**GUARDRAILS**

**NEMOGUARDRAILS -Nvidia:**

Guardrails (or "rails" for short) are specific ways of controlling the output of a large language model, such as not talking about politics, responding in a particular way to specific user requests, following a predefined dialog path, using a particular language style, extracting structured data, and more.

**Read any Medium blog for free**(as some blogs referred here are paywalled ):

<https://www.youtube.com/shorts/T52uUUPkRNI>

**Start with:** Youtube Tutorial then ref docs and blogs

<https://youtu.be/SwqusllMCnE?feature=shared>

<https://youtu.be/3Vg5bEtaepQ?feature=shared>

<https://youtu.be/alDcHItL6yo?feature=shared>

**Above tutorial associated Notebooks:**

<https://www.pinecone.io/learn/nemo-guardrails-intro/>

<https://github.com/pinecone-io/examples/tree/master/learn/generation/chatbots/nemo-guardrails>

**Must read:**

<https://ksankar.medium.com/generative-ai-guardrails-functional-compositional-to-address-the-enhanced-treat-surface-43366685e466>

**Docs:**

<https://github.com/NVIDIA/NeMo-Guardrails/tree/develop/docs>

**NemoGuardrails Architecture Diagram:**

<https://github.com/NVIDIA/NeMo-Guardrails/tree/develop/docs/architecture>

**Setting up Pipelines for various HF models:**

<https://github.com/NVIDIA/NeMo-Guardrails/tree/develop/examples/configs/llm>

**Colang or Canonical form vs Semantic:**

* Canonical language is the "how" of saying things, while semantic is the "what" of it.
* Canonical language is preferred for clarity and precision, while semantic analysis considers all possible meanings and interpretations.

**NemoGuardrails Library:**

<https://github.com/NVIDIA/NeMoGuardrails/blob/develop/docs/user_guides/guardrails-library.md>

**llm\_call function calling:**

<https://github.com/NVIDIA/NeMo-Guardrails/blob/develop/nemoguardrails/actions/llm/utils.py>

**NemoGuardrails Implementations:**

**With OpenAI:**

<https://towardsdatascience.com/configuring-nemo-guardrails-your-way-an-alternative-method-for-large-language-models-c82aaff78f6e>

**RAG with OpenAI and NemoGuardrails:**

<https://medium.aiplanet.com/retrieval-augmented-pipeline-with-actions-using-nemo-gaurdrails-447b84a5334b>

**With Llama2:**

<https://blog.marvik.ai/2023/10/09/enhancing-llama2-conversations-with-nemo-guardrails-a-practical-guide/>

Above blog repo: <https://github.com/marvik-ai/llama2-nemo-guardrails>

Issues faced with above implementation community discussion thread:

<https://github.com/NVIDIA/NeMo-Guardrails/issues/238>

My version of above repo fixed:

<https://github.com/Jaykumaran/NemoGuardrails/tree/main/GuardRAILS%20LLAMA2/llama2-nemo-guardrails> #Read Markdown headings in notebooks carefully to understand setup.

**Guardrails AI:**

**Docs**: <https://www.guardrailsai.com/docs>

**Example Notebooks:**

<https://github.com/guardrails-ai/guardrails/tree/main/docs/examples>

**SEMANTIC ROUTER: (Different from Guardrails)**

Tutorial: <https://www.youtube.com/watch?v=ro312jDqAh0>

Notebooks: <https://github.com/aurelio-labs/semantic-router/tree/main/docs>

Blogs: <https://medium.com/ai-insights-cobet/beyond-basic-chatbots-how-semantic-router-is-changing-the-game-783dd959a32d>

**LLAMA GUARD: (Purple Llama)**

<https://towardsdatascience.com/safeguarding-your-rag-pipelines-a-step-by-step-guide-to-implementing-llama-guard-with-llamaindex-6f80a2e07756>

**LLM EVALUATIONS:**

**RAGAS :**

**Docs**: <https://docs.ragas.io/en/latest/howtos/customisations/llms.html>

**Tutoria**l: <https://youtu.be/mEv-2Xnb_Wk?feature=shared>

**Notebooks:**

<https://github.com/explodinggradients/ragas/tree/main/docs/howtos>

**With OpenAI**

<https://colab.research.google.com/drive/1TZo2sgf1YFzI4_U-tGppg_ylHAR3MXF_?usp=sharing>

**With HF models and VLLM:**

<https://github.com/Jaykumaran/RAGAS-Langchain-Evaluation-VLLM-HFTextInference/blob/main/Working_Copy_of_Evaluating_Advanced_Retrieval_Methods_in_VLLM_%2B_LangChain_with_RAGAS%2BLiteLLM.ipynb>

