



SES206

From Prompt Engineering to Auto-Prompt Optimization

Philipp Kaindl

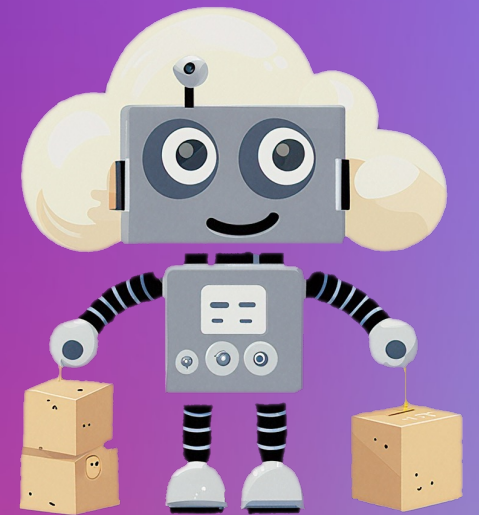
(he/him)

Sr. AI/ML Specialist Solutions Architect
Amazon Web Services

Stefan Christoph

(he/him)

Principal Solution Architect
Amazon Web Services



Agenda

- Introduction
- Hard Prompting Optimization
- Soft Prompting Optimization
- Deep Dive Time – Solution Walkthrough



Introduction



Your speakers today



Philipp Kaindl

(he/him)

Sr. AI/ML Specialist Solutions
Architect
AWS



Stefan Christoph

(he/him)

Principal Solutions Architect
AWS



A lay of the land – what wider techniques are available to tune model output?

Hard Prompt tuning

Titan Text G1 - Express
Change model

User: List me three different properties for hard prompts:

Bot: Hard prompts are a type of prompt that are designed to be more challenging for a language model to generate responses to. Here are three different properties:

1. Specificity: Hard prompts should be highly specific and detailed. They should provide clear instructions to the language model on what response is expected.
2. Complexity: Hard prompts should be complex and nuanced. They should require the language model to engage in higher-level thinking and generate a more sophisticated response.
3. Uncertainty: Hard prompts should contain uncertainty or ambiguity. This can be achieved by providing multiple possible answers or scenarios, or by asking questions that encourage the language model to explore different possibilities and generate creative solutions.

By incorporating these three properties into hard prompts, you can create more challenging and engaging tasks for language models to solve.

User: You are an Generative AI expert. List me three different properties for hard prompts. Be crisp and prescriptive.

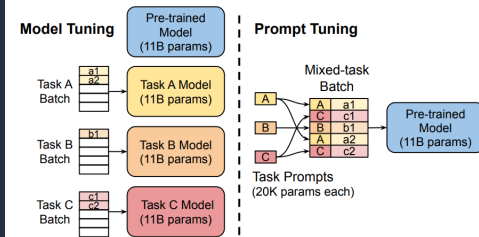
Bot: Hard prompts are a specific type of prompt designed to challenge a language model to generate creative and original responses. Here are three key properties:

1. Specificity: Hard prompts should be highly specific and detailed, providing clear instructions to the language model on what response is expected. This includes providing constraints, and specifying the desired tone, style, and structure of the response.
2. Complexity: Hard prompts should be complex and nuanced, requiring the language model to engage in higher-level thinking and generate a more sophisticated response.
3. Uncertainty: Hard prompts should contain uncertainty or ambiguity, which can be achieved by providing multiple possible answers or scenarios, or by asking questions that encourage the language model to explore different possibilities and generate creative solutions.

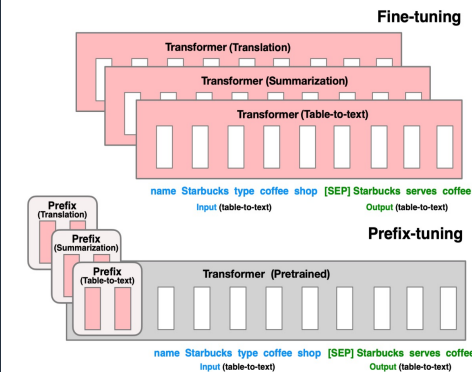
By incorporating these three properties into hard prompts, you can create tasks that are more challenging and engaging for language models, leading to more original responses.

