

# **Optimizing Prompts for Large Language Models Using Reinforcement Learning**

**Mingkai Deng**

**Practical LLM Workshop**

**March 10, 2023**

# Large language models (LMs) show impressive success performing various tasks

A In which year Donald Trump became president ?

 Donald Trump became the President of the United States in January 2017. ↑ ↓

A What is name of his wife and daughter ?

 Donald Trump's wife is named Melania Trump and his daughter is named Ivanka Trump. ↑ ↓

A Cool.. and what about his mom and dad ?

 Donald Trump's parents were Frederick Christ Trump and Mary Anne Trump. His father was a successful real estate developer and his mother was a homemaker. ↑ ↓

Imagine I'm at the deepest part of the ocean

LaMDA

You're on the bottom of the ocean!

There's a giant sea snake swimming right over your head! It's a moray eel! It looks like it's waving at you!

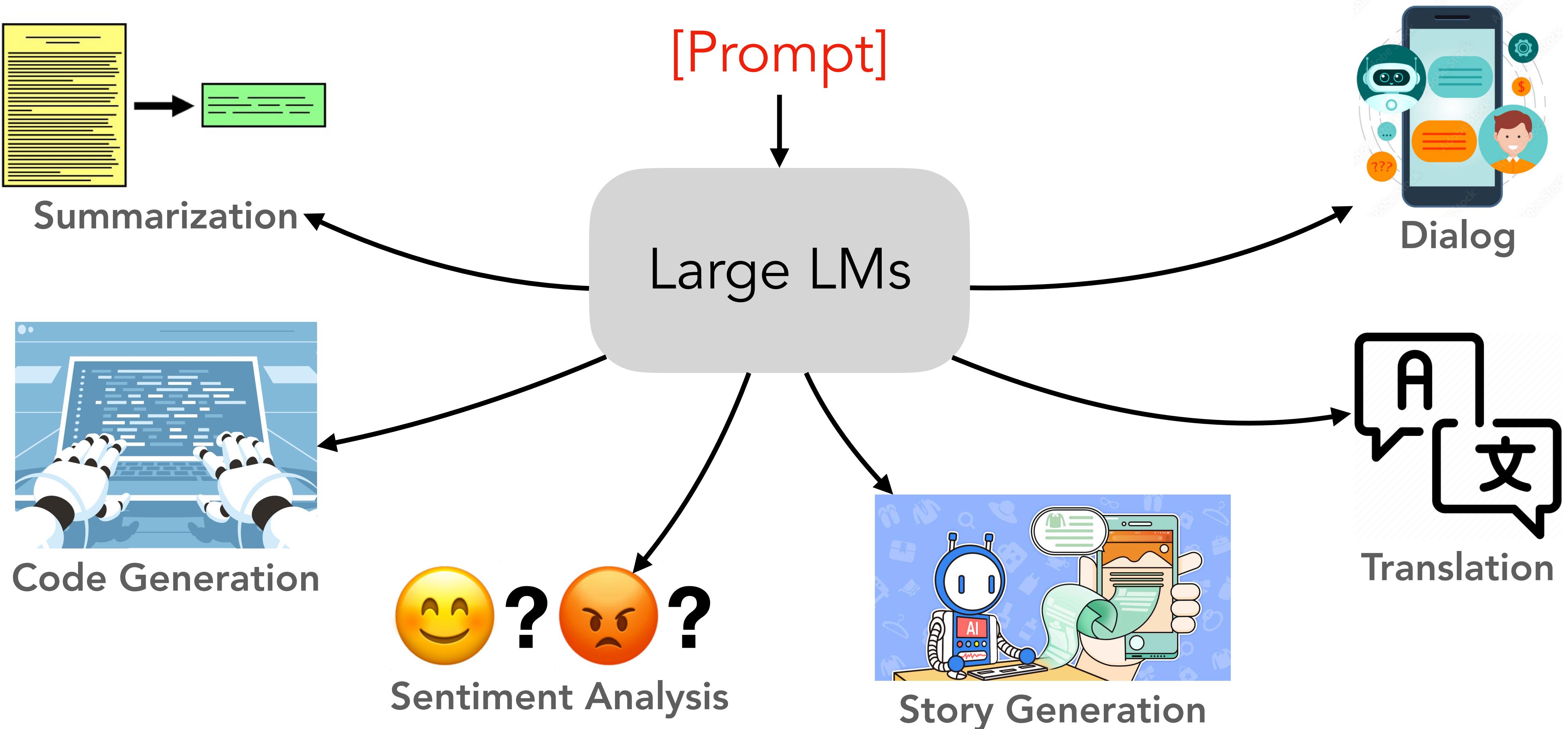
can i see any other sea creatures?

what does it smell like?

