neosoftia

# LLM-Based Reasoning on Your Business Data

Albert Vilamala
Antoine Valverde

Barcelona, 20/03/2024

neosoftia

*“Empowering SMEs with AI Excellence”*

**Albert Vilamala, PhD**

*Co-founder & AI Consultant*

Recommenders / Computer Vision / NLP / Price Prediction / Signal Processing

**Antoine Valverde, MSc**

*Co-founder & AI Consultant*

Computer Vision / NLP / Information Retrieval

https://neosoftia.com

neosoftia

# Agenda

- Basic introduction to Large Language Models (LLMs)
- How can those LLMs help in my business?
- Adoption
- Retrieval Augmented Generation (RAGs)
- Advanced RAGs
- Essential Extras
- Conclusions
- Q & A

# 1.

# Large Language Models

# What is an LLM? Let's prompt

**You**

Explain what is an LLM in layman's term

**ChatGPT**

An LLM, or Large Language Model, is like a super-smart robot that has read a vast amount of text from books, websites, and all sorts of written materials. Imagine it as a sponge that has soaked up almost everything written down somewhere. Now, when you ask it a question or talk to it, it uses everything it has learned to come up with responses. It can write stories, answer questions, and even help with homework by recalling all the information it has absorbed. It's not perfect and doesn't know everything, especially new events or very niche topics, but it's like having a really knowledgeable friend who's read a lot of books and can chat about a wide range of topics.

# What is an LLM?

**You**

Explain it by creating a small comic

neosoftia

# Guess the prompt!

**You**
Explain what is an LLM to Homer Simpson

**You**
great, but make it shorter

**You**
repeat with the donut and beer

**ChatGPT**
Imagine a magical doughnut soaked in beer, which has read every book in the world and can chat with you about anything, just by munching on it!

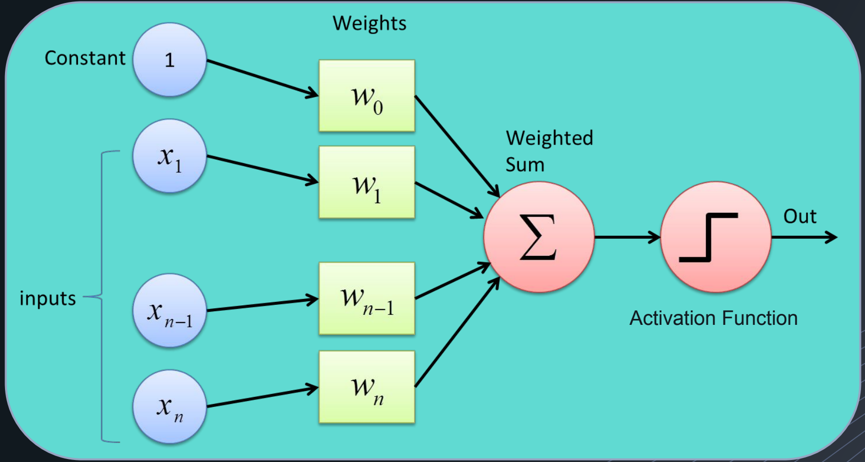
# What is an LLM?

Highly **sophisticated statistical models** that learn the complex probabilistic structure of language from large datasets.

$$\text{word}_1, \text{word}_2, \ldots, \text{word}_N \Rightarrow \text{word}_{N+1}$$

LLMs can perform a **wide range of language-related tasks,** such as text generation, translation, summarization, question answering, and more.

