Prompt Engineering: Classification of Techniques and Prompt Tuning

Godel Insights - Blog - Prompt Engineering: Classification of Techniques and Prompt Tuning

Kara Goward • 14 min read • 2024-07-11
https://www.godeltech.com/blog/prompt-engineering-classification-of-techniques-and-prompt-tuning/

By Katsiaryna Ruksha, Lead Data Scientist

Being an emerging field of study prompt engineering lacks definitive classification of techniques. When you look through different articles and websites discussing these techniques, you'll find that they vary and lack structure. As a result of this mess, practitioners often stick to the simplest approaches. In this article I propose an overview and a clear classification of prompt engineering techniques that will help you grasp their concepts and use them effectively in your applications. Additionally, I'll talk about conducting prompt engineering as a Data Science process that involves prompt tuning and evaluation.

Pitfalls of Large Language Models

Despite many researches' efforts, large language models still struggle with some issues. Their major pitfalls are:

- **Citing resources.** LLMs can generated a content that looks very trustworthy and has references to external resources but it's important to remember that they cannot cite resources having neither access to the Internet not,
- **Bias.** LLMs can exhibit bias in their responses, often generating stereotypical or prejudiced content,
- **Hallucinations.** LLMs can sometimes "hallucinate" or generate false information when asked a question they do not know the answer to,
- **Math and commonsense problems.** Despite their advanced capabilities, LLM's often struggle with solving even the simplest mathematical or commonsense problems,
- **Prompt hacking**.LLMs can be manipulated or "hacked" by users to ignore developers' instructions and generate specific content.

Most prompt engineering techniques address two issues: hallucinations and solving math and commonsense tasks. There are specific techniques that aim to mitigate prompt hacking but this is a topic for separate discussion.

Common rules

Before discussing the specific techniques, let's talk about common rules of prompting that will help you to write clear and specific instructions:

1. Be precise in saying what do do (write, summarize, extract information),

2. Avoid saying what not to do and say what to do instead,

3. Be specific: instead of saying “in a few sentences”, say “in 2-3 sentences”,

4. Add tages or delimiters to structurize the prompt,

5. Ask for a structured output (JSON, HTML) if needed,

6. Ask the model to verify whether the conditions are satisfied (e.g. “if you do not know the answer, say “No information”),

7. Ask a model to first explain and then provide the answer (otherwise a model may try to justify an incorrect answer).

Classification

My proposal of taxonomy of prompt engineering techniques

Most of existing techniques can be divided into three groups:

• **single prompt techniques** aim at optimizing the response given to one prompt,

• next are the techniques that **combine a few prompts**. Their common concept is in querying a model (or models) a few times in order to solve the task,

• finally, there is a group of approaches that combines large language models **with external tools**.

Single Prompt Techniques

Input-Output or Single prompt techniques

What techniques aim to solve your task in a single prompt?

• Zero-Shot,

• Few-Shot,

• Chain of Thought,

• Program Aided Language.

Let’s study them one by one.

Zero-Shot Prompting

This is the simplest technique that uses natural language instructions.

One-Shot Learning. Example is from promptingguide.ai

Few-Shot Prompting

LLMs are extremely good at one-shot learning but they still may fail at complicated tasks. The idea of few-shot learning is to demonstrate to the model similar tasks with correct answers (Brown et al. (2020)).

Few-Shot Learning. Example is from promptingguide.ai

In Min et al. (2022) paper, it is shown that incorrectness of labels of demonstrations barely hurts

the performance on a range of classification
and multi-choice tasks. Instead, it's essential to ensure that demonstrations provide a few examples of the label space, the distribution of the input test and the overall format of the sequence.

Chain of Thought Prompting (CoT)

Chain-of-Thought prompting enables complex reasoning capabilities through intermediate reasoning steps. This technique aims to make the model iterate and reason each step.

Zero-shot, Few-shot and Chain-of-Thought prompting techniques. Example is from Kojima et al. (2022).

