

Prompt Engineering RAG Reasoning

Ramaseshan Ramachandran

February 19, 2025

Questions

- ▶ How does the prompt relate to the text it generates? Is there any logical reasoning involved in checking whether the context of the prompt corresponds to the generated text?
- ▶ Is the reasoning engine module utilized in Gemini and other AI engines?
- ▶ Given that the output is probabilistic and potentially multiple outcomes are possible, how does AI engine select the most probable output from the generated outputs?
- ▶ Provide a comprehensive explanation of the vertex reasoning engine, accompanied by an illustrative example that elucidates its functionality.

- ▶ **Prompting:** Using natural language instructions to guide LLMs.
- ▶ Enables tasks like:
 - ▶ Translation
 - ▶ Summarization
 - ▶ Classification
- ▶ Crucial for effective interaction and desired outputs.

Image here



- ▶ **In-Context Learning:** Prompts act as learning signals, improving performance without parameter updates.
- ▶ **Instruction Tuning:** Fine-tuning LLMs on instruction-response pairs to enhance instruction following.
- ▶ Both techniques improve LLM's ability to understand and respond to user requests.

- ▶ LLMs can generate harmful, false, or toxic content.
- ▶ Examples include:
 - ▶ Unsafe advice
 - ▶ Verbal attacks
 - ▶ Hate speech
- ▶ Requires careful mitigation strategies.

- ▶ **Safety Training:** Training models to avoid generating harmful content.
- ▶ **Preference Alignment:** Aligning model objectives with human goals.
- ▶ Common methods:
 - ▶ Reinforcement Learning from Human Feedback (RLHF)
 - ▶ Direct Preference Optimization (DPO)



- ▶ A separate model is trained to assess the alignment of a candidate response with human preferences.
- ▶ This model guides the LLM towards generating more desirable outputs.
- ▶ Crucial for ensuring that LLMs are helpful and harmless.

Image here

Prompting

- ▶ A prompt is a text string issued to a language model.
- ▶ It guides the model to perform a specific task.
- ▶ The model generates tokens iteratively, conditioned on the prompt.
- ▶ Prompt engineering is the process of crafting effective prompts.

- ▶ Tokenization
- ▶ Embedding
- ▶ Transformer Networks
- ▶ Contextual Understanding
- ▶ Language Generation

- ▶ Breaking text into smaller units (tokens).
- ▶ Tokens can be words, subwords, or characters.
- ▶ Transforms raw text into numerical format.

- ▶ Converting tokens into numerical vectors (embeddings).
- ▶ Embeddings capture semantic meaning.
- ▶ Similar meanings have close embeddings.

- ▶ Neural network architecture for sequential data.
- ▶ Key component: Attention mechanism.
- ▶ Attention weighs the importance of different tokens.
- ▶ Enables understanding of context.

- ▶ Transformer processes embedded tokens.
- ▶ Attention builds contextual understanding.
- ▶ Understanding represented by neuron activation patterns.

- ▶ Predicting the next token in the sequence.
- ▶ Based on contextual understanding and learned patterns.
- ▶ Probability of each token is calculated.
- ▶ Process repeated iteratively.

- ▶ Embeddings: $v_i \in \mathbb{R}^d$
- ▶ Attention:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

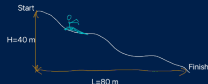
- ▶ Probability distributions over vocabularies.



Prompt

Here's a solution to a Physics problem by a student.

23. A skier slides down the frictionless slope as shown. What is the skier's speed at the bottom?



Solution:

The conservation of energy means that the energy of the skier is same at the beginning and at the end.

$$\text{Start: } E = m g L$$

$$\text{End: } E = \frac{1}{2} m v^2$$

$$\Rightarrow m g L = \frac{1}{2} m v^2$$

$$\Rightarrow v = \sqrt{2 g L} = \sqrt{2 \times 9.81 \times 80} = 39.6 \text{ m s}^{-1}$$

Try to reason about the question step by step.

Did the student get the correct answer? If the solution is wrong, please explain what is wrong and solve the problem.

Make sure to use LaTeX for math and round off the final answer to two decimal places.



Response

The student did not get the correct answer. The student made a mistake in the calculation of the potential energy at the start of the slope. The potential energy at the start of the slope is given by mgh , where m is the mass of the skier, g is the acceleration due to gravity, and h is the height of the slope. The student used the length of the slope, L , instead of the height, H , in the calculation of the potential energy.

