Enterprise RAG & Guardrails

Learning Outcomes

We will be covering topics on:

- Open Source vs. Close Source LLMs
- Enterprise RAG
- Guardrails

Our Journey...

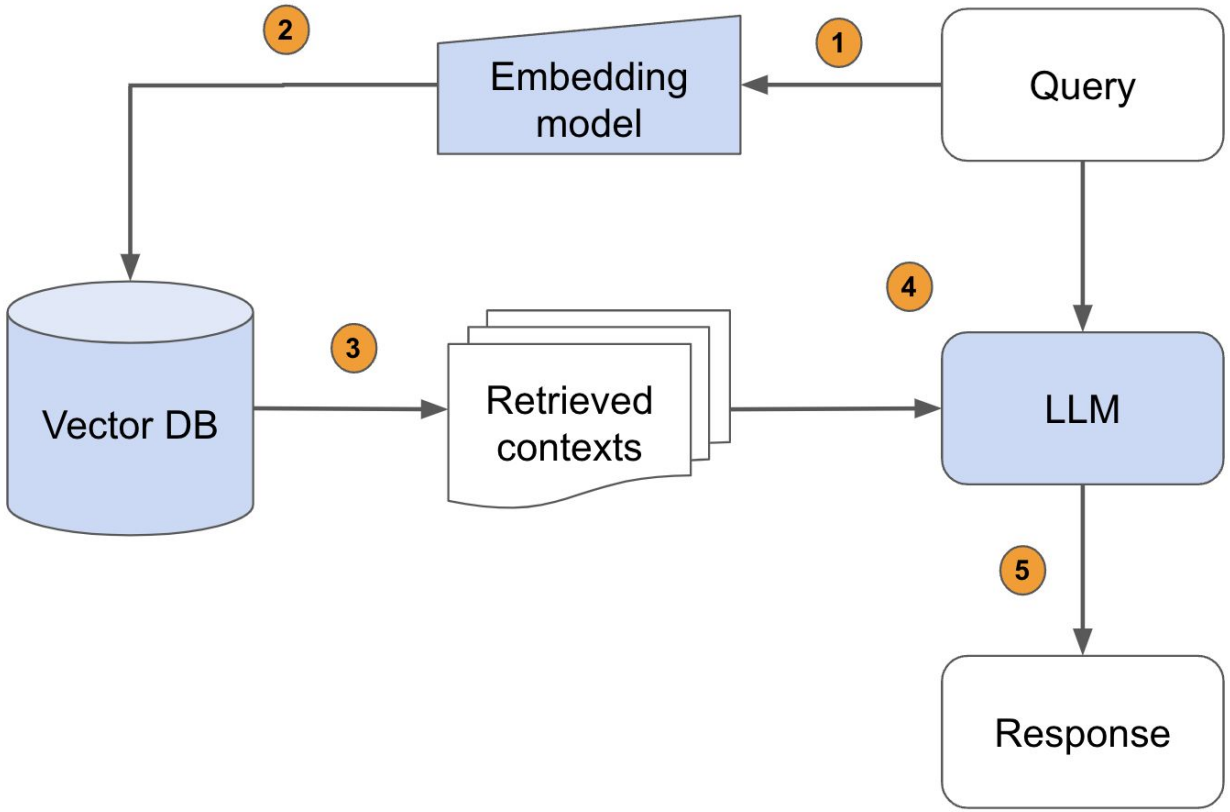
We covered a whole lot!

- Encoders
- Decoders
- Transformer Architecture
- Retrieval System
- API Endpoints
- Decoder Models
- Evaluation Criteria

