
2	0.94	0.000153	100	0.94	0.000153	100	0.001127	FrugalGPT	FrugalGPT	1
3	0.94	0.000153	100	0.94	0.000153	100	0.001680	FrugalGPT	FrugalGPT	1
4	0.94	0.000153	100	0.94	0.000153	100	0.002233	FrugalGPT	FrugalGPT	1
5	0.94	0.000153	100	0.94	0.000153	100	0.002787	FrugalGPT	FrugalGPT	1
6	0.94	0.000153	100	0.94	0.000153	100	0.003340	FrugalGPT	FrugalGPT	1
7	0.94	0.000153	100	0.94	0.000153	100	0.003893	FrugalGPT	FrugalGPT	1
8	0.94	0.000153	100	0.94	0.000153	100	0.004447	FrugalGPT	FrugalGPT	1
9	0.94	0.000153	100	0.94	0.000153	100	0.005000	FrugalGPT	FrugalGPT	1
0	0.88	0.000295	100	0.90	0.000294	100	10.000000	gpt-3.5-turbo	azure_openai	1
1	0.75	0.000432	100	0.71	0.000430	100	10.000000	llama-3-70b-instruct	azure_llama	1
2	0.90	0.001477	100	0.89	0.001474	100	10.000000	gpt-4o	azure_gpt4o	1
3	0.91	0.000091	100	0.87	0.000091	100	10.000000	mistral-nemo	azure_mistral_nemo	1
4	0.77	0.000024	100	0.79	0.000024	100	10.000000	ministral	azure_ministral	1
5	0.92	0.000089	100	0.87	0.000089	100	10.000000	gpt-4o-mini	azure_gpt4o_mini	1

```
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT>
```

The cost given is the train cost for 100 queries & answers

94% of accuracy for FrugalGPT

2.2. LLM Approximation



2.2. LLM Approximation

Tech part of the Cache Completion

Architecture of the Cache

Cache JSON
Index FAISS (Facebook AI Similarity Search)

Tech used

Sentence Transformer
Index FAISS (Facebook AI Similarity Search)
Similarity Threshold

Advantages

Reduced API call costs
Time saving
Dynamic Optimization

Sentence Transformer & all-mpnet-base-v2

- Text into dense vector embeddings for semantic comparison.
- MPNet (natural language understanding)

FAISS

- O(logN) search complexity, much faster than O(N)
- Finds the closest cached response based on vector similarity.

2.3. Cost comparison w/ & w/o the cascade

```
[2025-03-19 15:12:49 +0100] [26528] [INFO] 127.0.0.1:60151 POST /chat 1.1 200 - 5972644
[2025-03-19 15:12:49 +0100] [26528] [INFO] 127.0.0.1:60151 POST /chat 1.1 - - 5972644
Cascade total cost: 5.74e-05
GPT-4o total cost: 0.00017250000000000002
[2025-03-19 15:14:45 +0100] [26528] [INFO] 127.0.0.1:60397 POST /chat 1.1 200 - 10494007
[2025-03-19 15:14:45 +0100] [26528] [INFO] 127.0.0.1:60397 POST /chat 1.1 - - 10494007
Cascade total cost: 0
GPT-4o total cost: 0.0001775
[2025-03-19 15:15:15 +0100] [26528] [INFO] 127.0.0.1:59882 POST /chat 1.1 200 - 6550885
[2025-03-19 15:15:15 +0100] [26528] [INFO] 127.0.0.1:59882 POST /chat 1.1 - - 6552162
```

```
INFO:     127.0.0.1:60152 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
---invoke new API call---
CRITICAL:root:Using service: azure_openai/gpt-3.5-turbo
CRITICAL:root:Using service: azure_openai/gpt-3.5-turbo
---invoke new API call---
azure_openai/gpt-3.5-turbo
INFO:     127.0.0.1:60398 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:60398 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
INFO:     127.0.0.1:59883 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:59883 - "POST /execute_gpt4o HTTP/1.1" 200 OK
```

Legal Copilot — 127.0.0.1:50505

EY Azure GPT — Chat — Evaluate

Clear chat — Developer settings

What is the capital of Spain ?

The capital of Spain is Madrid.

Did you like this response?

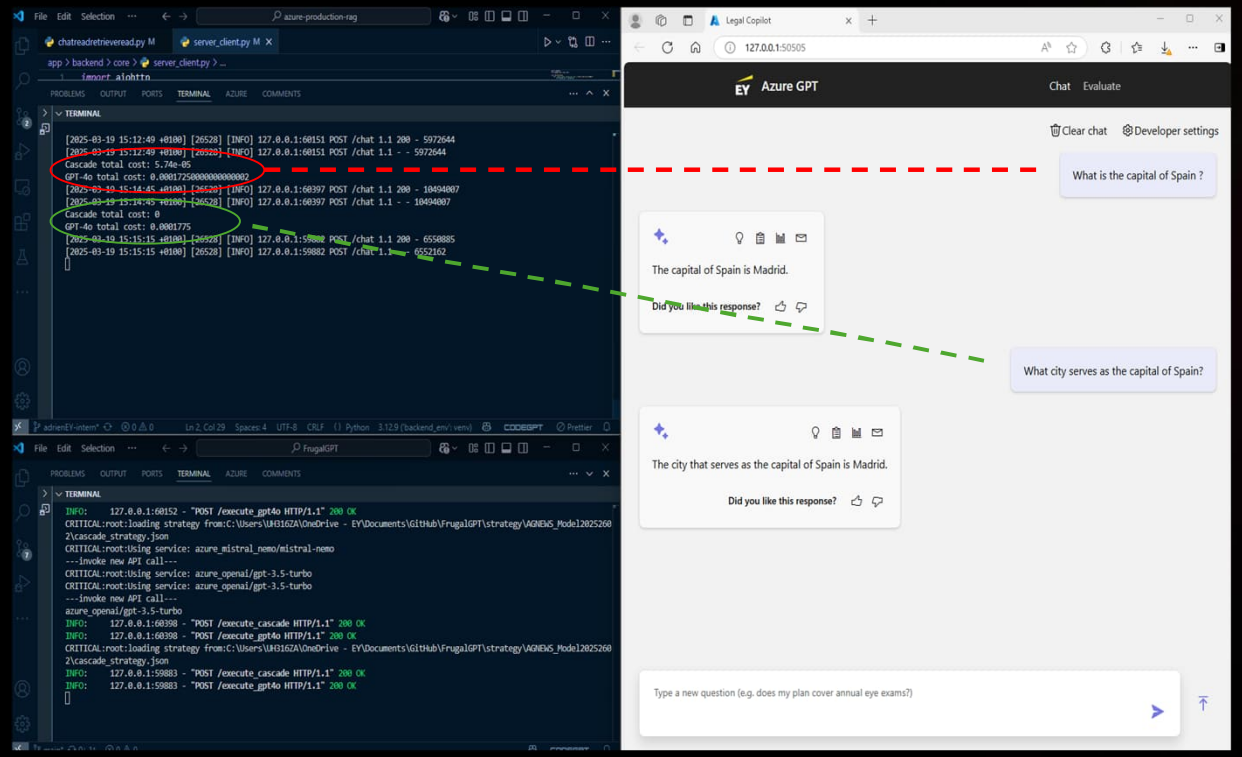
What city serves as the capital of Spain?

The city that serves as the capital of Spain is Madrid.

Did you like this response?

Type a new question (e.g. does my plan cover annual eye exams?)

2.3. Cost comparison w/ & w/o the cascade



Cost of the query and the answer :

- With the cascade :
  - 0.0000574$
- With GPT4o :
  - 0.000175$
- With the cache :
  - 0$ (for a similar question)

Cost reduction of 66% for the given query

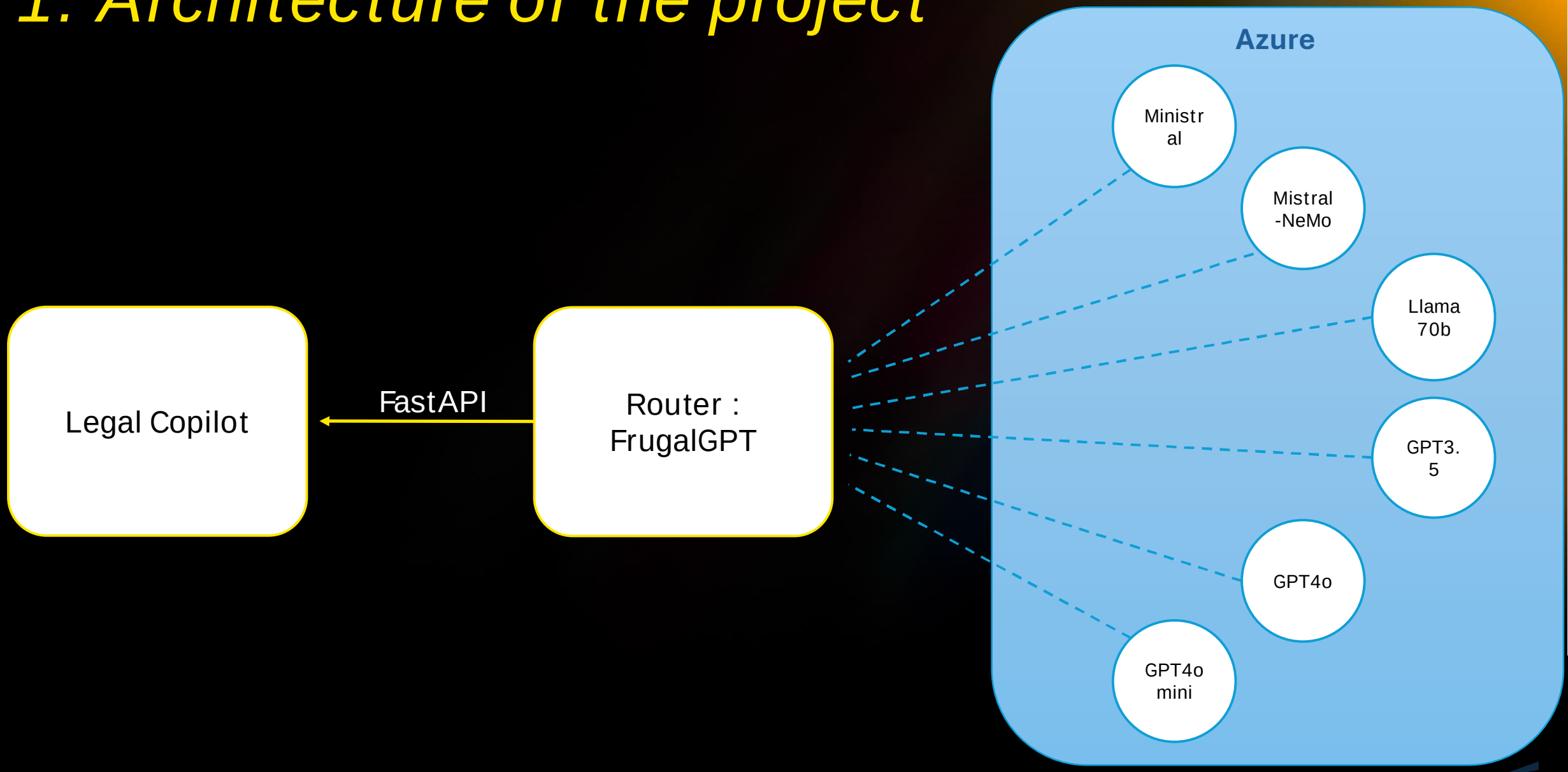
Week 9
Advanced development of the technical solution

Contents

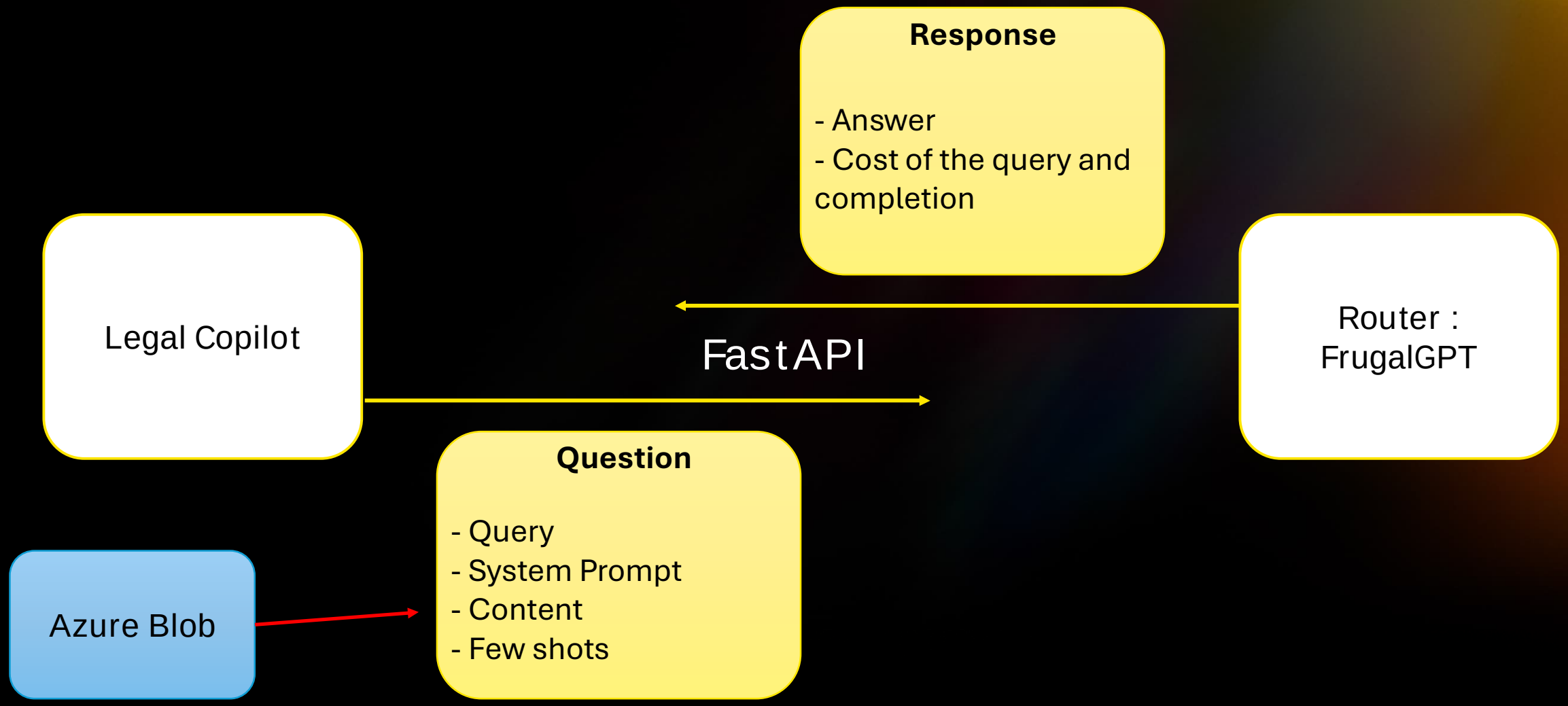
FrugalGPT as a Router

1. Architecture of the project
2. FrugalGPT
    - 2.1 LLM Cascade
    - 2.2 LLM Approximation
    - 2.3 Cost comparison w/ & w/o the cascade
3. Demo
4. Cost Analysis

1. Architecture of the project



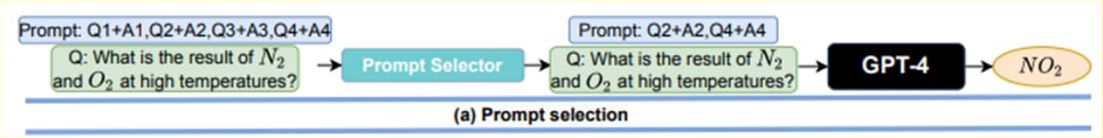
1. Architecture of the project



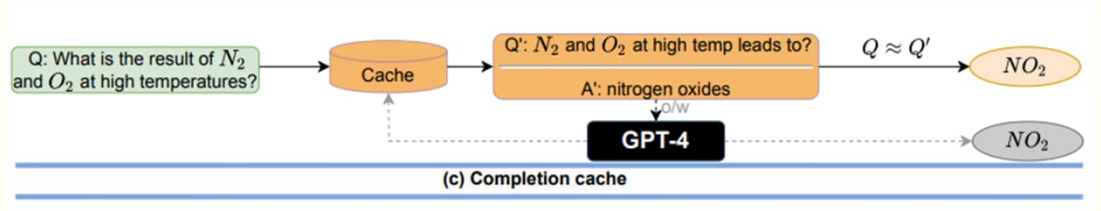
2. FrugalGPT

FrugalGPT

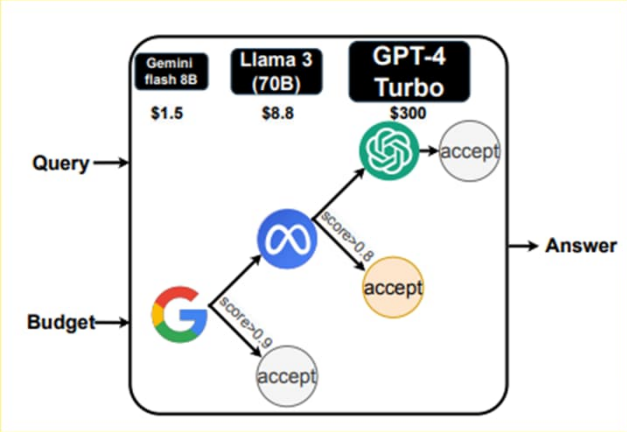
Prompt Adaptation



LLM Approximation



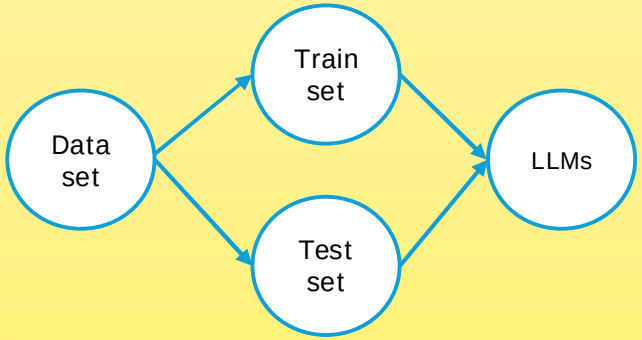
LLM Cascade



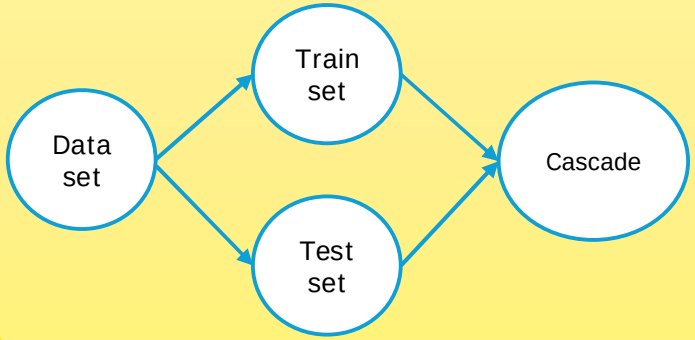
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



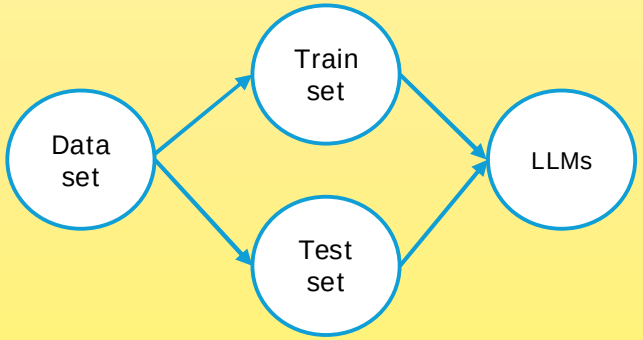
Cascade

1. Optimal strategy by testing LLM permutations that meet budget and threshold constraints.
2. Derives a strategy that aligns with the various cost requirements.
3. Reading the cascade to respond to a new query.
4. BERT used to evaluate the response quality checking if the threshold is exceeded. If so, the response is retained.

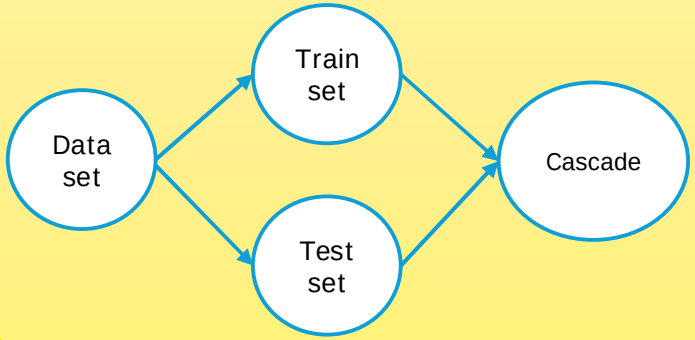
2.1. LLM Cascade

LLM Cascade

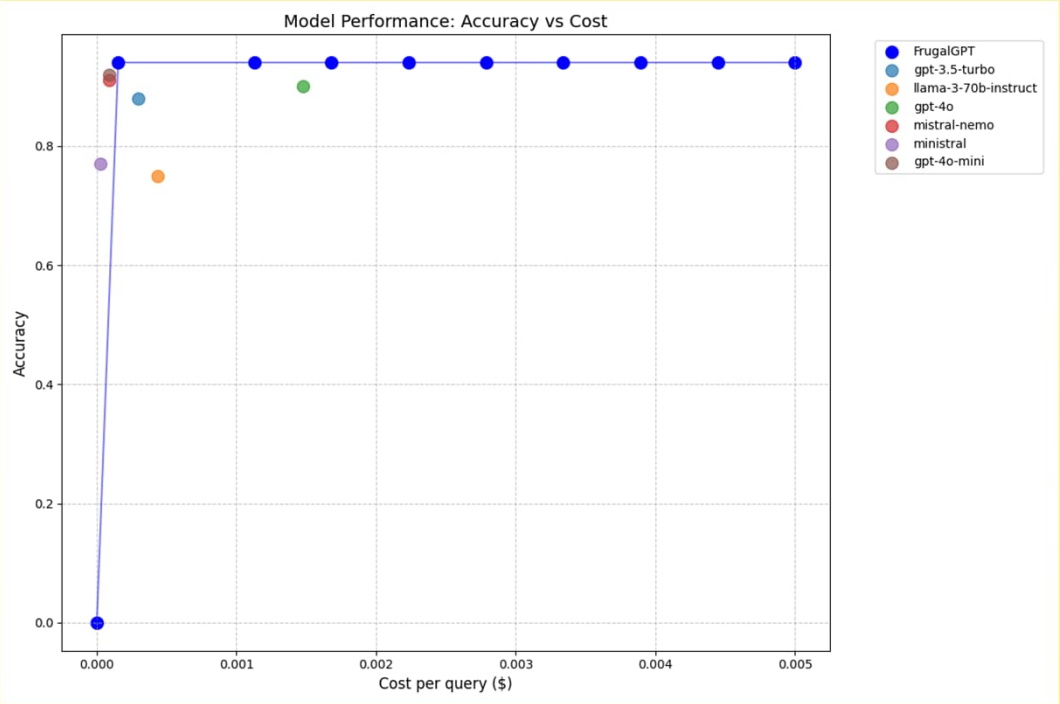
Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



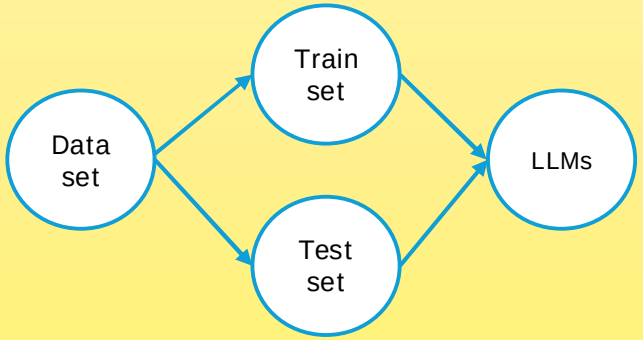
Cascade



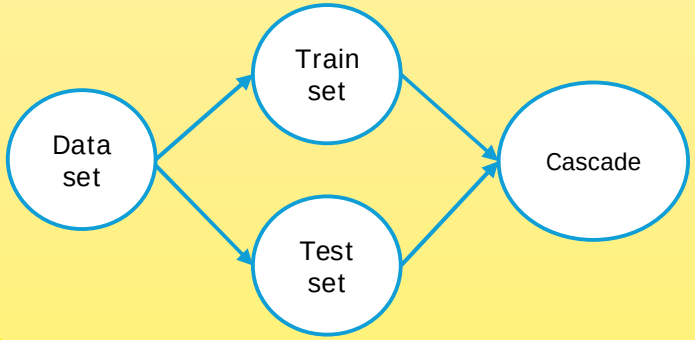
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