Write a prompt... (Shift + ENTER to start a new line, and ENTER to generate a response)

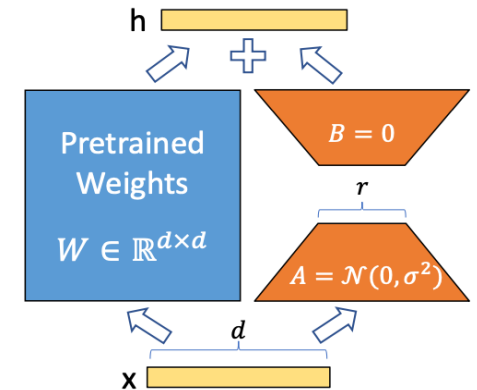
Soft Prompt tuning



Prefix tuning



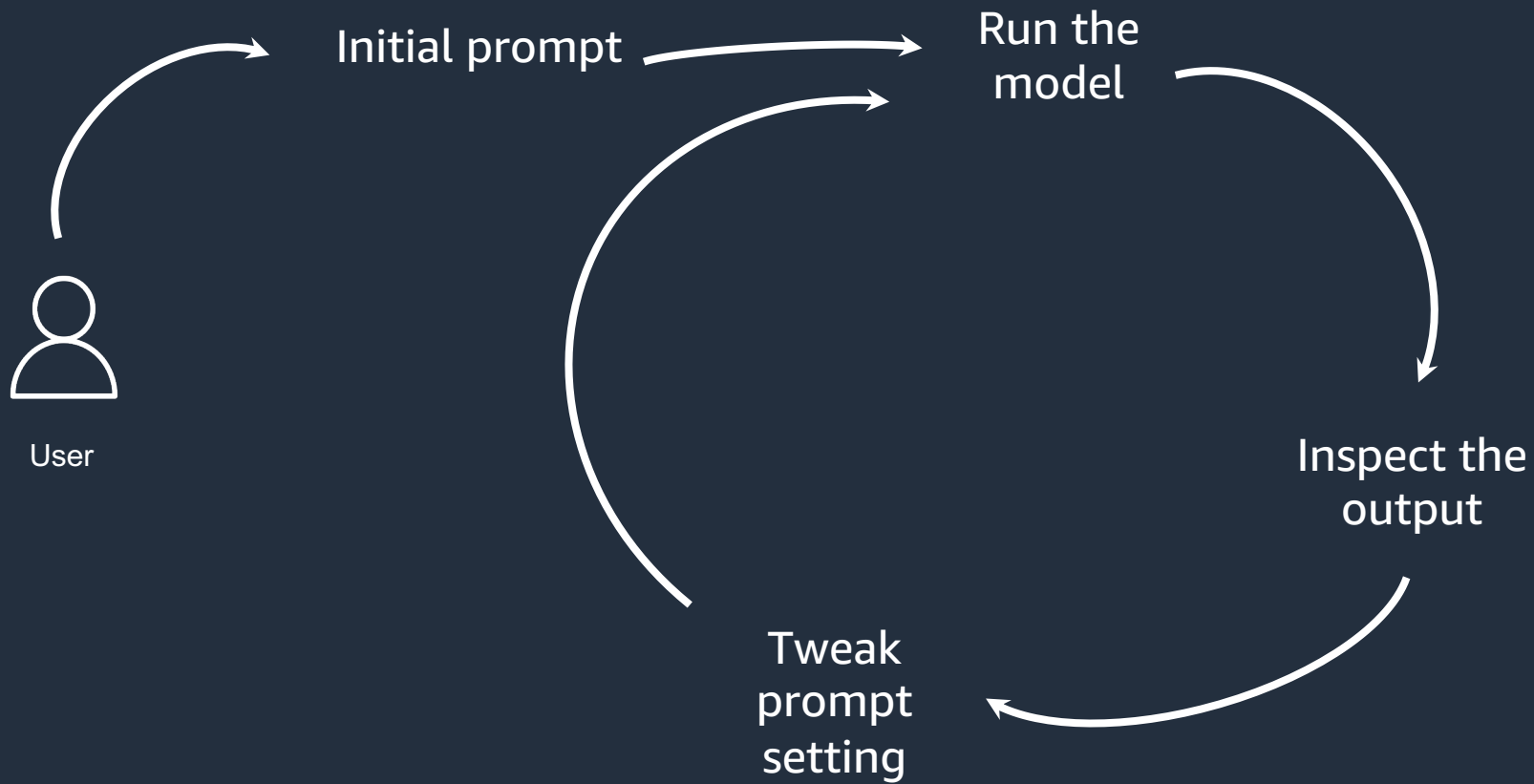
LoRA fine tuning



Hard Prompt Optimization



Hard prompt optimization from a user perspective



Hard Prompt Optimization

Advantages:

