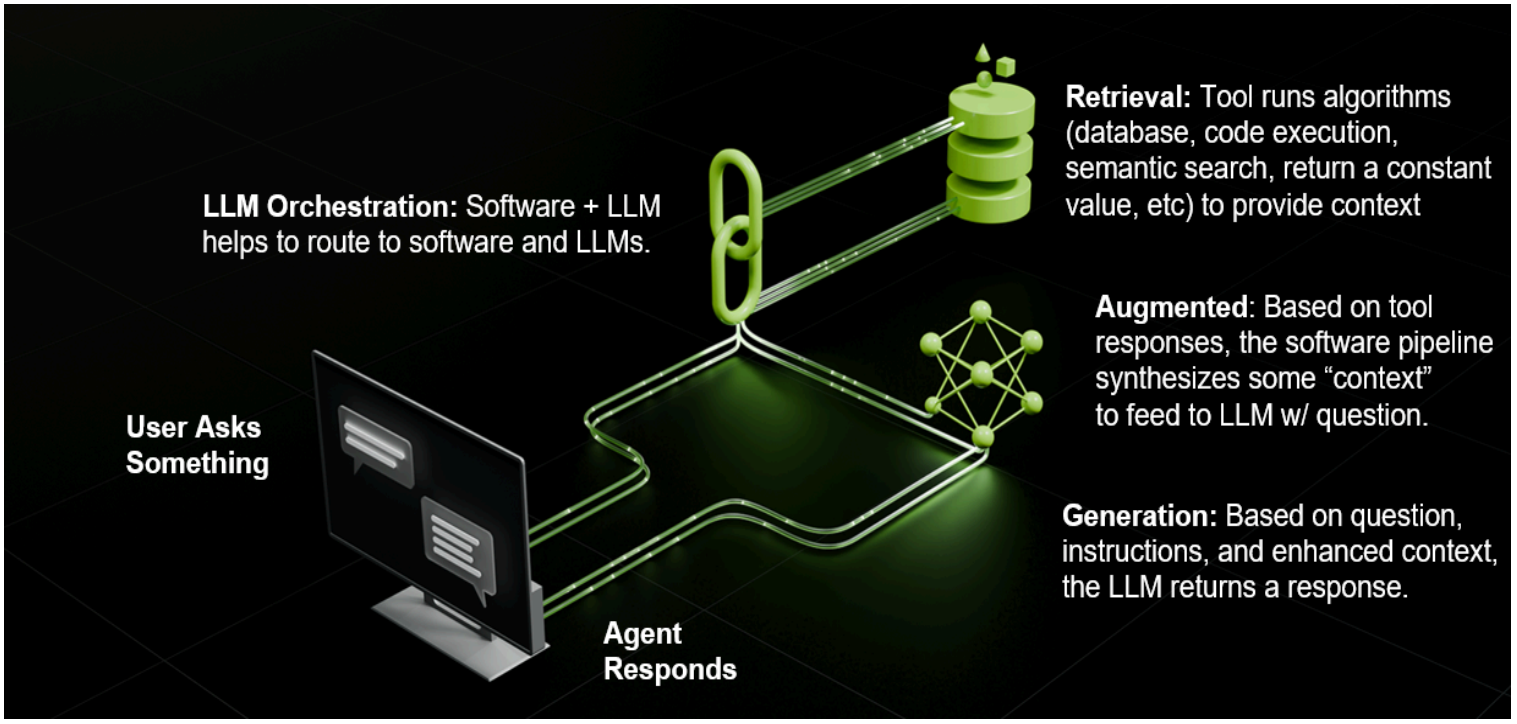
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
*** * ***
Université de Carthage
*** * ***
**Institut National des Sciences
Appliquées et de Technologie**

Building RAG Agents with LLMs

Prompt Engineering

- Presented by : Malek Zitouni

- Academic Year : 2024-2025



This image explains the concept of **Retrieval-Augmented Generation (RAG)** in the context of how large language models (LLMs) interact with software systems to respond to user queries.

Here's a breakdown of the diagram:

---

1. User Asks Something

- The process begins when a user asks a question or submits a request.

---

2. LLM Orchestration

- The system orchestrates between **software** and the **LLM** (Large Language Model). It ensures that the question or request is routed appropriately, deciding whether to involve external tools or rely directly on the LLM for a response.

---

3. Retrieval

- **What happens here?**
    - Algorithms or tools retrieve relevant information from various sources. Examples of these sources include:
        - **Databases**: For structured data.
        - **Code execution**: For running small programs or calculations.
        - **Semantic search**: For finding relevant documents or knowledge using meaning-based search techniques.
        - **Constant values**: For retrieving static or predefined data.
    - The goal is to collect context or information that will help the LLM provide a more accurate and specific response.