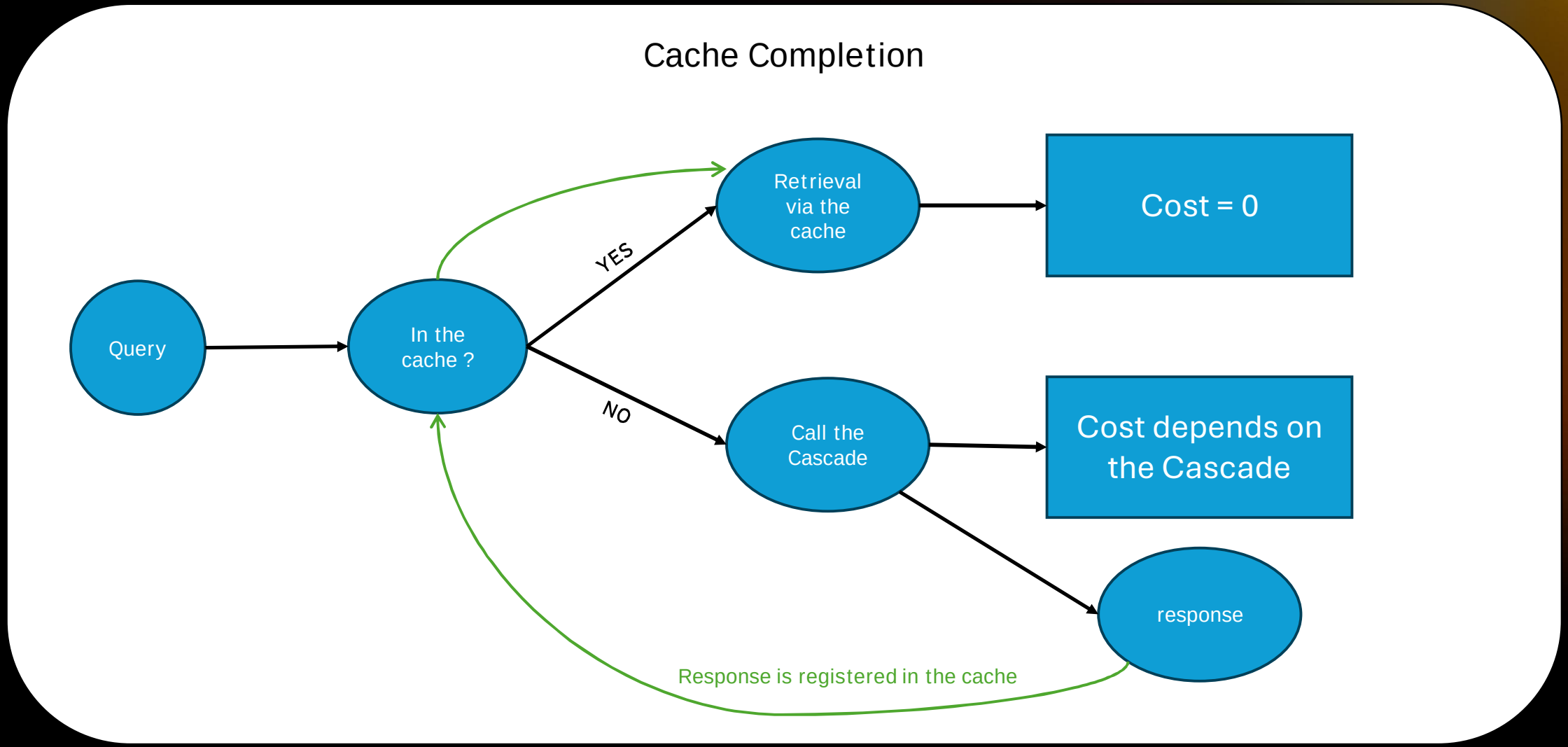
	Test_acc	Test_cost	Test_size	Train_acc	Train_cost	Train_size	Budget	Method	Provider	Marker
0	0.00	0.000000	100	0.00	0.000000	100	0.000010	FrugalGPT	FrugalGPT	1
1	0.94	0.000153	100	0.94	0.000153	100	0.000573	FrugalGPT	FrugalGPT	1
2	0.94	0.000153	100	0.94	0.000153	100	0.001127	FrugalGPT	FrugalGPT	1
3	0.94	0.000153	100	0.94	0.000153	100	0.001680	FrugalGPT	FrugalGPT	1
4	0.94	0.000153	100	0.94	0.000153	100	0.002233	FrugalGPT	FrugalGPT	1
5	0.94	0.000153	100	0.94	0.000153	100	0.002787	FrugalGPT	FrugalGPT	1
6	0.94	0.000153	100	0.94	0.000153	100	0.003340	FrugalGPT	FrugalGPT	1
7	0.94	0.000153	100	0.94	0.000153	100	0.003893	FrugalGPT	FrugalGPT	1
8	0.94	0.000153	100	0.94	0.000153	100	0.004447	FrugalGPT	FrugalGPT	1
9	0.94	0.000153	100	0.94	0.000153	100	0.005000	FrugalGPT	FrugalGPT	1
0	0.88	0.000295	100	0.90	0.000294	100	10.000000	gpt-3.5-turbo	azure_openai	1
1	0.75	0.000432	100	0.71	0.000430	100	10.000000	llama-3-70b-instruct	azure_llama	1
2	0.90	0.001477	100	0.89	0.001474	100	10.000000	gpt-4o	azure_gpt4o	1
3	0.91	0.000091	100	0.87	0.000091	100	10.000000	mistral-nemo	azure_mistral_nemo	1
4	0.77	0.000024	100	0.79	0.000024	100	10.000000	ministral	azure_ministral	1
5	0.92	0.000089	100	0.87	0.000089	100	10.000000	gpt-4o-mini	azure_gpt4o_mini	1

```
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT>
```

The cost given is the train cost for 100 queries & answers

94% of accuracy for FrugalGPT

2.2. LLM Approximation



2.2. LLM Approximation

Tech part of the Cache Completion

Architecture of the Cache

Cache JSON
Index FAISS (Facebook AI Similarity Search)

Tech used

Sentence Transformer
Index FAISS (Facebook AI Similarity Search)
Similarity Threshold

Advantages

Reduced API call costs
Time saving
Dynamic Optimization

Sentence Transformer & all-mpnet-base-v2

- Text into dense vector embeddings for semantic comparison.
- MPNet (natural language understanding)

FAISS

- O(logN) search complexity, much faster than O(N)
- Finds the closest cached response based on vector similarity.

2.3. Cost comparison w/ & w/o the cascade

```
[2025-03-19 15:12:49 +0100] [26528] [INFO] 127.0.0.1:60151 POST /chat 1.1 200 - 5972644
[2025-03-19 15:12:49 +0100] [26528] [INFO] 127.0.0.1:60151 POST /chat 1.1 - - 5972644
Cascade total cost: 5.74e-05
GPT-4o total cost: 0.00017250000000000002
[2025-03-19 15:14:45 +0100] [26528] [INFO] 127.0.0.1:60397 POST /chat 1.1 200 - 10494007
[2025-03-19 15:14:45 +0100] [26528] [INFO] 127.0.0.1:60397 POST /chat 1.1 - - 10494007
Cascade total cost: 0
GPT-4o total cost: 0.0001775
[2025-03-19 15:15:15 +0100] [26528] [INFO] 127.0.0.1:59882 POST /chat 1.1 200 - 6550885
[2025-03-19 15:15:15 +0100] [26528] [INFO] 127.0.0.1:59882 POST /chat 1.1 - - 6552162
```

```
INFO:     127.0.0.1:60152 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
---invoke new API call---
CRITICAL:root:Using service: azure_openai/gpt-3.5-turbo
CRITICAL:root:Using service: azure_openai/gpt-3.5-turbo
---invoke new API call---
azure_openai/gpt-3.5-turbo
INFO:     127.0.0.1:60398 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:60398 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
INFO:     127.0.0.1:59883 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:59883 - "POST /execute_gpt4o HTTP/1.1" 200 OK
```

Legal Copilot — 127.0.0.1:50505

EY Azure GPT — Chat — Evaluate

Clear chat — Developer settings

What is the capital of Spain ?

The capital of Spain is Madrid.

Did you like this response?

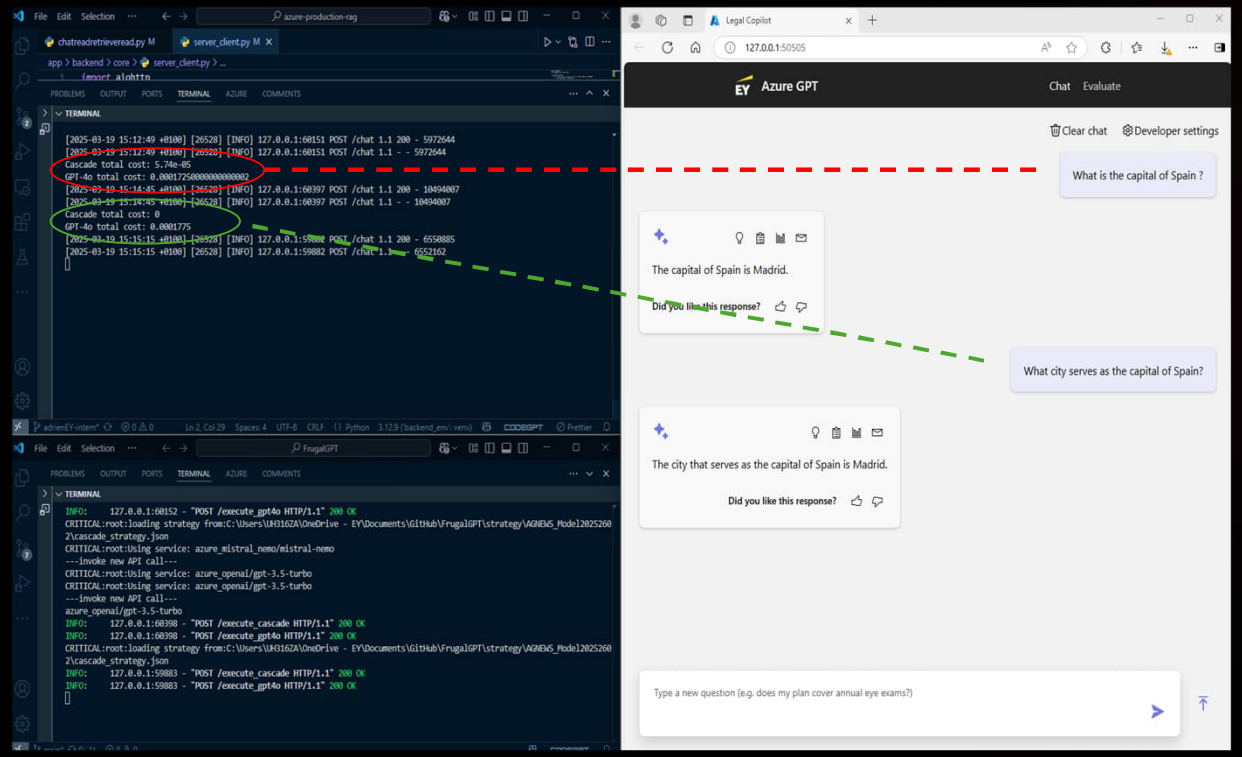
What city serves as the capital of Spain?

The city that serves as the capital of Spain is Madrid.

Did you like this response?

Type a new question (e.g. does my plan cover annual eye exams?)

2.3. Cost comparison w/ & w/o the cascade



Cost of the query and the answer :

- With the cascade :
  - 0.0000574$
- With GPT4o :
  - 0.000175$
- With the cache :
  - 0$ (for a similar question)

Cost reduction of 66% for the given query

2.3. Cost comparison w/ & w/o the cascade

```
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /auth_setup 1.1 200 386 2067
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /auth_setup 1.1 - - 2941
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /.auth/refresh 1.1 404 207 999
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /.auth/refresh 1.1 - - 999
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /config 1.1 200 83 503
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /config 1.1 - - 1765
Cascade total cost: 0.000225
GPT-4o total cost: 0.002365
[2025-03-25 10:34:38 +0100] [21748] [INFO] 127.0.0.1:50960 POST /chat 1.1 200 - 16559797
[2025-03-25 10:34:38 +0100] [21748] [INFO] 127.0.0.1:50960 POST /chat 1.1 - - 16559797
Cascade total cost: 0
GPT-4o total cost: 0.002345
[2025-03-25 10:36:21 +0100] [21748] [INFO] 127.0.0.1:51091 POST /chat 1.1 200 - 8359297
[2025-03-25 10:36:21 +0100] [21748] [INFO] 127.0.0.1:51091 POST /chat 1.1 - - 8360027
```

```
INFO:     Started server process [18804]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
INFO:     127.0.0.1:50752 - "POST /execute_cascade HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
INFO:     127.0.0.1:50989 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:50989 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
INFO:     127.0.0.1:51095 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:51095 - "POST /execute_gpt4o HTTP/1.1" 200 OK
```

Legal Copilot — 127.0.0.1:50505

Azure GPT — Chat — Evaluate

Clear chat — Developer settings

What is Northwind Health Plus?

Northwind Health Plus is a program offered by Northwind Health that provides personal health support programs to help individuals get the most out of their health benefits. The program includes access to Care Coordinators, Health Coaches, Care Managers, Health Educators, and Specialists to support individuals in managing their health [1]. Northwind Health is responsible for paying claims correctly by determining eligibility, obtaining pre-authorization for services when required, and paying for eligible services based on the terms of the plan. They also provide an explanation of benefits (EOB) after processing a claim, outlining the services provided and the amounts paid [2].

Citations: 1. Northwind_Standard_Benefits_Details.pdf#page=69
2. Northwind_Standard_Benefits_Details.pdf#page=99

Did you like this response?

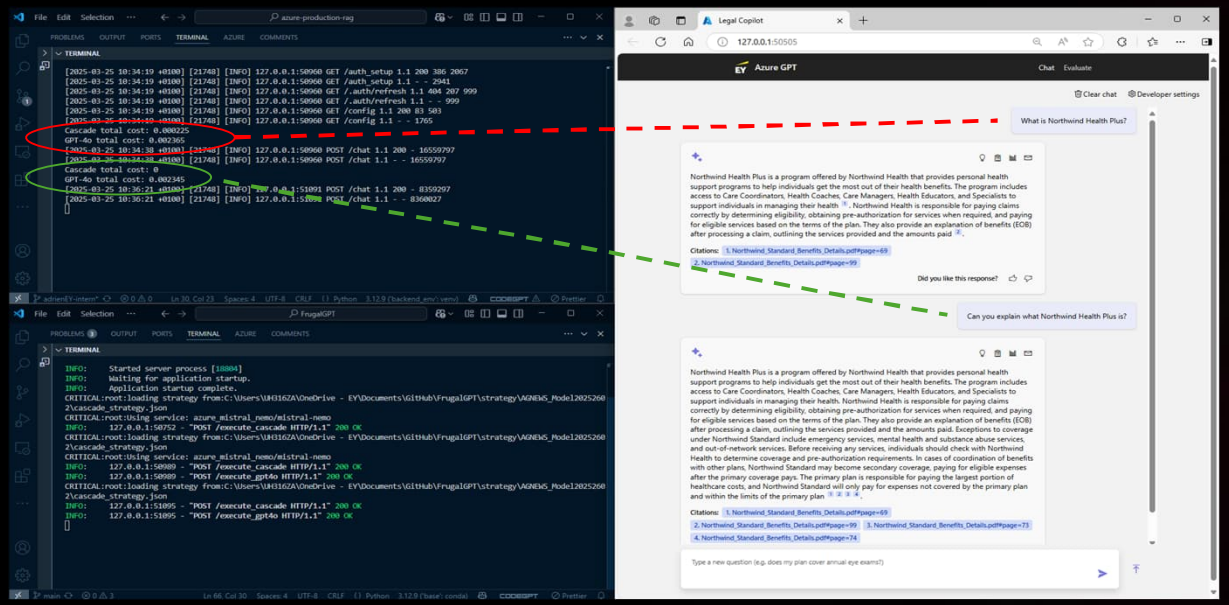
Can you explain what Northwind Health Plus is?

Northwind Health Plus is a program offered by Northwind Health that provides personal health support programs to help individuals get the most out of their health benefits. The program includes access to Care Coordinators, Health Coaches, Care Managers, Health Educators, and Specialists to support individuals in managing their health. Northwind Health is responsible for paying claims correctly by determining eligibility, obtaining pre-authorization for services when required, and paying for eligible services based on the terms of the plan. They also provide an explanation of benefits (EOB) after processing a claim, outlining the services provided and the amounts paid. Exceptions to coverage under Northwind Standard include emergency services, mental health and substance abuse services, and out-of-network services. Before receiving any services, individuals should check with Northwind Health to determine coverage and pre-authorization requirements. In cases of coordination of benefits with other plans, Northwind Standard may become secondary coverage, paying for eligible expenses after the primary coverage pays. The primary plan is responsible for paying the largest portion of healthcare costs, and Northwind Standard will only pay for expenses not covered by the primary plan and within the limits of the primary plan [1] [2] [3] [4].

Citations: 1. Northwind_Standard_Benefits_Details.pdf#page=69
2. Northwind_Standard_Benefits_Details.pdf#page=99
3. Northwind_Standard_Benefits_Details.pdf#page=73
4. Northwind_Standard_Benefits_Details.pdf#page=74

Type a new question (e.g. does my plan cover annual eye exams?)

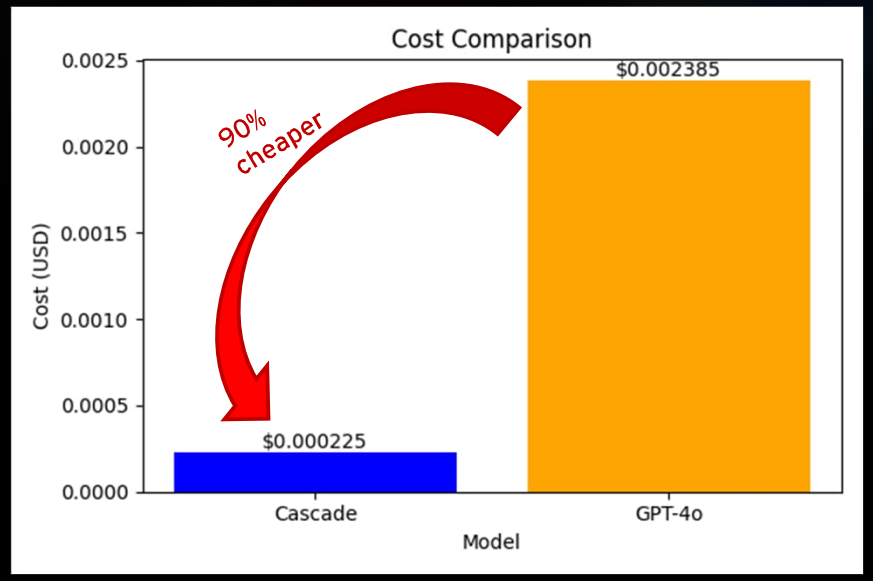
2.3. Cost comparison w/ & w/o the cascade



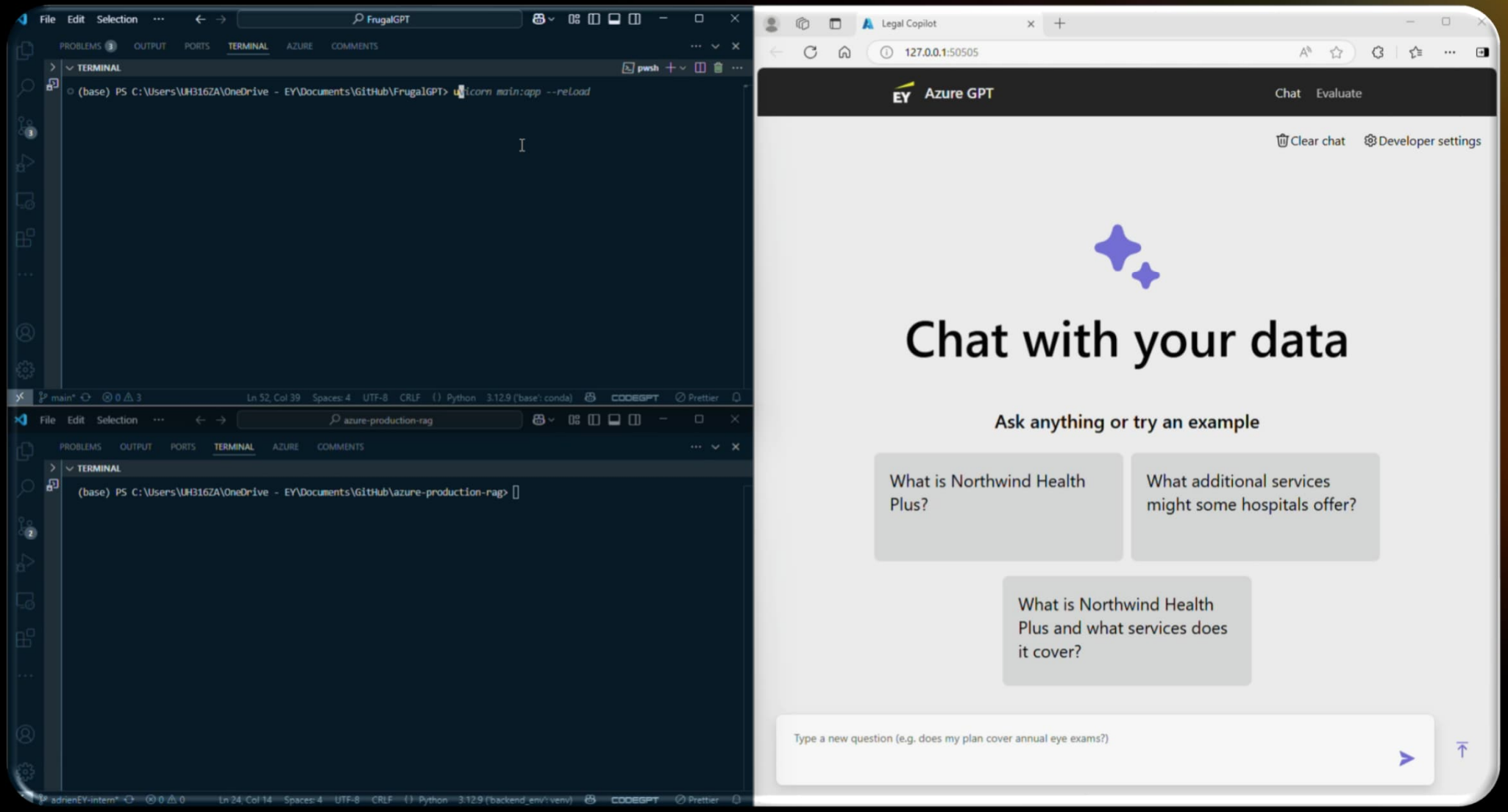
Cost of the query and the answer :

- With the cascade :
  - 0.000225$
- With GPT4o :
  - 0.002385$
- With the cache :
  - 0$ (for a similar question)

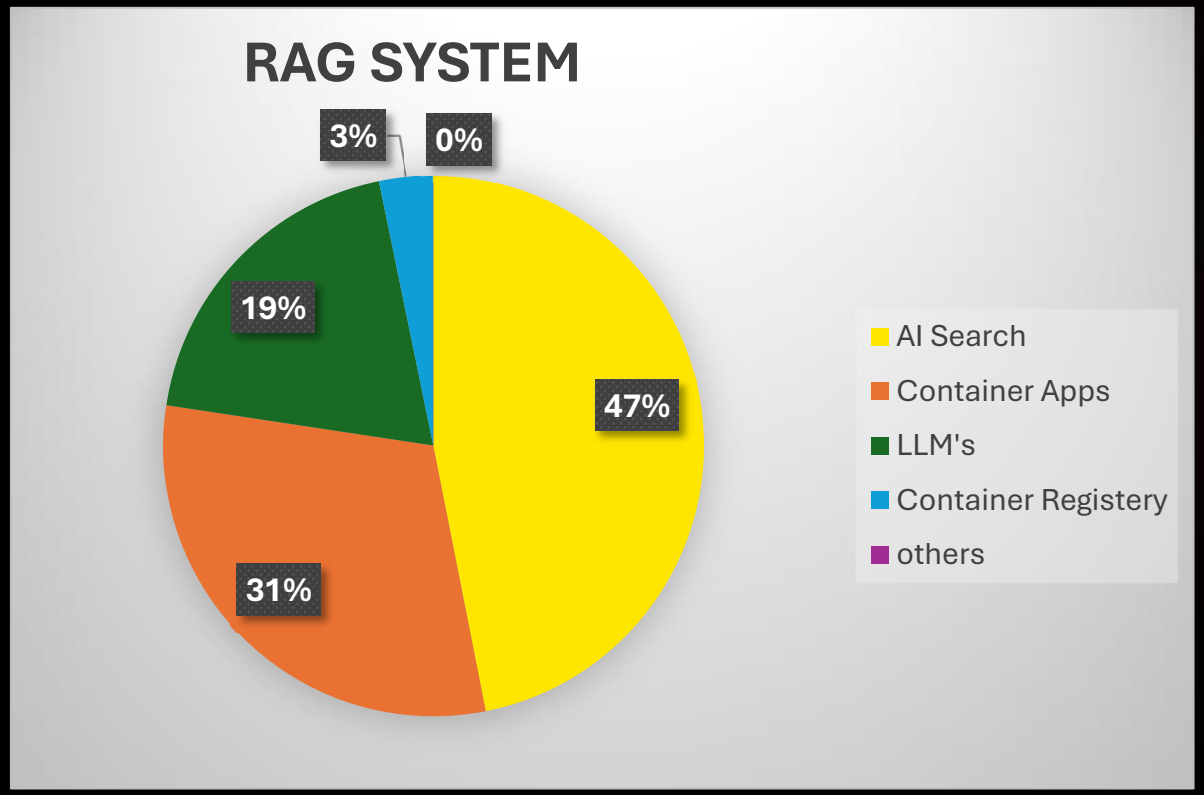
Cost reduction of 90% for the given query



3. DEMO


File Edit Selection
FrugalGPT
PROBLEMS OUTPUT PORTS TERMINAL AZURE COMMENTS
TERMINAL
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT> uvicorn main:app --reload
azure-production-rag
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\azure-production-rag>
Legal Copilot
127.0.0.1:50505
Azure GPT
Chat
Evaluate
Clear chat
Developer settings
Chat with your data
Ask anything or try an example
What is Northwind Health Plus?
What additional services might some hospitals offer?
What is Northwind Health Plus and what services does it cover?
Type a new question (e.g. does my plan cover annual eye exams?)