# The Perceptron




Invented in 1958 by Frank Rosenblatt

Reference: https://www.kaggle.com/code/ezzzio/introduction-to-the-perceptron

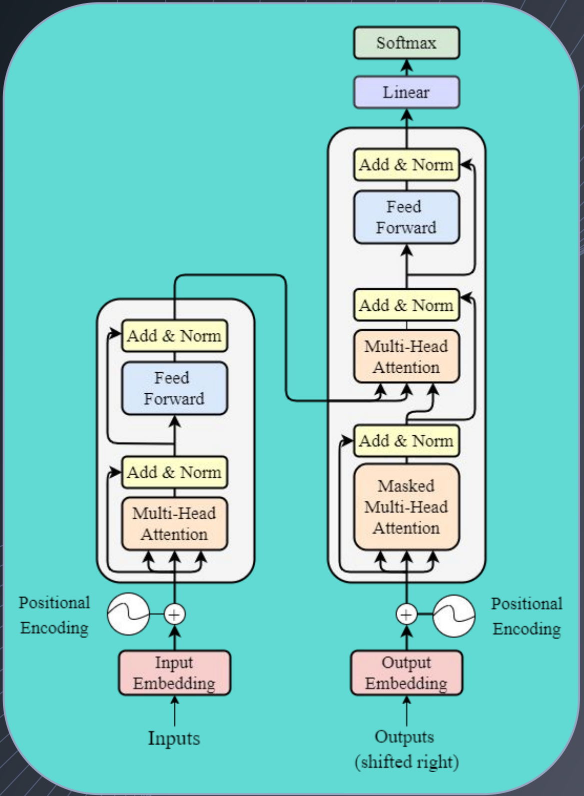
neosoftia

and after some years of cooking...

# Deep Neural Networks

- Huge amount of neurons smartly architectured
- 2012 (Computer Vision revolution): “ImageNet Classification with Deep Convolutional Neural Networks”, Krizhevsky et al.
- 2017 (NLP revolution): “Attention is all you need”, Vaswani et al.
- GPT-4 (Large Language Model): ~1.7 trillion weights