The correct solution is:

The conservation of energy means that the total energy of the skier is the same at the beginning and at the end of the slope.

At the start of the slope, the skier has only potential energy, which is given by mgh .

At the end of the slope, the skier has only kinetic energy, which is given by $\frac{1}{2} m v^2$

Therefore, we can write:

$$mgh = \frac{1}{2} m v^2$$

Canceling out the mass m on both sides, we get:

$$gh = \frac{1}{2} v^2$$

Solving for v , we get:

$$v = \sqrt{2gh}$$

Substituting the given values, we get:

$$v = \sqrt{2 \times 9.81 \times 40} = 28.01 \text{ m/s}$$

Therefore, the skier's speed at the bottom of the slope is 28.01 m/s.

Reference: Gemini: A Family of Highly Capable Multimodal Models

- ▶ Task: Classify the sentiment of a hotel review.
- ▶ Example Review (BLT Corpus):

Did not like the service that I was provided, when I entered the hotel. I also did not like the area, in which the hotel was located. Too much noise and events going on for me to feel relax.

- ▶ Append an incomplete statement to the review:

Did not like the service that I was provided, when
I entered the hotel. I also did not like the
area, in which the hotel was located. Too much
noise and events going on for me to feel relax.
In short, our stay was:

- ▶ This prompt provides context for the LLM.

- ▶ The LLM completes the statement by generating tokens.
- ▶ Examples of completions:
 - ▶ "... not a pleasant one. The staff at the front desk were not welcoming or friendly, and seemed disinterested in providing good customer service."
 - ▶ "... uncomfortable and not worth the price we paid. We will not be returning to this hotel."



- ▶ Negative completions reflect the negative sentiment of the review.
- ▶ Map completions to predefined sentiment classes (e.g., negative, positive).
- ▶ Example mappings:
 - ▶ `{excellent → positive}`
 - ▶ `{did not like → negative}`



- ▶ LLMs can perform various tasks with appropriate contextual nudges.
- ▶ For tasks like summarization and translation, we need reusable prompts.
- ▶ **Templates:** Task-specific prompting text with input slots.



- ▶ **Summarization:** {input}; tldr;
- ▶ **Translation:** {input}; translate to French:
- ▶ **Sentiment:** {input}; Overall, it was
- ▶ **Fine-Grained Sentiment:** {input}; What aspects were important in this review?
- ▶ {input} represents the input text.



- ▶ Templates are applied to inputs to create filled prompts.
- ▶ Example using the hotel review:
 - ▶ **Summarization:** Did not like the service...relax. tldr;
 - ▶ **Translation:** Did not like the service...relax. translate to French:
 - ▶ **Sentiment:** Did not like the service...relax. Overall, it was
- ▶ These are ready to be used with an LLM. (See Figure 12.1 in your original document for example outputs)

- ▶ Input followed by text to be completed is a common pattern.
- ▶ This constrains the generation effectively.
- ▶ **Bad Example:** Translate English to French: Did not like the service...
- ▶ This prompt is ambiguous and may not produce a translation.
- ▶ Prompts should be designed unambiguously.



- ▶ Specify the set of possible answers in the prompt.
- ▶ Example for sentiment analysis:

Human: Do you think that "input" has negative or positive sentiment?

Choices:

(P) Positive

(N) Negative

Assistant: I believe the best answer is:

- ▶ This provides even stronger guidance to the LLM.

- ▶ Templates provide structure and context to prompts.
- ▶ They guide LLMs to generate specific types of responses.
- ▶ We'll explore templates using the hotel feedback example.

Template: "I experienced [issue] at [location/time]. Specifically, [details]."

- ▶ **Example:** "I experienced poor service at check-in. Specifically, the process was slow and staff was unhelpful."
- ▶ **Example:** "I disliked the hotel's location. Specifically, it was too noisy."

Template: "I am writing to express my dissatisfaction with [aspect]. [Details]. As a result, [impact]."

- ▶ **Example:** "I am writing to express my dissatisfaction with the check-in service. The process was slow and the staff was unhelpful. As a result, I felt frustrated."
- ▶ **Example:** "I am writing to express my dissatisfaction with the hotel's location. There was constant noise from street events. As a result, I couldn't relax."



Template: "Problem: [issue]. Details: [details]. Recommendation: [solution]."

- ▶ **Example:** "Problem: Slow check-in. Details: 30-minute wait, unhelpful staff. Recommendation: Improve check-in efficiency and staff training."
- ▶ **Example:** "Problem: Noisy location. Details: Constant street events. Recommendation: Provide soundproof rooms or warn guests about the noise."