CoT is used either with zero-shot or few-shot learning. The idea of Zero-shot CoT is to suggest a model to think step by step in order to come to the solution. Authors of the approach (Kojima et al. (2022)) demonstrate that Zero-shot-CoT significantly outperforms zero-shot LLM performances on arithmetic, symbolic and other logical reasoning tasks.

If you select Few-shot CoT, you must ensure to have diverse examples with explanations (Wei et al. (2022)). The empirical gain of the approach is striking for arithmetic, commonsense, and symbolic reasoning tasks.

Program-Aided Language Models (PAL)

Program-Aided language models is an approach that extends Chain-of-Thought prompting by extending the explanation as a natural language with code (Gao et al. (2022)).

Program Aided Language prompting. Example is from Gao et al. (2022)

The technique can be implemented using LangChain PALChain class.

Multiple Prompt Techniques

The next group of prompts is based on different strategies of combining prompts of one or a few of models:

1. **Voting.** The idea is to apply voting to get the correct answer. Techniques: Self-Consistency.
2. **Divide and conquer.** This group of prompts is based on dividing a complicated task into a few prompts. Techniques: Directional Stimulus, Generate Knowledge, Prompt Chaining, Chain-of-Table and Least-to-Most prompting.
3. **Self-evaluation**. The approach suggests including into a framework a step of checking whether the output meets instructions. Techniques: Reflexion, Tree-of-Thoughts.

Self Consistency (SC)

Self-consistency is based on the intuition that "a complex reasoning problem typically admits multiple different ways of thinking leading to its unique correct answer" (Wang et al. (2022)).

Self Consistency prompting technique

It asks the same Chain-of-Thought prompt a few times thus generating a diverse set of reasoning paths and then select the most consistent answer by applying voting.

Example of Self-consistency prompting from Wang et al. (2022)

In Wang et al. (2022) the gain of applying self-consistency for arithmetic and commonsense tasks was 4%-18% on common benchmarks.

Directional Stimulus Prompting (DSP)

Next concept for combining prompts is “Divide and Conquer”. In DSP we have two steps: generate stimulus (e.g., keywords) and use them to improve quality of the response.

Directional Stimulus prompting was proposed in Li et al. (2023) for summarization, dialogue response generation, and chain-of-thought reasoning tasks. It includes two models:

- A small tunable policy LM is trained to generate a stimulus (a hint),
- A black box frozen large LM is used to generate a summary based on the question and the stimulus from the previous step.

Directional Stimulus prompting schema

The policy model can be optimized through supervised fine-tuning using labeled data and reinforcement learning from offline or online rewards based on the LLM’s output, for example:

DSP framework Li et al. (2023) where we learn a small tunable policy model to generate the directional stimulus (keywords in this case) that provide input-specific guidance for the LLM toward the desired target

Generated Knowledge prompting (GK)

The next prompting technique under the “Divide and Conquer” concept is Generated Knowledge proposed in Liu et al. (2022). The idea is to use a separate prompt to generate the knowledge first and then use it to get a better response.

Generated knowledge prompting involves two stages:

- **Knowledge generation**: using few-shot demonstrations to generate question related knowledge statements from a language model,
- **Knowledge integration**: using a second language model to make predictions with each knowledge statement, then selecting the highest-confidence prediction.

The schema of Generated Knowledge prompting

The method does not require task-specific supervision for knowledge integration, or access to a structured knowledge base, yet it improves performance of large-scale, state-of-the-art models on commonsense reasoning tasks.

Example of knowledge generation few-shot prompt from Liu et al. (2022) for QASC

Prompt Chaining

Prompt chaining is simple yet powerful technique in which you should split your task into subproblems and prompt the model with them one by one.

Prompt Chaining schema

Prompt chaining is useful to accomplish complex tasks which an LLM might struggle to address if prompted with a very detailed prompt. It also helps to boost the transparency of an LLM application, increases controllability, and reliability.