---

4. Augmented

- **What happens here?**
    - The software pipeline takes the retrieved information (from databases, code execution, or other tools) and synthesizes it into **context**.
    - This context is then sent to the LLM along with the user's original question.
    - For example:
        - If a user asks, "What is the latest stock price of Company X?"
        - The retrieval process gets the stock price from a financial database.

        - The augmented step formats this data into a context like: "The stock price of Company X as of today is $100."

---

5. Generation

- **What happens here?**
    - The LLM uses the question, instructions, and the enhanced context to generate a coherent and accurate response for the user.
    - For example, based on the previous step, the LLM might respond:
        - “The latest stock price of Company X is $100 as of today.”

---

6. Agent Responds

- Finally, the response is presented back to the user in a readable and understandable format.

Why Use RAG?

- **Efficiency**: The model does not need to “know” everything; it can fetch data dynamically.
- **Accuracy**: Incorporates real-time or external information, making responses more precise.
- **Scalability**: Can handle diverse tasks, from answering queries to performing calculations.

Welcome to Prompt Engineering with LLaMA-2

Prompt engineering drastically increases the capabilities of large language models (LLMs) with little effort. With a robust prompt engineering toolkit at your disposal you can customize LLM behavior to perform a diverse set of tasks and get more out of LLMs regardless of their size.

What is Prompt Engineering?

Prompt engineering is the practice of designing and optimizing input prompts to guide large language models (LLMs) like GPT-4, LLaMA-2, or Bard to produce desired outputs. Since LLMs generate responses based on the input they receive, crafting an effective prompt can drastically improve their performance for various tasks.

In simpler terms, it's about figuring out **how to ask the model the right way** to get the best results.

Why is Prompt Engineering Important?

- **Enhances LLM Performance:** A well-crafted prompt allows the model to perform complex tasks better without modifying or retraining the underlying model.
- **Unlocks Versatility:** Helps models perform a wide range of tasks like writing, summarization, coding, or answering questions.
- **Cost-Effective:** Eliminates the need for extensive training or fine-tuning of models.
- **Customizable Behavior:** You can shape how the model behaves or formats its output with precise instructions.

Key Concepts in Prompt Engineering

1. **Instructions:**
   Provide clear, explicit commands to the model.
   - Example: *"Summarize the following text in two sentences."*
2. **Context:**
   Include background information or details to guide the model.
   - Example: *"You are an expert chef. Explain how to cook pasta perfectly."*
3. **Examples (Few-shot Learning):**
   Show the model examples of the desired input-output behavior.

Example:
Q: What is the capital of France?
A: Paris
Q: What is the capital of Germany?
A:

-

4. **Constraints:**
   Set limits or boundaries for the output.

   - Example: *"Write a 50-word professional email declining a meeting request."*
5. **Formatting:**
   Use structured prompts for specific outputs.

Example:
Task: Translate from English to Spanish.
Input: "How are you?"
Output:

-

Applications of Prompt Engineering

1. **Text Generation:**

   - *"Write a short story about a futuristic city in 2050."*
2. **Summarization:**

   - *"Summarize this article in three bullet points."*
3. **Coding:**

   - *"Write a Python function to calculate the factorial of a number."*
4. **Content Creation:**

   - *"Generate a blog post outline about the benefits of AI in education."*
5. **Conversational AI:**

   - *"You are a friendly customer support agent. Respond to this question: 'How do I reset my password?'"*

6. **Translation:**
    - *"Translate this sentence from English to French: 'Good morning!'"*

Benefits of Prompt Engineering

- Maximizes the utility of LLMs without requiring advanced technical skills.
- Allows you to control tone, style, or structure of the output.
- Works across domains and industries (education, healthcare, programming, customer service, etc.).

Key Takeaway:

Prompt engineering is a powerful tool for getting the best results out of large language models. It's all about crafting the right inputs to guide the model toward performing tasks effectively and efficiently.

Introducing LLaMA: A foundational,

65-billion-parameter large language model

As part of Meta's commitment to open science, today we are publicly releasing LLaMA (Large Language Model Meta AI), a state-of-the-art foundational large language model designed to help researchers advance their work in this subfield of AI. Smaller, more performant models such as LLaMA enable others in the research community who don't have access to large amounts of infrastructure to study these models, further democratizing access in this important, fast-changing field.