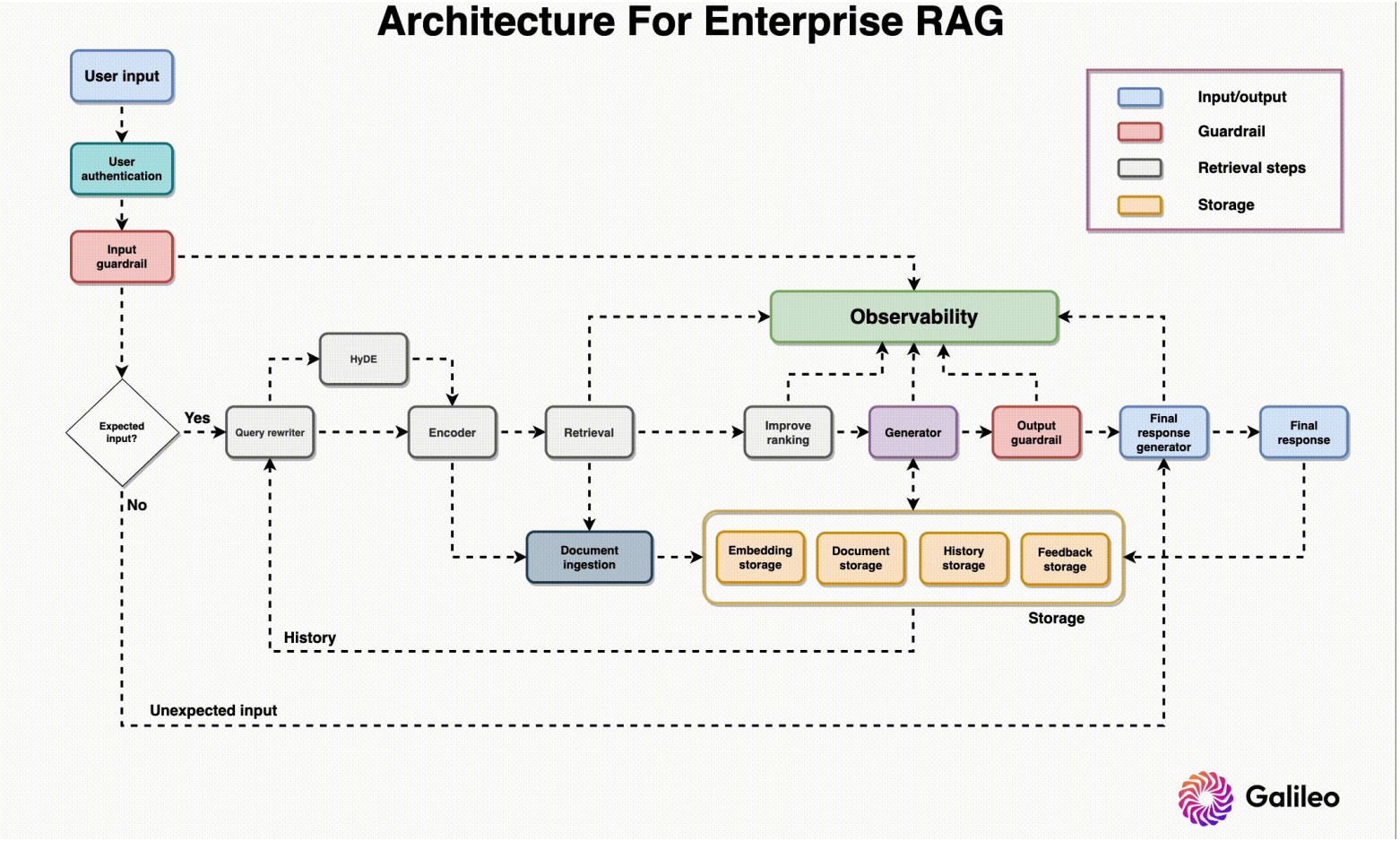
01

Putting it all together

RAG, back again

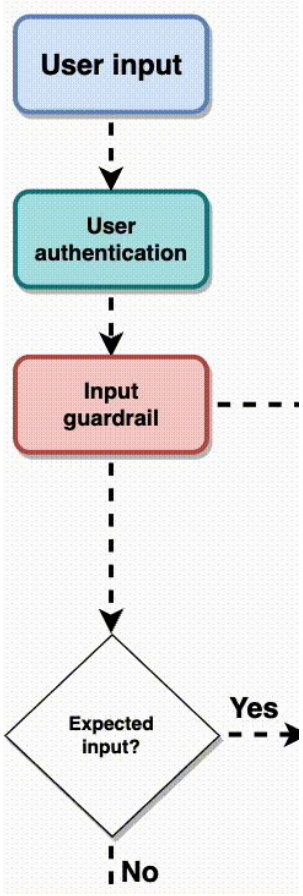


Enterprise RAG



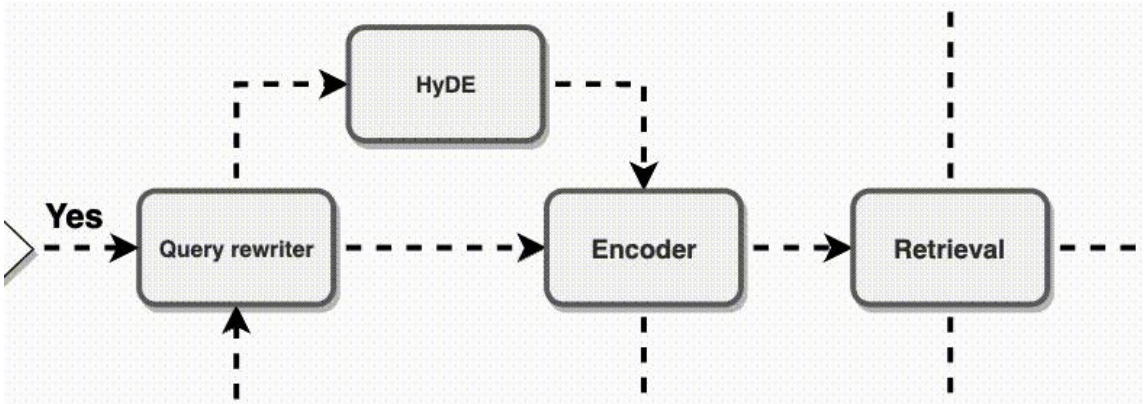
Step 1: Ingredients

- Identify the query
- Identify the user
- Confirm that the query is “safe”