- Human readable
- Open to non-technical users

Potential:

- Flexibility by using templates

Drawbacks:

- Optimization often requires trial & error => Large Effort
- Prompt formats differs slightly between models
- Static in nature and hence inflexible

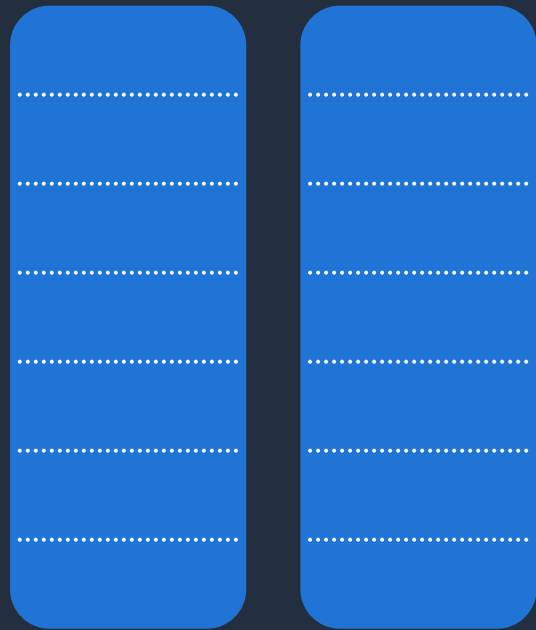
Soft Prompt Optimization



Soft Prompts

Trainable Parameters

Virtual Tokens



Input Prompt

