Here's a re-formatted version of the notes, incorporating more detail and aiming for greater clarity:

Prompt Engineering

- Prompt engineering is designing inputs (prompts) to get the best results from Large Language Models (LLMs).
- An LLM is a prediction engine that takes text as an input and predicts the next token based on its training data.
- The way you write prompts impacts the LLM’s output. Factors include the model, training data, configurations, word choice, style, tone, structure, and context.
- Prompt engineering is an iterative process.

LLM Output Configuration

- After choosing a model, you configure it.
- Most LLMs have configuration options that control the LLM's output.
- Effective prompt engineering requires setting these configurations optimally.
- Key configurations are output length and sampling controls.

Output Length

- Output length is the number of tokens in a response.
- More tokens increase computation, energy consumption, response time, and costs.
- Reducing output length doesn't make the LLM more succinct; it just stops token prediction.
- If a short output is needed, the prompt may need adjustment.
- Output length restriction is important for some LLM prompting techniques, like ReAct, where the LLM can emit useless tokens.

Sampling Controls

- LLMs predict probabilities for the next possible token.
- Token probabilities are sampled to determine the next token.
- Temperature, top-K, and top-P are common settings for this.

Temperature

- Temperature controls randomness in token selection.
- Lower temperatures are for more deterministic responses; higher ones for more diverse output.
- A temperature of 0 (greedy decoding) is deterministic, selecting the highest probability token.
- Temperatures near the maximum produce more random output.

- As temperature increases, all tokens become equally likely.

Top-K and Top-P

- Top-K and top-P restrict the next token to those with the highest predicted probabilities.
- Like temperature, these control the randomness and diversity of generated text.
- Top-K sampling selects the top K most likely tokens. Higher K means more varied output; lower K, more restrictive output.
- A top-K of 1 is equivalent to greedy decoding.
- Top-P sampling selects the top tokens whose cumulative probability doesn't exceed a value (P), ranging from 0 (greedy decoding) to 1 (all tokens).
- The best way to choose between top-K and top-P is to experiment.

Putting it all together

- The choice between top-K, top-P, temperature, and output length depends on the application and desired outcome, as these settings impact each other.
- It’s also important to understand how your chosen model combines the different sampling settings.
- If temperature, top-K, and top-P are all available, tokens that meet both the top-K and top-P criteria are candidates for the next predicted token, and then temperature is applied to sample from the tokens that passed the top-K and top-P criteria.
- If only top-K or top-P is available, the behavior is the same, but only the one top-K or P setting is used.
- If temperature is not available, whatever tokens meet the top-K and/or top-P criteria are then randomly selected from to produce a single next predicted token.
- Extreme settings of one sampling configuration value can cancel out others or become irrelevant.
    - If temperature is 0, top-K and top-P are irrelevant; the most probable token is predicted.
    - If temperature is extremely high (above 1), it becomes irrelevant, and tokens meeting top-K and/or top-P criteria are randomly sampled.
    - If top-K is 1, temperature and top-P are irrelevant.
    - If top-K is extremely high, any token with a nonzero probability will meet the top-K criteria.
    - If top-P is 0, temperature and top-K are irrelevant.
    - If top-P is 1, any token with a nonzero probability will meet the top-P criteria.
- A temperature of 0.2, top-P of 0.95, and top-K of 30 gives coherent, moderately creative results.

- For more creative results, start with a temperature of 0.9, top-P of 0.99, and top-K of 40.
- For less creative results, start with a temperature of 0.1, top-P of 0.9, and top-K of 20.
- For tasks with a single correct answer (e.g., math), start with a temperature of 0.
- More freedom in LLM settings can generate less relevant text.

WARNING: The Repetition Loop Bug

- A common issue in LLMs is the repetition loop bug, where the model repeats a word, phrase, or sentence, often due to inappropriate temperature and top-K/top-P settings.
- This occurs at both low and high temperatures.
- At low temperatures, the model becomes overly deterministic, which can lead to a loop.
- At high temperatures, the model's output becomes excessively random, increasing the probability of repetition.
- In both cases, the model's sampling process gets "stuck," resulting in monotonous and unhelpful output.
- Solving this requires carefully balancing determinism and randomness by adjusting temperature and top-K/top-P values.