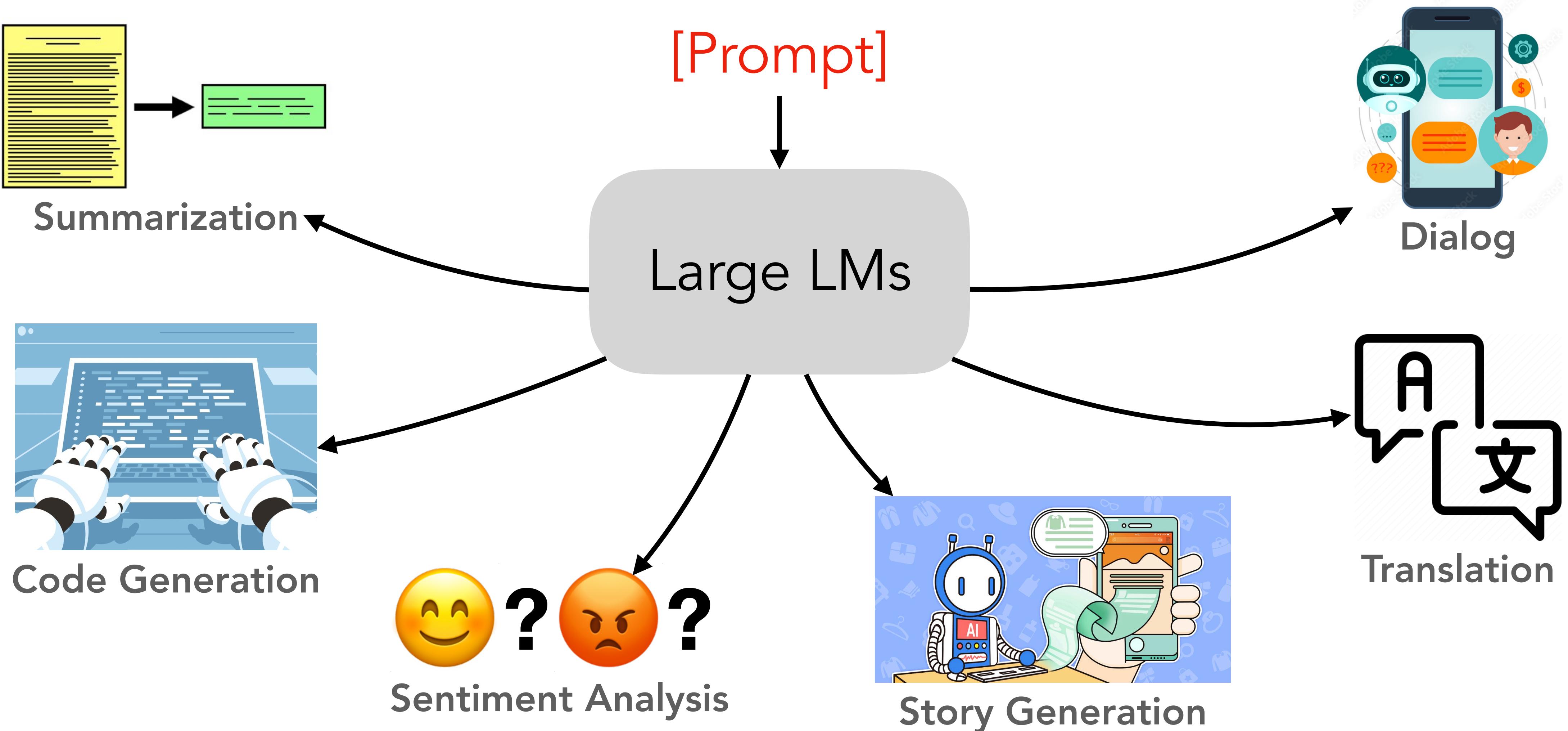
what color is the moray eel?

 more ideas...

# Large language models (LMs) show impressive success performing various tasks



# LLMs can be sensitive to prompt wordings

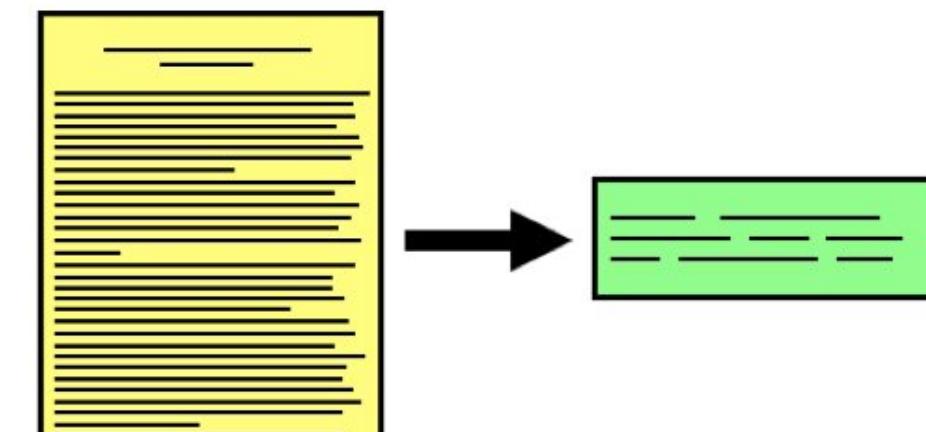


# LLMs can be sensitive to prompt wordings

"Summa[~~Prompt~~ this] article"

"The summary is"