4. Cost Analysis



Impact on Total Expenses

- **Context**: LLMs account for **20% of the total estimated cost of $130**.
- **Potential Optimization**: Cost reduction of LLMs by **50 to 90%**.
- **Impact on Total Cost**: This would lead to a **maximum reduction of 10 to 15%**. (13 to 23$)
- **Other Factors to Consider**: Number of documents for search, query frequency, infrastructure architecture…

Week 10
In-depth testing of the beta version

The better the question. The better the answer. The better the world works.

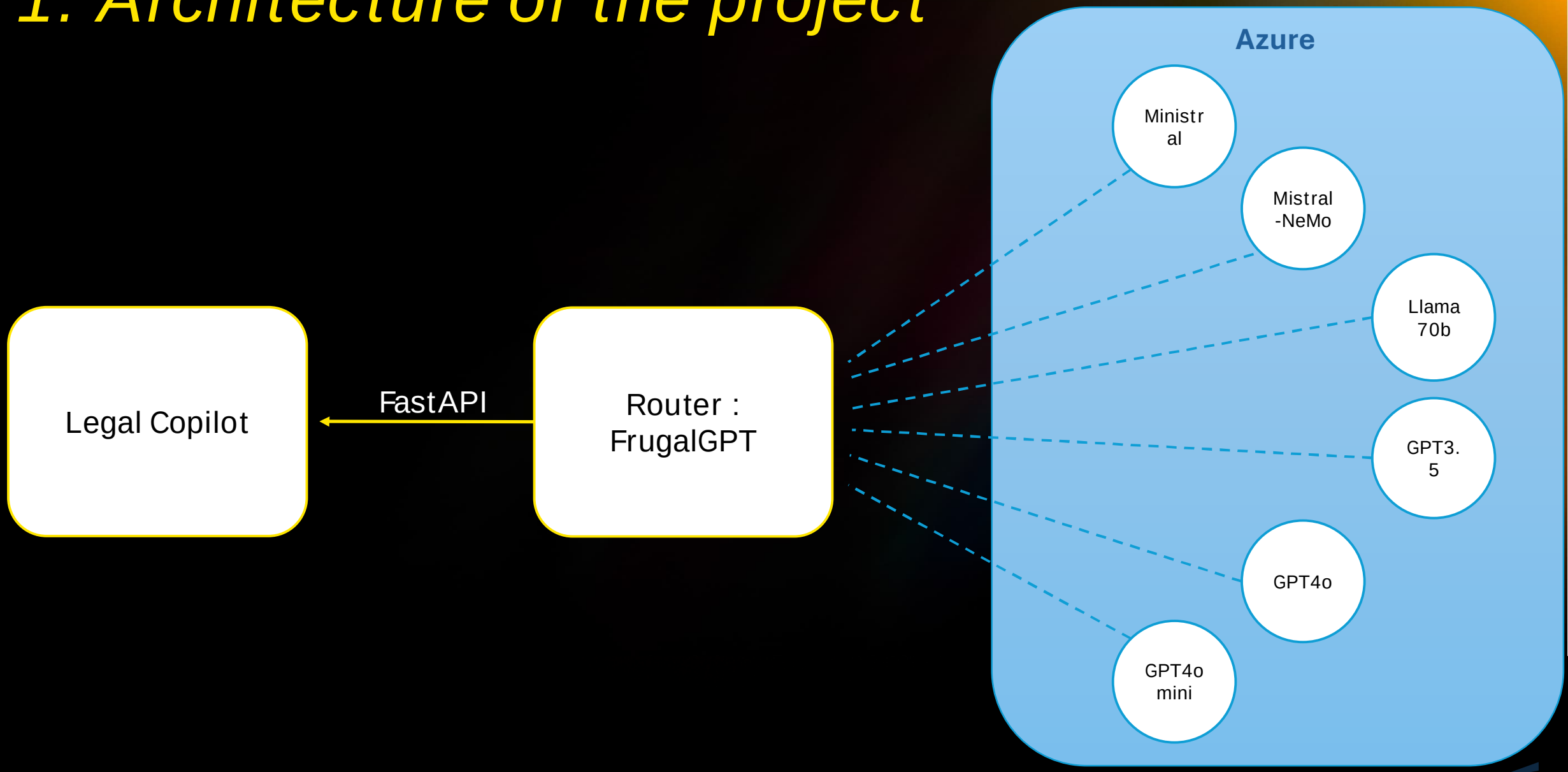
Shape the future with confidence

Contents

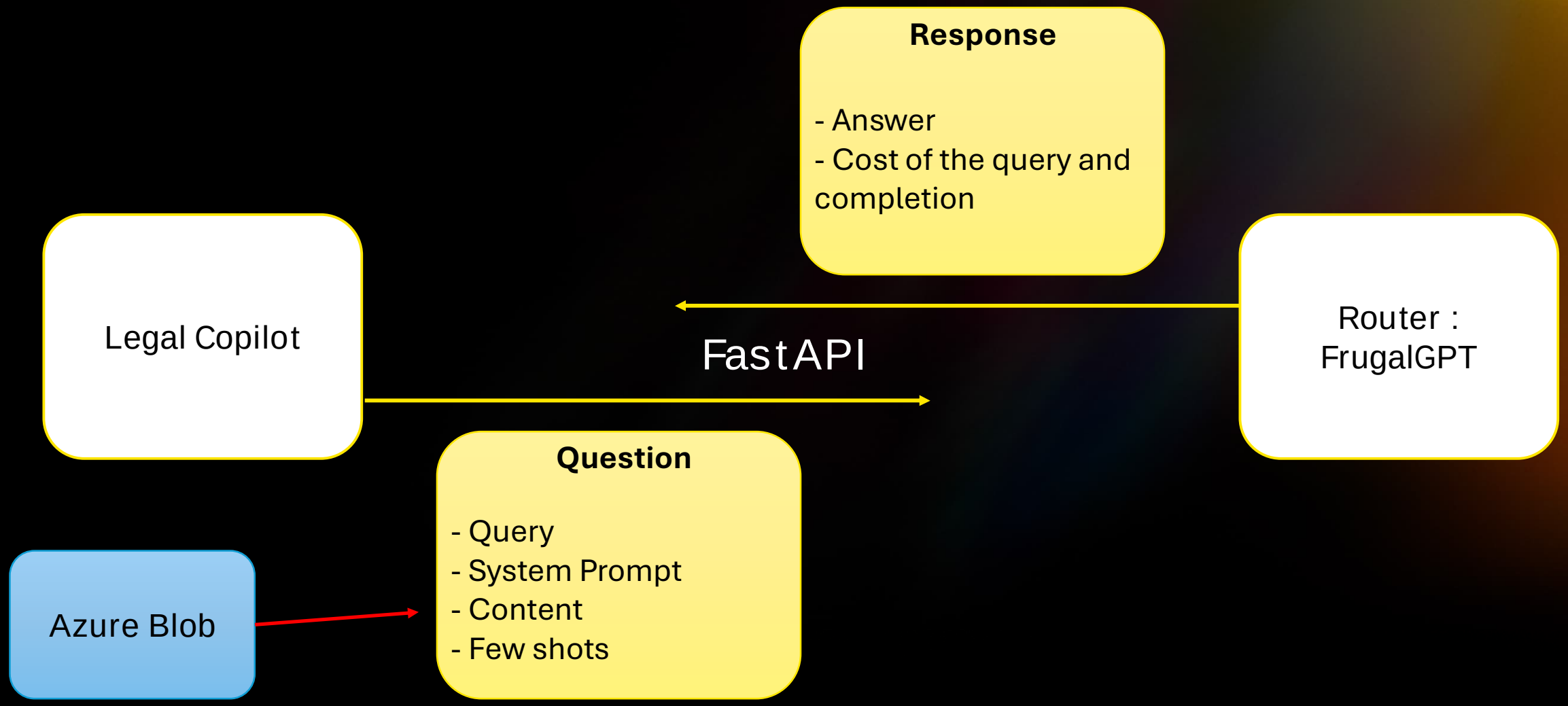
FrugalGPT as a Router

1. Architecture of the project
2. FrugalGPT
    - 2.1 LLM Cascade
    - 2.2 LLM Approximation
    - 2.3 Cost comparison w/ & w/o the cascade
3. Demo
4. Cost Analysis
5. What’s Next
6. FrugalGPT into SFPIM

1. Architecture of the project



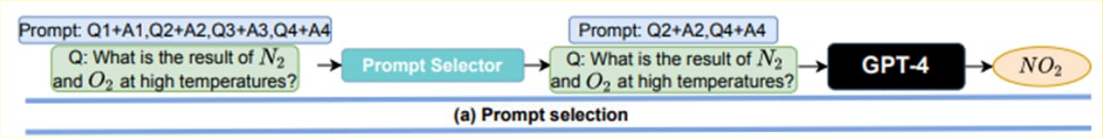
1. Architecture of the project



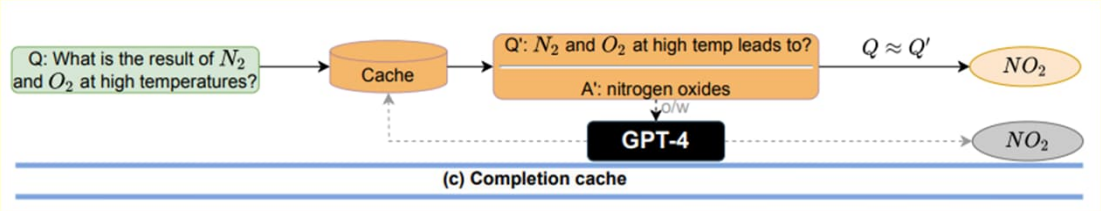
2. FrugalGPT

FrugalGPT

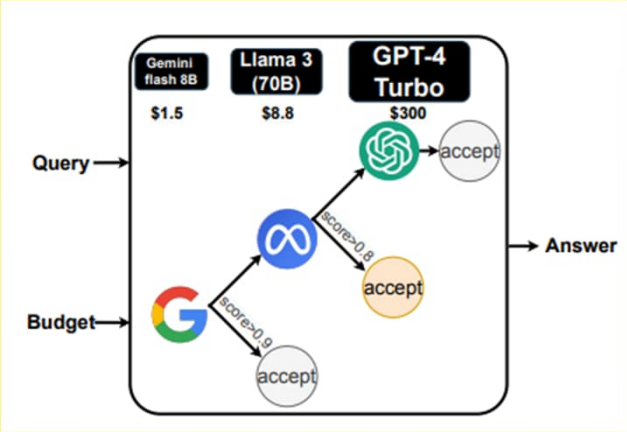
Prompt Adaptation



LLM Approximation



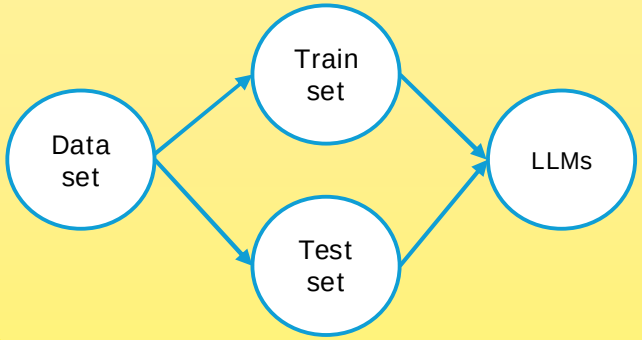
LLM Cascade



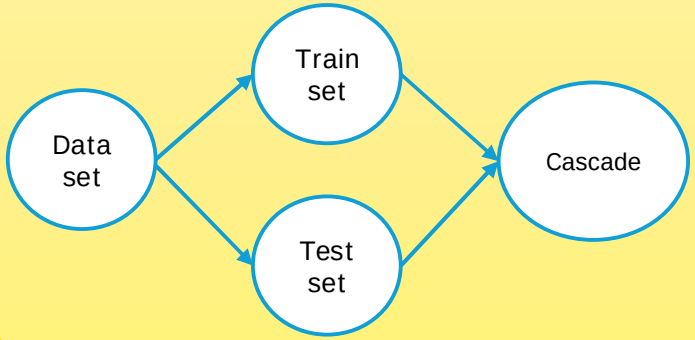
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



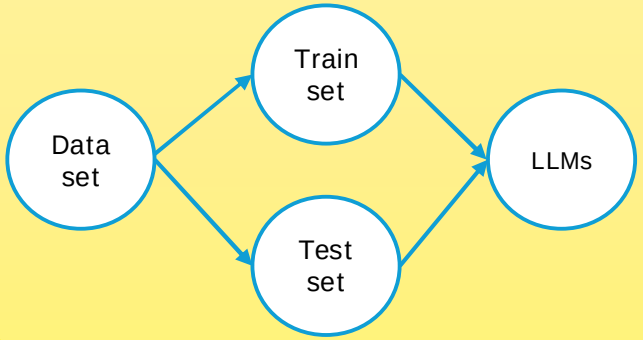
Cascade

1. Optimal strategy by testing LLM permutations that meet budget and threshold constraints.
2. Derives a strategy that aligns with the various cost requirements.
3. Reading the cascade to respond to a new query.
4. BERT used to evaluate the response quality checking if the threshold is exceeded. If so, the response is retained.

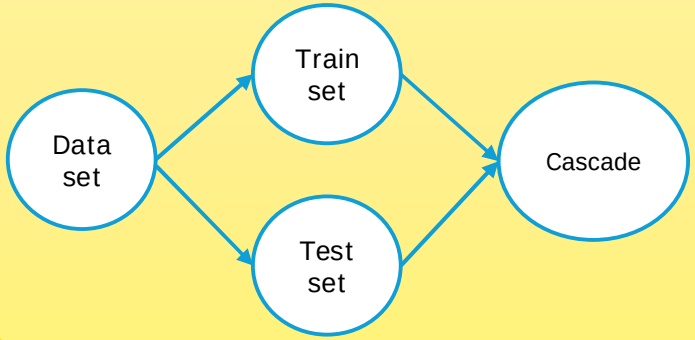
2.1. LLM Cascade

LLM Cascade

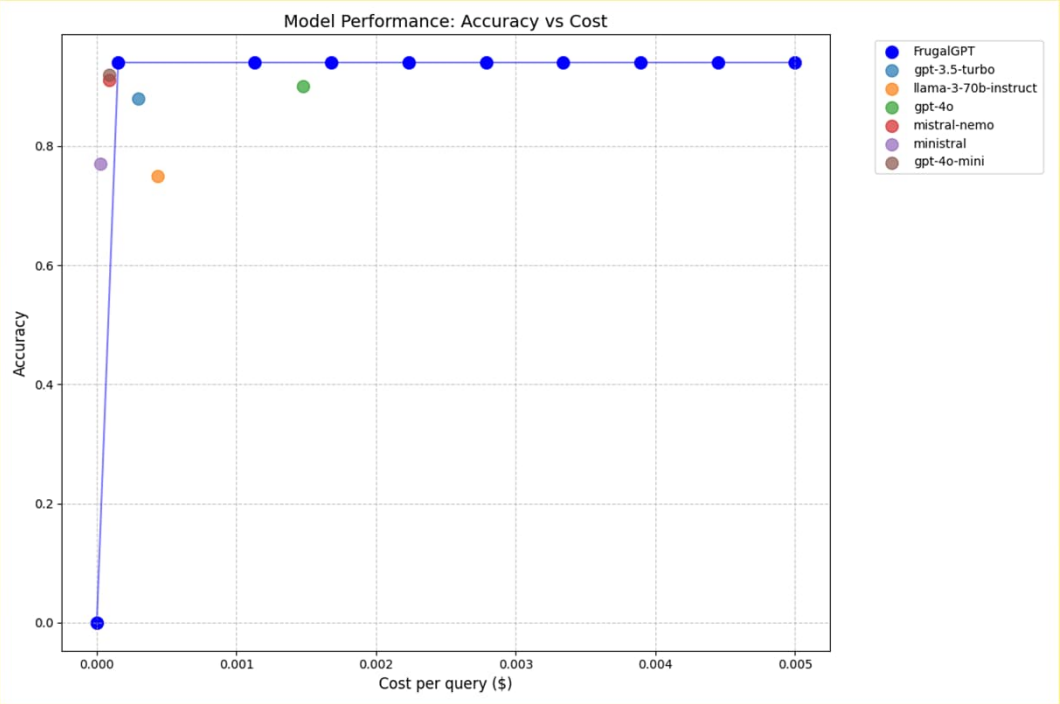
Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



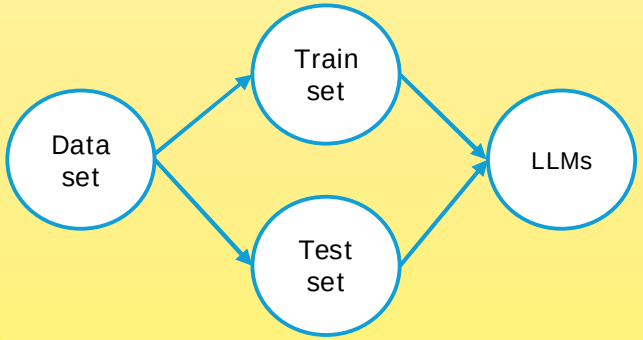
Cascade



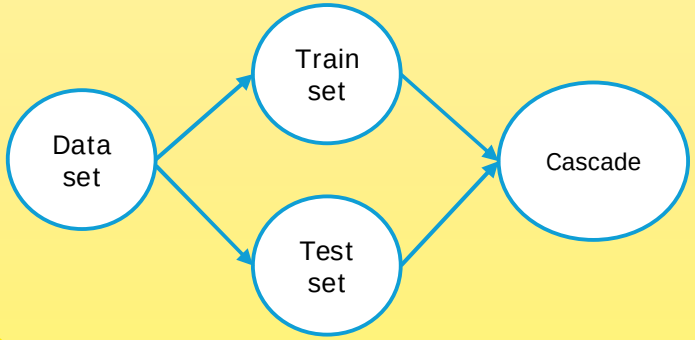
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



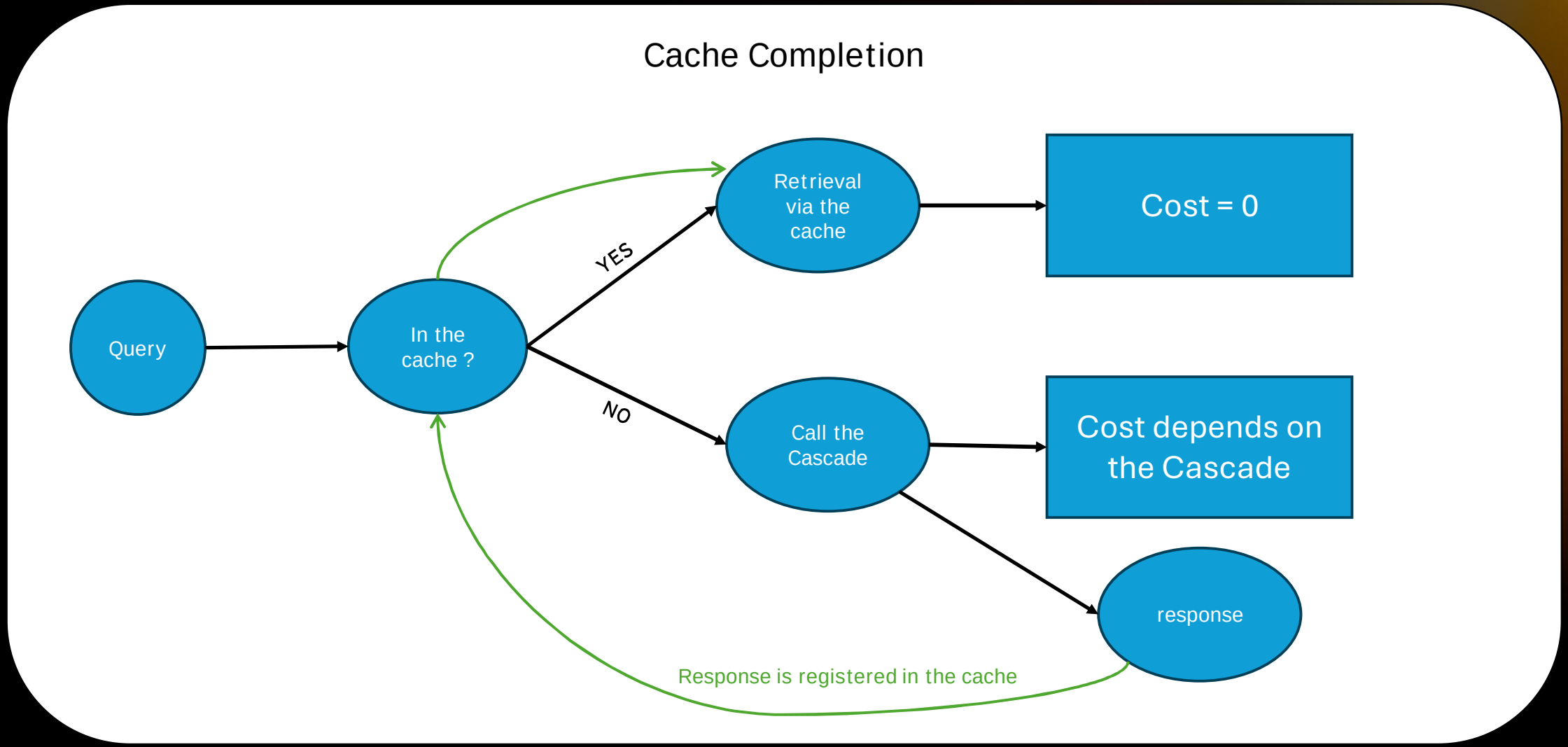
	Test_acc	Test_cost	Test_size	Train_acc	Train_cost	Train_size	Budget	Method	Provider	Marker
0	0.00	0.000000	100	0.00	0.000000	100	0.000010	FrugalGPT	FrugalGPT	1
1	0.94	0.000153	100	0.94	0.000153	100	0.000573	FrugalGPT	FrugalGPT	1
2	0.94	0.000153	100	0.94	0.000153	100	0.001127	FrugalGPT	FrugalGPT	1
3	0.94	0.000153	100	0.94	0.000153	100	0.001680	FrugalGPT	FrugalGPT	1
4	0.94	0.000153	100	0.94	0.000153	100	0.002233	FrugalGPT	FrugalGPT	1
5	0.94	0.000153	100	0.94	0.000153	100	0.002787	FrugalGPT	FrugalGPT	1
6	0.94	0.000153	100	0.94	0.000153	100	0.003340	FrugalGPT	FrugalGPT	1
7	0.94	0.000153	100	0.94	0.000153	100	0.003893	FrugalGPT	FrugalGPT	1
8	0.94	0.000153	100	0.94	0.000153	100	0.004447	FrugalGPT	FrugalGPT	1
9	0.94	0.000153	100	0.94	0.000153	100	0.005000	FrugalGPT	FrugalGPT	1
0	0.88	0.000295	100	0.90	0.000294	100	10.000000	gpt-3.5-turbo	azure_openai	1
1	0.75	0.000432	100	0.71	0.000430	100	10.000000	llama-3-70b-instruct	azure_llama	1
2	0.90	0.001477	100	0.89	0.001474	100	10.000000	gpt-4o	azure_gpt4o	1
3	0.91	0.000091	100	0.87	0.000091	100	10.000000	mistral-nemo	azure_mistral_nemo	1
4	0.77	0.000024	100	0.79	0.000024	100	10.000000	ministral	azure_ministral	1
5	0.92	0.000089	100	0.87	0.000089	100	10.000000	gpt-4o-mini	azure_gpt4o_mini	1

```
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT>
```

The cost given is the train cost for 100 queries & answers

94% of accuracy for FrugalGPT

2.2. LLM Approximation



2.2. LLM Approximation

Tech part of the Cache Completion

Architecture of the Cache

Cache JSON
Index FAISS (Facebook AI Similarity Search)

Tech used

Sentence Transformer
Index FAISS (Facebook AI Similarity Search)
Similarity Threshold

Advantages

Reduced API call costs
Time saving
Dynamic Optimization

Sentence Transformer & all-mpnet-base-v2

- Text into dense vector embeddings for semantic comparison.
- MPNet (natural language understanding)

FAISS

- O(logN) search complexity, much faster than O(N)
- Finds the closest cached response based on vector similarity.