Training smaller foundation models like LLaMA is desirable in the large language model space because it requires far less computing power and resources to test new approaches, validate others' work, and explore new use cases. Foundation models train on a large set of unlabeled data, which makes them ideal for fine-tuning for a variety of tasks. We are making LLaMA available at several sizes (7B, 13B, 33B, and 65B parameters) and also sharing a LLaMA model card that details how we built the model in keeping with our approach to Responsible AI practices.

Over the last year, large language models — natural language processing (NLP) systems with billions of parameters — have shown new capabilities to generate creative text, solve mathematical theorems, predict protein structures, answer reading comprehension questions, and more. They are one of the clearest cases of the substantial potential benefits AI can offer at scale to billions of people.

Even with all the recent advancements in large language models, full research access to them remains limited because of the resources that are required to train and run such large models. This restricted access has limited researchers' ability to understand how and why these large language models work, hindering progress on efforts to improve their robustness and mitigate known issues, such as bias, toxicity, and the potential for generating misinformation.

Smaller models trained on more tokens — which are pieces of words — are easier to retrain and fine-tune for specific potential product use cases. We trained LLaMA 65B and LLaMA 33B on 1.4 trillion tokens. Our smallest model, LLaMA 7B, is trained on one trillion tokens.

Like other large language models, LLaMA works by taking a sequence of words as an input and predicts a next word to recursively generate text. To train our model, we chose text from the 20 languages with the most speakers, focusing on those with Latin and Cyrillic alphabets.

There is still more research that needs to be done to address the risks of bias, toxic comments, and hallucinations in large language models. Like other models, LLaMA shares these challenges. As a foundation model, LLaMA is designed to be versatile and can be applied to many different use cases, versus a fine-tuned model that is designed for a specific task. By sharing the code for LLaMA, other researchers can more easily test new approaches to limiting or eliminating these problems in large language models. We also provide in the paper a set of evaluations on benchmarks evaluating model biases and toxicity to show the model's limitations and to support further research in this crucial area.

To maintain integrity and prevent misuse, we are releasing our model under a noncommercial license focused on research use cases. Access to the model will be granted on a case-by-case basis to academic researchers; those affiliated with organizations in government, civil society, and academia; and industry research laboratories around the world. People interested in applying for access can find the link to the application in our research paper.

We believe that the entire AI community — academic researchers, civil society, policymakers, and industry — must work together to develop clear guidelines around responsible AI in general and responsible large language models in particular. We look forward to seeing what the community can learn — and eventually build — using LLaMA.

Key Characteristics of a Foundation Model:

A **foundation model** is a large machine learning model that is trained on a broad and diverse dataset at scale and can be adapted (fine-tuned) for a wide range of downstream tasks. These models are "foundational" because they serve as a base upon which other, more specific applications are built.

---

1. **Pretraining on Large-Scale Data**:
    - The model is trained on massive datasets, often sourced from the internet (e.g., text, images, code, etc.), using self-supervised learning techniques.
    - The goal is to create a general understanding of the domain (e.g., language, images).
2. **General Purpose**:
    - Foundation models are not designed for a specific task. Instead, they learn general representations that can be reused and specialized for many tasks, such as text classification, summarization, question answering, or image recognition.
3. **Scalability**:
    - They often have billions or even trillions of parameters, making them highly powerful but resource-intensive.
    - Their size and training approach enable them to learn nuanced patterns and complex relationships in data.
4. **Adaptability (Fine-Tuning)**:
    - Once pretrained, a foundation model can be fine-tuned with task-specific data to achieve excellent performance on specific applications.
    - Examples:
        - Fine-tuning GPT for writing code or answering medical questions.
        - Fine-tuning a vision model for object detection in specific industries.

5. **Multimodal Capabilities** (optional):
    - Some foundation models are capable of working across multiple modalities, such as text, images, and audio (e.g., CLIP or GPT-4 Vision).

Examples of Foundation Models:

1. **NLP Models**:
    - **BERT**: A foundation model for understanding text.
    - **GPT (Generative Pre-trained Transformer)**: A family of models designed for generating text.
    - **LLaMA**: A model optimized for efficient natural language processing.
2. **Vision Models**:
    - **CLIP**: A multimodal model capable of understanding images and text together.
    - **DINO**: A model for unsupervised image representation learning.
3. **Multimodal Models**:
    - **DALL-E**: Generates images from text prompts.
    - **GPT-4 Vision**: Combines text and image understanding.

Why Are They Important?