Large LMs



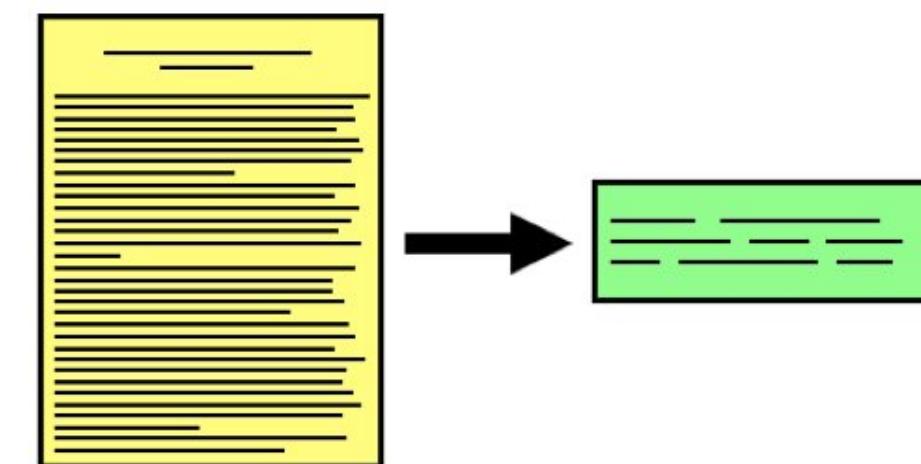
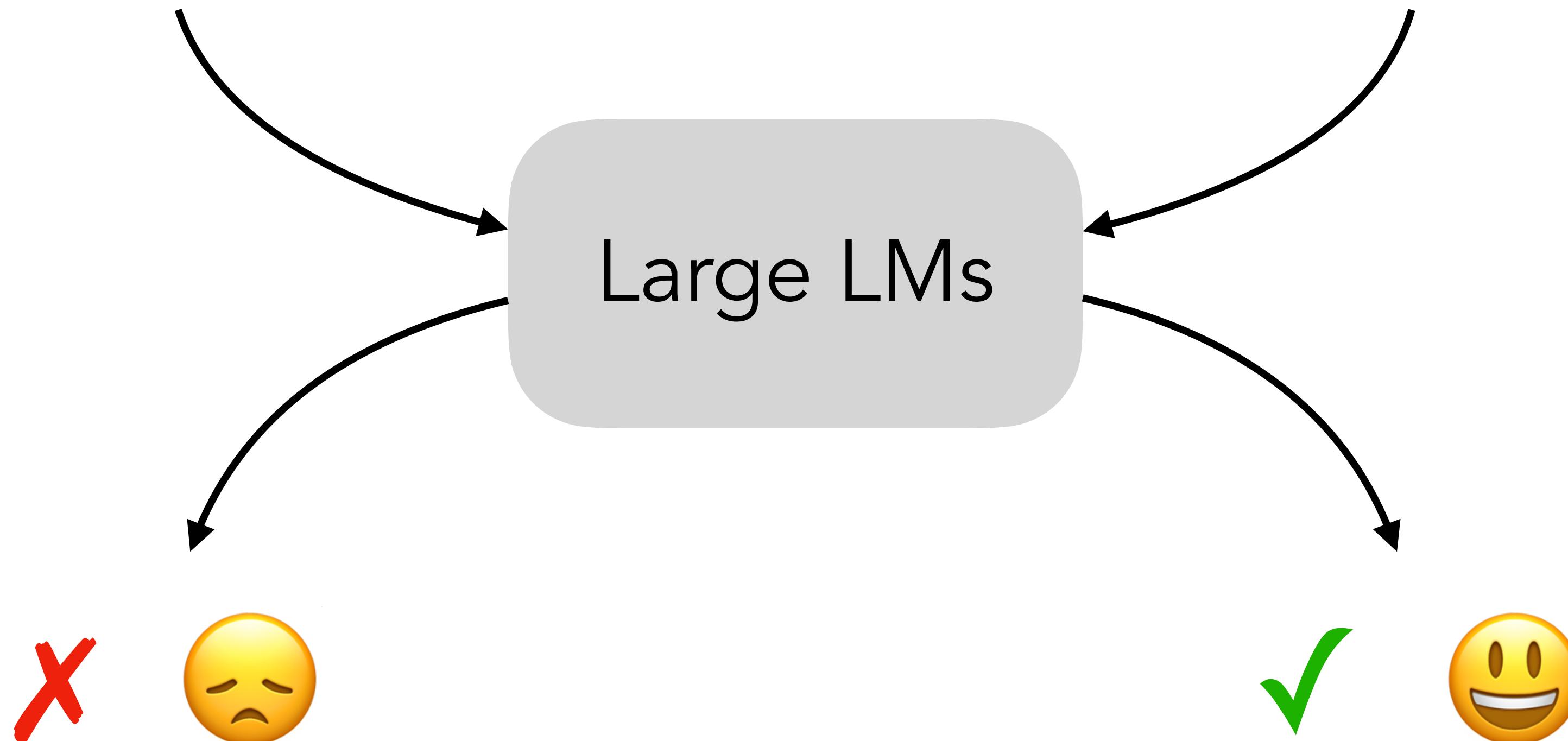
Summarization

What is the best way to talk to them?

How can we make LLMs do what we want?

