Inference Techniques for Cost Optimization

Hamza Salem - Innopolis university

Agenda

Introduction to Inference Techniques

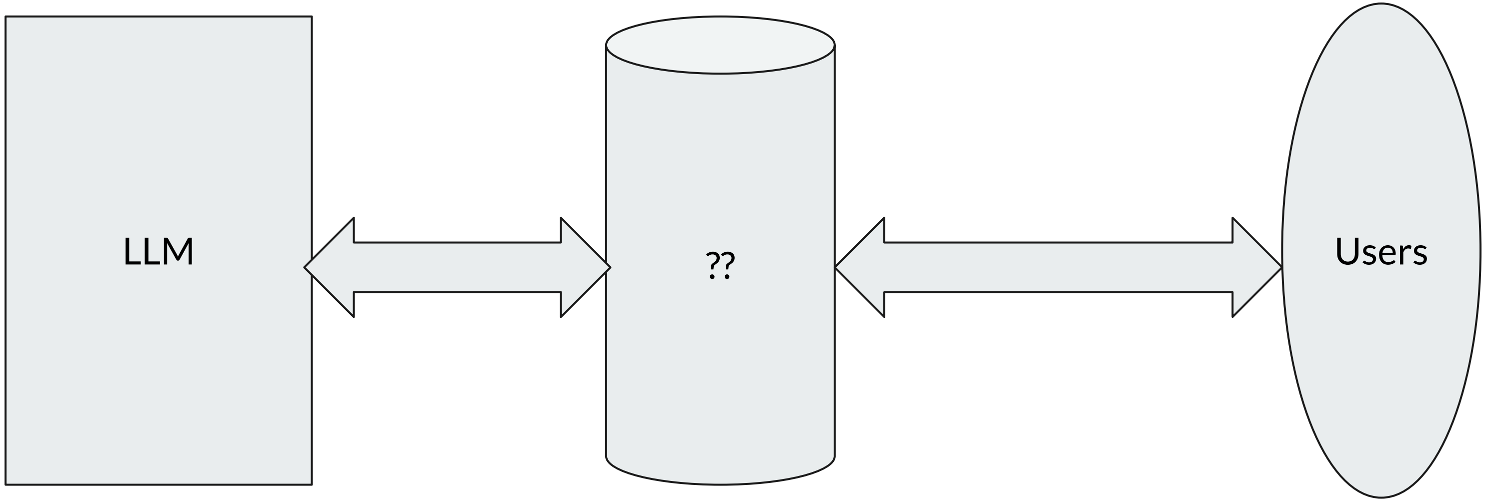
Advanced Inference Techniques

Handling Long Documents and Summarization

Batch Prompting Strategies

Retrieval Augmented Generation

Summary and Q&A


LLM
??
Users

Understanding Inference in LLMs

Inference refers to the process of using a trained model to generate predictions or outputs based on new input data.

Importance: Efficient inference is crucial for reducing operational costs, especially in real-time applications.

Challenges: Balancing speed, accuracy, and cost during inference.

Scenario: Imagine using Doctor Strange’s Time Stone to predict future events. Inference is like the process of quickly scanning possible futures (outputs) based on the current situation (input), and the faster you do it, the less energy (cost) you use.

Techniques for Efficient Inference

Prompt Engineering: Crafting specific input prompts to guide the model towards more accurate and efficient outputs.

Example: Asking a question in a precise way to get the most relevant answer, similar to how Tony Stark gives clear commands to JARVIS to get the exact information he needs.

Caching with Vector Stores: Storing previously computed outputs to reuse for similar future inputs, reducing the need for repeated inference.