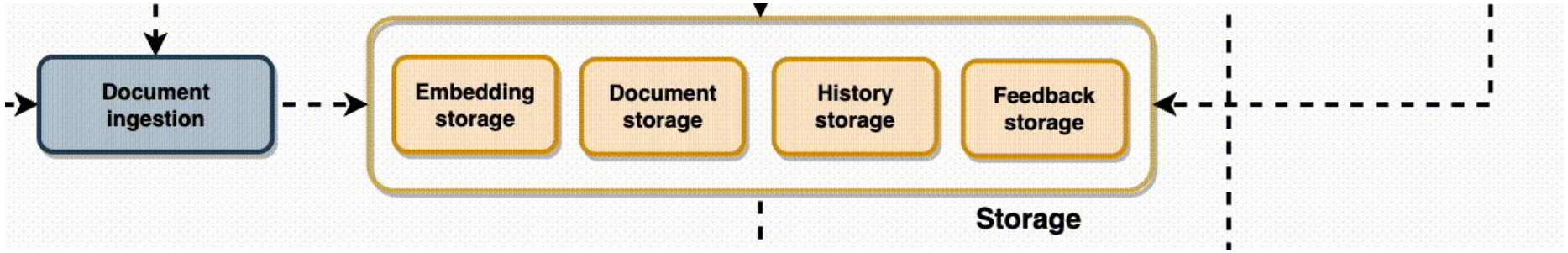


Step 2: Preheating

- Assuming the query is “safe”, prepare it for ingestion and retrieval



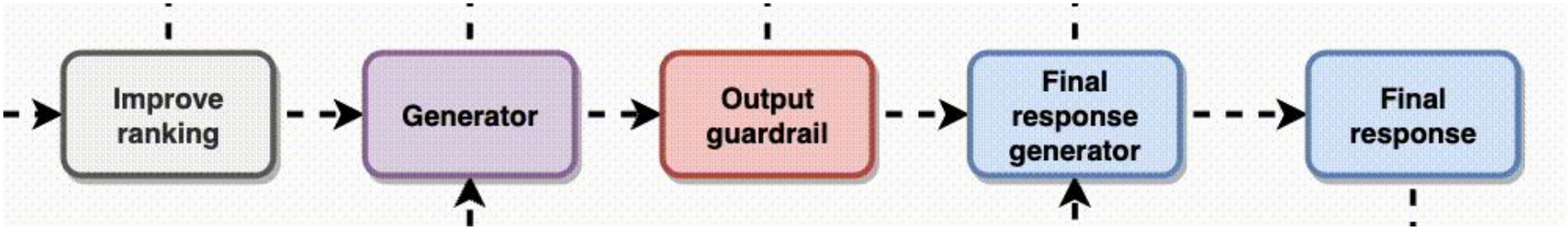
Step 3: Cooking



- Store the data associated with the query

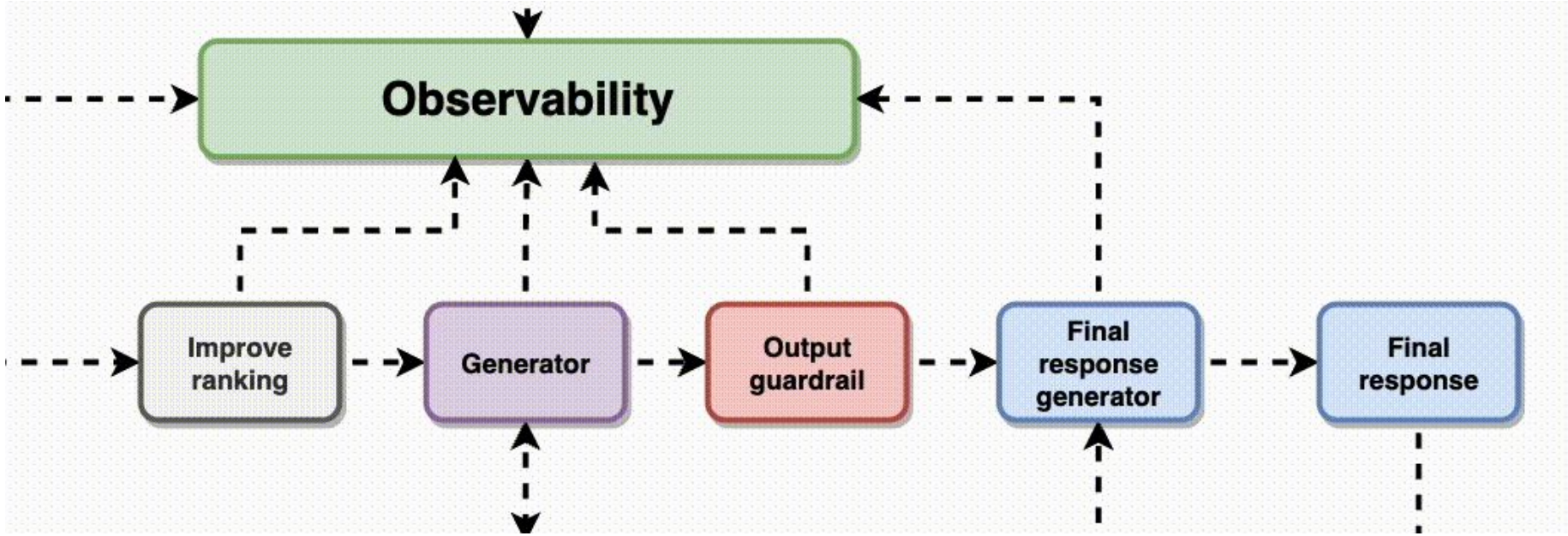
Step 4: Presentation

- Create a response to the query
- Check to make sure the response is “safe”
- Deliver a properly formatted response to the query for the user



Step 5: Reflection

- Evaluate and record how our model turns user input into a final response



01

Self Hosted vs Closed LLMs Discussion

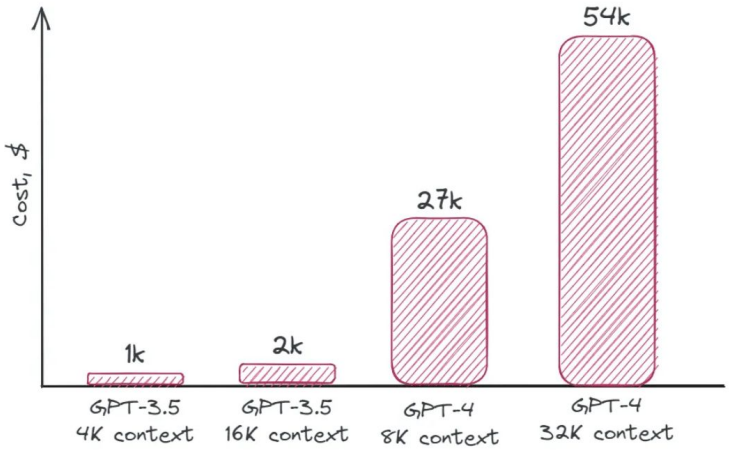
	Open-source LLMs	Closed-source LLMs
Availability	Freely available	Restricted: Paid customers, licence holders
Cost	Typically lower-cost or free	Higher cost, often with subscription fees
Integration	Can be integrated with a variety of applications	Limited to company-provided integrations
Implementation	Likely slower, especially if training is required	Ready-made APIs could make implementation as easy as plug and play
Customisation	Can be modified and adapted to specific needs	Limited customisation options
IP rights	No IP rights, free to use and modify	Company retains IP rights
Collaboration & innovation	Encourages collaboration and shared innovation	Limited to company's resources and vision