Prompt Chaining example from txt.cohere.com

Least to Most prompting

Least to Most prompting goes a little bit further adding a step where a model should decide on how to split your task into subproblems.

Schema of Least to Most prompting

Experimental results in in Zhou et al. (2022) reveal that least-to-most prompting is performing well on tasks related to symbolic manipulation, compositional generalization, and math reasoning.

Example of Least to Most prompting from Zhou et al. (2022)

Chain-of-Table prompting

In the recent study (Wang et al. (2024)) a new approach is proposed where tabular data is explicitly used in the reasoning chain as a proxy for intermediate thoughts.

Schema of Chain-of-Table prompting

The algorithm includes a cycle of two steps:

1. **Dynamic planning** at which an LLM samples the next operation from the Operation pool based on the input query and the history of previous operations (Operation Chain),
2. **Argument generation** involves generating arguments for the selected at the previous step operation (such as new column names) using an LLM and application of programming languages to execute the operation and create the corresponding intermediate tables.

Example and comparison of Chain-of-Thought prompting from Wang et al. (2024)

The next two approaches implement the concept of Self-Check - there's a step in the framework that checks the solution. Example of Cgain-of-Table implementation can be found by the link.

Tree of Thoughts (ToT)

Tree of Thoughts generalizes over the Chain-of-Thought approach allowing the model to explore multiple reasoning steps and self-evaluate choices.

To implement ToT technique, we must decide on four questions:

1. How to decompose the intermediate process into thought steps,

2. How to generate potential thoughts from each state,

3. How to heuristically evaluate states (using a state evaluator prompt),

4. What search algorithm to use (Yao et el. (2023))

Tree-of-Thoughts prompting technique

The **input prompt** must include description of the intermediate steps to solve the problem and either sampled thoughts, or instructions on their generation. The **state evaluator prompt** must

provide instructions on which prompts to select for the next step.

Example of Tree-of-Thought for Game of 24 from Yao et el. (2023)

Experiments in Yao et el. (2023) show success of the ToT for tasks requiring non-trivial planning or search. The LangChain has implementation of Tree-of-Thought technique in langchain_experimental.tot.base.ToTChain class.

Reflexion

Reflexion is a framework that reinforces language agents through linguistic feedback. Reflexion agents verbally reflect on task feedback signals, then maintain their own reflective text in an episodic memory buffer to induce better decision-making in subsequent trials (Shinn et al. (2023)) .

Schema of reflexion from from Shinn et al. (2023)

Reflexion framework consists of three distinct models:

- **Actor**: an LLM model that generates text and actions based on the state observations (uses CoT and ReAct),
- **Evaluator**: an LLM model that scores outputs produced by the Actor,
- **Self-Reflection**: an LLM model that generates verbal reinforcement cues to assist the Actor in self-improvement.

Example of Reflexion for different tasks from Shinn et al. (2023)

Reflexion performs well in tasks that require sequential decision-making, coding, language reasoning.

Check out the implementation by the link.

LLM frameworks with external tools

I'll cover two approaches in this section - Retrieval Augmented Generation and ReAct.

Retrieval-Augmented Generation (RAG)

RAG combines an information retrieval component with a text generator model:

- **Retrieval**. At the retrieval step the system searches for relevant documents that might answer the question typically using the vector search,
- **Generation**. Next the relevant documents are passed as a context to an LLM together with the initial question (Lewis et al. (2021)).

In most cases RAG-sequence approach is used meaning that we retrieve *k* documents, and use them to generate all the output tokens that answer a user query.

Schema of RAG

Language models in RAG can be fine-tuned but in reality it's rarely done as pre-trained LLMs are good enough to use as is, and too costly to fine-tune ourselves. Moreover the internal knowledge in RAG can be modified in an efficient manner and without needing retraining of the entire model.

Example of RAG pipeline from deepset.ai

RAG generates responses that are more factual, specific, and diverse, improves results on fact verification.