EY

2.3. Cost comparison w/ & w/o the cascade

```
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /auth_setup 1.1 200 386 2067
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /auth_setup 1.1 - - 2941
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /.auth/refresh 1.1 404 207 999
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /.auth/refresh 1.1 - - 999
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /config 1.1 200 83 503
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /config 1.1 - - 1765
Cascade total cost: 0.000225
GPT-4o total cost: 0.002365
[2025-03-25 10:34:38 +0100] [21748] [INFO] 127.0.0.1:50960 POST /chat 1.1 200 - 16559797
[2025-03-25 10:34:38 +0100] [21748] [INFO] 127.0.0.1:50960 POST /chat 1.1 - - 16559797
Cascade total cost: 0
GPT-4o total cost: 0.002345
[2025-03-25 10:36:21 +0100] [21748] [INFO] 127.0.0.1:51091 POST /chat 1.1 200 - 8359297
[2025-03-25 10:36:21 +0100] [21748] [INFO] 127.0.0.1:51091 POST /chat 1.1 - - 8360027
```

```
INFO:     Started server process [18804]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
INFO:     127.0.0.1:50752 - "POST /execute_cascade HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
INFO:     127.0.0.1:50989 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:50989 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
INFO:     127.0.0.1:51095 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:51095 - "POST /execute_gpt4o HTTP/1.1" 200 OK
```

Legal Copilot — 127.0.0.1:50505

Azure GPT — Chat — Evaluate

Clear chat — Developer settings

What is Northwind Health Plus?

Northwind Health Plus is a program offered by Northwind Health that provides personal health support programs to help individuals get the most out of their health benefits. The program includes access to Care Coordinators, Health Coaches, Care Managers, Health Educators, and Specialists to support individuals in managing their health [1]. Northwind Health is responsible for paying claims correctly by determining eligibility, obtaining pre-authorization for services when required, and paying for eligible services based on the terms of the plan. They also provide an explanation of benefits (EOB) after processing a claim, outlining the services provided and the amounts paid [2].

Citations: 1. Northwind_Standard_Benefits_Details.pdf#page=69
2. Northwind_Standard_Benefits_Details.pdf#page=99

Did you like this response?

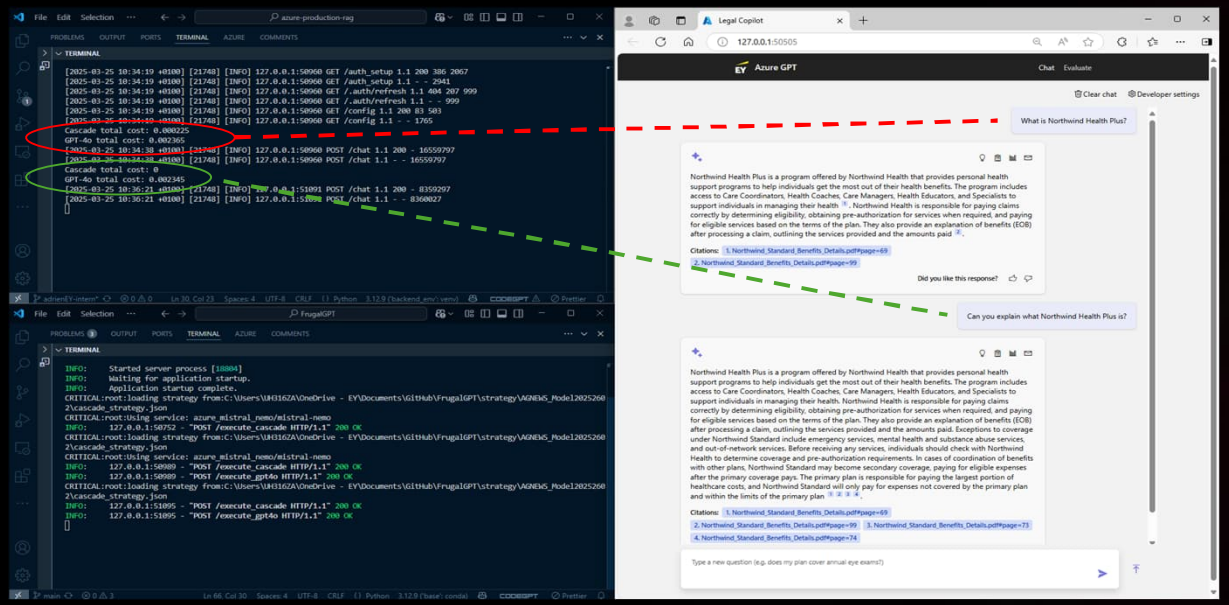
Can you explain what Northwind Health Plus is?

Northwind Health Plus is a program offered by Northwind Health that provides personal health support programs to help individuals get the most out of their health benefits. The program includes access to Care Coordinators, Health Coaches, Care Managers, Health Educators, and Specialists to support individuals in managing their health. Northwind Health is responsible for paying claims correctly by determining eligibility, obtaining pre-authorization for services when required, and paying for eligible services based on the terms of the plan. They also provide an explanation of benefits (EOB) after processing a claim, outlining the services provided and the amounts paid. Exceptions to coverage under Northwind Standard include emergency services, mental health and substance abuse services, and out-of-network services. Before receiving any services, individuals should check with Northwind Health to determine coverage and pre-authorization requirements. In cases of coordination of benefits with other plans, Northwind Standard may become secondary coverage, paying for eligible expenses after the primary coverage pays. The primary plan is responsible for paying the largest portion of healthcare costs, and Northwind Standard will only pay for expenses not covered by the primary plan and within the limits of the primary plan [1] [2] [3] [4].

Citations: 1. Northwind_Standard_Benefits_Details.pdf#page=69
2. Northwind_Standard_Benefits_Details.pdf#page=99
3. Northwind_Standard_Benefits_Details.pdf#page=73
4. Northwind_Standard_Benefits_Details.pdf#page=74

Type a new question (e.g. does my plan cover annual eye exams?)

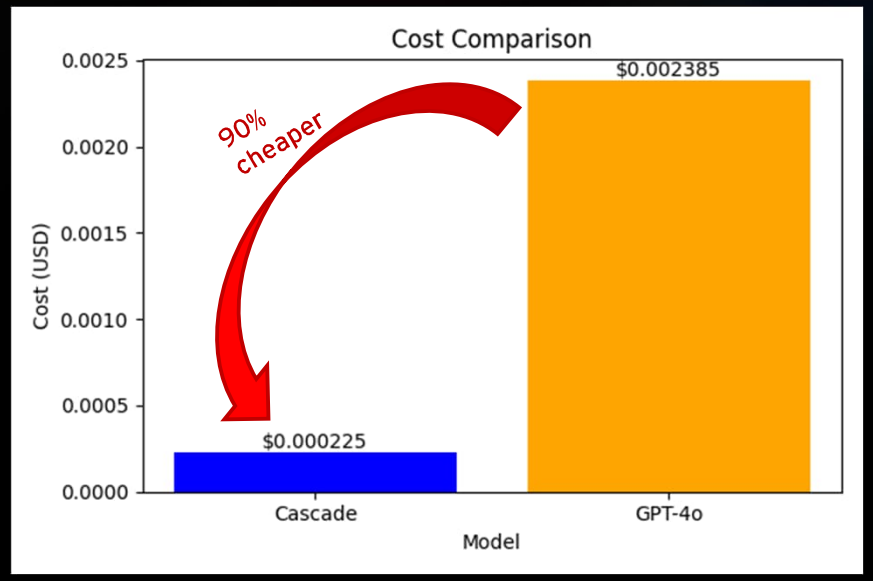
2.3. Cost comparison w/ & w/o the cascade



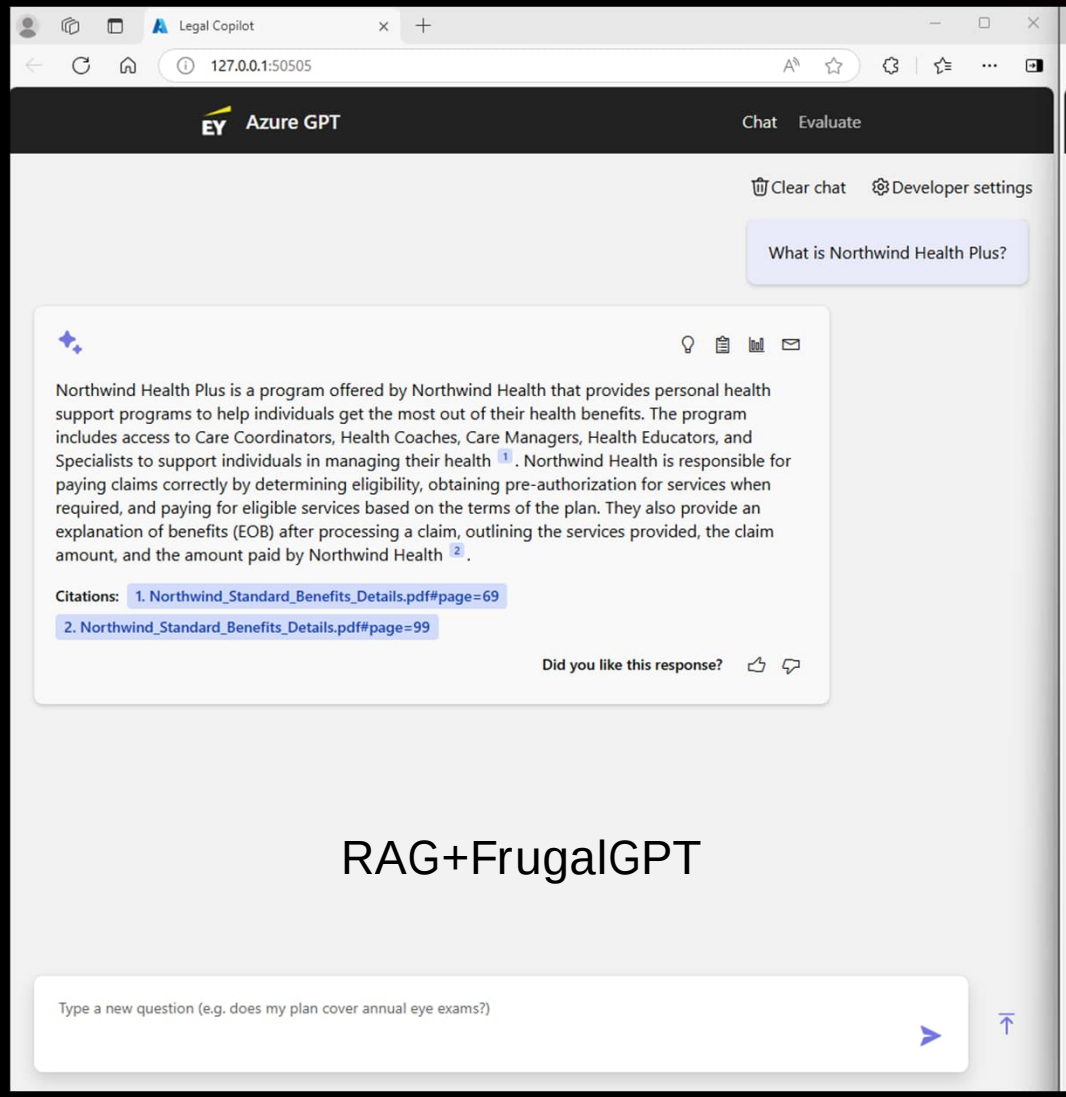
Cost of the query and the answer :

- With the cascade :
  - 0.000225$
- With GPT4o :
  - 0.002385$
- With the cache :
  - 0$ (for a similar question)

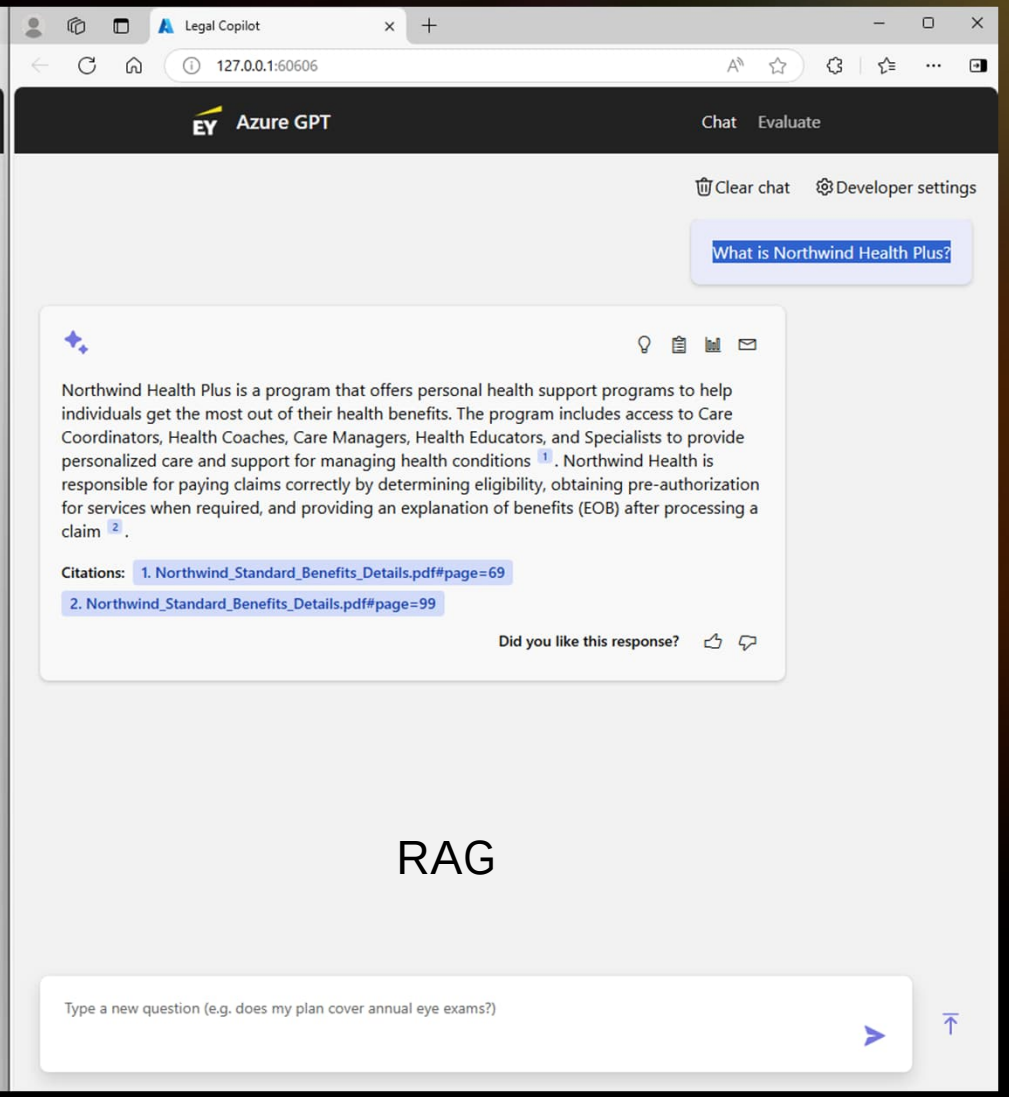
Cost reduction of 90% for the given query



2.4 RAG+FrugalGPT vs RAG

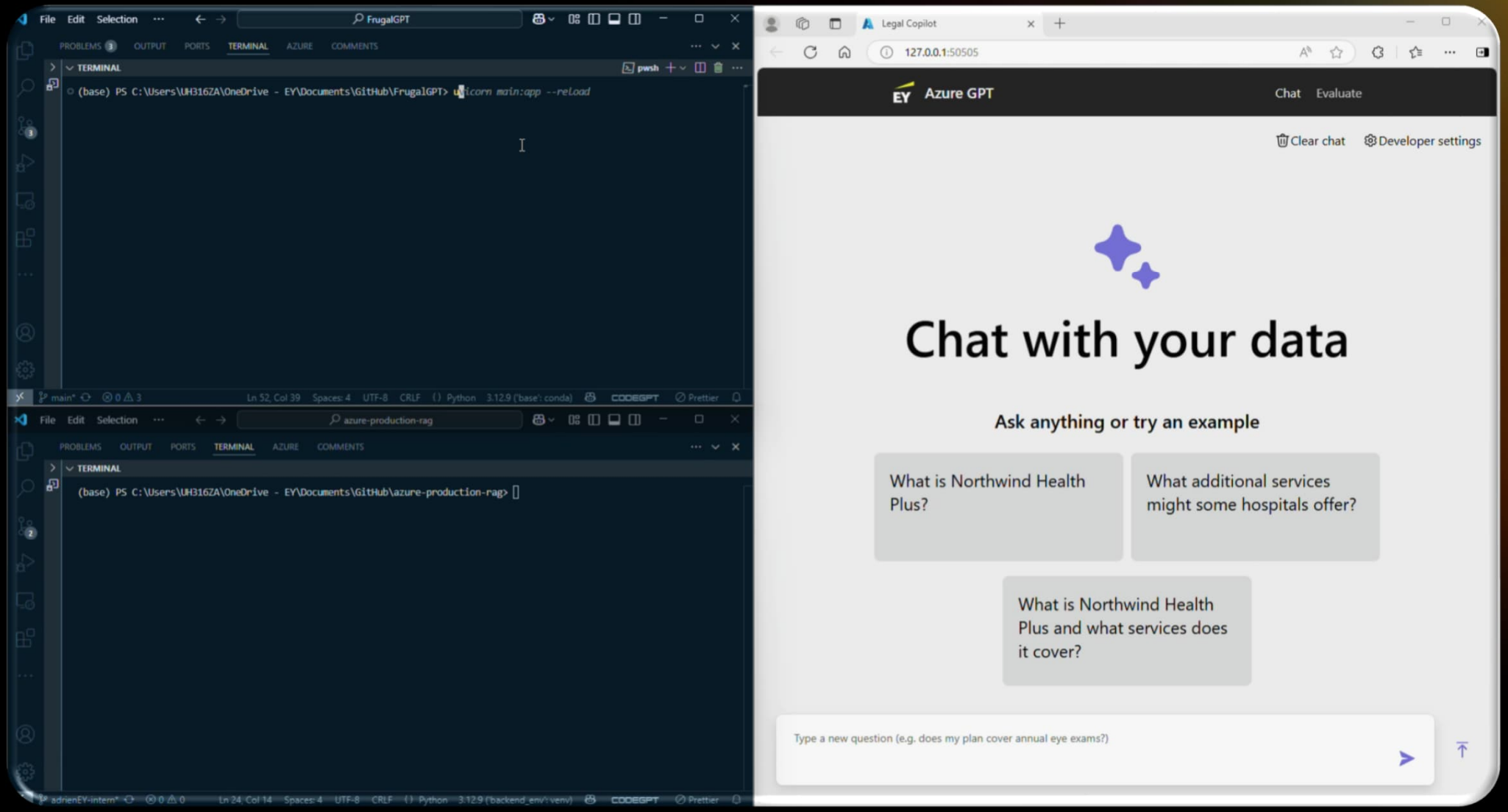


RAG+FrugalGPT

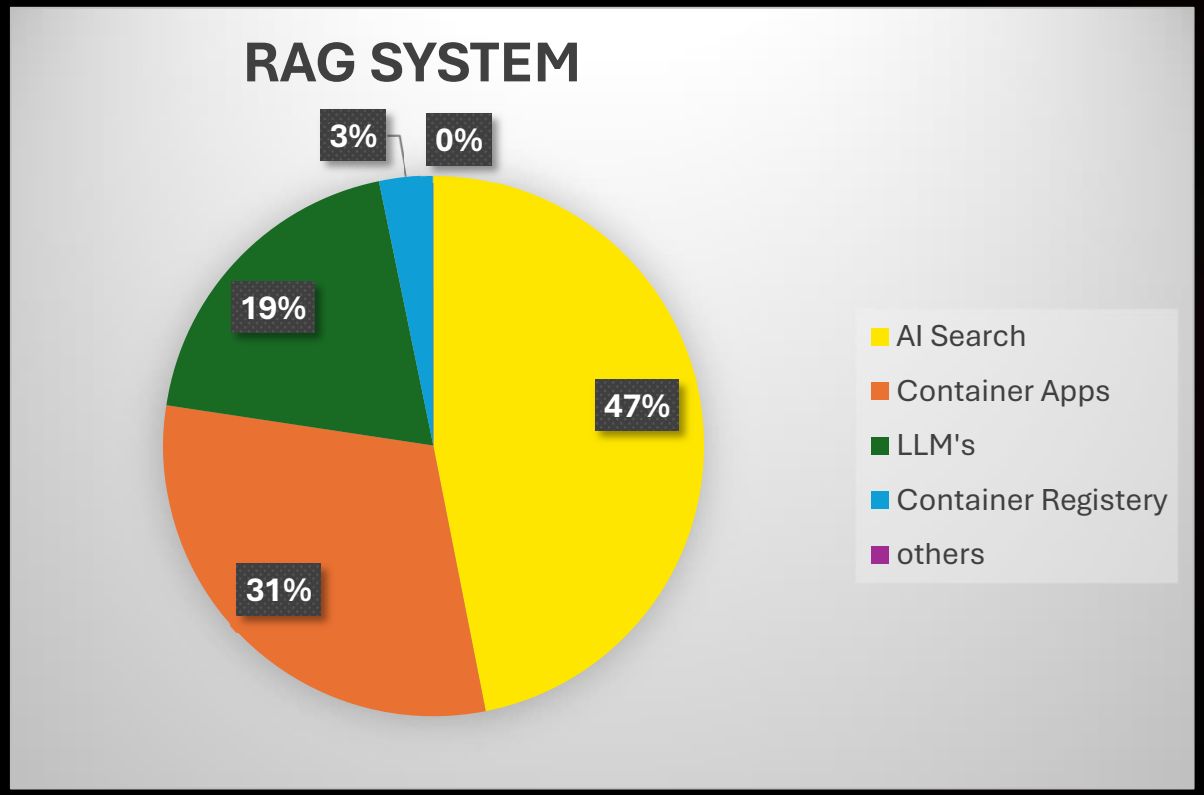


RAG

3. Demo


File Edit Selection
FrugalGPT
PROBLEMS OUTPUT PORTS TERMINAL AZURE COMMENTS
TERMINAL
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT> uvicorn main:app --reload
azure-production-rag
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\azure-production-rag>
Legal Copilot
127.0.0.1:50505
Azure GPT
Chat
Evaluate
Clear chat
Developer settings
Chat with your data
Ask anything or try an example
What is Northwind Health Plus?
What additional services might some hospitals offer?
What is Northwind Health Plus and what services does it cover?
Type a new question (e.g. does my plan cover annual eye exams?)

4. Cost Analysis



Impact on Total Expenses

- **Context**: LLMs account for **20% of the total estimated cost of $130**.
- **Potential Optimization**: Cost reduction of LLMs by **50 to 90%**.
- **Impact on Total Cost**: This would lead to a **maximum reduction of 10 to 15%**. (13 to 23$)
- **Other Factors to Consider**: Number of documents for search, query frequency, infrastructure architecture…

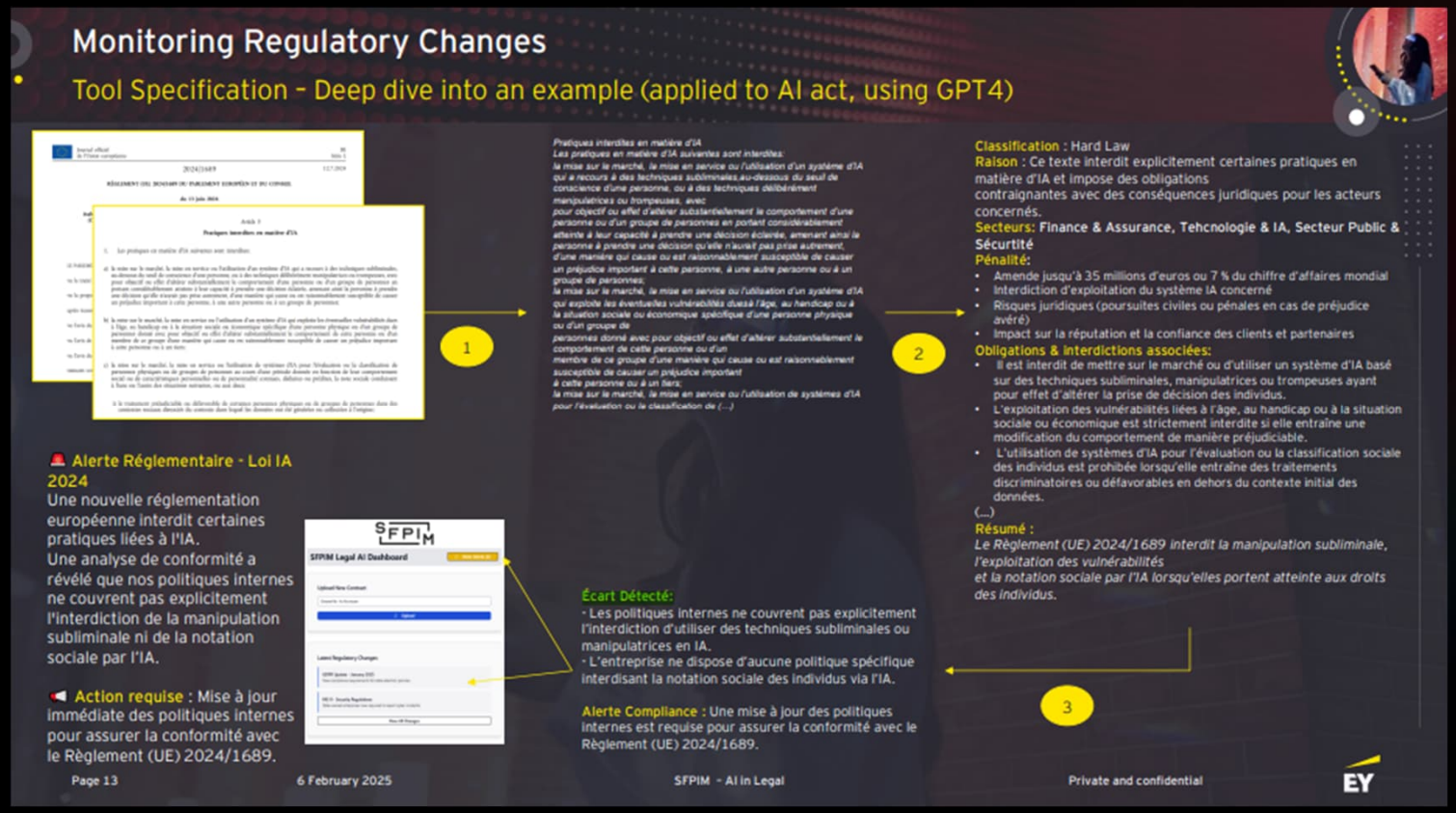
5. What’s Next

- 1. Alternatives to Azure AI Search due to High Costs
  - Given the current costs, it would be beneficial to explore other options in order to minimize the expenses associated with using Azure AI Search, which currently accounts for 47% of our total costs. Several alternatives are available, such as Weaviate, Elasticsearch, and Redisearch. It is worth evaluating whether these solutions would meet the client's needs.