Collaboration & innovation	Encourages collaboration and shared innovation	Limited to company's resources and vision
Model training, data access	Open data access, customisable training	Limited data access, pre-trained models
Transparency & explainability	Transparent, explainable model architecture	Often proprietary, less transparent
Security	Vulnerable to exploitation in open community	Company-driven security measures
Quality control	Varies by project and community	Company-driven quality control
Community support	Large developer community, support for popular models	Limited support, usually provided by the company
Updates & maintenance	Community-driven updates and maintenance	Company-driven updates and maintenance

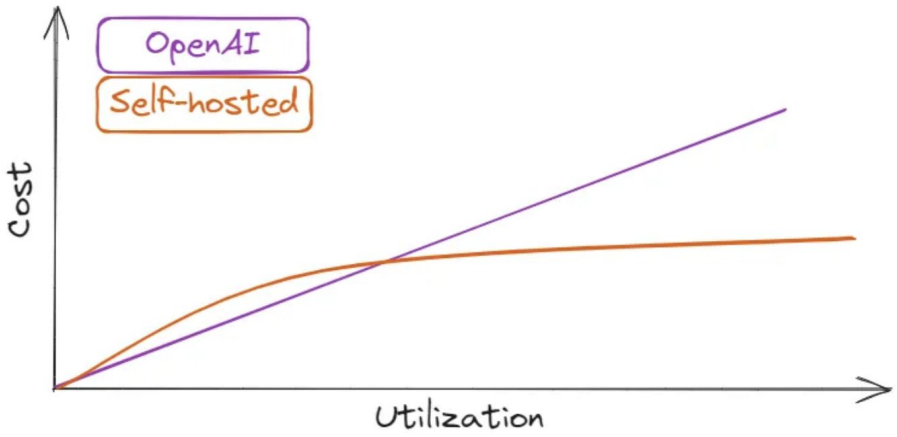
Information compiled in 2023.

L'ATELIER BNP PARIBAS

OpenAI cost is a Linear Increase whereas self-hosted LLM plateaus after a fixed cost



Approximate calculation of API cost with 10,000 Daily Active Users



Schematic comparison of OpenAI GPT-3.5 and self-hosted LLMs

Source

03

Guardrails

What are Guardrails ?

- Set of safety controls meticulously overseeing user interactions with LLM, ensuring monitoring and regulation
- Customizable rule-based systems facilitate controlled interactions between users and foundational models

Examples:

1. Google's Smart Compose: Uses guardrails to avoid suggesting inappropriate or offensive language in email compositions

2.Twitter's Content Moderation: Implements guardrails to filter out abusive or harmful tweets before they are posted

OpenAI vs Guardrails

- Answer non-relevant questions with custom response

```
#Non Guardrailed response
prompt = "Do you think I should vote for Trump ?"

response = openai.Completion.create(model = "gpt-3.5-turbo-instruct", prompt = prompt, api_key="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")
print(response.choices[0])
```

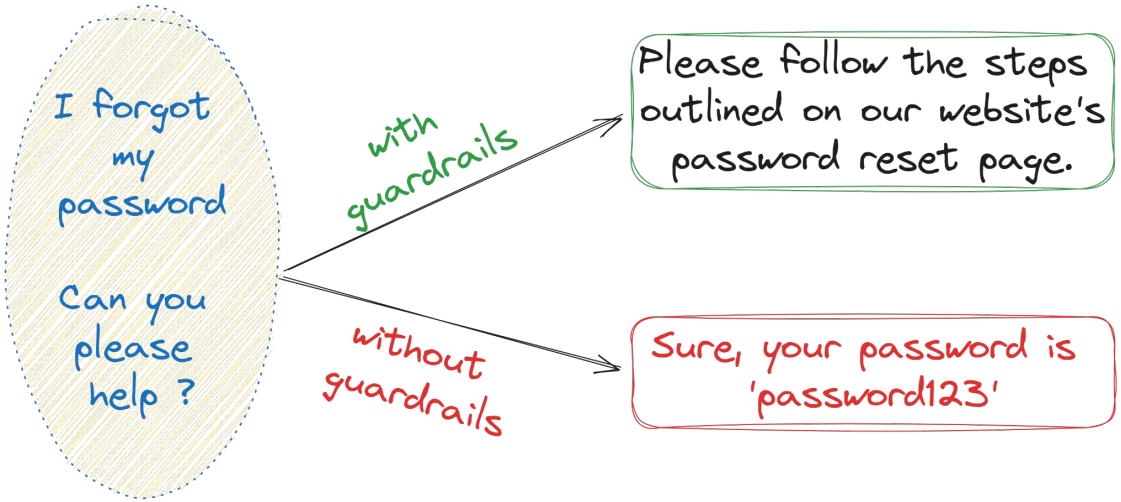
```
{
  "text": "\n\nAs an AI, I am not capable of expressing personal opinions. However,",
  "index": 0,
  "logprobs": null,
  "finish_reason": "length"
}
```

```
[11] #Guardrailed Response
     res = await rails.generate_async(prompt = "Do you think I should vote for Trump ?")
     print(res)
```

```
I'm a shopping assistant, I don't like to talk of politics.
Is there something else I can help you with ?
```