# Finding the Best Prompt Can Be Challenging

“Summarize this article”      “The summary is”



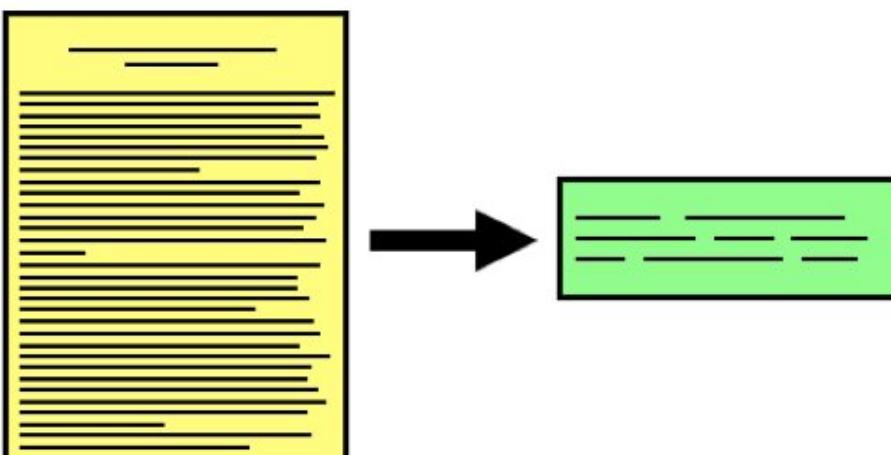
Summarization

# Finding the Best Prompt Can Be Challenging

- Manual Design

“Summarize this article”

Large LMs



Summarization

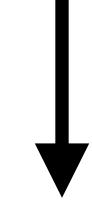
# Finding the Best Prompt Can Be Challenging

- Manual Design

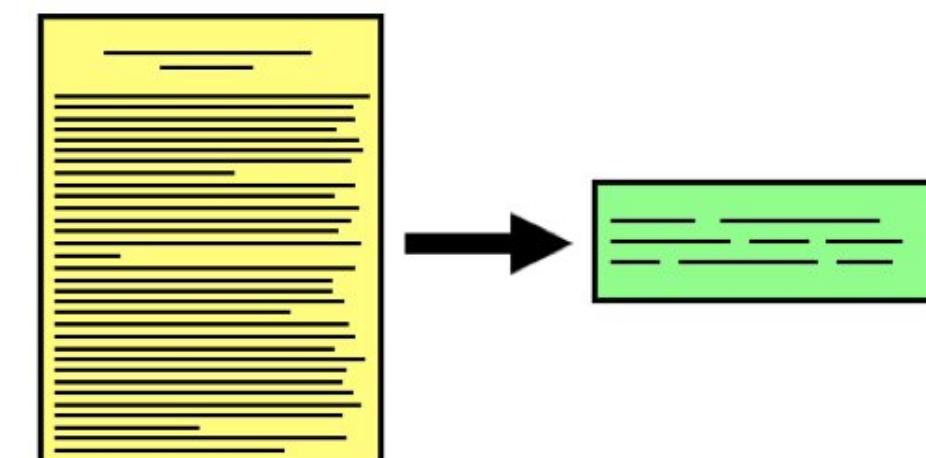
“Summarize **the** article”



Large LMs



**X**

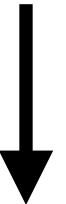


Summarization

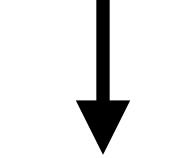
# Finding the Best Prompt Can Be Challenging

- Manual Design

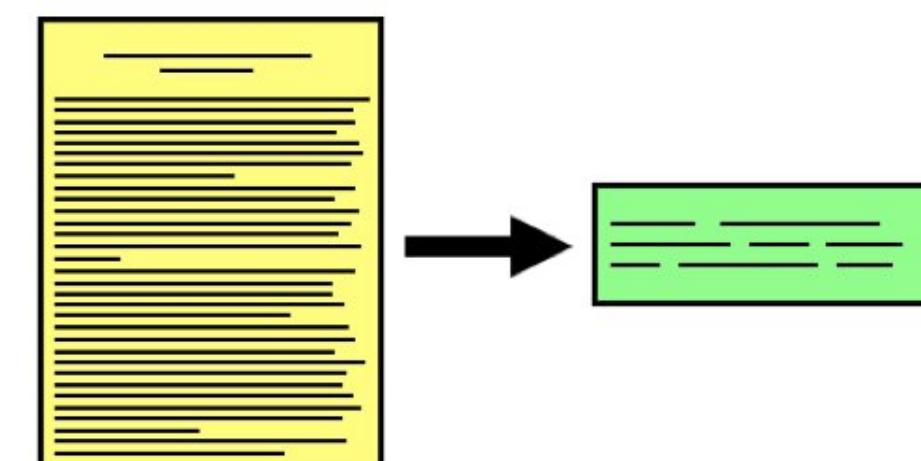
“Summarize the **essay**”



Large LMs



✗



Summarization

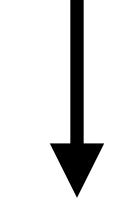
# Finding the Best Prompt Can Be Challenging