**LANGSMITH:**

**Docs:** <https://docs.smith.langchain.com/overview>

**ARIZE PHOENIX**:

**Docs**: <https://docs.arize.com/phoenix/>

**Blog:**

<https://arize.com/blog-course/llm-evaluation-the-definitive-guide/>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**WEIGHTS AND BIASES:**

<https://wandb.ai/ayush-thakur/llm-eval-sweep/reports/How-to-Evaluate-Compare-and-Optimize-LLM-Systems--Vmlldzo0NzgyMTQz>

**Other evaluation Blogs:**

<https://www.analyticsvidhya.com/blog/2023/05/how-to-evaluate-a-large-language-model-llm/>

**PROMPT ENGINEERING**

**Best Guide:**

<https://www.promptingguide.ai/>

**26 Prompt Techniques:**

<https://www.youtube.com/watch?v=chAQGTBMXXQ>

**OpenAI Guide:**

<https://platform.openai.com/docs/guides/prompt-engineering/six-strategies-for-getting-better-results>

<https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api>

<https://help.openai.com/en/collections/3675942-prompt-engineering>

**Cookbook:**

<https://github.com/dair-ai/Prompt-Engineering-Guide>

**Effectiveness of prompt Structure in Healthcare:**  
<https://arxiv.org/abs/2305.13614>

**Best channel to learn LLM:**

Umar Jamil <https://www.youtube.com/@umarjamilai> -Best For LLM Architectures

AI Anytime <https://www.youtube.com/@AIAnytime>

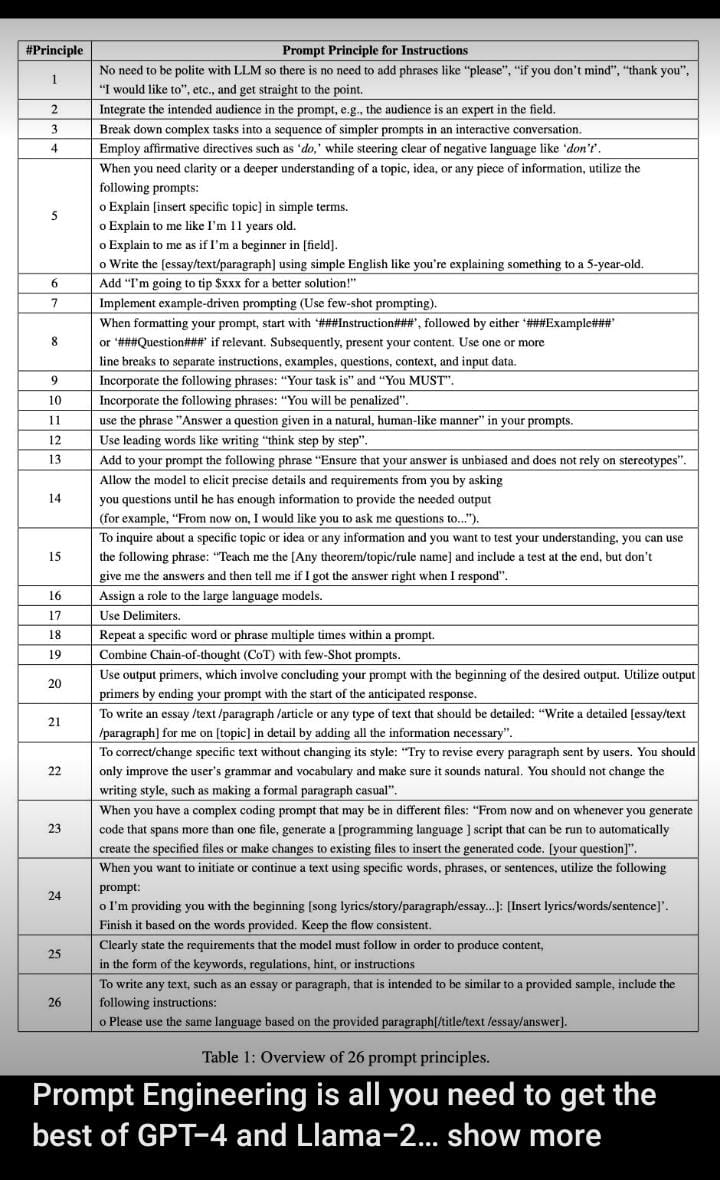
code\_your\_own\_ai <https://www.youtube.com/@code4AI>

1littlecoder <https://www.youtube.com/@1littlecoder>

PromptEngineer <https://www.youtube.com/@engineerprompt>

Samwitteveenai <https://www.youtube.com/@samwitteveenai>

James Briggs <https://www.youtube.com/results?search_query=james+briggs>

****