Transformer architecture

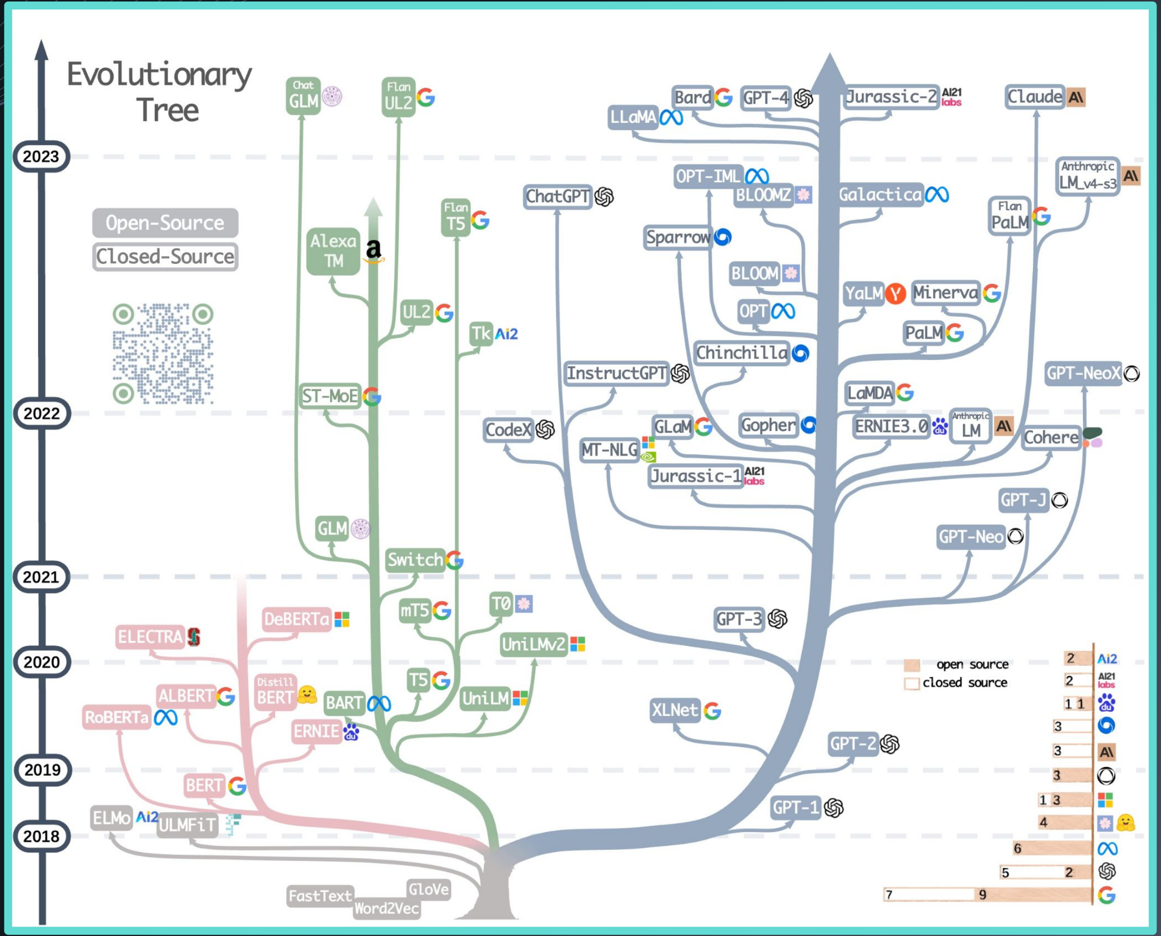



Reference: https://arxiv.org/abs/1706.03762

neosoftia

# Hype of LLM Models

2022-2023 pivotal year




Reference: https://github.com/Mooler0410/LLMsPracticalGuide

# Trends

- MultiModal (e.g. GPT-5, Gemini)
- Small Language Model - SML (e.g. Mistral-7B, Falcon-7B, and Phi 2)
- Specific LLM (e.g. BloombergGPT, Google's Med-Palm)
- Looking for Freshness (e.g. Perplexity)
- AI agents towards AGI

neosoftia

# How are LLMs being trained?

1. Pre-train on large chunk of internet (self-supervised learning)

   Once | upon | a | <MASK> — **Guess the next word**

2. Fine-tune for downstream task (supervised learning)

   E.g. Instruction fine-tuning

3. Align to generate human-preferred outputs (RL learning)

   E.g. Use user feedback

neosoftia

# 2.

# How can those LLMs help in my business?

# Use Cases

- Customer Support Automation
  - *Zendesk:* LLM chatbot  to provide customer support
- Content Creation and Summarization
  - *Copy.ai:* platform that creates blog posts
  - BBC uses CharismaAI to create interactive stories
- Data Analysis and Insight Generation
  - *JPMorgan* (DocLLM): classifies legal/finance docs and supports Question/Answering
- Personalization
  - *Spotify* is partnering with Google to extend recommendation to podcasts and audiobooks
- Language Translation
  - *Meta platform* uses LLM to translate social media content
- Development Support
  - Github Copilot, Amazon Code Whisperer, etc.

neosoftia

# Challenges

- Ethical and Safety concerns
- Bias
- Computational Resources
- Interpretability and Transparency
- Robustness / Hallucinations

neosoftia

# Use Case

## What defines a good use case?

- High volume of Textual unstructured data
- Complexity and semantic nuance in text
- Personalization requirements
- High effort of human generation vs Low effort of LLM validation
- Automate Routine type of tasks

## Checklist

- Define success metrics (Time savings,  process efficiency, etc.)
- Involve Domain Experts and Stakeholders
- Viability analysis (risk vs long-term potential ROI, data availability, and compliance challenges.)
- Keep updated
- Iterate (start small, constraint list of users, gather feedback, etc..)

# 3.

# Adoption

neosoftia

# Four ways to use an LLM

### Out-of-the-box

Use any available LLM as is

### Pre-Training

Train a whole LLM from scratch

### Fine-Tuning

Adapt a pre-trained LLM to a specific domain

### Retrieval Augmented Generation (RAG)

Combine the LLM with an external knowledge base

neosoftia

# Out-of-the-box

Prompt Engineering: The art of defining specialized instructions to guide the LLM behavior.

## Zero-shot Prompting

**System**: Classify the text into neutral, negative or positive.

**Text**: I think the vacation is okay.
**Sentiment**:

## Few-shot Prompting

**System**: Classify the text into neutral, negative or positive.

**Text**: This is awesome!
**Sentiment**: Positive

**Text**:This is bad!
**Sentiment**: Negative

**Text**: What a horrible show!
**Sentiment**:

## Chain-of-Thought Prompting

I went to the market and bought 10 apples. I gave 2 apples to the neighbor and 2 to the repairman. I then went and bought 5 more apples and ate 1. How many apples did I remain with?

Let's think step by step.

Reference: https://www.promptingguide.ai

# Prompt Engineering tips

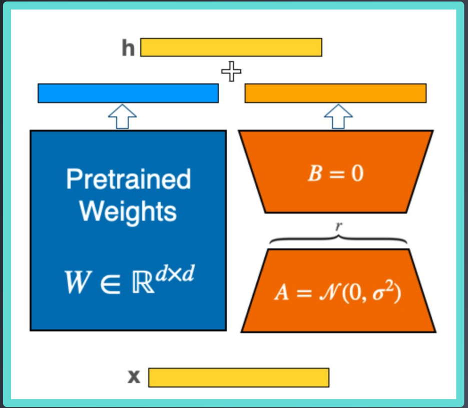
- Be specific and clear
  - Assign roles
  - Give context
- Specify language tone
- Constraint hallucinations
  - Allow the LLM to say "I don't know"
  - Only answer if the LLM is confident
- Put the questions at the end of the context in long queries (~10K tokens)