- Manual Design
- Enumerate and Select

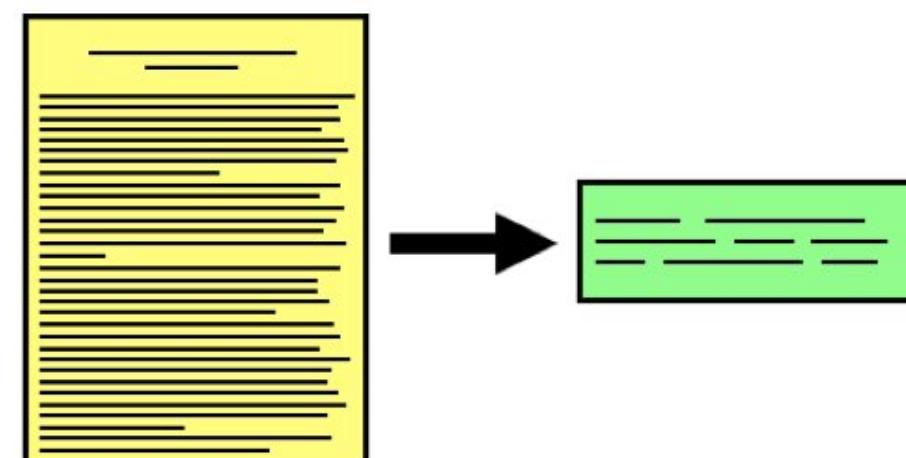
“Summarize the **essay**”