- 2. Securing User Input Prompt and Output Filtering (OWASP)
  - A part of this has already been implemented with the query prompt template.

```
query_prompt_template = """Below is a history of the conversation so far, and a new question asked by the user that needs to be answered by searching in a knowledge base.
You have access to Azure AI Search index with 100's of documents.
Generate a search query based on the conversation and the new question.
Do not include cited source filenames and document names e.g info.txt or doc.pdf in the search query terms.
Do not include any text inside [] or <<>> in the search query terms.
Do not include any special characters like '+'.
If the question is not in English, translate the question to English before generating the search query.
If you cannot generate a search query, return just the number 0.
"""
```

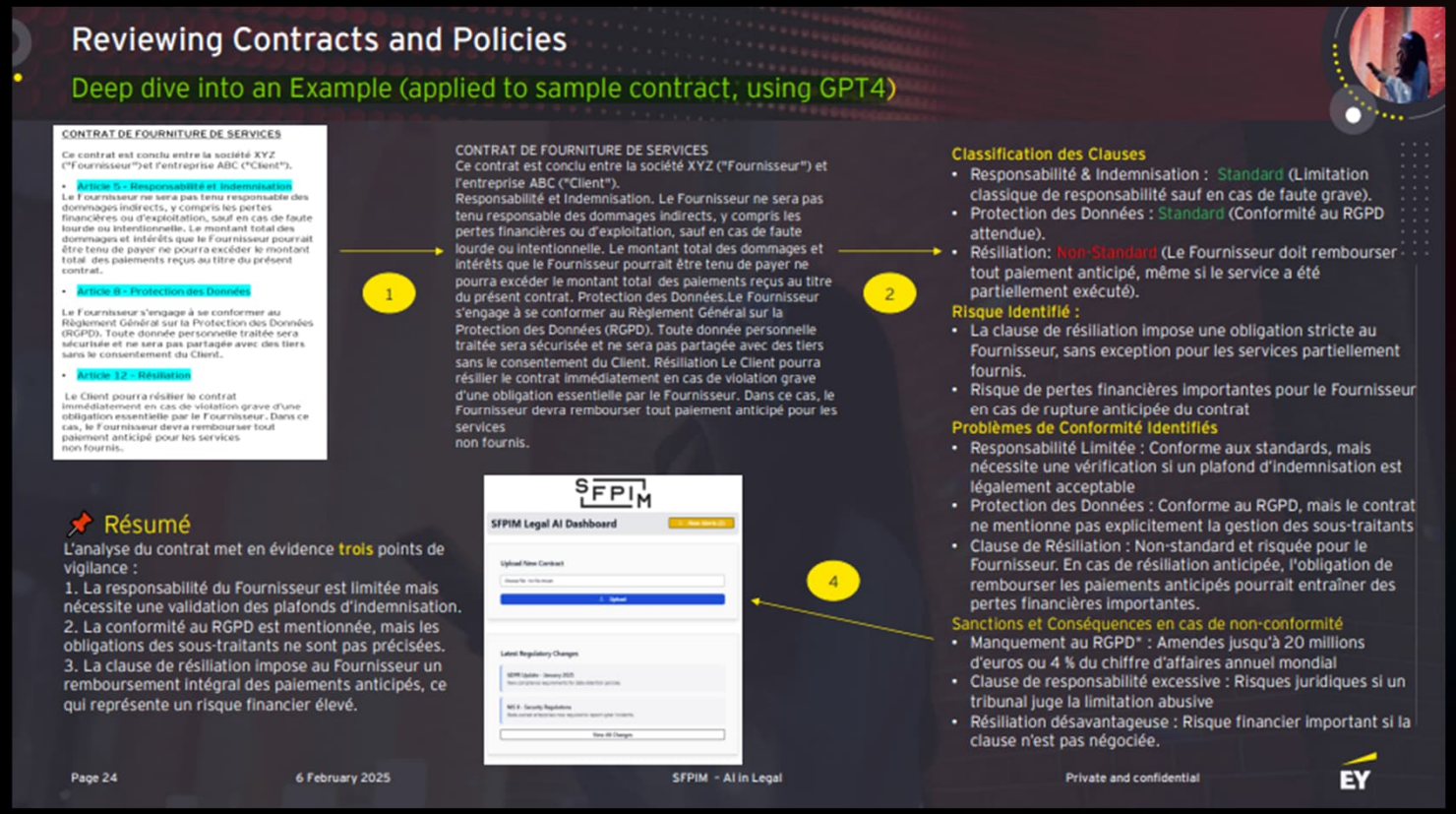
6. FrugalGPT into SFPIM



STEPS :

- Data generated by GPT-4 to create a ground truth dataset
- Enables evaluation of response accuracy for different LLMs
- Creation of a cascade with the optimal strategy to adapt to policy changes
- With a cache solution, the model can store the regulation internally to avoid repeating the process

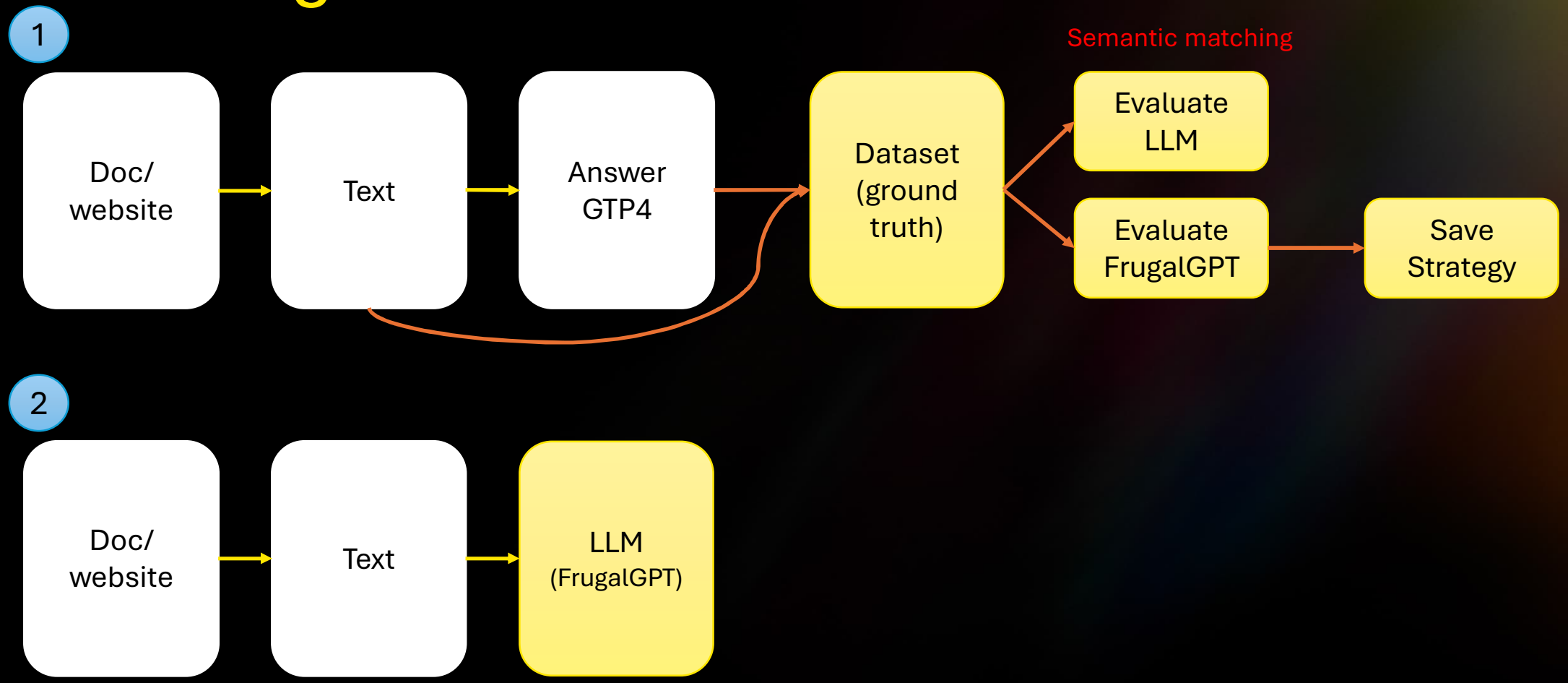
6. FrugalGPT into SFPIM



STEPS :

- Data generated by GPT-4 to create a ground truth dataset
- Enables evaluation of response accuracy for different LLMs
- Creation of a cascade with the optimal strategy to adapt to policy changes
- With a cache solution, the model can store the regulation internally to avoid repeating the process

6. FrugalGPT into SFPIM



Week 11
Finalization of the code

The better the question. The better the answer. The better the world works.

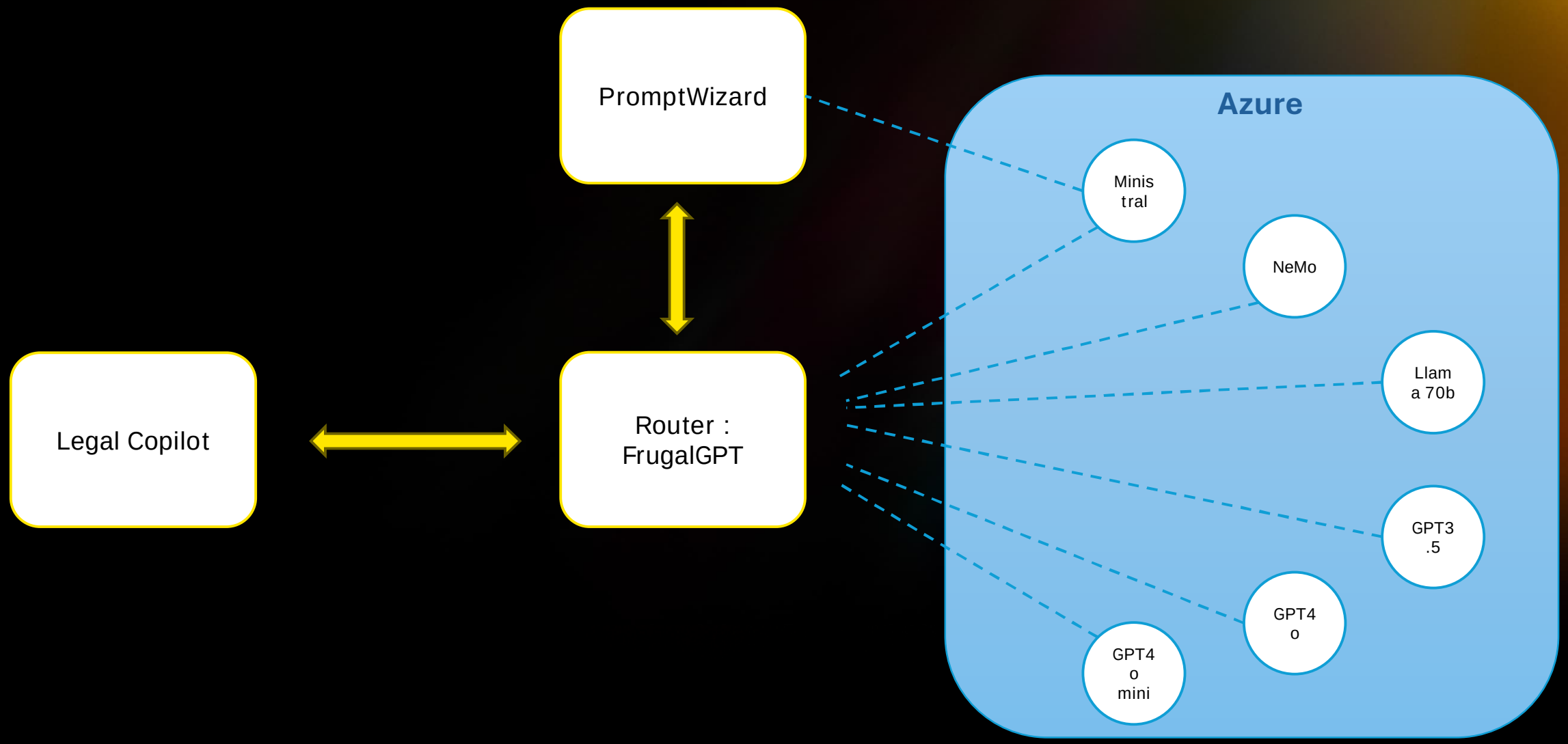
Shape the future
with confidence

Contents

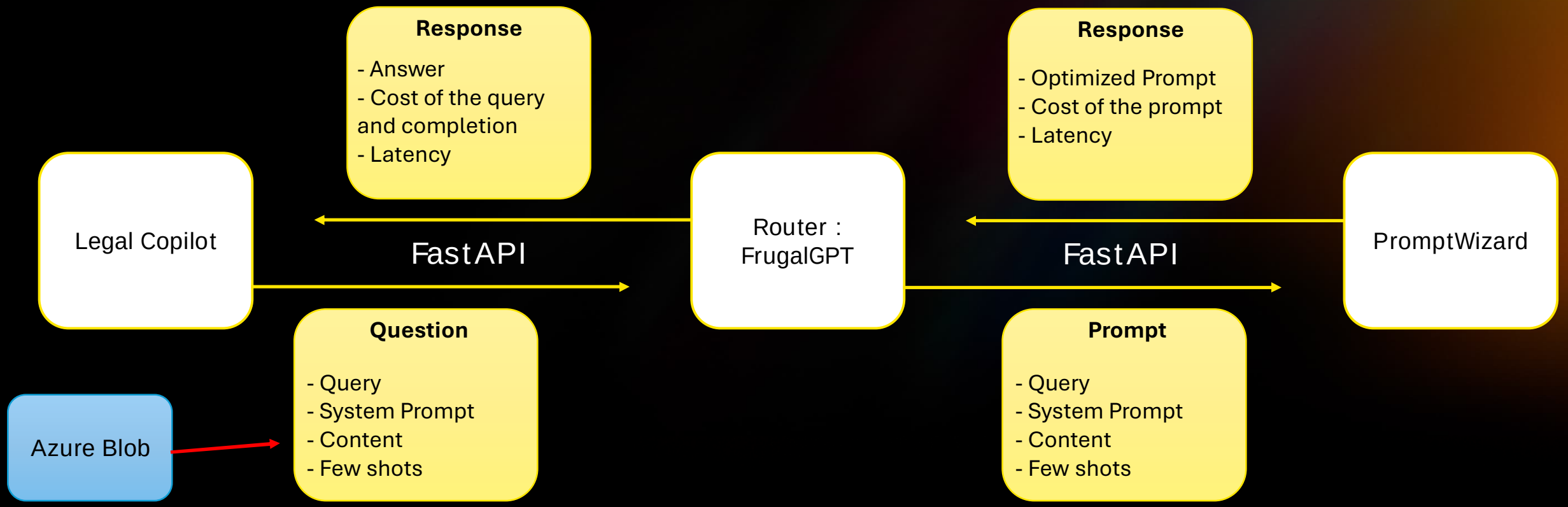
FrugalGPT as a Router

1. Architecture of the project
2. FrugalGPT
    - 2.1 LLM Cascade
    - 2.2 Prompt Adaptation
    - 2.3 LLM Approximation
    - 2.4 Cost & Latency comparison w/ & w/o the cascade
3. Demo
4. Cost Analysis
5. What’s Next
6. FrugalGPT into SFPIM

1. Architecture of the project



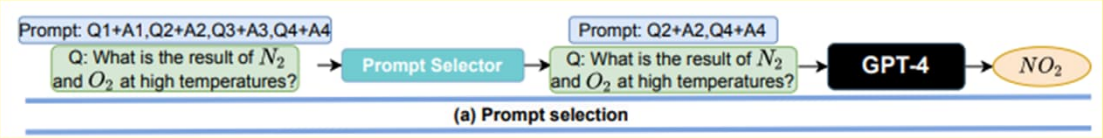
1. Architecture of the project



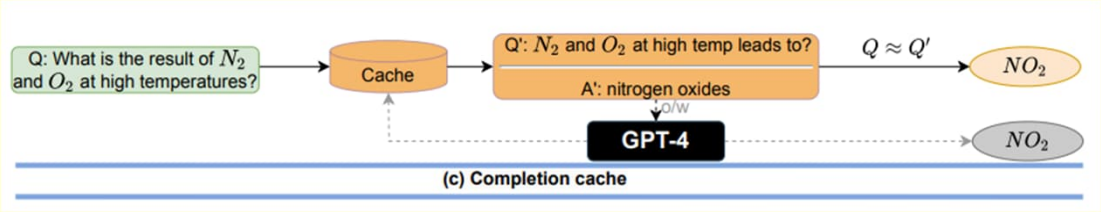
2. FrugalGPT

FrugalGPT

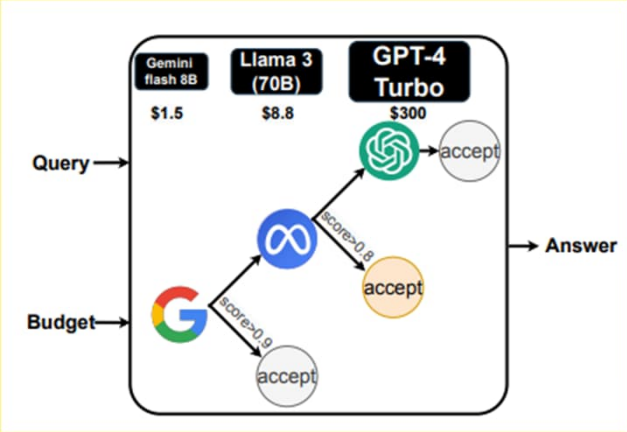
Prompt Adaptation



LLM Approximation



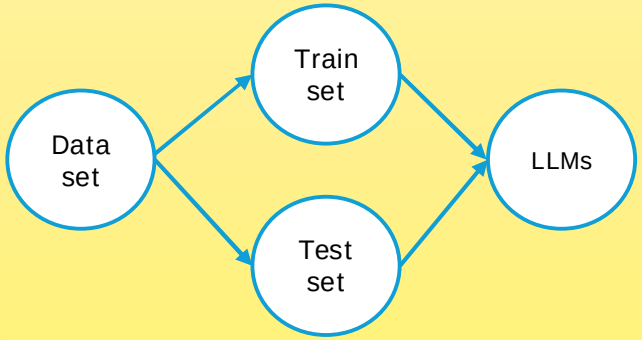
LLM Cascade



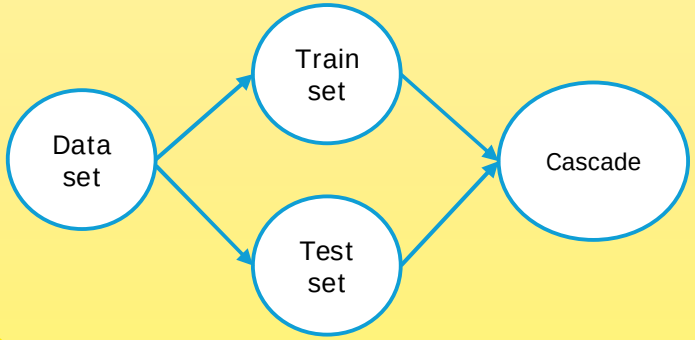
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



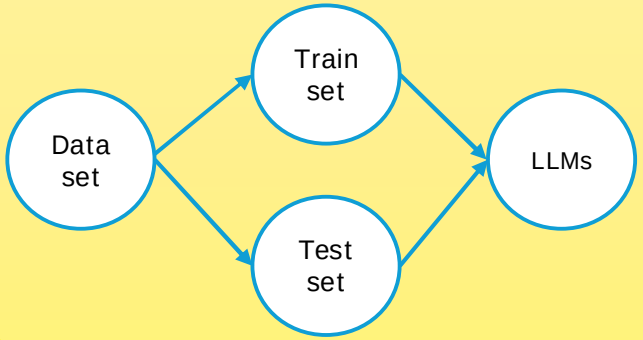
Cascade

1. Optimal strategy by testing LLM permutations that meet budget and threshold constraints.
2. Derives a strategy that aligns with the various cost requirements.
3. Reading the cascade to respond to a new query.
4. BERT used to evaluate the response quality checking if the threshold is exceeded. If so, the response is retained.

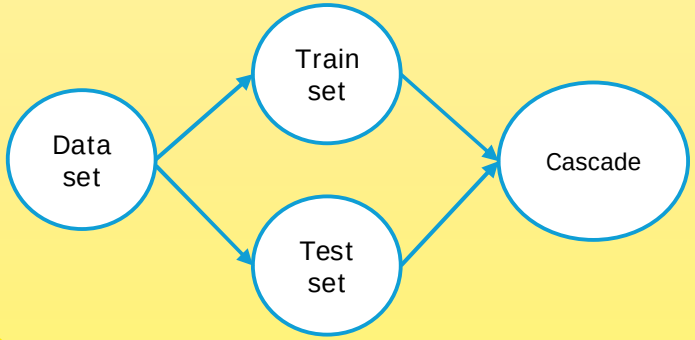
2.1. LLM Cascade

LLM Cascade

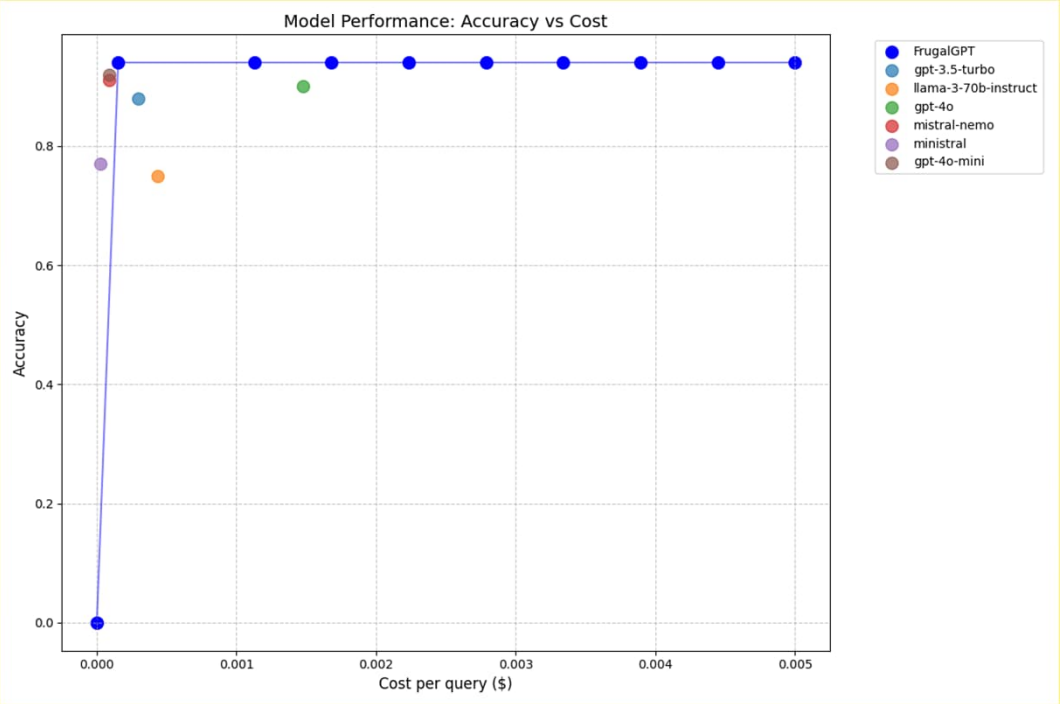
Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