Experimentation & iteration are key

# Fine-Tuning

Adapt a pre-trained LLM to specific datasets or domains

- Gather domain-specific labelled dataset
- Update the weights of the model
  - Parameter-Efficient Fine-Tuning (PEFT)
  - E.g. Low-Rank Adaptation (LORA)




Reference: https://huggingface.co/docs/peft/main/en/conceptual_guides/lora

neosoftia

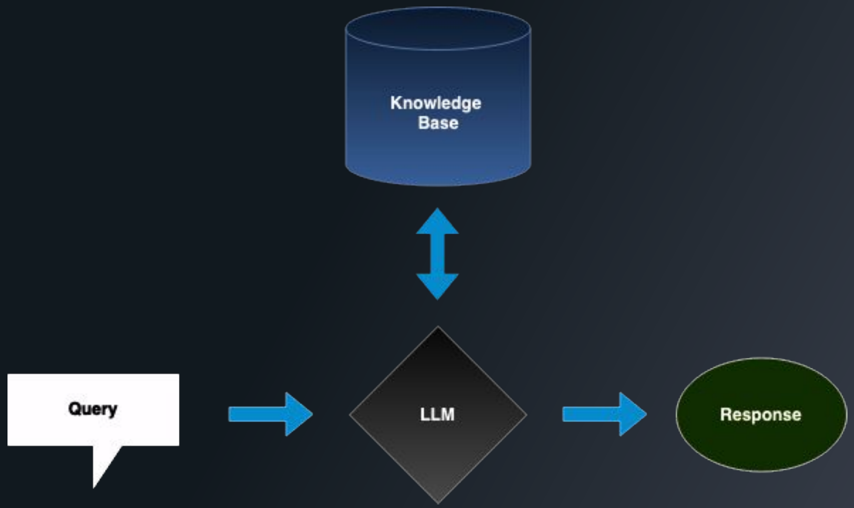
# Pre-Training

Train an LLM from scratch

- Requires huge amount of data
- Requires large amounts of computing resources
- Three step training:
    - Self-supervised pre-training
    - Supervised fine-tuning
    - Human-preferred alignment

# Retrieval Augmented Generation (RAG)

Combine the LLM with an external knowledge base



# Benefits & drawbacks

### Out-of-the-box LLM

- Ready to be used
- Access through API
- Prompt engineering

- Outdated knowledge
- No business data
- Low control
- Hallucinations

### Fine-tune an LLM

- Adapt to your specific business vocabulary
- Less prompting

- Need labelled data
- Need compute resources (GPUs)

### Train an LLM from scratch

- Completely tailored to your needs
- Full control of what it knows

- Huge demand for computing resources

### Retrieval Augmented Generation (RAG)

- Reason on your business data
- Less prone to hallucinations
- Updated knowledge
- Interpretability

- Requires well-engineered pipelines
- Higher latency

neosoftia

# Costs

- An online retailing company wants to expose an LLM based chatbot to interact with its customers in the FAQ section of its website.
- The expected traffic is 1M messages a month
- The Knowledge Base is composed of 1M document paragraphs

### Out-of-the-box

- LLM API calls: ~100$

### Fine-tune

- Training: > 100$
- LLM API calls: ~600$

### Train an LLM from scratch

- Several million dollars !