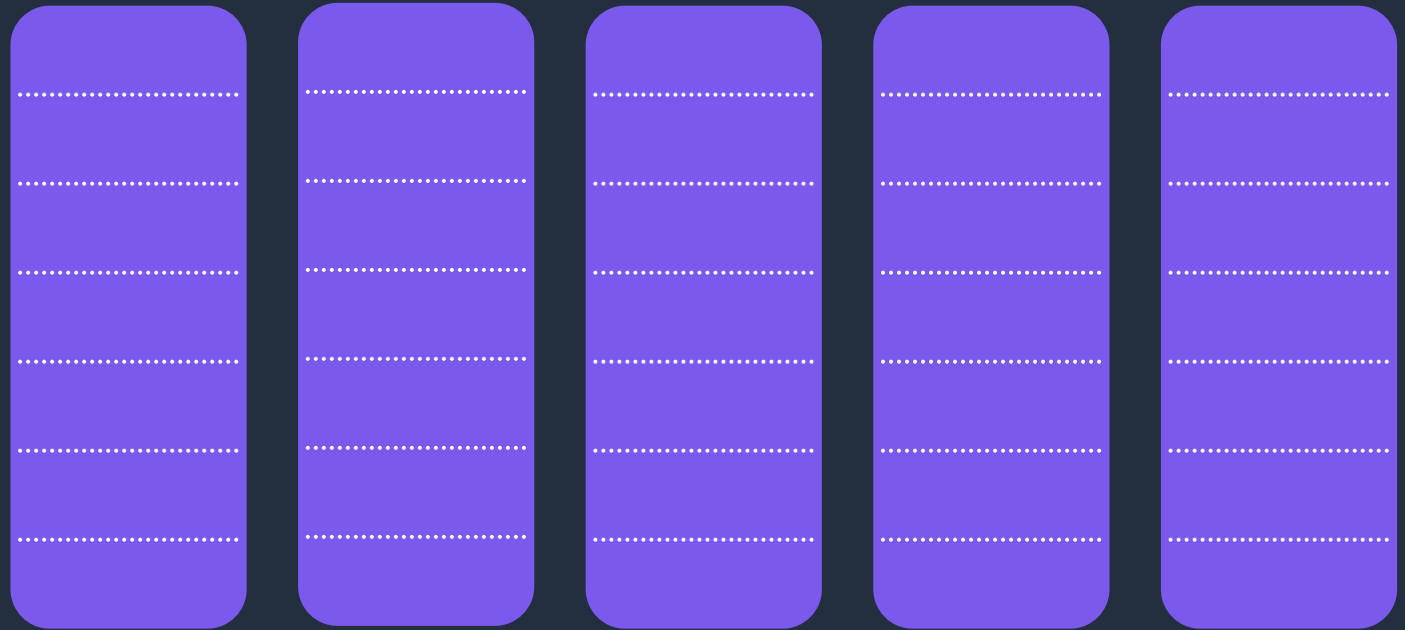
Always

take

the

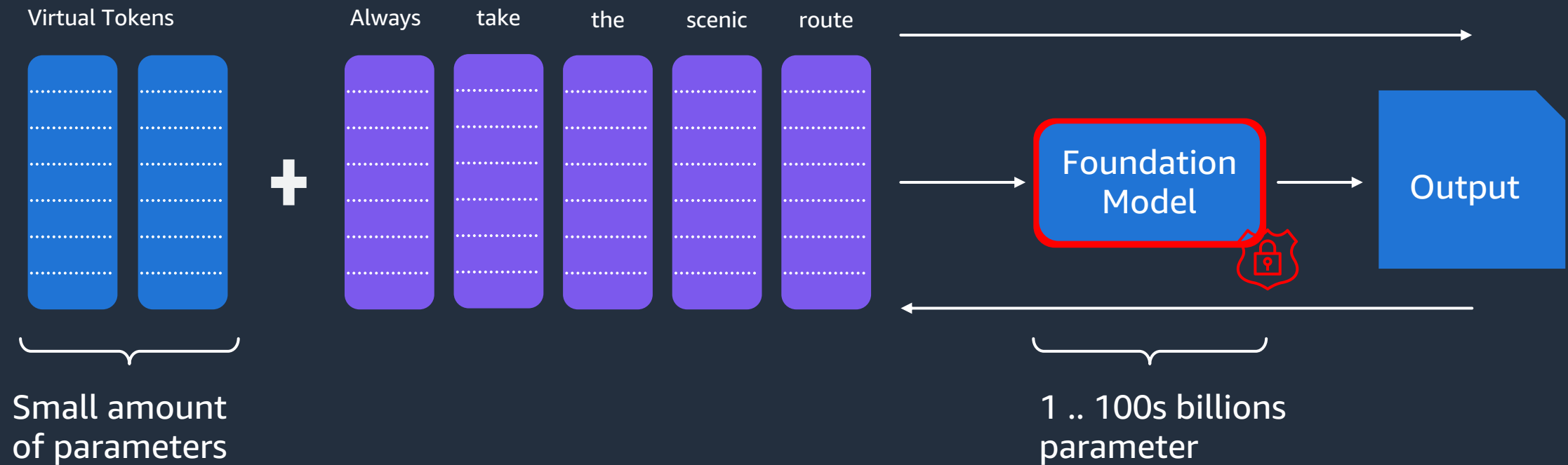
scenic

route

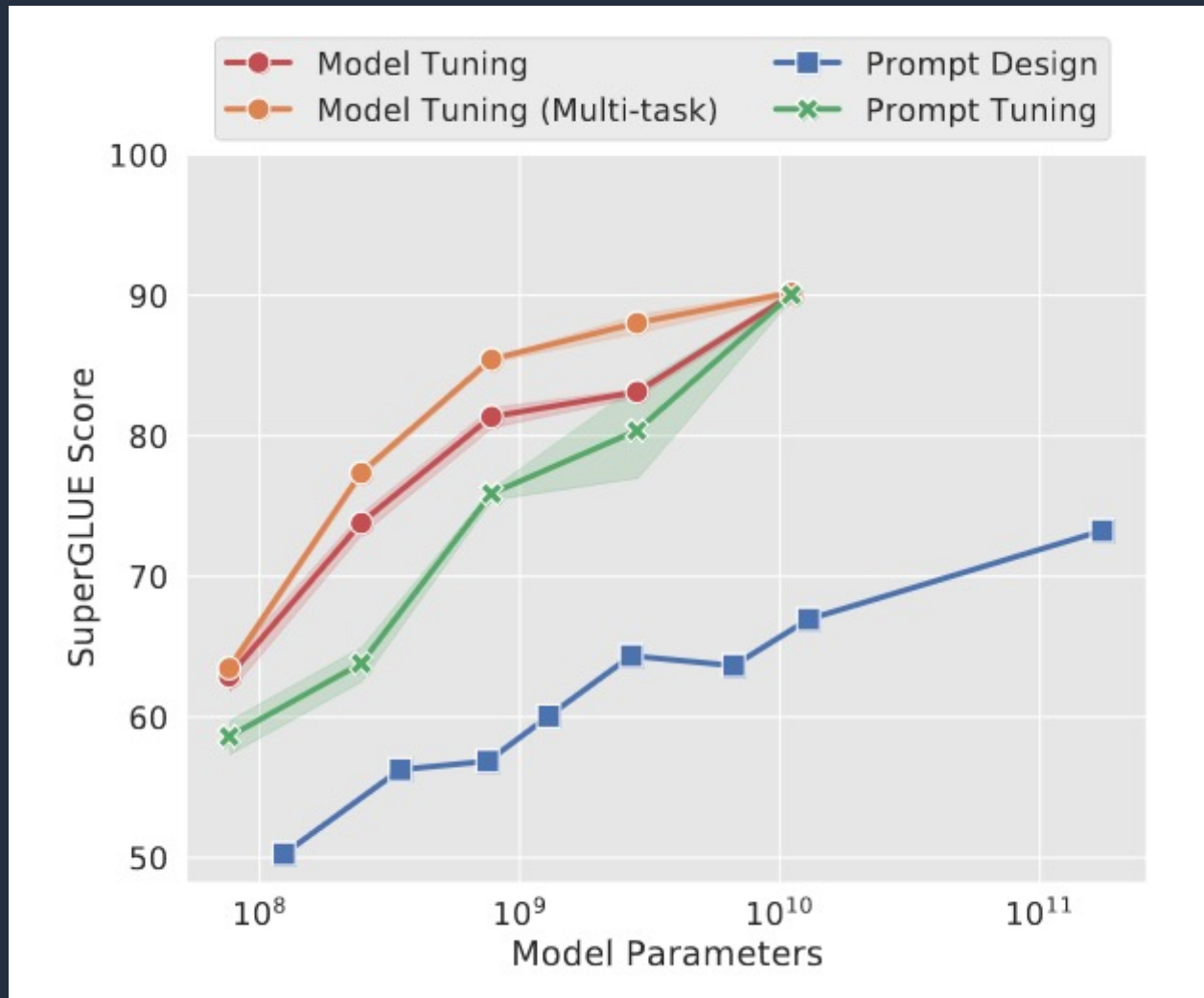


Task specific

Soft Prompts - Finetuning



Soft Prompts Finetuning – does it work?



Source: [The Power of Scale for Parameter-Efficient Prompt Tuning](#)

Soft Prompts - Summary

Advantages:

- Flexibility - easy adaptation across different tasks without changing entire model