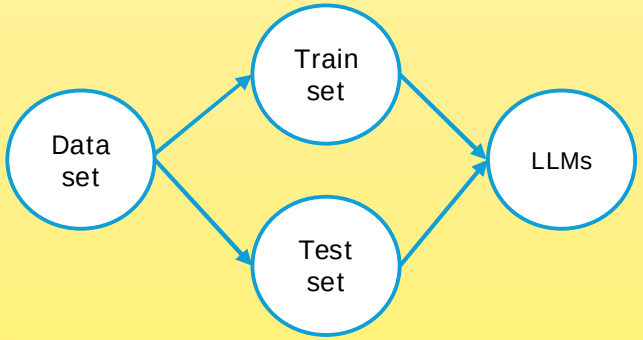
Cascade



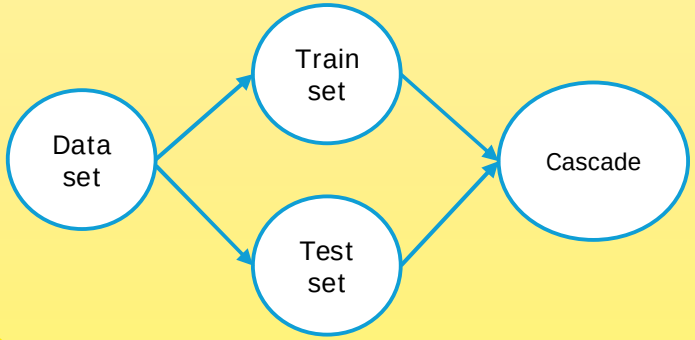
2.1. LLM Cascade

LLM Cascade

Evaluate the accuracy of the LLMs



Evaluate the accuracy of the Cascade



	Test_acc	Test_cost	Test_size	Train_acc	Train_cost	Train_size	Budget	Method	Provider	Marker
0	0.00	0.000000	100	0.00	0.000000	100	0.000010	FrugalGPT	FrugalGPT	1
1	0.94	0.000153	100	0.94	0.000153	100	0.000573	FrugalGPT	FrugalGPT	1
2	0.94	0.000153	100	0.94	0.000153	100	0.001127	FrugalGPT	FrugalGPT	1
3	0.94	0.000153	100	0.94	0.000153	100	0.001680	FrugalGPT	FrugalGPT	1
4	0.94	0.000153	100	0.94	0.000153	100	0.002233	FrugalGPT	FrugalGPT	1
5	0.94	0.000153	100	0.94	0.000153	100	0.002787	FrugalGPT	FrugalGPT	1
6	0.94	0.000153	100	0.94	0.000153	100	0.003340	FrugalGPT	FrugalGPT	1
7	0.94	0.000153	100	0.94	0.000153	100	0.003893	FrugalGPT	FrugalGPT	1
8	0.94	0.000153	100	0.94	0.000153	100	0.004447	FrugalGPT	FrugalGPT	1
9	0.94	0.000153	100	0.94	0.000153	100	0.005000	FrugalGPT	FrugalGPT	1
0	0.88	0.000295	100	0.90	0.000294	100	10.000000	gpt-3.5-turbo	azure_openai	1
1	0.75	0.000432	100	0.71	0.000430	100	10.000000	llama-3-70b-instruct	azure_llama	1
2	0.90	0.001477	100	0.89	0.001474	100	10.000000	gpt-4o	azure_gpt4o	1
3	0.91	0.000091	100	0.87	0.000091	100	10.000000	mistral-nemo	azure_mistral_nemo	1
4	0.77	0.000024	100	0.79	0.000024	100	10.000000	ministral	azure_ministral	1
5	0.92	0.000089	100	0.87	0.000089	100	10.000000	gpt-4o-mini	azure_gpt4o_mini	1

```
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT>
```

The cost given is the train cost for 100 queries & answers

94% of accuracy for FrugalGPT

2.2. Prompt Adaptation

PromptWizard

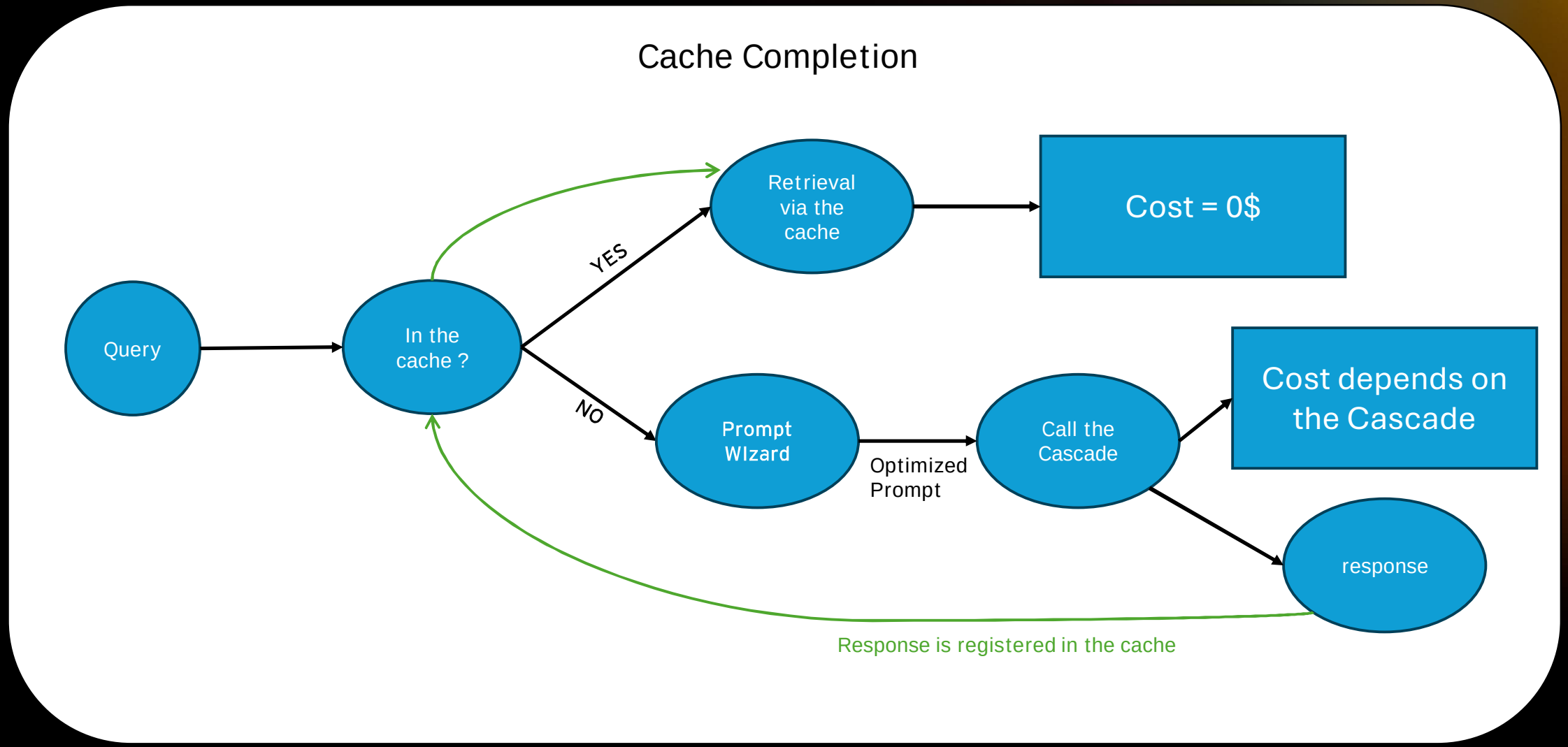
What is Prompt Wizard ?

Definition : P.W. is a tool or feature designed to help generate, refine and optimize prompts for AI models to achieve more accurate and tailored results.

How does it work ?

1. The system is configured with specific optimization parameters.
2. Task : Optimize the prompt for clarity and relevance (assisting employees with healthcare plan queries).
3. Prompt mutations are generated by modifying the original prompt.
4. Each iteration is tested, and the most effective prompt is selected based on the quality of the generated response.
5. The optimized prompt is sent to FrugalGPT.

2.3. LLM Approximation



2.3. LLM Approximation

Tech part of the Cache Completion

Architecture of the Cache

Cache JSON
Index FAISS (Facebook AI Similarity Search)

Tech used

Sentence Transformer
Index FAISS (Facebook AI Similarity Search)
Similarity Threshold (85%)

Advantages

Reduced API call costs
Time saving
Dynamic Optimization

Sentence Transformer & all-mpnet-base-v2

- Text into dense vector embeddings for semantic comparison.
- MPNet (natural language understanding)

FAISS

- O(logN) search complexity, much faster than O(N)
- Finds the closest cached response based on vector similarity.

EY

2.4. Cost & Latency comparison w/ & w/o the cascade

```
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /auth_setup 1.1 200 386 2067
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /auth_setup 1.1 - - 2941
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /.auth/refresh 1.1 404 207 999
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /.auth/refresh 1.1 - - 999
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /config 1.1 200 83 503
[2025-03-25 10:34:19 +0100] [21748] [INFO] 127.0.0.1:50960 GET /config 1.1 - - 1765
Cascade total cost: 0.000225
GPT-4o total cost: 0.002365
[2025-03-25 10:34:38 +0100] [21748] [INFO] 127.0.0.1:50960 POST /chat 1.1 200 - 16559797
[2025-03-25 10:34:38 +0100] [21748] [INFO] 127.0.0.1:50960 POST /chat 1.1 - - 16559797
Cascade total cost: 0
GPT-4o total cost: 0.002345
[2025-03-25 10:36:21 +0100] [21748] [INFO] 127.0.0.1:51091 POST /chat 1.1 200 - 8359297
[2025-03-25 10:36:21 +0100] [21748] [INFO] 127.0.0.1:51091 POST /chat 1.1 - - 8360027
```

```
INFO:     Started server process [18804]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
INFO:     127.0.0.1:50752 - "POST /execute_cascade HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
CRITICAL:root:Using service: azure_mistral_nemo/mistral-nemo
INFO:     127.0.0.1:50989 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:50989 - "POST /execute_gpt4o HTTP/1.1" 200 OK
CRITICAL:root:loading strategy from:C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\FrugalGPT\strategy\AGNEWS_Model20252602\cascade_strategy.json
INFO:     127.0.0.1:51095 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:     127.0.0.1:51095 - "POST /execute_gpt4o HTTP/1.1" 200 OK
```

Legal Copilot — 127.0.0.1:50505

Azure GPT — Chat — Evaluate

Clear chat — Developer settings

What is Northwind Health Plus?

Northwind Health Plus is a program offered by Northwind Health that provides personal health support programs to help individuals get the most out of their health benefits. The program includes access to Care Coordinators, Health Coaches, Care Managers, Health Educators, and Specialists to support individuals in managing their health [1]. Northwind Health is responsible for paying claims correctly by determining eligibility, obtaining pre-authorization for services when required, and paying for eligible services based on the terms of the plan. They also provide an explanation of benefits (EOB) after processing a claim, outlining the services provided and the amounts paid [2].

Citations: 1. Northwind_Standard_Benefits_Details.pdf#page=69
2. Northwind_Standard_Benefits_Details.pdf#page=99

Did you like this response?

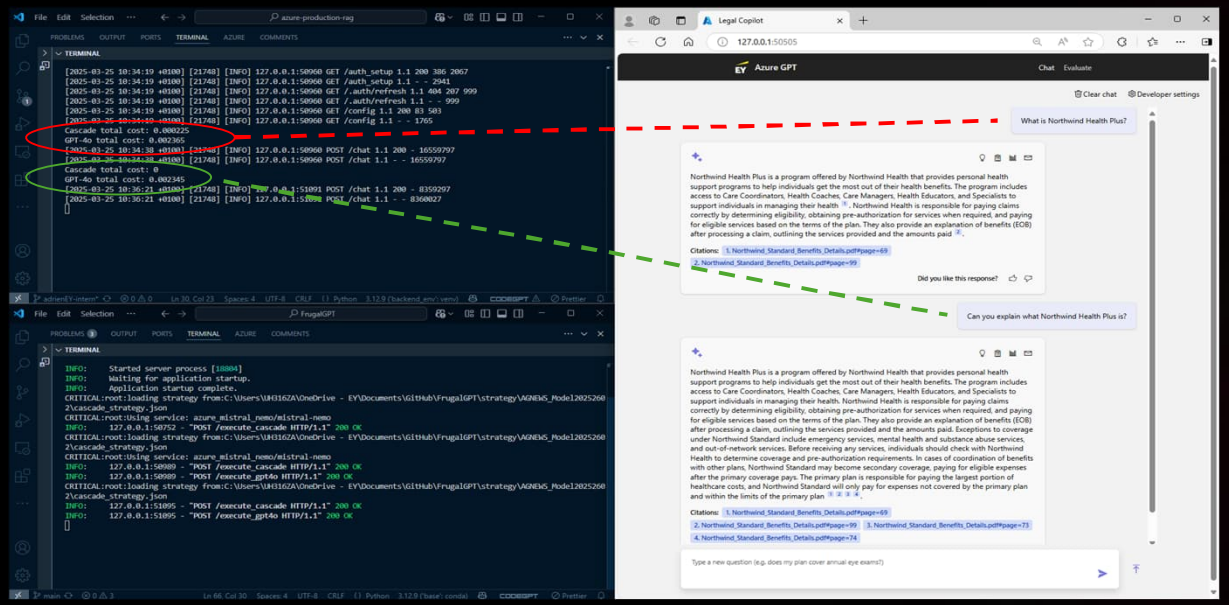
Can you explain what Northwind Health Plus is?

Northwind Health Plus is a program offered by Northwind Health that provides personal health support programs to help individuals get the most out of their health benefits. The program includes access to Care Coordinators, Health Coaches, Care Managers, Health Educators, and Specialists to support individuals in managing their health. Northwind Health is responsible for paying claims correctly by determining eligibility, obtaining pre-authorization for services when required, and paying for eligible services based on the terms of the plan. They also provide an explanation of benefits (EOB) after processing a claim, outlining the services provided and the amounts paid. Exceptions to coverage under Northwind Standard include emergency services, mental health and substance abuse services, and out-of-network services. Before receiving any services, individuals should check with Northwind Health to determine coverage and pre-authorization requirements. In cases of coordination of benefits with other plans, Northwind Standard may become secondary coverage, paying for eligible expenses after the primary coverage pays. The primary plan is responsible for paying the largest portion of healthcare costs, and Northwind Standard will only pay for expenses not covered by the primary plan and within the limits of the primary plan [1][2][3][4].

Citations: 1. Northwind_Standard_Benefits_Details.pdf#page=69
2. Northwind_Standard_Benefits_Details.pdf#page=99
3. Northwind_Standard_Benefits_Details.pdf#page=73
4. Northwind_Standard_Benefits_Details.pdf#page=74

Type a new question (e.g. does my plan cover annual eye exams?)

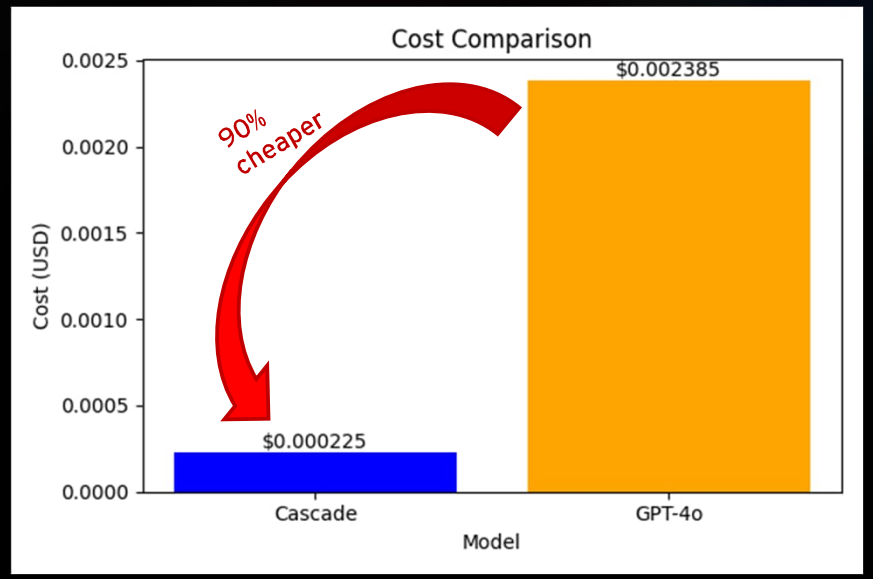
2.4. Cost & Latency comparison w/ & w/o the cascade



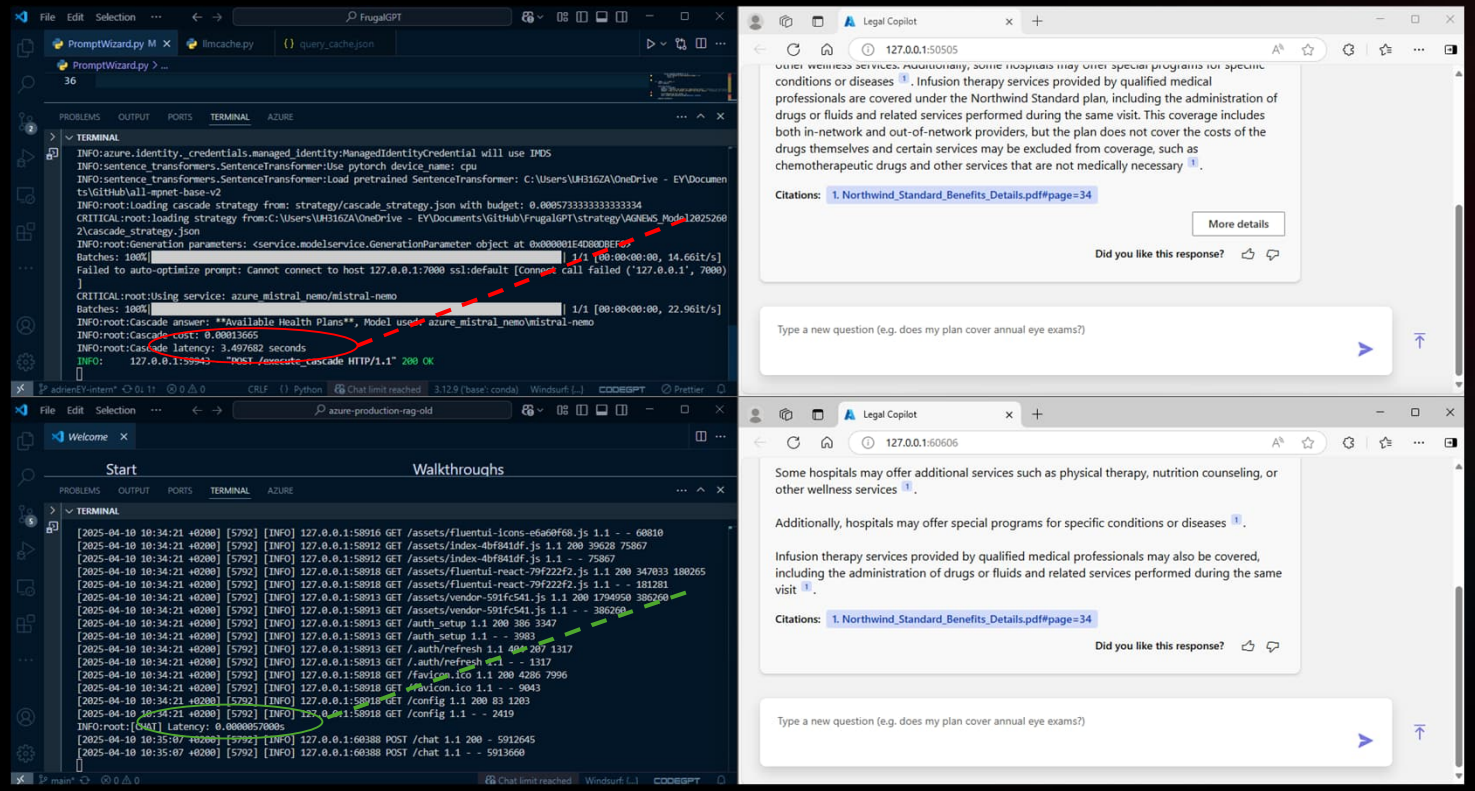
<u>Cost of the query and the answer :</u>

- With the cascade :
  - 0.000225$
- With GPT4o :
  - 0.002385$
- With the cache :
  - 0$ (for a similar question)

Cost reduction of 90% for the given query



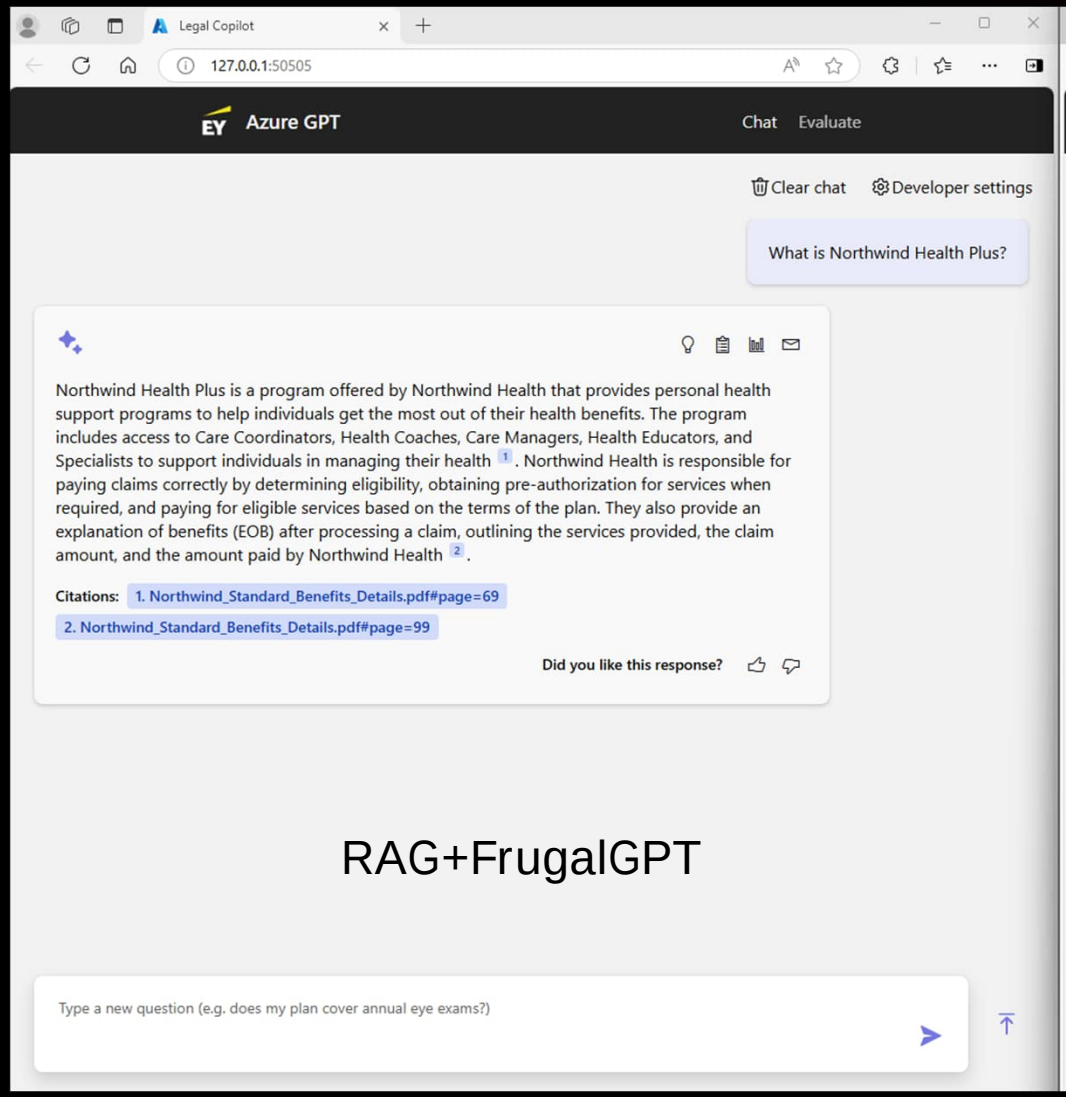
2.4. Cost & Latency comparison w/ & w/o the cascade