Prompting Techniques

- LLMs are tuned to follow instructions and are trained on large amounts of data.
- Clearer prompts help LLMs predict the next likely text.
- Specific techniques leverage LLM training to get relevant results.

General Prompting / Zero Shot

- A zero-shot prompt is the simplest, providing only a task description.
- The input could be a question, story start, or instructions.
- "Zero-shot" means 'no examples' are provided.

One-Shot & Few-Shot

- Providing examples helps AI models understand what is asked.
- Examples are useful to steer the model to a certain output structure or pattern.
- A one-shot prompt provides a single example for the model to imitate to complete the task.
- A few-shot prompt provides multiple examples, showing a pattern for the model to follow.
- The number of examples needed depends on task complexity, example quality, and gen AI model capabilities.

- Use at least three to five examples for few-shot prompting.
- Examples should be relevant, diverse, high-quality, and well-written.
- Include edge cases for robust output.

System, Contextual, and Role Prompting

- These guide LLMs to generate text, focusing on different aspects:
  - System prompting sets the overall context and purpose.
  - Contextual prompting provides specific, relevant details.
  - Role prompting assigns a character or identity.
- Distinguishing these provides a framework for designing prompts with clear intent and makes it easier to analyze how each prompt type influences the language model's output.

System Prompting

- System prompts generate output that meets specific requirements.
- A 'system prompt' provides an additional task.

Role Prompting

- Role prompting guides LLMs by assigning a specific role or persona.
- This is effective in creative writing, Q&A, and interactive applications, helping the model generate relevant and contextually appropriate responses.

Contextual Prompting

- Contextual prompting provides specific details or background information.
- By grounding the LLM in a context, you can generate more focused and relevant outputs.

Step-Back Prompting

- Step-back prompting improves LLM reasoning.
- The LLM first considers a general question before a specific task.
- The general question's answer is then used in a prompt for the specific task.
- This is inspired by how humans break down complex problems.
- This technique can increase prompt accuracy.

Chain of Thought (CoT)

- Chain of Thought (CoT) prompting improves LLM reasoning by generating intermediate steps for more accurate answers.
- Combining it with few-shot prompting improves results on complex reasoning tasks.
- CoT is low-effort, effective, and works with off-the-shelf LLMs (no fine-tuning

needed).

- CoT improves interpretability, allowing you to learn from the LLM's responses and see the reasoning steps that were followed.
- LLM responses include chain of thought reasoning, which means more output tokens, which means predictions cost more money and take longer.

Self-Consistency

- Self-consistency improves LLM accuracy in reasoning or problem-solving tasks.
- It builds on Chain of Thought (CoT) prompting.
- Instead of a single chain of thought, it generates multiple diverse reasoning paths.
- The final answer is selected based on the consistency among these paths.
- If a model reasons correctly, it should arrive at the same answer through different lines of reasoning.
- Checking for agreement among multiple reasoning paths reduces the likelihood of a correct answer by chance or superficial pattern matching.

ReAct (Reason & Act)

- ReAct enhances LLM ability to perform complex tasks by combining reasoning and acting.
- It enables LLMs to interact with environments or tools to gather information and take actions, in addition to generating natural language.
- ReAct is inspired by how humans solve problems, combining deduction with actions that modify our environment.
- The LLM generates interleaved reasoning and action steps.
- Reasoning helps the model think through the problem, plan, and decide what actions to take.
- Action steps let the model interact with external environments like search engines or databases.
- By combining reasoning and acting, ReAct allows LLMs to tackle more complex and dynamic tasks.

Code Prompting

- The same regular large language model can be used for code prompting.

Multimodal Prompting

- Multimodal prompting uses multiple input formats, instead of just text, to guide a large language model.

Best Practices

- Effective prompt engineering involves providing examples, simplicity, clear

instructions, controlling output length, using variables, experimenting with formats, and adapting to model updates.
- For CoT prompting, put the answer after the reasoning.
- For CoT prompting, set the temperature to 0.
- Document prompt attempts in full detail.

JSON Repair

- JSON, while good for parsing, requires more tokens than plain text, increasing processing time and costs.
- JSON verbosity can consume the entire output window, and truncation can result in invalid JSON.
- Tools like json-repair can automatically fix incomplete or malformed JSON, which is crucial when dealing with potential truncation issues.

Working with Schemas

- JSON Schemas define the expected structure and data types of a JSON object, and can be used to structure LLM input.