Template: "I felt [emotion] due to [issue] at [location/time]. [Details]."

- ▶ **Example:** "I felt frustrated due to the slow check-in process. The staff was unhelpful and I waited for 30 minutes."
- ▶ **Example:** "I felt stressed due to the noisy location. Constant street events made it impossible to relax."

- ▶ Templates can be combined for comprehensive feedback.
- ▶ Example: "Problem: Slow check-in and noisy location. Details: 30-minute wait, unhelpful staff, constant street events. I felt frustrated and stressed. Recommendation: Improve efficiency, train staff, and address noise issues."

- ▶ **Consistency:** Ensures uniform structure across prompts.
- ▶ **Clarity:** Guides LLMs to generate specific responses.
- ▶ **Efficiency:** Reduces the time needed to craft effective prompts.

- ▶ How do LLMs ensure generated text aligns with the prompt?
- ▶ Is there logical reasoning involved?



- ▶ LLMs trained on massive datasets.
- ▶ They recognize statistical relationships between words.
- ▶ Given a prompt, they find matching patterns.
- ▶ Prediction of "what comes next" based on likelihood.



- ▶ Transformer architecture with attention mechanism.
- ▶ Attention weighs importance of prompt parts.
- ▶ Model focuses on relevant information.
- ▶ Maintains context throughout generation.

- ▶ Generates text by predicting next word's probability.
- ▶ Prediction based on prompt context and preceding words.
- ▶ Determines most likely word to follow.

- ▶ LLMs don't possess true reasoning abilities.
- ▶ Rely on learned patterns, leading to inconsistencies.
- ▶ Can produce factually incorrect statements.
- ▶ Active research to improve logical reasoning.

- ▶ "Context window" limits preceding text consideration.
- ▶ Maintains running context for consistency.
- ▶ Context window limitations can affect coherence.



- ▶ LLMs generate probabilistic outputs.
- ▶ Multiple probable outcomes exist.
- ▶ How does the AI engine select the final output?



- ▶ LLMs produce probability distributions over possible tokens.
- ▶ Each token has an associated probability score.
- ▶ Higher scores indicate more likely tokens.

▶ Greedy Decoding:

- ▶ Selects the token with the highest probability at each step.
- ▶ Simple but can lead to suboptimal results.

▶ Sampling:

- ▶ Randomly samples tokens based on their probabilities.
- ▶ Introduces diversity but can be inconsistent.

▶ Beam Search:

- ▶ Keeps track of multiple promising sequences (beams).
- ▶ Selects the sequence with the highest overall probability.
- ▶ Balances quality and diversity.

▶ Top-k and Top-p Sampling:

- ▶ Restricts sampling to the top k most probable tokens or tokens whose cumulative probability exceeds p.
- ▶ Controls diversity and reduces the risk of low-probability outputs.

- ▶ Selected output may undergo post-processing.
- ▶ Techniques like:
 - ▶ Filtering for safety and relevance.
 - ▶ Re-ranking based on additional criteria.
 - ▶ Paraphrasing or editing for clarity.

- ▶ Task requirements (e.g., creativity vs. accuracy).
- ▶ Model parameters (e.g., temperature).
- ▶ User preferences (e.g., length, style).



- ▶ Understanding the interaction between Prompts, Retrieval Augmented Generation (RAG), and Large Language Models (LLMs).
- ▶ Exploring the role of Gemini API, Vertex AI API, and seminal papers.

Retrieval Augmented Generation (RAG)