- Efficiency - more versatile strategy, particularly useful in multi-task learning

Potential:

- Enable more nuanced and adaptable responses from AI models

Drawbacks:

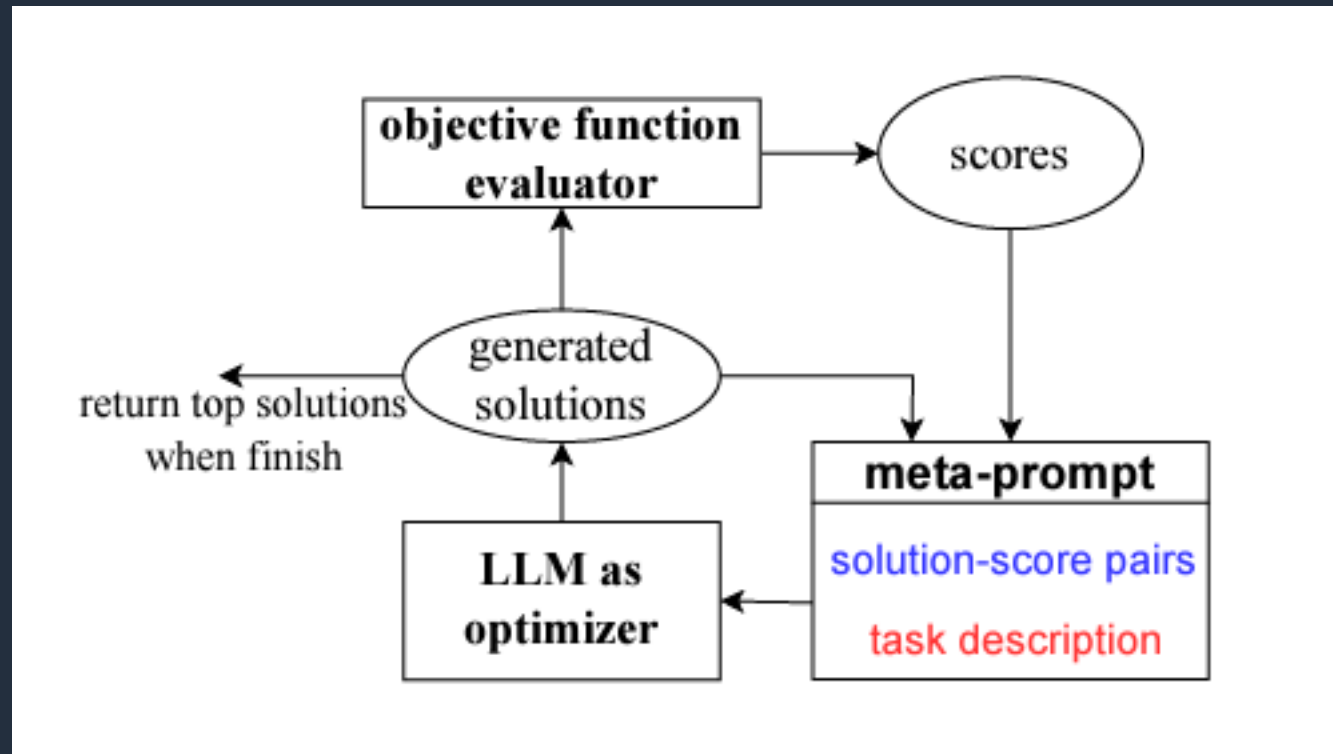
- Less interpretable by humans due to their non-linguistic nature
- Work best if model is open-source and user can take full control.
- Might be less relevant with LoRA Adapters