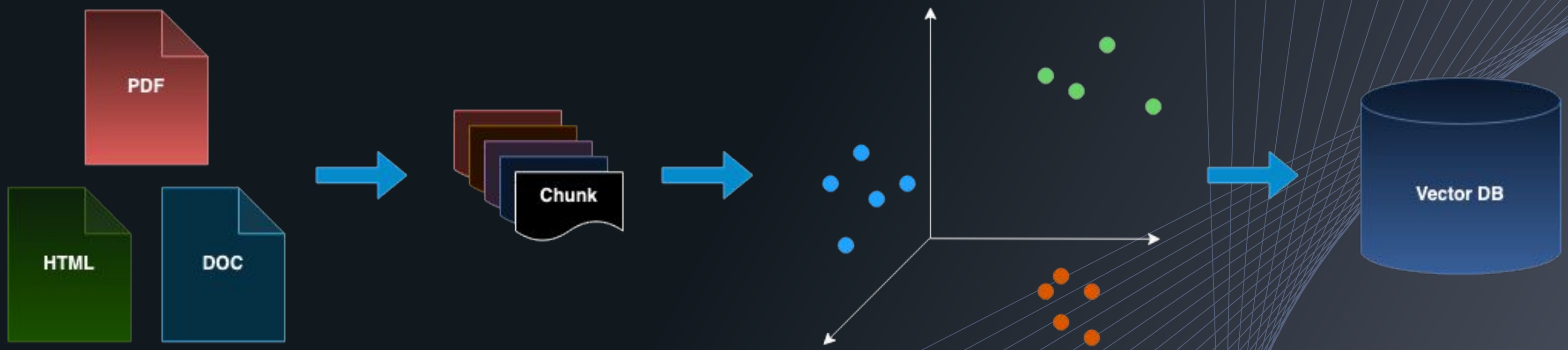
### RAG

- LLM API calls: ~400$
- Knowledge Base: ~150$

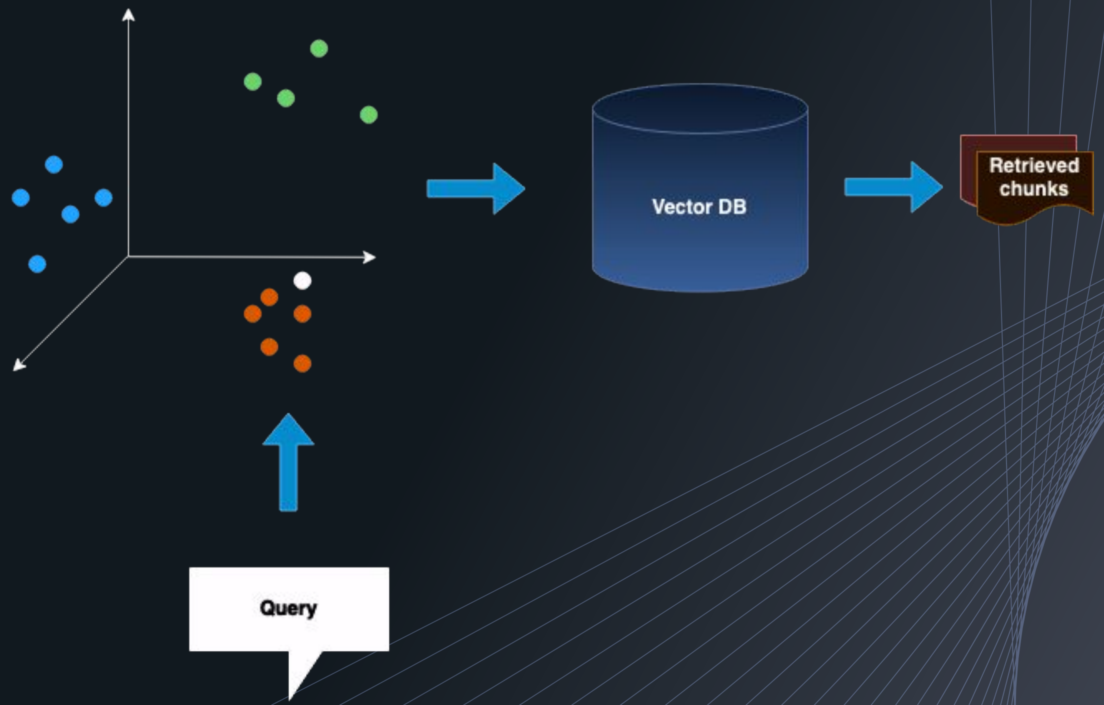
Tool: token counter

# 4.
# Naive RAG

# Phase 1: Indexing



# Phase 2: Retrieval



# Phase 3: Generation


How to orchestrate these components?

# Limitations of Naive RAG

### Retrieval

- Low precision
- Low recall
- Outdated information

### Augmentation

- Incoherent integration
- Redundancy / repetition
- Differences in writing style

### Generation

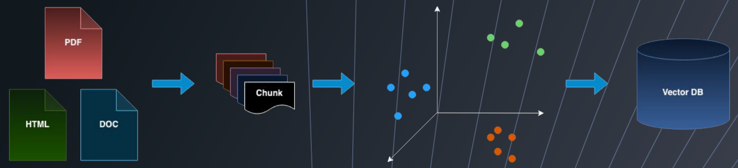
- Hallucinations
- Irrelevant content
- Toxicity / bias
- Reiterate retrieved context

