Kaggle | Google

5-Day Gen AI Intensive Course with Google 2025

Whitepaper Companion Podcast (Notes): Prompt Engineering Techniques

Prompt Engineering Techniques for Large Language Models

Introduction to Prompt Engineering

- Prompt engineering is the art of communicating effectively with Large Language Models (LLMs) to achieve specific outputs.
- While anyone can write a prompt, creating effective prompts requires skill and understanding, especially for data-centric tasks on platforms like Kaggle.
- The objective is to provide Kaggle users with practical techniques to enhance their coding and data analysis capabilities.
- Topics covered range from basic concepts to advanced techniques like Chain of Thought and React, tailored for Kaggle challenges.

Configuring Model Output

- Understanding how to configure the output of language models is crucial, as both input and model settings influence the output.
- **Output Length:**
    - The number of tokens generated impacts processing time and costs, which is especially important for Kaggle users with output limits.
    - Targeted prompts are necessary for concise responses, particularly when using the React technique for iterative actions.
- **Sampling Controls:**
    - Influence the randomness of the model's outputs.

    - **Temperature:**
        - Lower temperatures (e.g., 0.1) yield more predictable results, suitable for generating specific code with correct syntax.
        - Higher temperatures (e.g., 0.9) allow for more creativity, useful for brainstorming novel features or exploring different algorithms.
    - **Top K and Top P:**
        - Refine word selection by limiting the next word to the most probable candidates.
        - Top K focuses on a fixed number of candidates.
        - Top P is based on cumulative probabilities.
        - Experimentation with these settings is crucial, as different tasks benefit from different configurations. Combining them can lead to optimal outputs.
- **Repetition Loop Bug:**
    - The model gets stuck repeating the same words or phrases.
    - Fine-tuning temperature, Top K, and Top P is key to avoiding this.
- **Recommendations for Kaggle:**
    - Coherent results with creativity: temperature around 0.2, Top P of 0.95, and Top K of 30.
    - Pushing for creative output: temperature of 0.9, Top P of 0.99, and Top K of 40.
    - Factual accuracy: temperature of 0.1, Top P of 0.9, and Top K of 20.
    - Single correct answer: temperature of zero.

Prompt Engineering Techniques

- Crafting clear prompts is fundamental to obtaining accurate predictions from LLMs.

- **General Prompting (Zero-Shot Prompting):**
    - Involves providing a task description without examples.
    - Effective for generating code snippets based on the model's training.
- **Documenting Prompts:**
    - Vital for Kaggle users to track what works and what doesn’t, facilitating continuous improvement.
- **One-Shot and Few-Shot Prompting:**
    - Provide examples within the prompt to guide the model, improving its understanding of the desired output format and task.
    - The quality of examples is crucial; poorly chosen examples can confuse the model.
    - Include examples of edge cases.
- **System, Role, and Contextual Prompting:**
    - Advanced techniques that provide additional guidance, setting the context and tone for the model's responses.
    - **System Prompting:** Setting the overall context and purpose.
    - **Role Prompting:** Giving the LLM a specific persona or identity to influence the style and tone of responses.
    - **Contextual Prompting:** Providing specific background information relevant to the task.
- **Step-Back Prompting:**
    - Encourages the model to consider broader questions before diving into specific tasks, potentially leading to more insightful outputs.

Advanced Reasoning Techniques

- **Chain of Thought (CoT) Prompting:**
    - Enhances the model's reasoning capabilities by requiring it to articulate intermediate reasoning steps before arriving at a conclusion.
    - Valuable for multi-step reasoning problems, improving transparency and reliability.
- **Self-Consistency:**
    - Involves generating multiple reasoning paths for the same prompt, allowing users to select the most consistent answer, thereby improving reliability.
- **Tree of Thoughts (ToT):**
    - Expands on CoT by enabling the model to explore multiple reasoning paths simultaneously.
    - Suitable for complex and open-ended problems.
- **React (Reason and Act):**
    - Combines the model's reasoning capabilities with the ability to interact with external tools.
    - Enables dynamic responses in Kaggle workflows.
- **Automatic Prompt Engineering (APE):**
    - Allows the model to generate its own prompts.
    - Streamlines the process of finding effective prompts for various tasks.

Code Prompting Applications

- Code prompting includes generating, explaining, translating, and debugging code.
- **Code Generation:**
    - Can significantly accelerate development, but it is crucial to review and test the generated code to ensure accuracy and functionality.

- **Explaining Code:**
    - Helps users understand unfamiliar code snippets, facilitating collaboration and knowledge sharing.
- **Translating Code:**
    - Aids users who encounter algorithms in languages they are not familiar with. Verification of the translated code is necessary.
- **Debugging and Reviewing Code:**
    - Assists users in identifying errors and suggesting improvements, enhancing the robustness and efficiency of their code.
- **Multimodal Prompting:**
    - Includes inputs beyond text.
    - Recognized as an emerging area that may become increasingly relevant in Kaggle competitions.

Best Practices for Prompt Engineering

- Provide examples through one-shot and few-shot prompting to guide the model.
- Design prompts with simplicity in mind to ensure clarity and ease of understanding.
- Be specific about desired outputs, such as required formats, to help the model produce relevant results without ambiguity.
- Use positive instructions rather than constraints to lead to more effective prompts, framing requests positively encourages desired behaviors.
- Control the maximum token length to stay within Kaggle's output limits and manage processing time effectively.
- Create dynamic prompts using variables, which allows for adaptability across different datasets and tasks, enhancing reusability.
- Experiment with different input formats and styles to discover the most effective prompting strategies for various tasks.

- Collaborate with other prompt engineers to exchange ideas and successful strategies, accelerating learning and innovation.
- Document prompt attempts and results to track progress, understand what works best, and debug future issues.
- Adapt to model updates.
- Experiment with output formats for Kaggle structured formats.
- For logical tasks in Kaggle, put the final answer after the reasoning steps and set the temperature to zero.

Conclusion and Future Considerations

- Mastering prompt engineering techniques can provide a competitive advantage in Kaggle competitions.
- Staying updated with new models and features is essential for success in Kaggle.
- Experiment, iterate, and push the boundaries of capabilities with LLMs, fostering a mindset of continuous learning and adaptation.