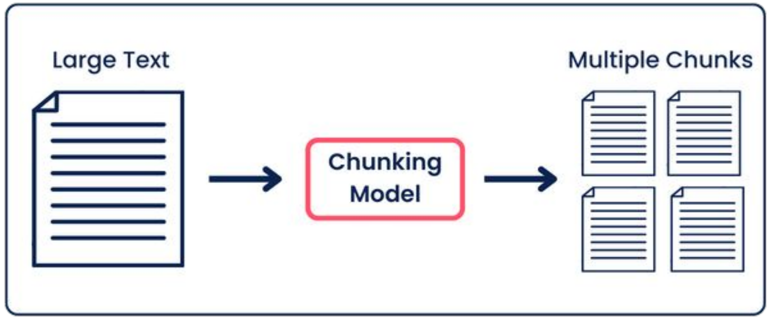
Example: Think of it like storing Thor’s previous battles in the Avengers database, so when a similar threat arises, you don’t need to re-strategize from scratch.

Handling Long Documents and Summarization

Challenge: Long documents can overwhelm LLMs, leading to high inference costs and slower performance.

Solutions:

- **Chunking: Breaking down long documents into smaller, manageable sections.**
- **Summarization: Generating concise summaries to reduce the amount of text the model needs to process.**

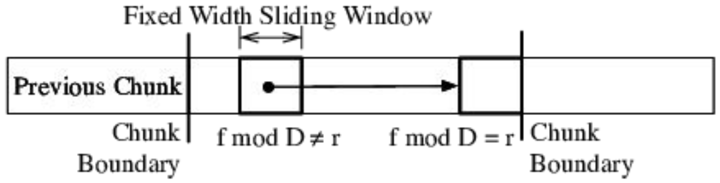


Handling Long Documents and Summarization

Challenge: Long documents can overwhelm LLMs, leading to high inference costs and slower performance.

Solutions:

- **Chunking: Breaking down long documents into smaller, manageable sections.**
- **Summarization: Generating concise summaries to reduce the amount of text the model needs to process.**
- **Sliding Window Approach: Moving a fixed-size window over the document to ensure all parts are processed without overwhelming the model.**



Batch Prompting Strategies

Sending multiple prompts in a single batch during inference to improve efficiency.

Benefits:

- Reduced Latency: Faster processing by grouping similar prompts.
- Lower Costs: Fewer calls to the model, leading to lower computational expenses.
- Implementation: Best suited for scenarios where multiple similar queries can be processed together.