Reference: Retrieval-Augmented Generation for Large Language Models: A Survey

# 5.

# Advanced RAG

# Phase 1: Indexing

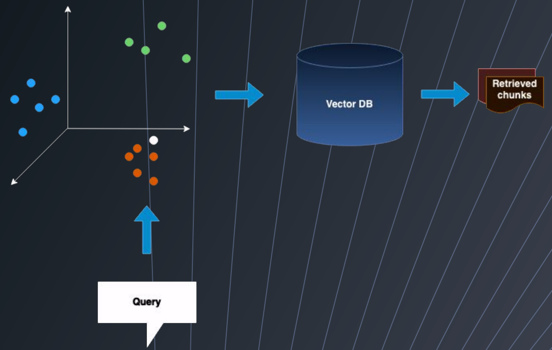



- Enhancing data granularity (e.g. removing irrelevant information)
- Optimizing index structures (e.g. adjusting the size of chunks, chunks relationship)
- Adding metadata (e.g. integrating referenced chapters for filtering purposes)
- Alignment optimization (e.g. hypothetical questions)

Reference: Retrieval-Augmented Generation for Large Language Models: A Survey

neosoftia

# Phase 2: Retrieval




- Query routing (e.g. retrieve information from Vector DB or Relational DB depending on the query)
- Query rewriting (e.g. rewrite the user query to better match the domain's jargon)
- Fine-tuning embeddings (i.e. customize embedding models for your domain)
- Dynamic embeddings (e.g. the same word can have varied embedding depending on surrounding context)

Reference: Retrieval-Augmented Generation for Large Language Models: A Survey

# Phase 3: Generation


- Reranking (i.e. rearrange retrieved documents to prioritize the most relevant ones at the top or the bottom of the context length)
- Prompt compression (e.g. condense retrieved information to fit context length)
- Fine-tune the LLM for RAG (i.e. fine-tuning the model to adapt to the input of both query and retrieved documents)

Reference: Retrieval-Augmented Generation for Large Language Models: A Survey

# Complex PDFs tips

- Complex Layout (nested, multi-column), Tables, Images, etc.

- Reader libraries:
  - PyPDF
  - PDFMiner
  - PDFPlumber

- Reader APIs:
  - Adobe PDF Extractor
  - Azure Document Intelligence
  - Llama Parse

- Vision LLMs:
  - GPT-V
  - LLaVA

neosoftia

# Complex PDFs tips

- Parser:
  - PDF ➡ MarkDown ➡ Graph-like structure
- Chunker:
  - Ensure every table chunk contains the headers
- Embedder:
  - For tables:
    - Consider use of regular DB search instead of embeddings
    - Consider embedding a summary of the table
- Mind the context size:
  - Although LLMs can interpret structural tags, they might blow up the number of tokens

# 6.
# Essential Extras

# LLM Evaluation

Measuring the performance of LLM apps is a critical step in the path from development to production.

- Provide tools to handle fine-tuning of these models (iterative process)
- Assure reliability and efficiency of model to optimize user experiences

Anecdote: The "generous" Bot of Air canada

# LLM Evaluation

LLM Model evaluation IS NOT LLM System evaluation

Model evaluation

- Overall performance and intelligence of the LLM on multiple tasks
- Testing in divergent cases

System evaluation (RAGs)

- Specific use-case effectiveness and integration within a system
- Test effectiveness within specific contexts

**Looking for standardisation!!**

# LLM Model Evaluation

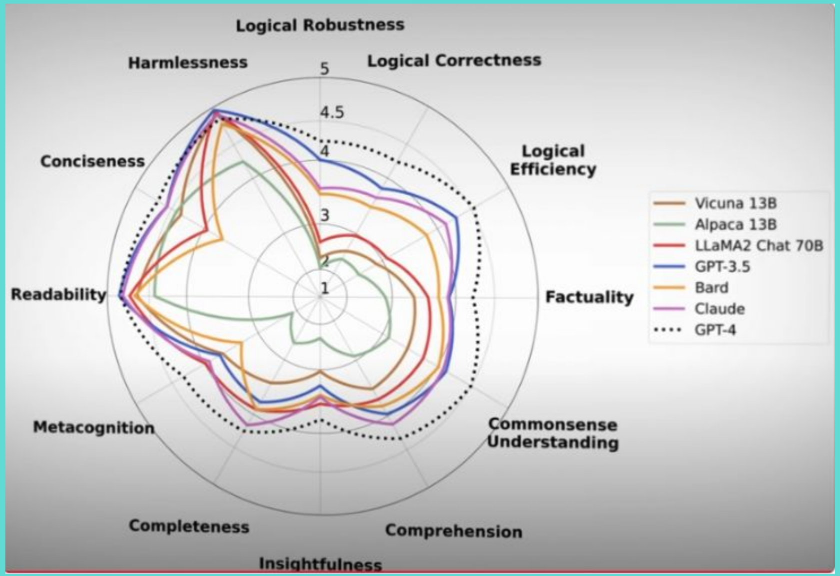
Understand the effectiveness of foundational models

Famous benchmarks:

- FLASK (Fine-grained Language Model Evaluation based on Alignment SKill Sets)
- HELM (Holistic Evaluation of Language Models).

