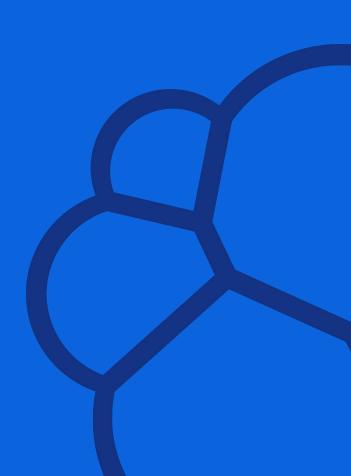
Clean Your Filthy RAGS!

Optimizing, Accelerating, and Evaluating RAG Applications

Justin Castilla







Justin Castilla

Developer Advocate @ Elastic

justin.castilla@elastic.co



Agenda

This lesson will have the majority of the text here, and the **repository** will house the code.



01-Recap

What is a RAG?



Learn why the use of chunking is advised for all text documents of a certain size

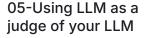


When you have no words to embed but plenty of data points, you need to get creative.

04-Semantic Caching

Don't waste your tokens and user's time on the same questions over and over!





Work on the response by exploring prompt engineering and fine-tuning our Natural Language requests around the context sent

06-Conversation

Connect the code to a Gradio UI library to have a conversation with our RAG application



Recap - What is a RAG?



Chunking

Tokens

 not quite words, but building blocks of understanding

Token Limitations

 processing these token groupings is expensive

Vector Limitations

 vectors can only hold so much information

Chunking

 meeting the vector requirements and reducing cost



Hypothetical Documents

Challenge:

- You have plenty of data in your documents: dimensions, costs, location, product_name, category
- No real semantic or contextual information
- Vectors don't run well against non-contextual data

Opportunity:

- Have an LLM consume your document and create the ideal customer query about that product
- Store that query as a vector attached to your document
- Search your documents with user query and LLM query!



Semantic Caching

Cost considerations:

- You pay for every token going to the LLM
- Users ask a lot of questions.
 - Many are the same **
 - You are paying multiple times for the same query

Cost savings:

- Park your previous question and answer pairs between your VectorDB and LLM
- Check it first to see if a similar question has been asked
- If it is similar enough (within a threshold of similarity), send it!



Using LLM-as-a-Judge to evaluate your LLM

Evaluating your RAG

- Ensure you keep a low quantity of hallucinations, incorrect answers, curtness, or non-sequitur responses
- Identify lacks in knowledge where your data doesn't match what the user needs

Creating your Judge

- Use an LLM to evaluate the quality of your LLM's response
 - Create prompts that identify correctness/incorrectness
 - Train with simple tables of examples



Using LLM-as-a-Judge to evaluate your LLM

- Ask <u>Binary</u> questions, not quantitative
 - Is the tone professional?
 - Is the response concise?
 - o Is the answer faithful to the source material?

- Keep a consistent evaluation criteria
 - Ask your judge to explain reasoning step by step.
 - Have them give reasoning
 - create an evaluation matrix for each one



Using LLM-as-a-Judge to evaluate your LLM

How: create prompts for your Judge LLM

- They won't have large result sets sent to them like your main LLM, so they can take big prompts with lots of instructions
- Iterate on your prompts
- Act on your gathered metrics with updated prompts for your main LLM

This may or may not work for your use case

- Try a jury of LLMs instead of a judge to meet quorum
- It might be to have a simple thumbs up/down near your answer
- Evaluation is a fuzzy area



Thank you!