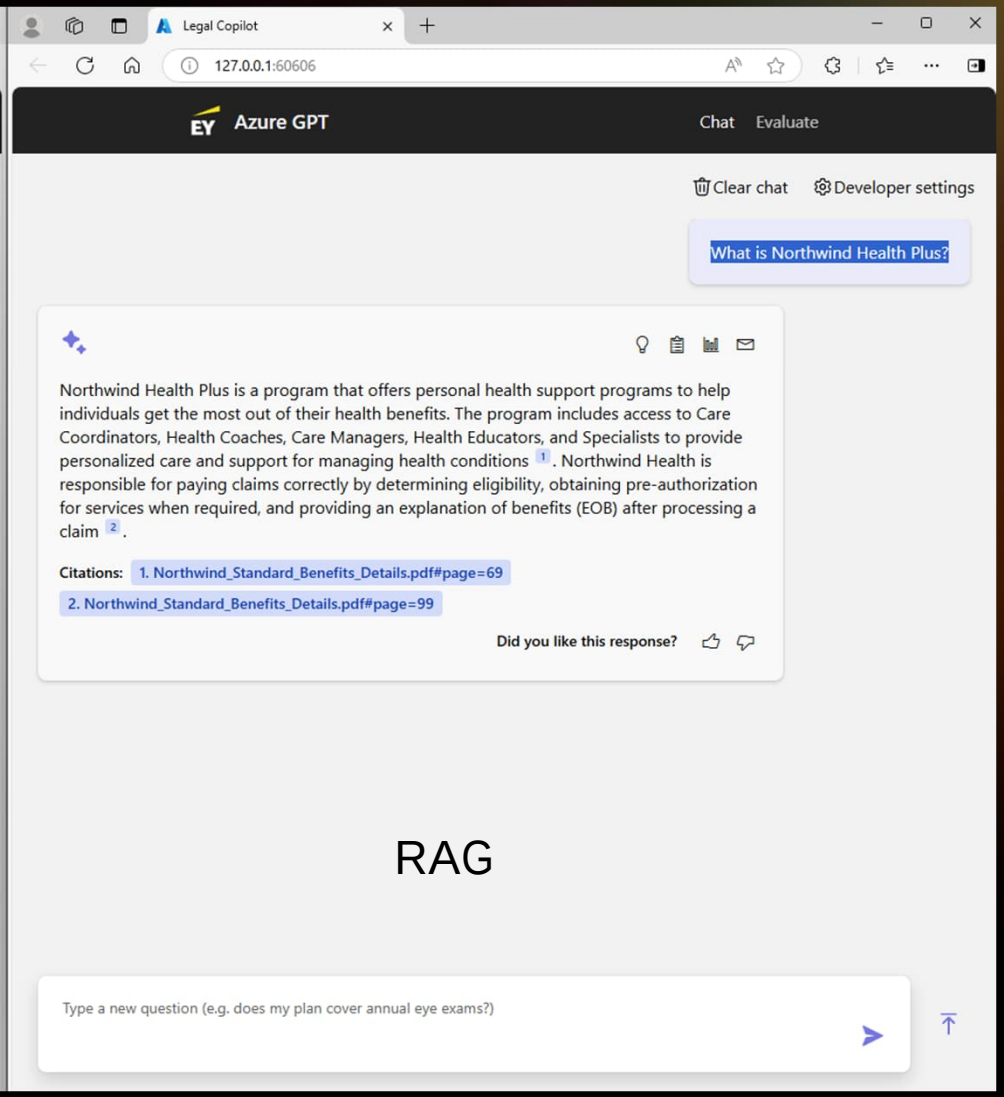
Latency of the request :

- With the cascade :
  - 3.5s
- With GPT4o :
  - 0.0000057s
- Ratio :
  - GPT4o is 99% faster

2.4 RAG+FrugalGPT vs RAG

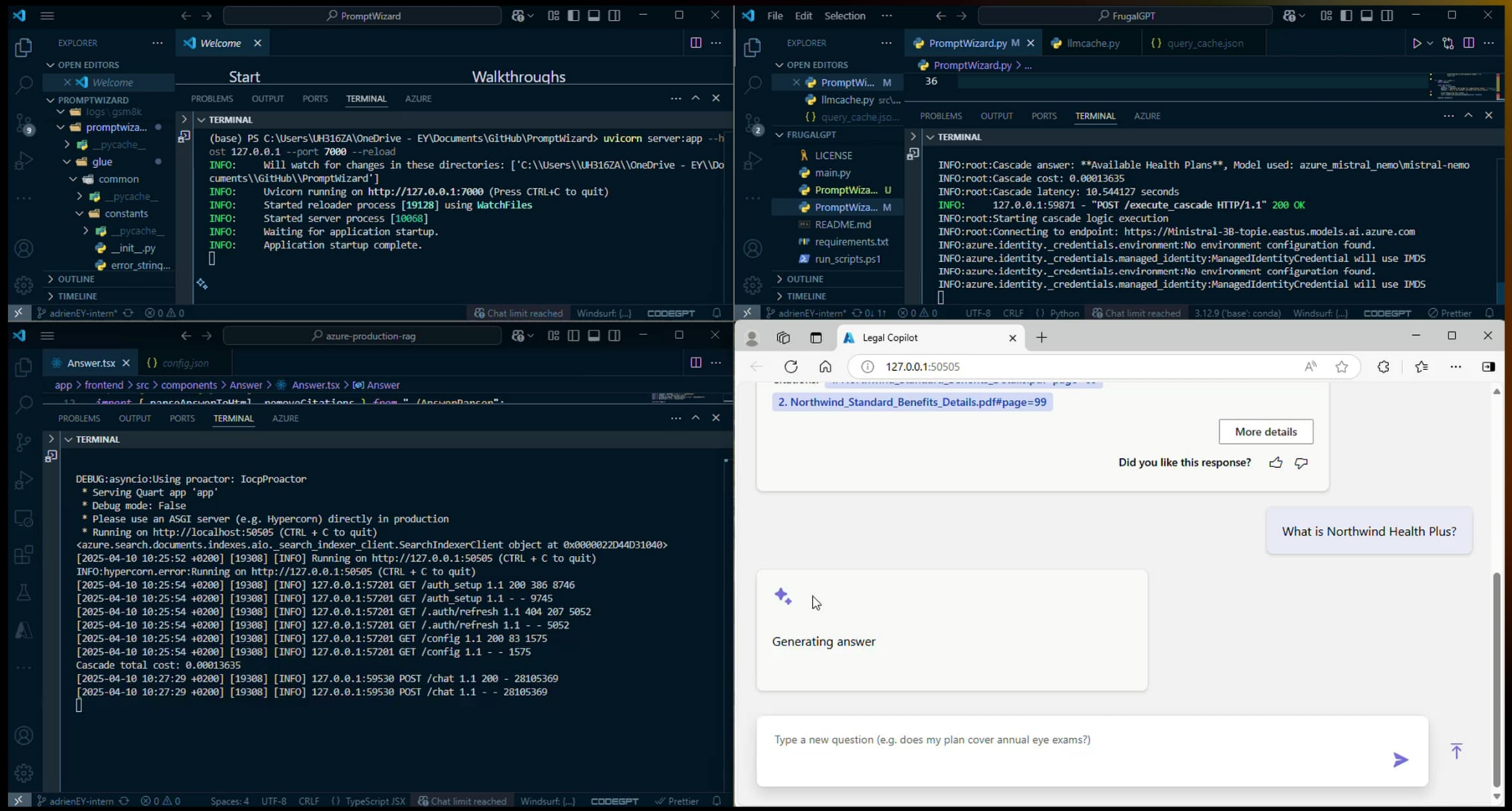


RAG+FrugalGPT

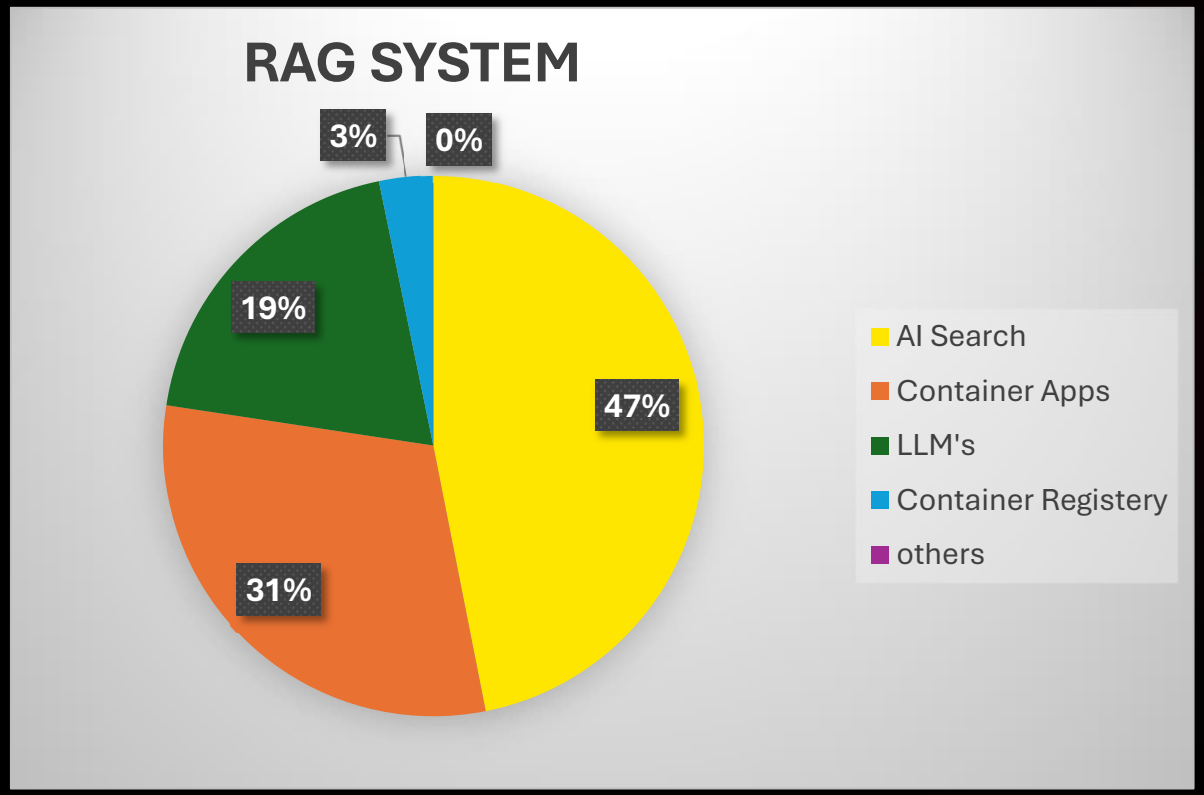


RAG

3. Demo


PromptWizard
Welcome
Start
Walkthroughs
PROBLEMS OUTPUT PORTS TERMINAL AZURE
TERMINAL
(base) PS C:\Users\UH316ZA\OneDrive - EY\Documents\GitHub\PromptWizard> uvicorn server:app --host 127.0.0.1 --port 7000 --reload
INFO: Will watch for changes in these directories: ['C:\\Users\\UH316ZA\\OneDrive - EY\\Documents\\GitHub\\PromptWizard']
INFO: Uvicorn running on http://127.0.0.1:7000 (Press CTRL+C to quit)
INFO: Started reloader process [19128] using WatchFiles
INFO: Started server process [10068]
INFO: Waiting for application startup.
INFO: Application startup complete.
FrugalGPT
PromptWizard.py
llmcache.py
query_cache.json
INFO:root:Cascade answer: **Available Health Plans**, Model used: azure_mistral_nemo\mistral-nemo
INFO:root:Cascade cost: 0.00013635
INFO:root:Cascade latency: 10.544127 seconds
INFO: 127.0.0.1:59871 - "POST /execute_cascade HTTP/1.1" 200 OK
INFO:root:Starting cascade logic execution
INFO:root:Connecting to endpoint: https://Ministral-3B-topie.eastus.models.ai.azure.com
INFO:azure.identity._credentials.environment:No environment configuration found.
INFO:azure.identity._credentials.managed_identity:ManagedIdentityCredential will use IMDS
INFO:azure.identity._credentials.environment:No environment configuration found.
INFO:azure.identity._credentials.managed_identity:ManagedIdentityCredential will use IMDS
azure-production-rag
Answer.tsx
app > frontend > src > components > Answer > Answer.tsx > Answer
DEBUG:asyncio:Using proactor: IocpProactor
* Serving Quart app 'app'
* Debug mode: False
* Please use an ASGI server (e.g. Hypercorn) directly in production
* Running on http://localhost:50505 (CTRL + C to quit)
[2025-04-10 10:25:52 +0200] [19308] [INFO] Running on http://127.0.0.1:50505 (CTRL + C to quit)
INFO:hypercorn.error:Running on http://127.0.0.1:50505 (CTRL + C to quit)
[2025-04-10 10:25:54 +0200] [19308] [INFO] 127.0.0.1:57201 GET /auth_setup 1.1 200 386 8746
[2025-04-10 10:25:54 +0200] [19308] [INFO] 127.0.0.1:57201 GET /auth_setup 1.1 - - 9745
[2025-04-10 10:25:54 +0200] [19308] [INFO] 127.0.0.1:57201 GET /.auth/refresh 1.1 404 207 5052
[2025-04-10 10:25:54 +0200] [19308] [INFO] 127.0.0.1:57201 GET /.auth/refresh 1.1 - - 5052
[2025-04-10 10:25:54 +0200] [19308] [INFO] 127.0.0.1:57201 GET /config 1.1 200 83 1575
[2025-04-10 10:25:54 +0200] [19308] [INFO] 127.0.0.1:57201 GET /config 1.1 - - 1575
Cascade total cost: 0.00013635
[2025-04-10 10:27:29 +0200] [19308] [INFO] 127.0.0.1:59530 POST /chat 1.1 200 - 28105369
[2025-04-10 10:27:29 +0200] [19308] [INFO] 127.0.0.1:59530 POST /chat 1.1 - - 28105369
Legal Copilot
127.0.0.1:50505
2. Northwind_Standard_Benefits_Details.pdf#page=99
More details
Did you like this response?
What is Northwind Health Plus?
Generating answer
Type a new question (e.g. does my plan cover annual eye exams?)

4. Cost Analysis



Impact on Total Expenses

- **Context**: LLMs account for **20% of the total estimated cost of $130**.
- **Potential Optimization**: Cost reduction of LLMs by **50 to 90%**.
- **Impact on Total Cost**: This would lead to a **maximum reduction of 10 to 15%**. (13 to 23$)
- **Other Factors to Consider**: Number of documents for search, query frequency, infrastructure architecture…

Week 12
Complete technical and operational documentation

The better the question. The better the answer. The better the world works.

Shape the future with confidence

Contents

1. Context
2. Architecture of the project
3. RAG As Is / To Be
4. Recommendations

1. Context

RAG (Retrieval-Augmented Generation) systems are powerful but expensive, especially at scale.

Why reduce costs?

- EY aims to offer cost-effective AI solutions to clients without compromising performance.

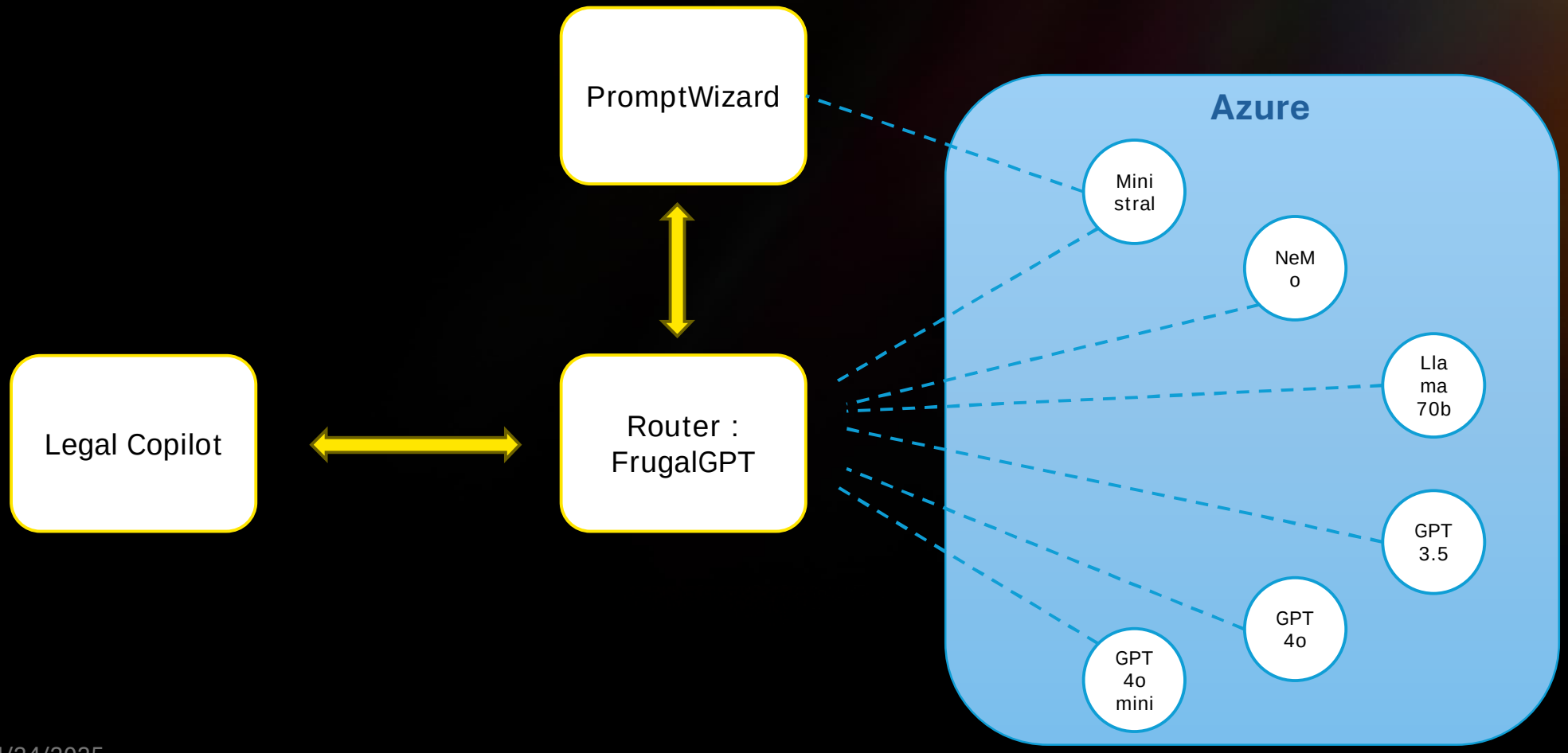
Solution	Description	Focus
FrugalGPT	Smart model routing based on query complexity. Uses small models first, escalates only when needed.	Cost-effective model usage
Cache-Augmented Generation	Speeds up text generation by reusing cached past outputs.	Avoids redundant LLM calls
Cohere Rerank	Improves document ranking in retrieval phase.	Better context for LLM
PromptWizard	Optimizes prompts to reduce token use and improve LLM output.	Prompt efficiency

1. Context

✅ Why Build on FrugalGPT?

- **Up to 90% cost reduction** per query.
- Combines well with caching + prompt optimization.
- Lightweight integration via REST API.
- Offers **traceability and control** over per-query costs.
- Aligns with sustainability and digital sobriety goals.

2. Architecture of the project



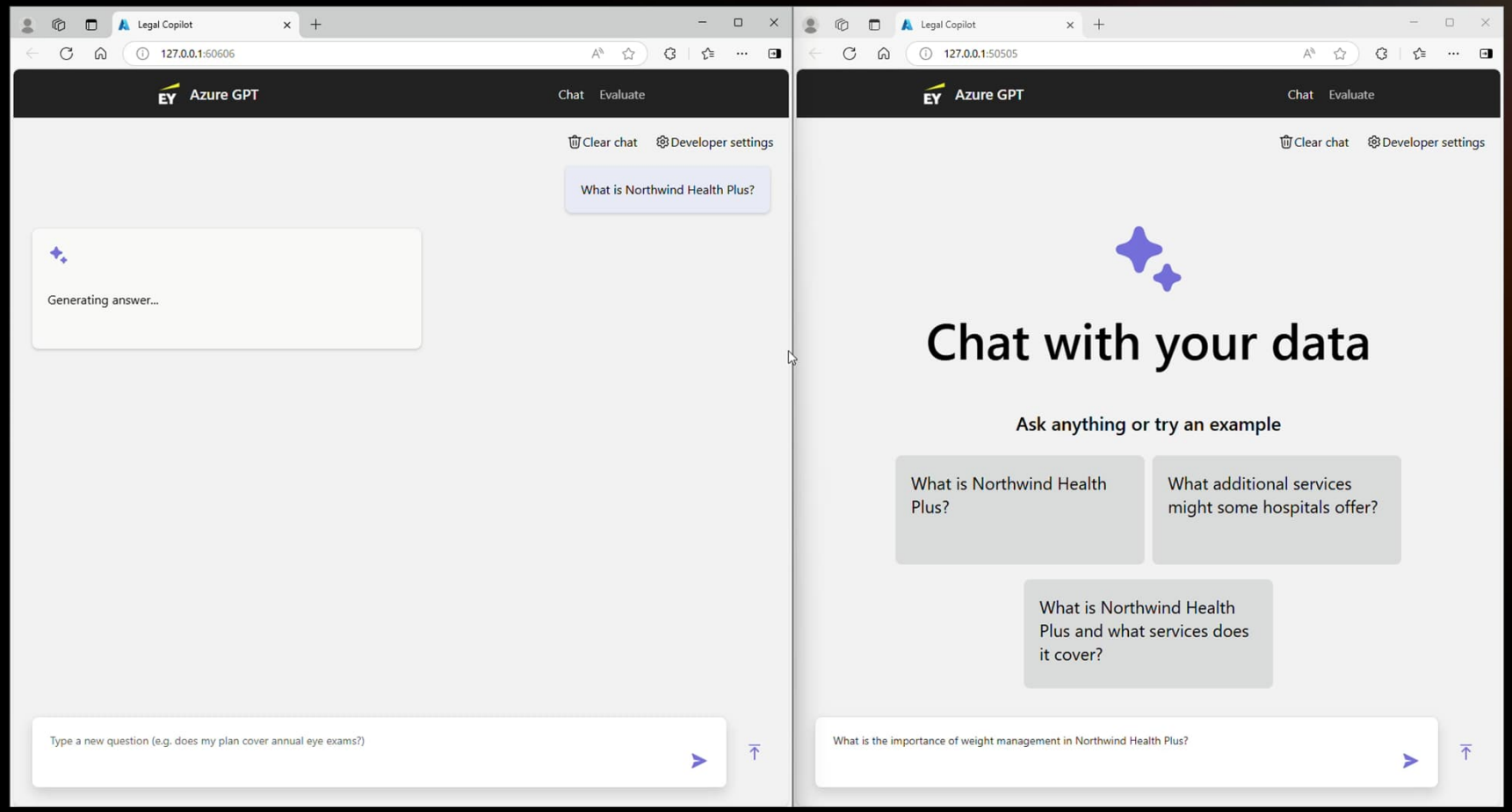
3. RAG As Is / To Be

Aspect	Before (Standard RAG)	After (Optimized RAG)
Model Usage	Always GPT-4o, regardless of query complexity	FrugalGPT dynamically selects the cheapest suitable model (e.g., GPT-3.5, GPT-4o-mini)
Cost per Query	~$0.002385 per query (**no optimization**)	~$0.000225 per query (**with routing + cache**)
Caching	No caching — repeated queries hit the LLM each time	FAISS-based cache avoids unnecessary model calls
Prompt Design	Manually written	PromptWizard rewrites prompts to improve focus
Latency	Fast (single-shot GPT-4o response)	Slight delay (2–15s) due to model selection and prompt optimization
Scalability	Costs scale linearly with usage	Smart routing + caching make high-volume usage affordable
Flexibility	One-size-fits-all setup	Modular, easily adjustable depending on budget/performance
User Interaction	No “**more details button**”	"**Ask for more details**" button enables multi-turn clarification

3. RAG As Is / To Be

Technologies Used : RAG + FrugalGPT (w/o cache) + PromptWizard

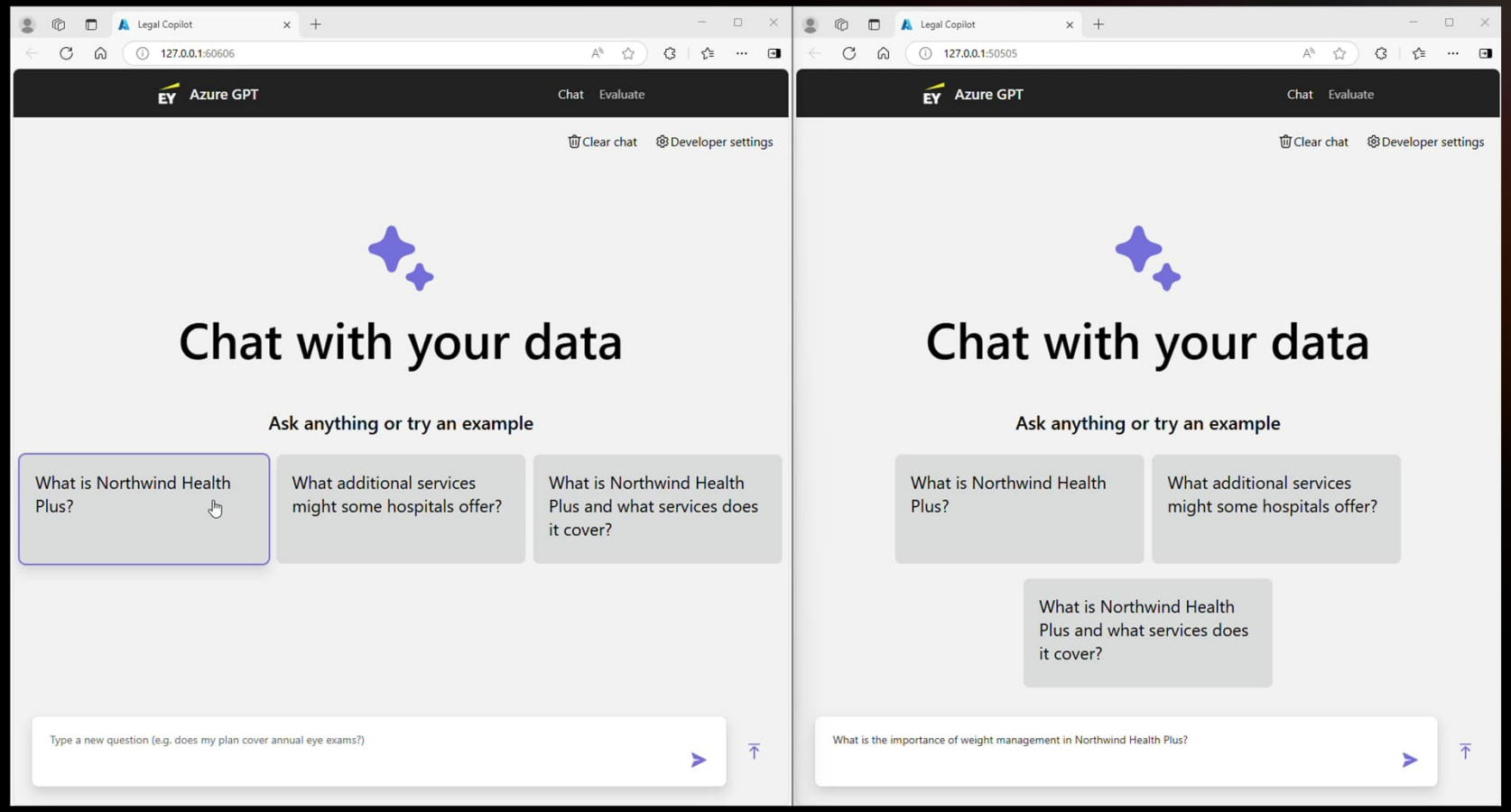
Latency : 14 seconds



3. RAG As Is / To Be

Technologies Used : RAG + FrugalGPT (w/o cache)