Automation of the optimization process



How to do this in a more automated fashion?

- Use LLMs as optimizer !



Source: [LARGE LANGUAGE MODELS AS OPTIMIZERS](#)

LLMs as optimizers

Prompting with natural language feedback. A recent line of work investigates approaches to improve the LLM performance by prompting with natural language feedback to revise the model output, which has shown effectiveness in reducing harmful LLM outputs (Bai et al., 2022; Ganguli et al., 2023), improving reasoning (Shinn et al., 2023; Madaan et al., 2023) and code generation performance (Chen et al., 2023e; Olausson et al., 2023; Shinn et al., 2023; Chen et al., 2023b), dialogue applications (Nair et al., 2023; Madaan et al., 2023; Yuan et al., 2023), and so on (Kim et al., 2023; Wang et al., 2023). Specifically, Yuan et al. (2023) develops a human-in-the-loop framework for deriving system-level feedback from a collection of instance-level feedback, which is then used for refining data. In our work, the optimizer LLM utilizes the optimization trajectory in the prompt, which implicitly requires the LLM to summarize the common characteristics among solutions with similar scores. We consider incorporating explicit natural language feedback on generated solutions for later optimization steps as future work.

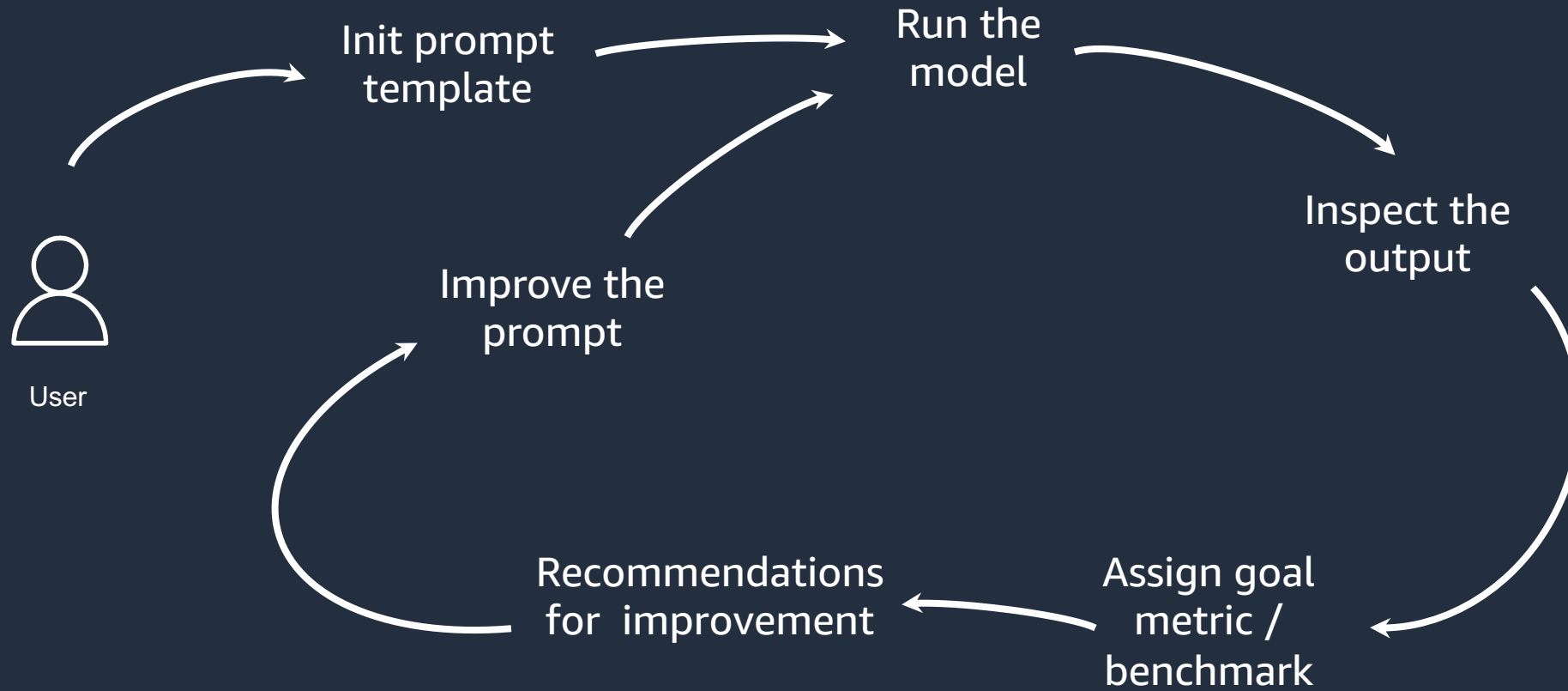


We do this

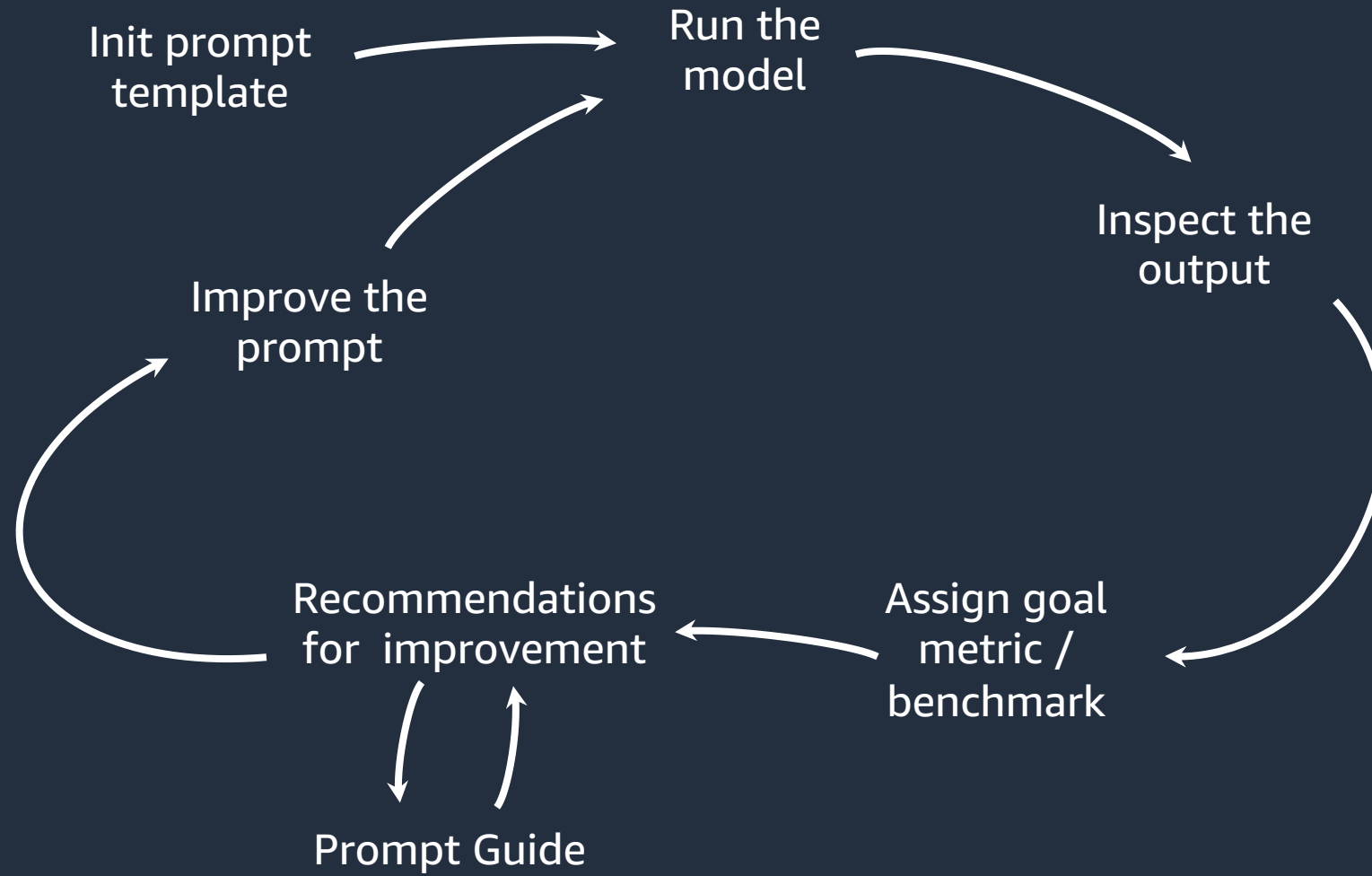
Tuning language models for optimization. Some previous works tune or prompt language models to behave as mutation and crossover operators in evolutionary algorithms. Meyerson et al. (2023) utilizes language models with few-shot exemplars to propose evolutionary cross-overs on tasks such as image and code generation. In Lehman et al. (2022), the large language model trained on code diff generation is used as the mutation operator, and they further design a fine-tuning method to improve performance in the Sodarace domain for robot simulation. EvoPrompting (Chen et al., 2023a) uses large language models to evolve neural network architectures, where they combine evolutionary search with soft prompt tuning. With respect to taking the trajectory as the input for optimization, OptFormer (Chen et al., 2022) trains a transformer model on large collections of hyperparameter optimization data. On the other hand, our work performs optimization solely by prompting without additional training.