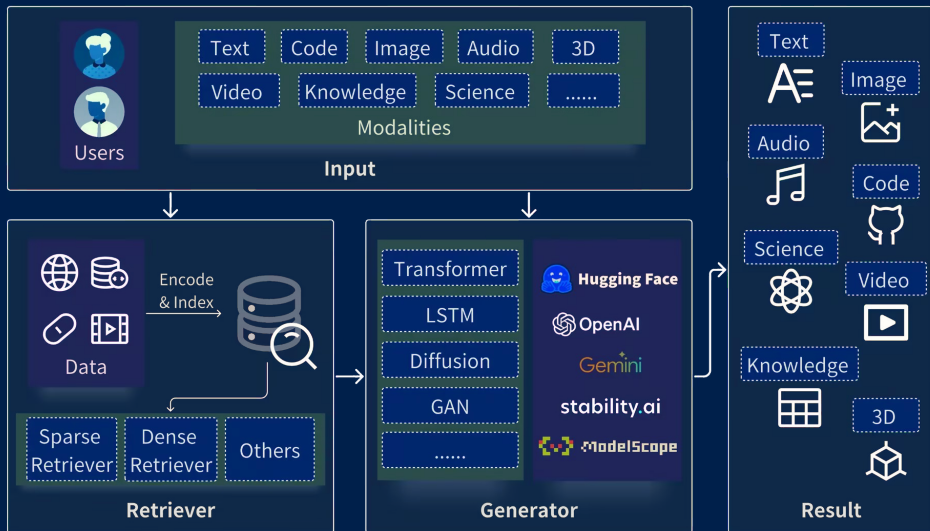


- ▶ Understanding the interaction between Prompts, Retrieval Augmented Generation (RAG), and Large Language Models (LLMs).
- ▶ Exploring the role of Gemini API, Vertex AI API, and seminal papers.



- ▶ Prompts are the user's input, guiding the LLM's response.
- ▶ They define the task, context, and desired output.
- ▶ Effective prompts are crucial for accurate and relevant LLM outputs.

- ▶ LLMs like Gemini/ChapGPT/DeepSeek/LLama are trained on vast datasets, enabling them to generate human-like text.
- ▶ They excel at understanding and generating language, but can suffer from:
 - ▶ Lack of up-to-date information.
 - ▶ "Hallucinations" (generating factually incorrect information).



- ▶ RAG enhances LLMs by retrieving relevant information from external knowledge sources.
- ▶ It addresses the limitations of LLMs by providing access to up-to-date and domain-specific information.
- ▶ Key steps:
 - ▶ Retrieval: Using an information retrieval engine to find relevant documents.
 - ▶ Augmentation: Injecting the retrieved information into the prompt.
 - ▶ Generation: LLM generates a response based on the augmented prompt.



- ▶ User provides a **prompt**.
- ▶ **RAG** system retrieves relevant information based on the prompt.
- ▶ Retrieved information is incorporated into an **augmented prompt**.
- ▶ **LLM** uses the augmented prompt to generate a response.
- ▶ This process ensures the response is grounded in external knowledge and contextually relevant.

Reasoning with LLMS



- ▶ Reasoning engines are systems designed to derive conclusions from given information.
- ▶ They use logical rules and inference mechanisms to process data.
- ▶ Traditionally, they have been rule-based systems.
- ▶ In the context of AI, they enhance decision-making and problem-solving.



- ▶ Rule-based systems: "If-then" logic.
- ▶ Well-defined rules for specific domains.
- ▶ Limitations:
 - ▶ Difficulty handling ambiguity.
 - ▶ Limited to predefined rules.



- ▶ Modern LLMs are integrating advanced reasoning capabilities.
- ▶ Moving beyond simple rule-based systems.
- ▶ Enhancing LLMs with:
 - ▶ Contextual reasoning.
 - ▶ Tool use.
 - ▶ Agentic workflows.

- ▶ LLMs analyze context using vast training data.
- ▶ Attention mechanisms process and weigh information.
- ▶ Enables understanding of complex relationships.
- ▶ Example: Answering questions based on the surrounding text.

- ▶ Reasoning engines involve logical inference.
- ▶ Mathematical frameworks can formalize these processes.
- ▶ This treatment focuses on propositional logic and first-order logic.

- ▶ Propositions: Statements that are either true or false.
- ▶ Connectives: \neg (negation), \wedge (conjunction), \vee (disjunction), \rightarrow (implication), \leftrightarrow (biconditional).
- ▶ Truth tables: Define the semantics of connectives.
- ▶ Inference rules: Modus ponens, modus tollens, etc.

- ▶ Let P, Q be propositions.
- ▶ Modus ponens: If P and $P \rightarrow Q$ are true, then Q is true.
- ▶ Symbolically: $(P \wedge (P \rightarrow Q)) \rightarrow Q$.
- ▶ This can be verified using truth tables.



- ▶ Extends propositional logic with predicates, quantifiers, and variables.
- ▶ Predicates: Represent properties or relations.
- ▶ Quantifiers: \forall (for all), \exists (there exists).
- ▶ Variables: Represent objects in a domain.



- ▶ Example: "All humans are mortal."
- ▶ Let $H(x)$ be "x is a human" and $M(x)$ be "x is mortal."
- ▶ Symbolically: $\forall x(H(x) \rightarrow M(x))$.
- ▶ Inference: If $H(John)$ is true, then $M(John)$ is true.