Large LMs



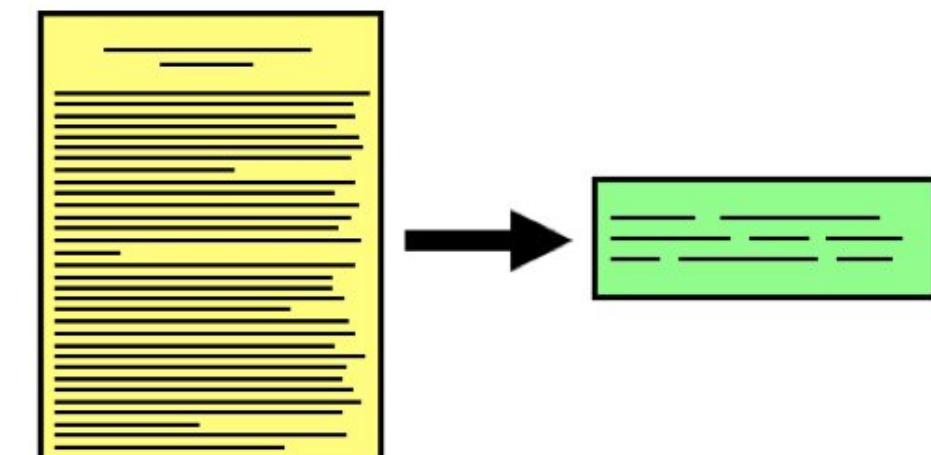
**X**



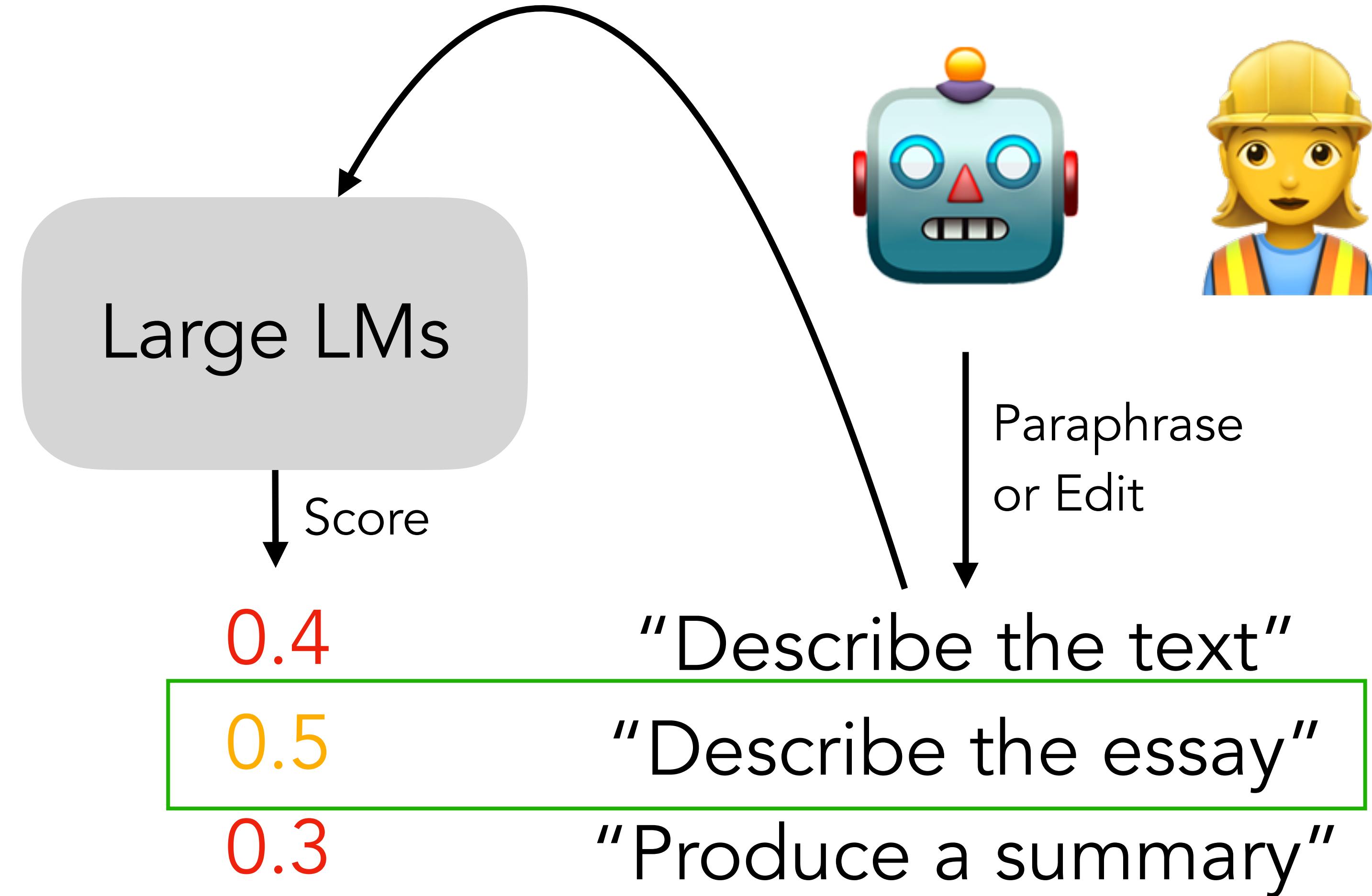
Summarization

# Finding the Best Prompt Can Be Challenging

- Manual Design
- Enumerate and Select

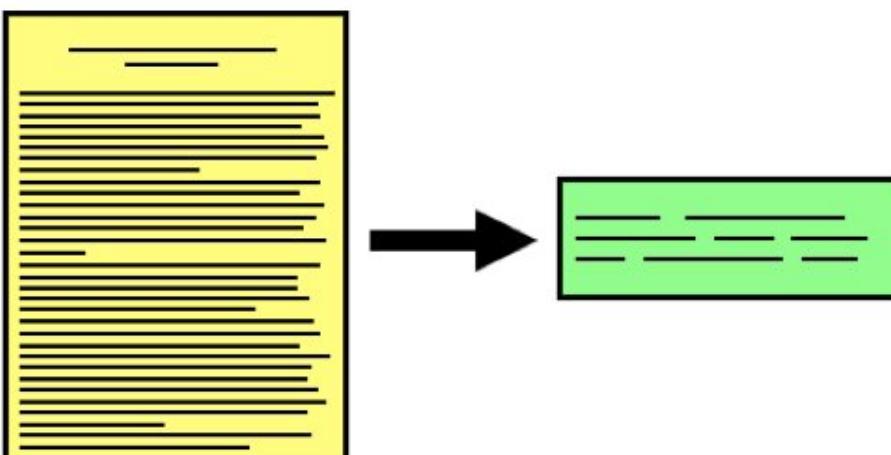


Summarization