Useful links:

- Hugging face embedding dashboard MTEB [link]
- Hugging face LLM dashboard [link]




Reference: Benchmark of large language model

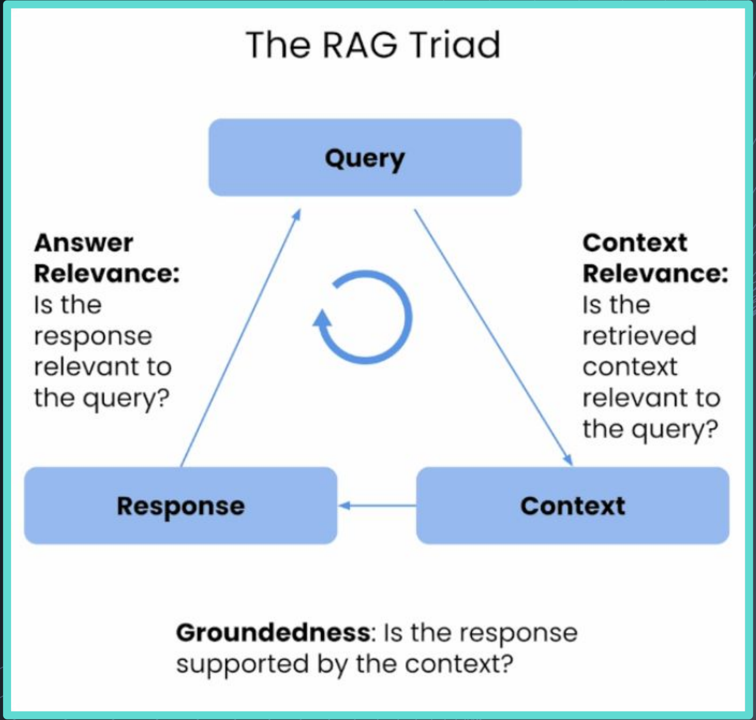
# System Evaluation: RAG Metrics

2 Targets:

- Retrieval
- Generation

3 primary Quality scores:

- Context Relevance (precision/recall)
- Groundedness / Faithfulness
- Answer Relevance




Reference: The RAG Triad

# System Evaluation: RAG Metrics

4 essential abilities:

- **Noise Robustness**
  - Capability to not be affected by data relevant but without information
- **Negative Rejection**
  - Capability to say "I do not know"
- **Information Integration**
  - Capability to integrate/summarize/gather different piece information
- **Counterfactual Robustness**
  - Capability to disregard errors in KB

Other: Toxicity, biais, etc..

Reference: Retrieval-Augmented Generation for Large Language Models: A Survey

# RAG Metrics: Practical insights

**Recommendation**

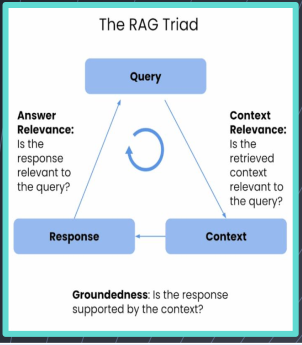
1. Optimize Retrieval metrics
2. Finetune Generation metrics

**Datasets**: Balance between curated manual and automatic dataset

- Create curated set of input/output [Dev/Test].
  - Agreement with Stakeholders (different from the ones that buy the project)
- Synthetic Datasets
  - Introduce noise to your curated dataset by modifying questions using an LLM
  - Create Question/Answer set with an LLM (e.g. using ragas tool)

neosoftia

# RAG Metrics: Practical insights

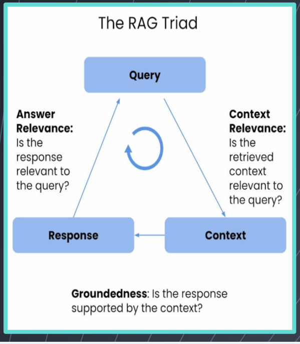



| Question | Response Gt | Context GT | RAG Context v1.0 | RAG Response v1.0 |
|---|---|---|---|---|
| Can my 11 years old child fly alone? | No, your child can not fly alone | - Kids < 12 y.o. can not fly alone | - Kids from 12 to 16 y.o. can fly alone<br>- Kids < 12 y.o. can not fly alone | Yes, your child can fly alone without their parents |