ReAct

Yao et al. (2022) introduced a framework named ReAct where LLMs are used to generate both *reasoning traces* and *task-specific actions* in an interleaved manner: reasoning traces help the model induce, track, and update action plans as well as handle exceptions, while actions allow it to interface with and gather additional information from external sources such as knowledge bases or environments.

Schema of ReAct

ReAct framework can select one of the available tools (such as Search engine, calculator, SQL agent), apply it and analyze the result to decide on the next action.

Example of ReAct from Yao et al. (2022)

ReAct overcomes prevalent issues of hallucination and error propagation in chain-of-thought reasoning by interacting with a simple Wikipedia API, and generating human-like task-solving trajectories that are more interpretable than baselines without reasoning traces (Yao et al. (2022)).

Check out an example of ReAct implementation with Langchain tools.

Prompt Tuning and Evaluation

Choice of prompt engineering technique highly depends on the application of your LLM and resources available. If you ever experimented with a prompt you know that large language models are sensitive to the smallest changes in human-generated prompts which are suboptimal and often subjective.

No matter what prompting technique you select, if you are building an application it's very important to think of prompt engineering as of a Data Science process. This means creating a test set and choosing metrics, tuning the prompts and evaluating its influence on the test set.

Prompt Engineering as a Data Science process

Metrics for testing a prompt will to high extent depend on the application but here are some guidelines (from Data Science Summit 2023):

Metrics for testing a prompt (from Data Science Summit 2023)

1. Faithfulness and relevancy:

• how factually accurate the generated answer is,

• how relevant the generated answer is to the question.

2. **Retrieval** - for RAG and ReAct pipelines mainly but can be applied to generated knowledge and directional stimulus prompting:

• Precision - how relevant the retrieved documents are,

• Recall - if all the relevant documents were retrieved.

3. **Internal thinking**:

• agents and tools selection accuracy - for ReAct,

• tool argument extraction - if correct arguments are retrieved from the context and properly transofrmed - for ReAct,

• remembering facts in long-turn conversation - for ReAct,

• correct logical steps - for ReAct and Chain-of-Thought prompting.

4. **Non-functional**:

• style and tone of the answer,

• absence of bias,

• compliance and safety checks,

• prompt injection testing.

Depending on your use case, select metrics and track influence of your prompt changes on the test set ensuring that any change doesn't degrade quality of the responses.

Summary

I don't claim to have covered all the existing techniques - there are so many of them that soon someone will publish a whole textbook. But if you read that far, you've noticed that the concepts of all the techniques are quite common and intuitive. I can summarize all the rules of writing a good prompt into a small list:

1. Be clear and precise so that a model doesn't have to guess your intentions,

2. Use delimiters or tags to add structure,

3. Help the model by showing examples and adding explanations,

4. Ask the model to think iteratively, explaining its solution,

5. In case of a complicated prompt, consider splitting it into subtasks,

6. Try asking the same prompt a few times,

7. Consider adding a step of a model self-check,

8. If needed, combine your LLM with external tools,

9. Treat prompt tuning as a Data Science process which is iterative and requires evaluation.

Read More

Godel Receives Official Certified Status for ISO 27001:2022

Godel Technologies has received ISO 27001:2022 certification for information security, cybersecurity and privacy protection. Godel has been certified by the certification body that holds accreditation from the United Kingdom Accreditation Service (UKAS) and the International Accreditation Forum (IAF), an internationally recognisable organisation for accrediting ISO certifications. The ISO/IEC 27001 standard provides companies of any size... Continue reading Prompt Engineering: Classification of Techniques and Prompt Tuning

!’

Modern Nearshore Part 2: How are IT Leaders Leveraging It?

In today’s rapidly evolving world, new technology will come and go, with some changing the technical landscape forever - just look at our previous blog about AI and how the growing phenomenon is dominating boardroom conversations. So, whenever a new tech breakthrough happens, we need the skill set within a business that can help deliver... Continue reading Prompt Engineering: Classification of Techniques and Prompt Tuning

!’