- ▶ Inference involves deriving new knowledge from existing knowledge.
- ▶ Mathematical logic provides formal rules for inference.
- ▶ Automated theorem proving: Algorithms that perform logical inference.

- ▶ LLMs can generate and manipulate logical expressions.
- ▶ Integrating formal logic with LLMs enhances reasoning capabilities.
- ▶ Techniques:
 - ▶ Symbolic reasoning with LLM generated logical rules.
 - ▶ LLMs used to generate or verify logical proofs.
 - ▶ Knowledge graphs and logical queries.



- ▶ LLMs can be trained on datasets containing logical expressions and proofs.
- ▶ Techniques like neural theorem proving are being explored.
- ▶ LLMs can use external tools to perform logical inference.
- ▶ Example: LLM generating logical queries for a knowledge graph.

- ▶ **Lewis et al. (2020):** "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." [**lewis2020retrieval**]
- ▶ **Komeili et al. (2021):** "Internet-Augmented Language Models through Interactive Retrieval." [**komeili2021internet**]
- ▶ P. Zhao, et al, "Retrieval- augmented generation for ai-generated content: A survey", 3 2024.



▶ **Vertex AI API:**

- ▶ Provides tools and services for building and deploying AI models, including RAG implementations.
- ▶ Offers access to LLMs like Gemini and information retrieval capabilities.

▶ **Gemini API:**

- ▶ Allows developers to integrate Gemini's capabilities into their applications.
- ▶ Can be used in conjunction with Vertex AI Search for RAG implementations.



- ▶ User asks a question.
- ▶ Vertex AI Search retrieves relevant documents.
- ▶ Gemini API uses the retrieved documents to generate an informed answer.
- ▶ Vertex AI reasoning engine can orchestrate this process.



- ▶ Prompts are the user's input, guiding the LLM's response.
- ▶ They define the task, context, and desired output.
- ▶ Effective prompts are crucial for accurate and relevant LLM outputs.

- ▶ LLMs like Gemini are trained on vast datasets, enabling them to generate human-like text.
- ▶ They excel at understanding and generating language, but can suffer from:
 - ▶ Lack of up-to-date information.
 - ▶ "Hallucinations" (generating factually incorrect information).

- ▶ RAG enhances LLMs by retrieving relevant information from external knowledge sources.
- ▶ It addresses the limitations of LLMs by providing access to up-to-date and domain-specific information.
- ▶ Key steps:
 - ▶ Retrieval: Using an information retrieval engine to find relevant documents.
 - ▶ Augmentation: Injecting the retrieved information into the prompt.
 - ▶ Generation: LLM generates a response based on the augmented prompt.



- ▶ User provides a **prompt**.
- ▶ **RAG** system retrieves relevant information based on the prompt.
- ▶ Retrieved information is incorporated into an **augmented prompt**.
- ▶ **LLM** uses the augmented prompt to generate a response.
- ▶ This process ensures the response is grounded in external knowledge and contextually relevant.



- ▶ **Lewis et al. (2020):** "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." [**lewis2020retrieval**]
- ▶ **Komeili et al. (2021):** "Internet-Augmented Language Models through Interactive Retrieval." [**komeili2021internet**]



▶ **Vertex AI API:**

- ▶ Provides tools and services for building and deploying AI models, including RAG implementations.
- ▶ Offers access to LLMs like Gemini and information retrieval capabilities.

▶ **Gemini API:**

- ▶ Allows developers to integrate Gemini's capabilities into their applications.
- ▶ Can be used in conjunction with Vertex AI Search for RAG implementations.

- ▶ User asks a question.
- ▶ Vertex AI Search retrieves relevant documents.
- ▶ Gemini API uses the retrieved documents to generate an informed answer.
- ▶ Vertex AI reasoning engine can orchestrate this process.

- ▶ Prompts, RAG, and LLMs work synergistically to create powerful AI applications.
- ▶ RAG addresses the limitations of LLMs by providing access to external knowledge.
- ▶ Google Cloud's Vertex AI and Gemini APIs provide the tools for building RAG-enabled applications.

- [1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, **33**, 9459-9474.
- [2] Komeili, M., Lewis, P., Shuster, K., Yih, W. T., & Riedel, S. (2021). Internet-augmented language models through interactive retrieval. *arXiv preprint arXiv:2103.07567*.
- [3] Guu, K., Lee, K., Tung, Z., & Chang, M. W. (2020). REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.