**Retrieval Metrics**:

- Context Relevance
  - Hit Rate (top-1, top-2), MRR, NDCG
  - Use an LLM to validate that the context is relevant to the question (e.g. Ragas, Trulens)

# RAG Metrics: Practical insights



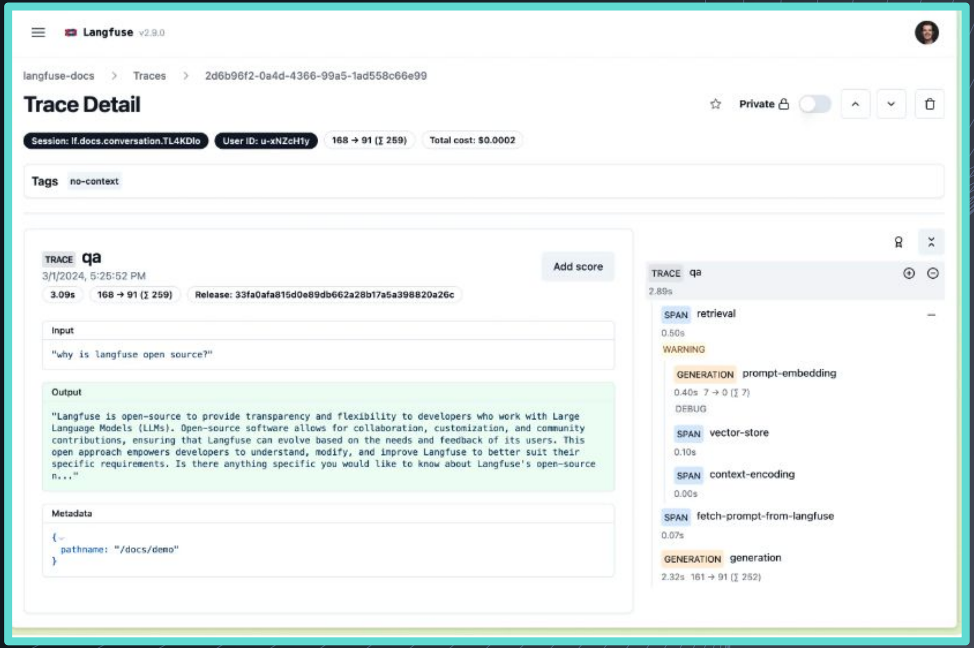

| Question | Response Gt | Context GT | RAG Context v1.0 | RAG Response v1.0 |
|---|---|---|---|---|
| Can my 11 years old child fly alone? | No, your child can not fly alone | - Kids < 12 y.o. can not fly alone | - Kids from 12 to 16 y.o. can fly alone<br>- Kids < 12 y.o. can not fly alone<br><br>Not | Yes, your child can fly alone without their parents |

**Generation Metrics**:
- Groundness:
  - Ask an LLM to validate if all claims of answers can be found in the retrieved context
- Answer Relevance:
  - Distance between GT and predicted answer
  - Distance between original question and set of questions generated by LLM, based on answer
  - Use an LLM to evaluate the relevance of the answer with respect to the question (e.g. Ragas)

# RAG Metrics: Practical insights

**Debug and Experiment tracking API/SAAS**

- Langfuse
- Langflow
- Trulens
- Weights & Biases



# Safeguards

- Hallucinations
- Mention of competitors
- Toxic language
- Accurate summarization
- PII Leakage
- Generating invalid / unsafe code
- Jailbreaking

### Tools

- GuardrailsAI
- WhyLabs
- NeMo Guardrails
- RAIL

Reference: "Safeguarding LLMs with Guardrails"

neosoftia

# 7.
# Conclusions

# Conclusions

- Choose the right use case, assess risk and ROI
- Adopt an iterative process
- Test it on high level system first (e.g. MS copilot 360)
- Put evaluation systems in place from the beginning
- Use LLMs to help you generate your development artifacts
- **Warning**: It’s easy to create a POC but difficult to get production compliant
- Hire a consultancy

neosoftia

# Thank you!

# Q & A

https://neosoftia.com