1. **Efficiency in Development**:
    - Training large models from scratch is costly and time-consuming.
    - Foundation models reduce the need to start from zero, enabling faster application development.
2. **Wide Applicability**:
    - They can be adapted to many tasks with minimal effort, making them highly versatile.
3. **Improved Performance**:

    - Because they learn from a large and diverse dataset, foundation models often outperform smaller, task-specific models on a wide range of tasks.
4. **Research Acceleration**:
    - Open foundation models like LLaMA democratize AI research by giving researchers access to powerful pretrained models.

Challenges and Considerations:

1. **Compute and Resource Requirements**:
    - Training foundation models requires massive computational resources and infrastructure.
2. **Ethical Concerns**:
    - Biases in the training data can propagate into the model, leading to potential misuse or unintended consequences.
3. **Interpretability**:
    - Understanding why foundation models make certain decisions can be challenging due to their size and complexity.
4. **Fine-Tuning Costs**:
    - While the base model is general-purpose, fine-tuning still requires additional compute and data.

In summary, **foundation models** form the backbone of modern AI research and applications. They provide general capabilities that, when fine-tuned, enable state-of-the-art performance across a wide variety of tasks and domains.

Random Sampling & Non-Determinism in LLMs

- **Random sampling** allows LLMs to generate varied responses by probabilistically selecting tokens (words/subwords) from a distribution of likely options, rather than rigidly choosing the "top" prediction. This introduces creativity and adaptability but ensures the same prompt can yield different outputs, making responses non-deterministic.
- **Temperature** adjusts the degree of randomness:
    - **High temperature** (e.g., >1.0) flattens the token probability distribution, encouraging exploration of less likely tokens for diverse, creative outputs (e.g., storytelling).
    - **Low temperature** (e.g., <0.5) sharpens the distribution, prioritizing high-probability tokens for focused, deterministic responses (e.g., factual answers).
      Temperature acts as a "creativity dial," balancing novelty and coherence based on use-case needs.

Versatile Text Generation with LLaMA-2

- **Text generation tasks**: LLaMA-2 can perform diverse tasks like creative writing, summarization, question answering, dialogue generation, and code/poetry creation. Its flexibility stems from its training on vast datasets, enabling context-aware outputs tailored to user prompts.
- **System context**: Defining a model's role via system prompts (e.g., *"You are a helpful tutor explaining concepts simply"*) steers its responses toward specific tones, styles, or objectives. This context acts as a "meta-instruction," ensuring outputs align with desired behaviors (e.g., formality, brevity, or expertise).
    - Example: A system prompt like *"Respond as a historian analyzing causes of events"* yields more analytical, evidence-driven answers vs. a generic query.
    - Use system context to reduce ambiguity and enhance task-specific consistency.

Trimming Conversation History

Purpose:
Language models (LLMs) have a maximum token limit they can process in a single request (e.g., 4,096 tokens for many models). Trimming ensures the conversation history stays within this limit, preventing errors and maintaining performance.

How It Works:

1. **Token Counting**:
    - Each input (user prompts, model responses, system instructions) is broken into tokens (words, subwords, or characters).
    - The total tokens in the conversation (including past interactions) are tracked.
2. **Trimming Strategy**:
    - When the token count nears the limit, older messages are removed *sequentially*, starting from the earliest exchanges.
    - Example: If a chat history exceeds 4,000 tokens, the first few messages (oldest) are deleted to free up space for new input.
3. **Prioritization**:
    - Critical context (e.g., system prompts defining the model's role) may be preserved to maintain coherence.
    - Recent messages are usually kept intact, as they often contain the most relevant context.

1. **Token**:
   - The basic unit of text processed by language models.
   - Represents words, subwords (e.g., `"un" + "happy"`), or punctuation.
   - Example: The sentence `"Hello, world!"` might tokenize to `["Hello", ",", " world", "!"]`.

2. **Token Limit**:
   - The maximum number of tokens a model can accept in a single input (e.g., 4,096 tokens for many models).
   - Exceeding this limit causes truncation or errors, requiring strategies like shortening prompts or trimming conversation history.
   - Impacts both input (prompts) and output (responses).

3. **Tokenizer**:
   - A tool that converts raw text into tokens (and vice versa).
   - Specific to each model (e.g., GPT-4 uses a different tokenizer than LLaMA-2).

  - Handles language-specific rules, rare words, and efficiency (e.g., splitting `"tokenization"` into `["token", "ization"]`).
  - Critical for aligning user inputs with model expectations.

Why These Matter:
- Tokens determine computational costs and model performance.
- Token limits enforce efficiency but require careful context management.
- Tokenizers ensure models interpret text accurately (garbage in = garbage out).