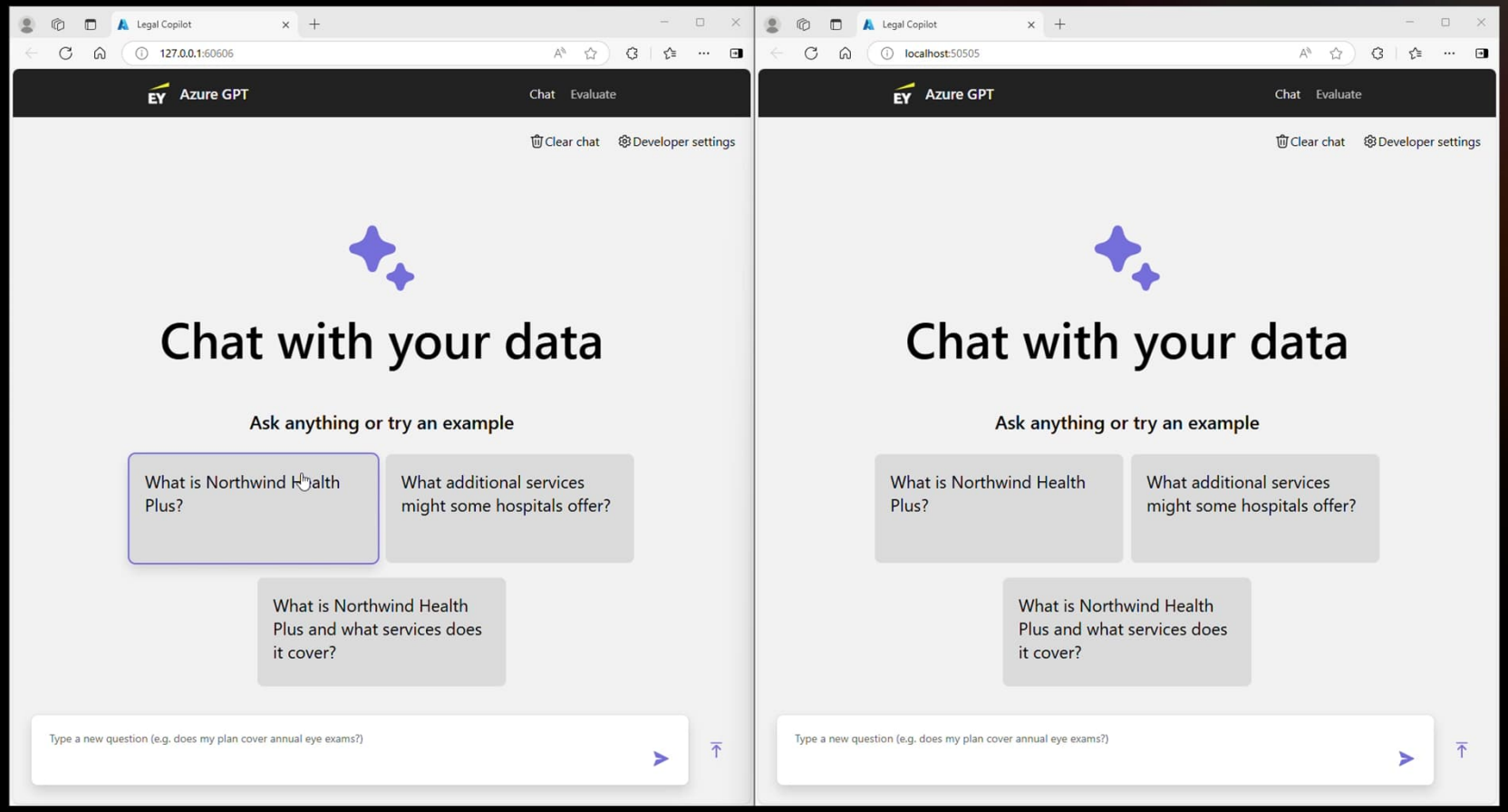
Latency : 8 seconds



3. RAG As Is / To Be

Technologies Used : RAG + FrugalGPT (w/ cache)

Latency : 1 second



4. Recommendations

🎯 Why Adopt This Solution?

- Reduce costs by up to 90% per query through caching, smart routing and prompt optimization.
- Maintain response quality even with lower-cost models.
- Automatic reuse of similar responses (via cache) → performance gain.
- Environmentally Friendly

4. Recommendations

🎯 Ideal Client Use Cases :

- High-volume internal assistants (e.g., Legal Copilot)
- Customer support chatbots with recurring FAQs
- Resource-constrained environments (limited budget)
- Sustainability-conscious clients looking to reduce environmental impact

4. Recommendations

FrugalGPT with cache

Aspect	Details
Description	Uses a cache to instantly respond to frequently asked or previously answered questions.
Response Time	~1 second
Typical Use Cases	- FAQs (interest rates, fees, schedules) - Common customer queries - Simple, high-volume requests
Example Queries	- "What are the international transfer fees?" - "How do I reset my password?"
Target Users	End customers via chatbot, mobile app, website FAQ
Advantages	- Ultra fast - Cost-effective - Highly scalable
Disadvantages	- Static answers - No personalization - Risk of outdated responses
When to Use	- When speed is critical - For very frequent and predictable questions

4. Recommendations

FrugalGPT w/o cache

Aspect	Details
Description	Uses FrugalGPT to dynamically respond to unique, context-rich, or personal queries without relying on cached answers.
Response Time	~5 to 7 seconds
Typical Use Cases	- Personalized client queries (account-specific, tax-related, portfolio info) - Sensitive or uncommon topics - Questions not reusable or cacheable
Example Queries	- "How has my portfolio performed over the past 3 years?" - "What are the tax implications for my insurance policy opened in 2010?"
Target Users	Clients through secure channels Customer advisors Middle/back-office staff
Advantages	- Context-aware, personalized answers - No cache pollution - Less costly than larger models
Disadvantages	- Slower than cache - Accuracy depends on prompt clarity
When to Use	- When the query is too specific or sensitive for cache - When personalization is required - For one-off or context-specific requests

4. Recommendations

FrugalGPT + PromptWizard

Aspect	Details
Description	The prompt is first optimized by PromptWizard (via a small LLM) before being sent to FrugalGPT, improving response quality for complex or poorly phrased queries.
Response Time	~10 to 15 seconds
Typical Use Cases	- Complex or unclear questions - Internal research - Structured content generation (e.g., summaries, reports)
Example Queries	- "Summarize 2024 AMF regulations for financial advisors." - "Draft a solvency analysis report for this client case."
Target Users	Analysts, legal teams, compliance officers, expert advisors
Advantages	- Improved response quality - Handles vague or poorly formulated prompts - Suitable for high-value tasks
Disadvantages	- Longer response time - Higher compute cost
When to Use	- When response quality is critical - For complex or structured tasks - When the input prompt needs refinement

Week 13
LowCost RAG : Final presentation at Team Meeting

Presented by : Adrien

The better the question. The better the answer. The better the world works.

Shape the future with confidence

1. What is RAG ?

- 🔄 What is RAG? A combo of retrieval (documents) + generation (via LLM).

- 🧩 Components:
  - Retrieval 📚 → finds relevant info using vector embeddings.

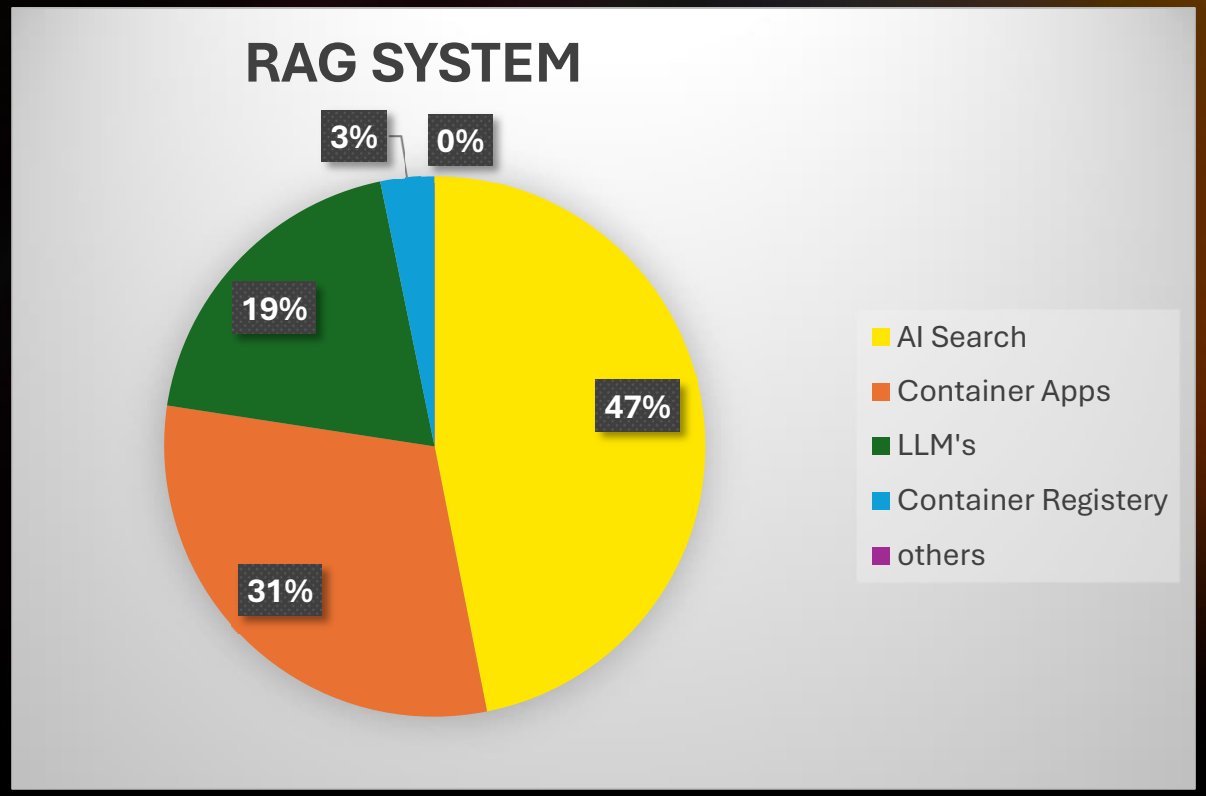
  - Generation 🤖 → LLM creates the final response using the fetched data.
    - 📈 Pros: Reduces hallucinations, improves transparency.
    - ⚠️ Cons: Costly, needs frequent updates.

2. Initial State of RAG & Cost Analysis

📄 Analyzed Azure usage (Feb–Mar 2025).

💥 Key Cost Areas:

- 🔍 Indexing: 47%
- 🤖 Generation (LLMs): 20%
- ☁️ App Hosting: 31%
- 🧩 Miscellaneous: Small %



3. Optimization Strategies

Goal: Cut LLM costs while keeping performance strong.

💡 Combined approaches:

- FrugalGPT 🪙
- PromptWizard ✍️
- Caching ♻️

FrugalGPT

🎓 Smart system from Stanford.

🧭 Logic:

- Easy queries → small models
- Complex queries → GPT-4o
- Caching avoids repeat generation

📉 Result: Up to 90% cost reduction.

PromptWizard

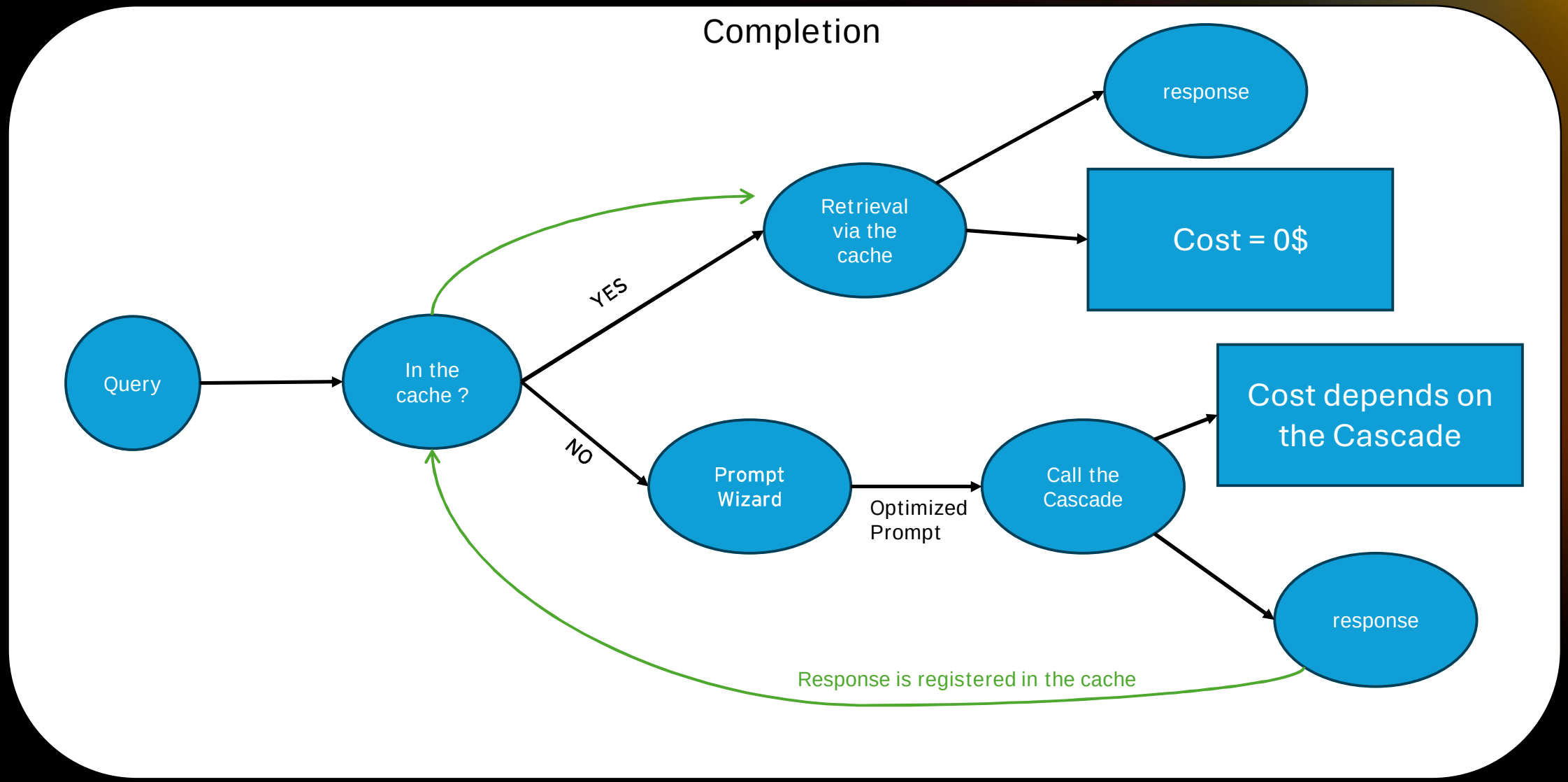
🧠 Microsoft tool to improve prompts.

🚀 Features:

- Semantic rewriting
- Context-aware prompts
- Avoids redundant queries

⚡ Slight latency but better answers.

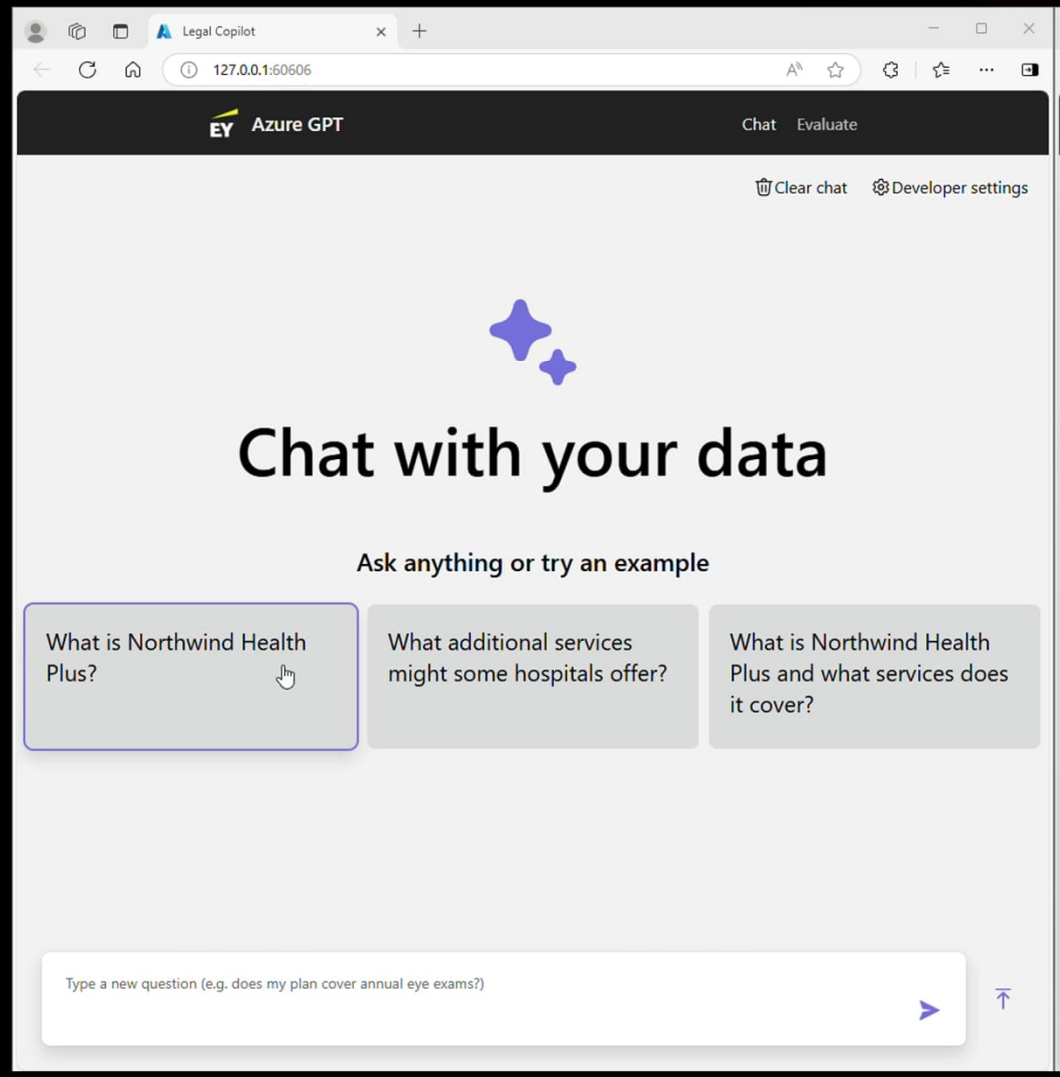
3. Optimization Strategies

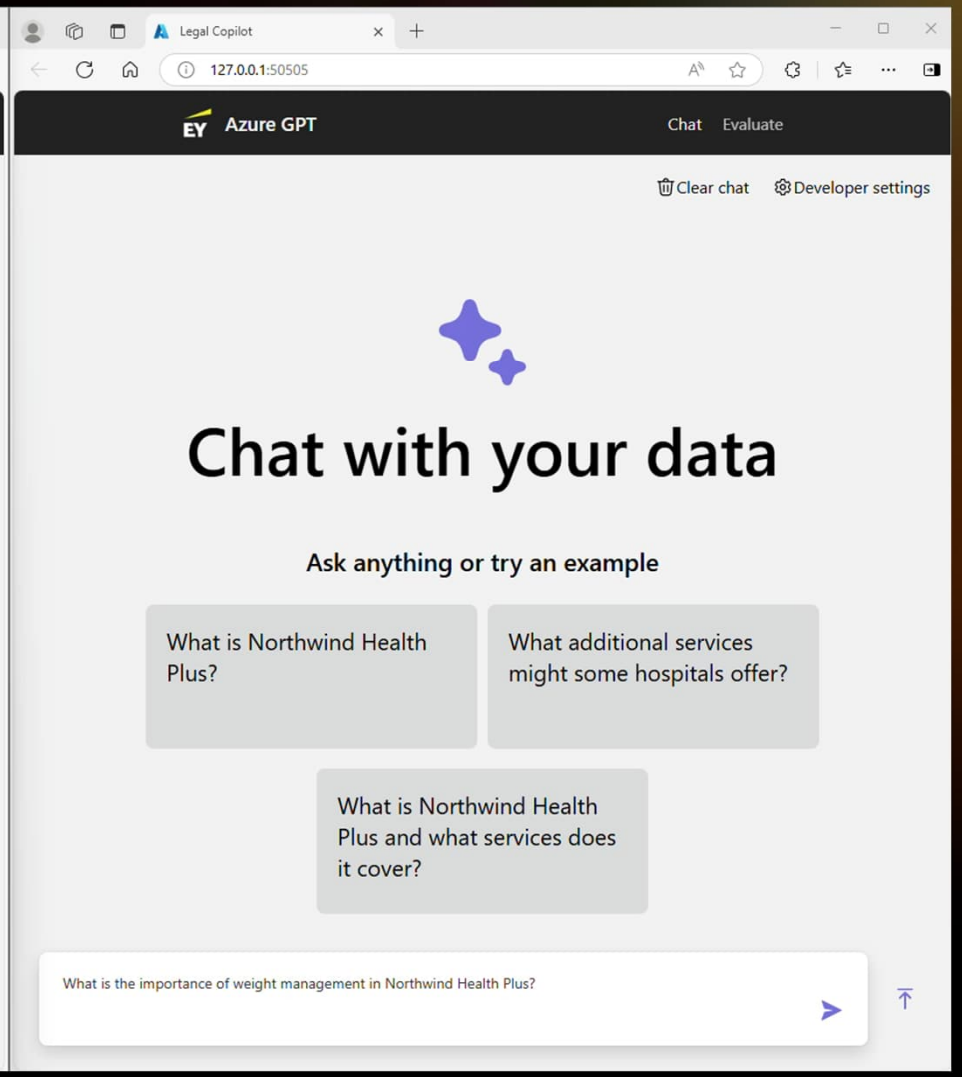


4. RAG As Is / To Be

Aspect	Before (Standard RAG)	After (Optimized RAG)
Model Usage	Always GPT-4o, regardless of query complexity	FrugalGPT dynamically selects the cheapest suitable model (e.g., GPT-3.5, GPT-4o-mini) ✅
Cost per Query	~$0.002385 per query (no optimization)	~$0.000225 per query (with routing + cache) ✅
Caching	No caching – repeated queries hit the LLM each time	FAISS-based cache avoids unnecessary model calls ✅
Prompt Design	Manually written	PromptWizard rewrites prompts to improve focus ✅
Latency	Fast (single-shot GPT-4o response) ✅	Slight delay (2–15s) due to model selection and prompt optimization
Scalability	Costs scale linearly with usage	Smart routing + caching make high-volume usage affordable ✅
Flexibility	One-size-fits-all setup	Modular, easily adjustable depending on budget/performance ✅
User Interaction	No “more details button”	"Ask for more details" button enables multi-turn clarification ✅

4. RAG As Is / To Be





5. Recommendations

Criteria	🟢 Frugal with Cache	🟡 Frugal without Cache	🔵 PromptWizard + Frugal (No Cache)
Response Time	⚡ ~1 sec	⏱️ ~5–7 sec	⏳ ~10–15 sec
Query Complexity	Very low, generic	Medium, personalized	High, complex, poorly formulated
Typical Use Cases	FAQs, procedures, product info	Client-specific questions, tax/legal cases	Reports, legal analysis, internal research
Example Queries	"What are interest rates?"	"What tax rules apply to my contract?"	"Summarize new regulations for advisors."
Personalization	❌ None	✅ Yes (contextual & personal)	✅ Yes (after prompt optimization)
Compute Cost	💰 0$	💰 Depends on the cascade	💰 Depends on the cascade
Target Users	Clients via chatbot/mobile	Advisors, internal agents	Analysts, legal, compliance
When to Avoid	For unique or sensitive questions	For long, structured content	For frequent, simple queries
When to Use	When speed is key	When personalization is required	When response quality matters most

5. Project Access

The complete project is available on the following organization repositories :

- https://github.com/dna-ey-fso/azure-production-rag/tree/adrienEY-intern
- https://github.com/dna-ey-fso/FrugalGPT/tree/adrienEY-intern
- https://github.com/dna-ey-fso/PromptWizard/tree/adrienEY-intern

Internship.md files with the link between all the project and how to launch the project is available.