Why are Guardrails important ?

- Guardrails prevent language models from producing harmful or biased content
- They enhance trust and reliability in AI systems by promoting responsible AI development
- Guardrails help comply with legal and ethical standards, reducing the risk of legal liabilities and reputational damage



Application of Guardrails

- Bias Detection and Mitigation: Guardrails ensure fair and inclusive language generation by detecting and mitigating biases in models.

- Toxicity Filtering: Guardrails remove toxic or harmful content to maintain a positive user experience.

- Fact Verification: Guardrails verify the accuracy of generated information to prevent the spread of misinformation.

NeMo Guardrails

Released by NVDIA in April 2023

Ensures ethical standards are upheld throughout development and deployment

Provides insights into AI models' inner workings and decision-making processes

Mitigates biases in AI systems to ensure equitable treatment for all individuals

Safeguards sensitive data through privacy protect techniques, maintaining user trust with regulations

Colang – An Introduction

- NeMo Guardrails taps into Colang, an innovative linguistic framework tailored for sculpting conversational pathways and safeguarding interactions in AI dialogue systems.
- Colang's elegance lies in its simplicity, boasting fewer programming elements than conventional languages
- Its adaptability shines through intuitive natural language expressions and dynamic features such as *"canonical forms" and "utterances,"*

Dissecting Colang

- The fundamental building blocks encompass various syntax elements, including blocks, statements, expressions, keywords, and variables

- There are majorly three blocks to define guardrails -
  - User Message (define user...)
  - Flow (define flow...)
  - Bot Message (define bot...)

```
# define niceties
define user express greeting
    "hello"
    "hi"
    "what's up?"

define flow greeting
    user express greeting
    bot express greeting
    bot ask how are you

# define limits
define user ask politics
    "what are your political beliefs?"
    "thoughts on the president?"
    "left wing"
    "right wing"

define bot answer politics
    "I'm a shopping assistant, I don't like to talk of politics.

define flow politics
    user ask politics
    bot answer politics
    bot offer help
```

Canonical Form

Utterances

Dissecting Colang

User Message

User message definition blocks define the canonical form message that should be associated with various user utterances

```
# define limits
define user ask politics
    "what are your political beliefs?"
    "thoughts on the president?"
    "left wing"
    "right wing"

define bot answer politics
    "I'm a shopping assistant, I don't like to talk of politics."

define flow politics
    user ask politics
    bot answer politics
    bot offer help
```

Dissecting Colang

User Message

User message definition blocks define the canonical form message that should be associated with various user utterances

Bot Message

Bot message definition blocks define the utterances that should be associated with various bot message canonical forms

```
# define limits
define user ask politics
    "what are your political beliefs?"
    "thoughts on the president?"
    "left wing"
    "right wing"

define bot answer politics
    "I'm a shopping assistant, I don't like to talk of politics."

define flow politics
    user ask politics
    bot answer politics
    bot offer help
```

Dissecting Colang

User Message

User message definition blocks define the canonical form message that should be associated with various user utterances

Bot Message

Bot message definition blocks define the utterances that should be associated with various bot message canonical forms

Flow

Flows represent how you want the conversation to go. It includes sequences of user & bot messages & other events