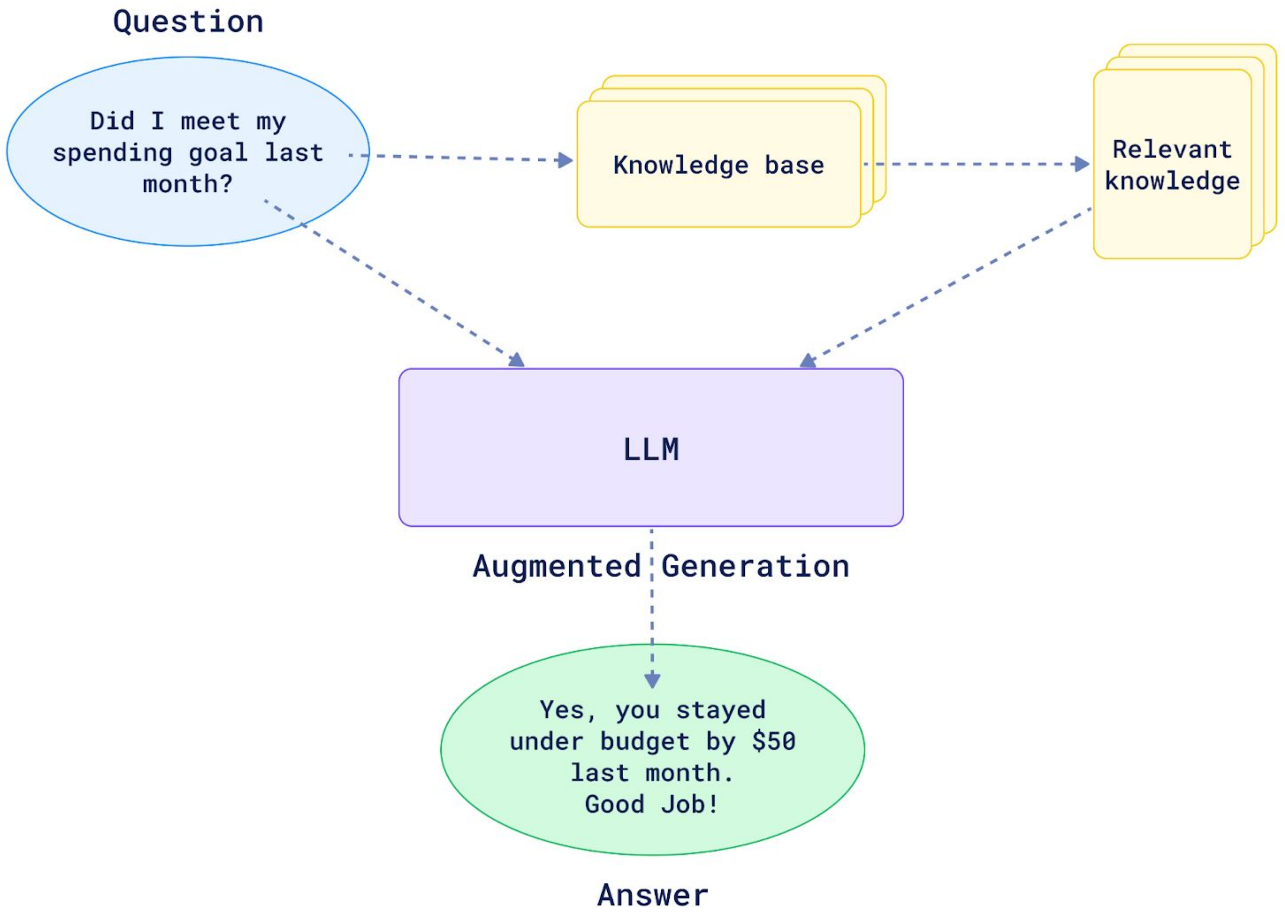
Key Takeaways

Inference Techniques are critical for managing costs in real-time and large-scale applications.

Advanced Strategies like prompt engineering, caching, and batch prompting can significantly enhance efficiency.

Practical Applications: Use these techniques to improve the efficiency of projects like chatbots, recommendation systems, or content summarization tools.

Retrieval Augmented Generation (RAG)?


Question
Did I meet my spending goal last month?
Knowledge base
Relevant knowledge
LLM
Augmented Generation
Yes, you stayed under budget by $50 last month. Good Job!
Answer

RAG

- Retrieving relevant information from a database or knowledge base using a query generated by the LLM
- Integrating the retrieved information into the LLM's input, enabling it to generate more contextually relevant text
- Leveraging vector databases to efficiently store and retrieve data based on semantic similarity

RAG

- Provides up-to-date and accurate responses by using external data sources instead of just static training data
- Reduces inaccurate or fabricated responses (hallucinations) by grounding the LLM's output in relevant external knowledge
- Generates domain-specific, relevant responses tailored to an organization's data
- Is efficient and cost-effective compared to fine-tuning or customizing LLMs

Demo

https://colab.research.google.com/drive/1R2w_YbjKfoOF_6fzCy2tswKUHU2HnQIR?usp=sharing

Pre Exercise

Modify the current code to use sentence transformers not Google Gemini.

pre-ex.*

Exercise 1: Telegram Bot using RAG + LLM

- By using the previous labs create a telegram bot that answer questions about our course (lecture 1, lecture 2, lecture3) slides:
- Google Gemini as LLM.
- Chromadb as VectorDB, feel free to use another type too.

Use our slides from the lecture as TEXT, every slide as Document.

Bot should answer only from the slides.

Bot should not call LLM if users write the same question and answer from previous answers.

Optional Exercise 2: Telegram Bot using RAG + LLM

Modify the previous bot to handle batch prompting, every three questions should be answer by one call to the LLM API.

hint : using Prompt engineering

Resources

- https://medium.com/the-ai-forum/semantic-chunking-for-rag-f4733025d5f5
- https://research.trychroma.com/evaluating-chunking
- https://www.perplexity.ai/search/what-is-rag-wCJdqvJCQGKs26yw8ffWZA