Source: [LARGE LANGUAGE MODELS AS OPTIMIZERS](#)

From Experimentation to a more systematic approach



How do we orchestrate the prompt optimization?



Different Ways to orchestrate

Complexity




1. Build it yourself from the ground up

...

2. Use a Library like  LangGraph

...

3. Use a Framework like  or autogen



Learning



Guide the Optimization Process by setting Goals

Qualitative Goals

- Example: Generate Marketing E-Mails

How well do the output adheres to the provided style guidelines?

- Example: Instruction Generation

How easy is it to follow the instructions?

Quantitative Goals

- How much of a score can I achieve?
- Potentially a lot of effort if you use entire data set => use a subset



This image was generated by Titan Image Generator G1 v1 on Amazon Bedrock.

On Quantitative Goals

Things to consider:

- Benchmark Data
- Evaluation Function
- Output Parsing
- Multithreading to reduce execution time

Priority



Deep Dive Time – Solution Walkthrough

A diver is swimming upside down in a clear blue ocean. The diver is wearing a black wetsuit and fins, and is holding a camera. The background shows a coral reef and many small fish swimming in the water.

“Dive Deep: Leaders operate at all levels, stay connected to the details, audit frequently, and are skeptical when metrics and anecdote differ. No task is beneath them.” – [Amazon Leadership Principle](#)

Wrap Up and References

Key Takeaways:

- Try to automate as much as possible
- Apart from Finetuning, optimization outside of models is possible
- Use available library and frameworks

References:

- Paper: [The Power of Scale for Parameter-Efficient Prompt Tuning](#)
- Paper: [LARGE LANGUAGE MODELS AS OPTIMIZERS](#)
- Blogpost: [From Prompt Engineering to Auto Prompt Optimisation](#)
- Code on GitHub: [Soft-Prompt Optimization](#)

Github Repo





Thank you!

Philipp Kaindl
(he/him)
Sr. AI/ML Specialist
Solutions Architect
Amazon Web Services

Stefan Christoph
(he/him)
Principal Solution Architect
Amazon Web Services



Please complete the session survey.