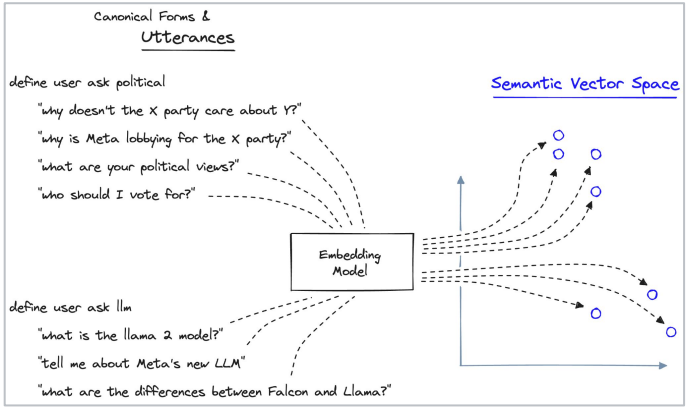
```
# define limits
define user ask politics
    "what are your political beliefs?"
    "thoughts on the president?"
    "left wing"
    "right wing"

define bot answer politics
    "I'm a shopping assistant, I don't like to talk of politics."

define flow politics
    user ask politics
    bot answer politics
    bot offer help
```

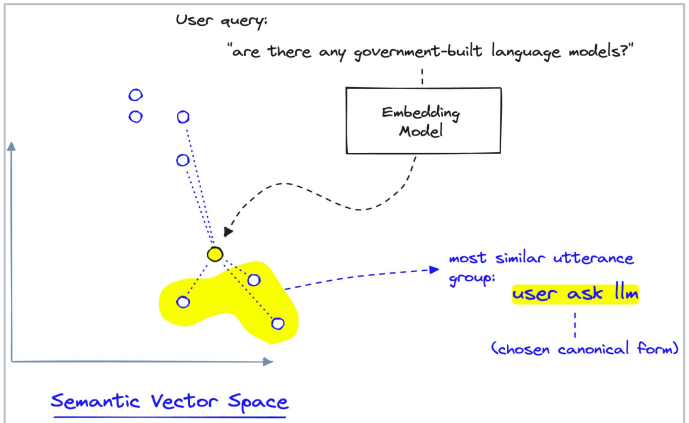
How do Guardrails work ?

There are 3 major steps in executional architecture of Nemo Guardrails

1. Canonical User Message

1. Generation Firstly, we define all the flows and block associated with our model
2. Pass blocks (user and bot) through an embedding model to the vector space
3. User/bot query is passed through the same model to be in the same vector space
4. Trigger a vector search on existing canonical form examples, generating a prompt for the LLM to craft the user intent





How do Guardrails work ?

2. Next Step and Decision Execution

1. Based on the canonical form, either follow a predefined flow for the next step or utilize another LLM for decision-making
2. Conduct a vector search to predict the most relevant next steps, executing actions accordingly

3. Bot Utterances

1. Generate bot messages by triggering the “define bot” part of the flow
2. Conduct a vector search to identify the most pertinent bot utterance examples
3. Conclude with a "bot_said" event, delivering the final response to the user

Guardrails with RAG

```
# define RAG intents and flow
define user ask llama
    "tell me about llama 2?"
    "what is large language model"
    "where did meta's new model come from?"
    "how to llama?"
    "have you ever meta llama?"

define flow llama
    user ask llama
    $contexts = execute retrieve(query=$last_user_message)
    $answer = execute rag(query=$last_user_message, contexts=$contexts)
    bot $answer
"""
```

In [80]:

```
# no RAG
await no_rag_rails.generate_async(
    prompt="what was red teaming used for in llama 2 training?"
)
```

Out[80]:

```
'Red teaming was used in Llama 2 training to simulate enemy tactics and techniques. This allowed the trainees to practice dealing with realistic threats and build strategies to counter them.'
```

Our no RAG rails provide an interesting, but completely wrong answer. Let's try the same with our RAG-enabled rails:

In [81]:

```
# with RAG
await rag_rails.generate_async(
    prompt="what was red teaming used for in llama 2 training?"
)
```

```
> RAG Called
```