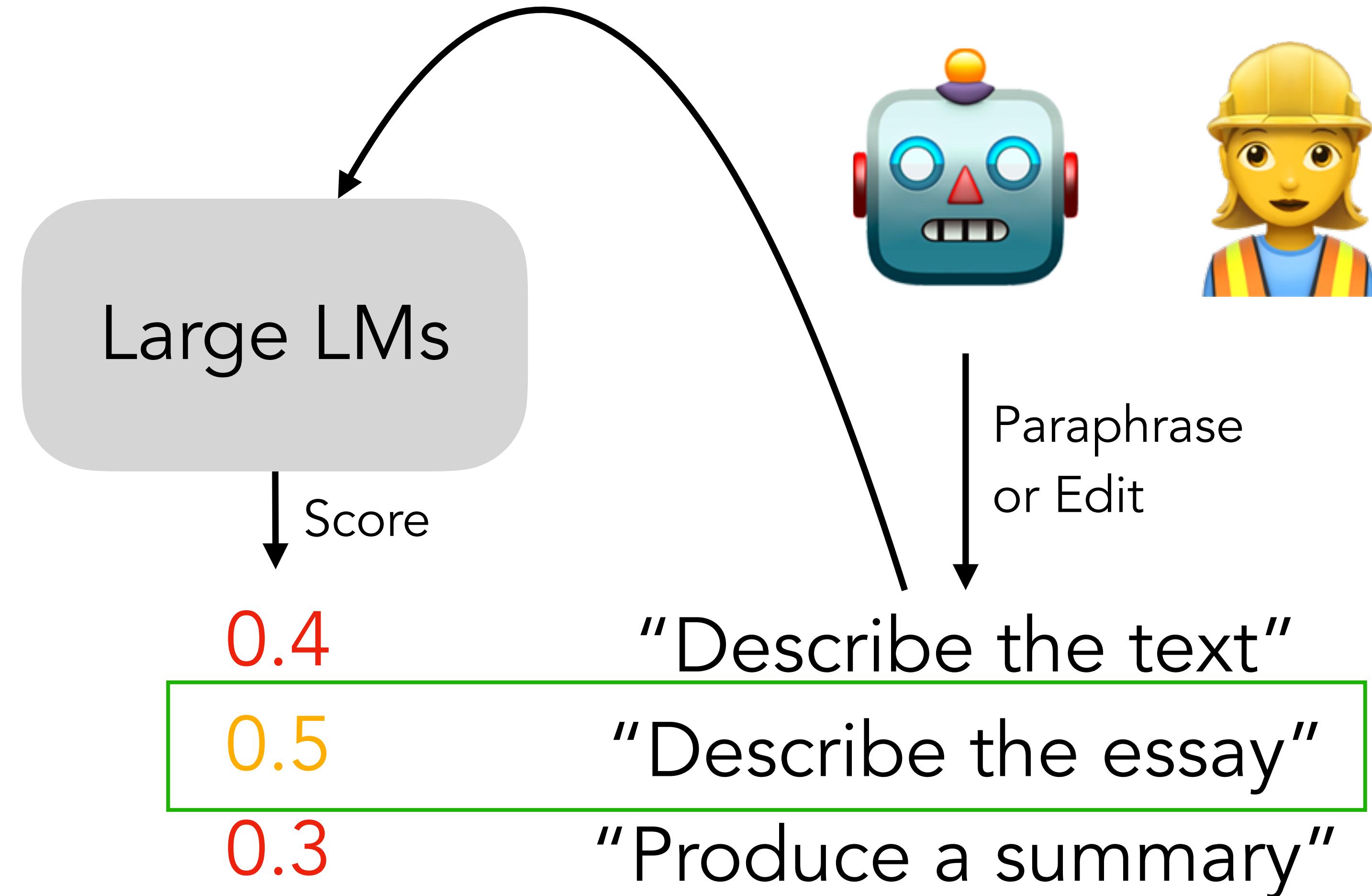


# Finding the Best Prompt Can Be Challenging

- Manual Design
- Enumerate and Select
- Numerical Tuning



Summarization



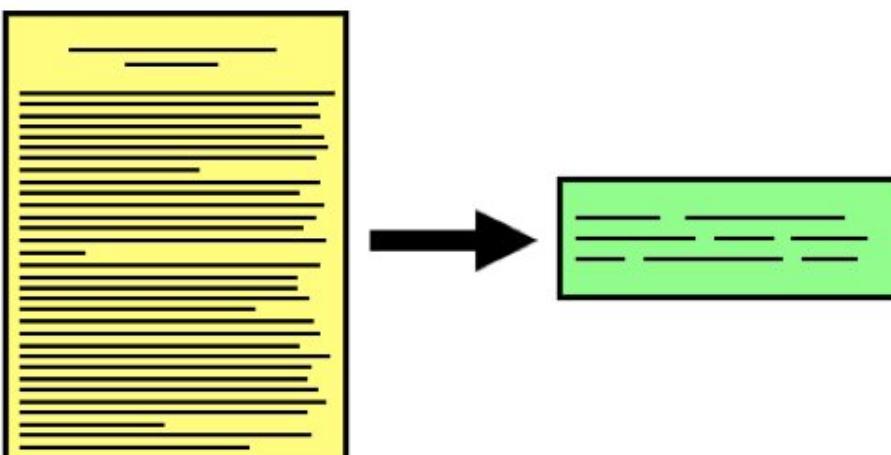
# Finding the Best Prompt Can Be Challenging