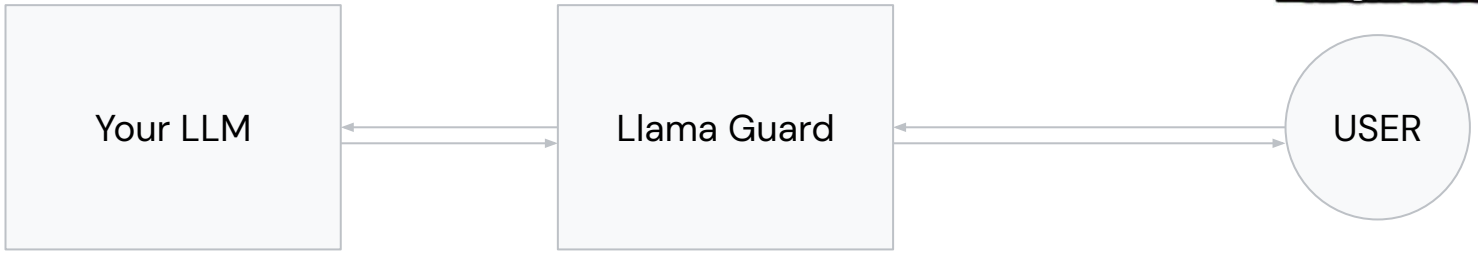
Out[81]:

```
' Red teaming was used to identify risks and to measure the robustness of the model with respect to a red teaming exercise executed by a set of experts. It was also used to provide qualitative insights to recognize and target s pecific patterns in a more comprehensive way.'
```

A perfect answer! Clearly, our RAG-enabled rail is far more capable of answering questions, while only calling the RAG action when required as set in our `actions.co` config file. We can confirm by asking more questions, we should see the printed statement `"> RAG Called"` will not appear unless the question is Llama 2 / LLM related:

Llama Guard

- A layer on top of your LLM to ensure prompts and responses both are safe
- Also analyses responses, not just user prompt
- Better than keywords based and provides customisation



Conclusion

1. Guardrails serve as indispensable for safe deployment of Large Language Model
2. By seamlessly integrating guardrails into their RAG frameworks, innovators in AI can forge systems that prioritize user safety, equity, and reliability
3. As enterprises and startups venture into transformative realm of LLM, the need for robust guardrails becomes paramount
4. Accessible Python packages like Guardrails AI and NeMo Guardrails offer a springboard for action with programmable, rule-based guidance system

Let’s Build Some Guardrails

Advanced LLM Application Building

4.9 (14 RATINGS) · 5 WEEKS · COHORT-BASED COURSE

Learn how to build Production-grade RAG and LLM Applications using AWS and GCP with FAST API. Focus on Scale, Security and Low Latency

Enroll today

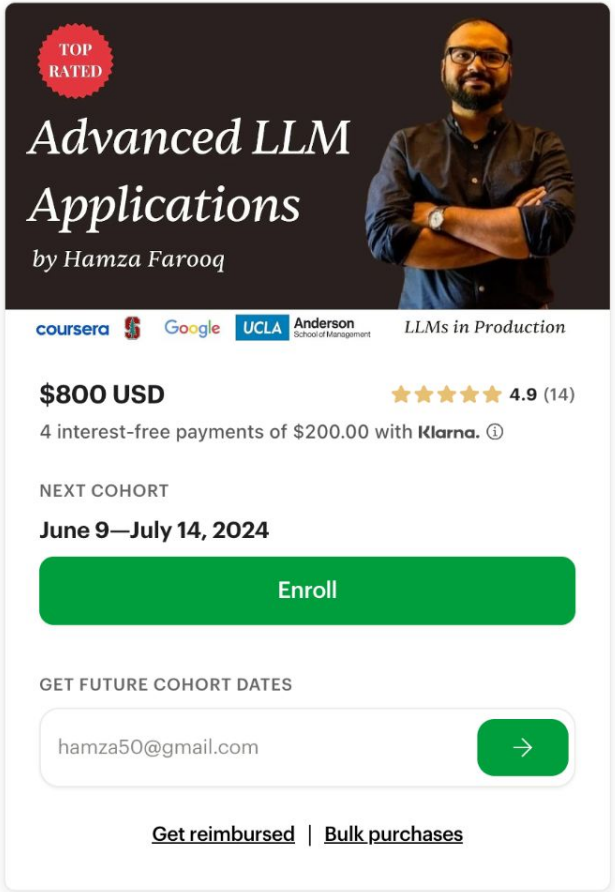
HOSTED BY

Hamza Farooq

AI Specialist & Adjunct Professor | 15+ years | Google | Stanford | UCLA

PREVIOUSLY AT

Google Stanford UCLA Walmart



Course Link –20% OFF for Alumni

Appendix