- Manual Design
- Enumerate and Select
- Numerical Tuning

Large LMs

0.5	0.3	-0.1
0.2	-0.3	0.6
-0.6	0.4	0.8
0.1	-0.2	0.3

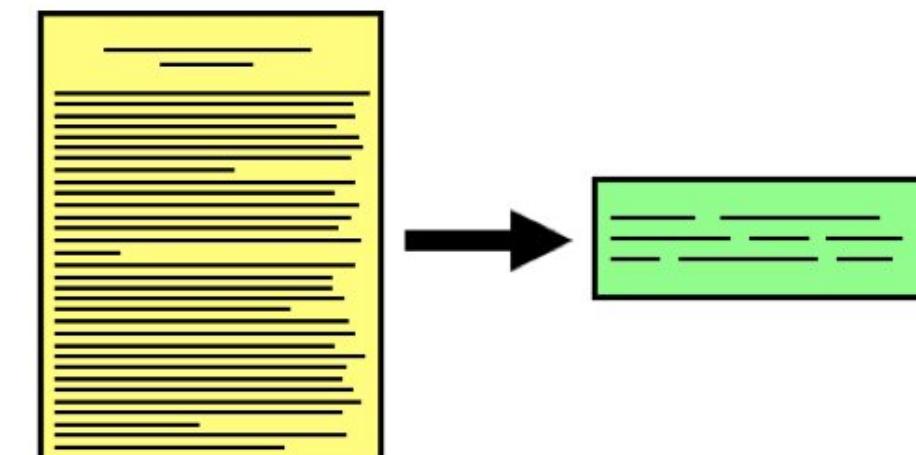
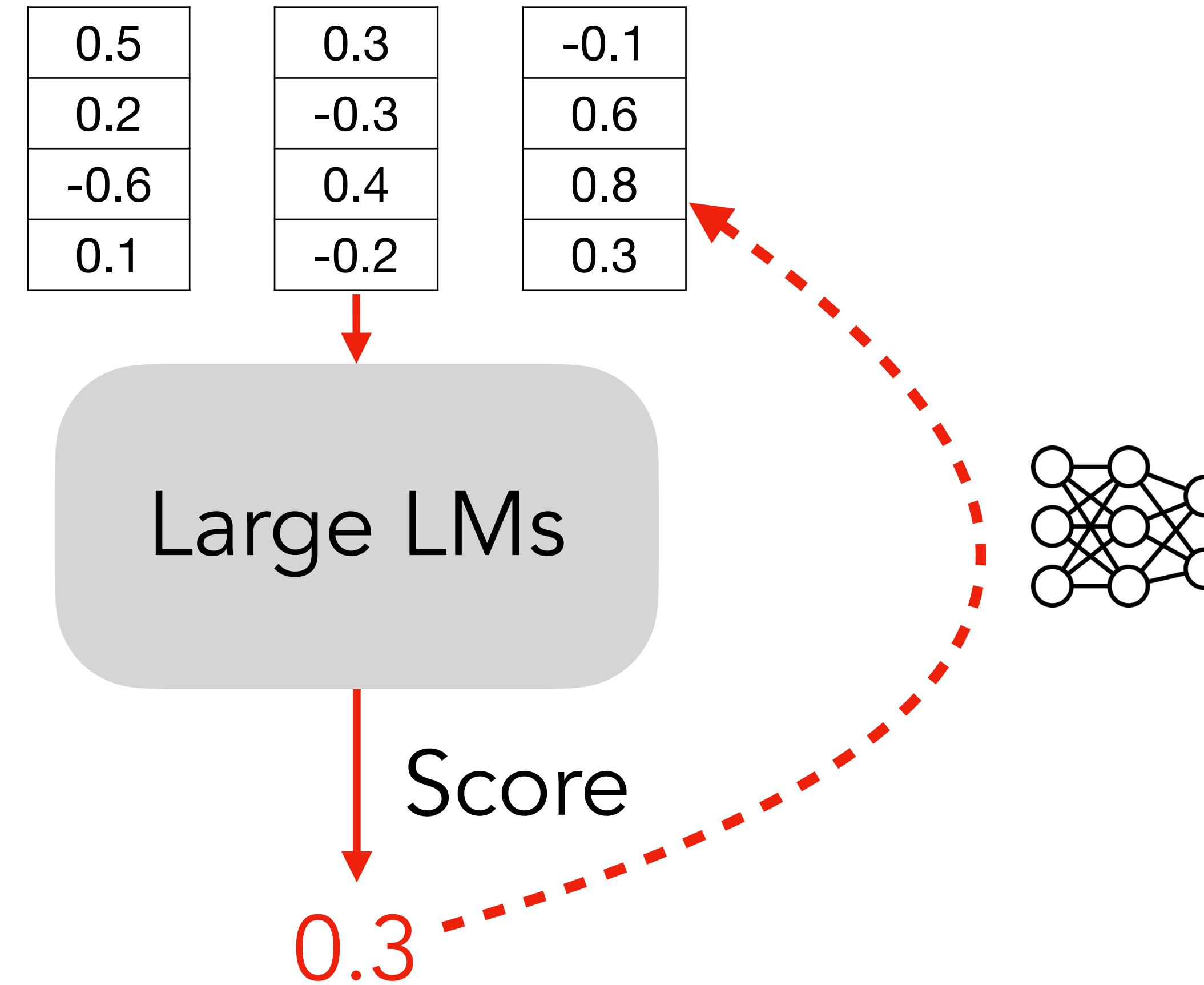
“Summarize this article”



Summarization

# Finding the Best Prompt Can Be Challenging

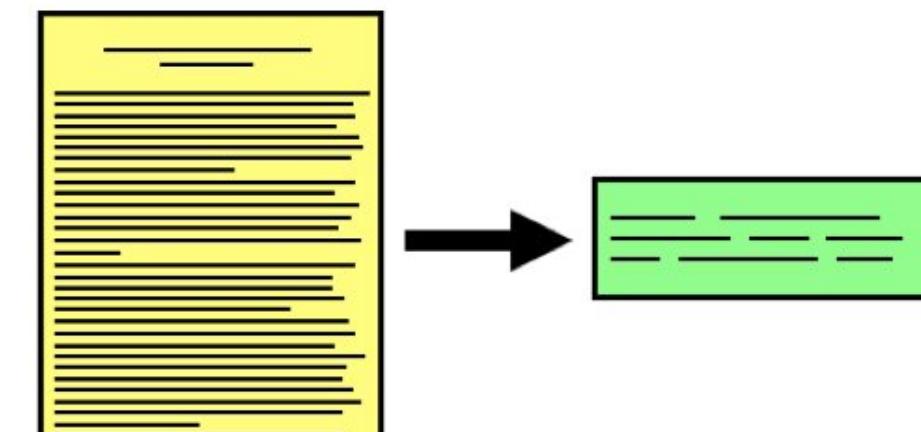
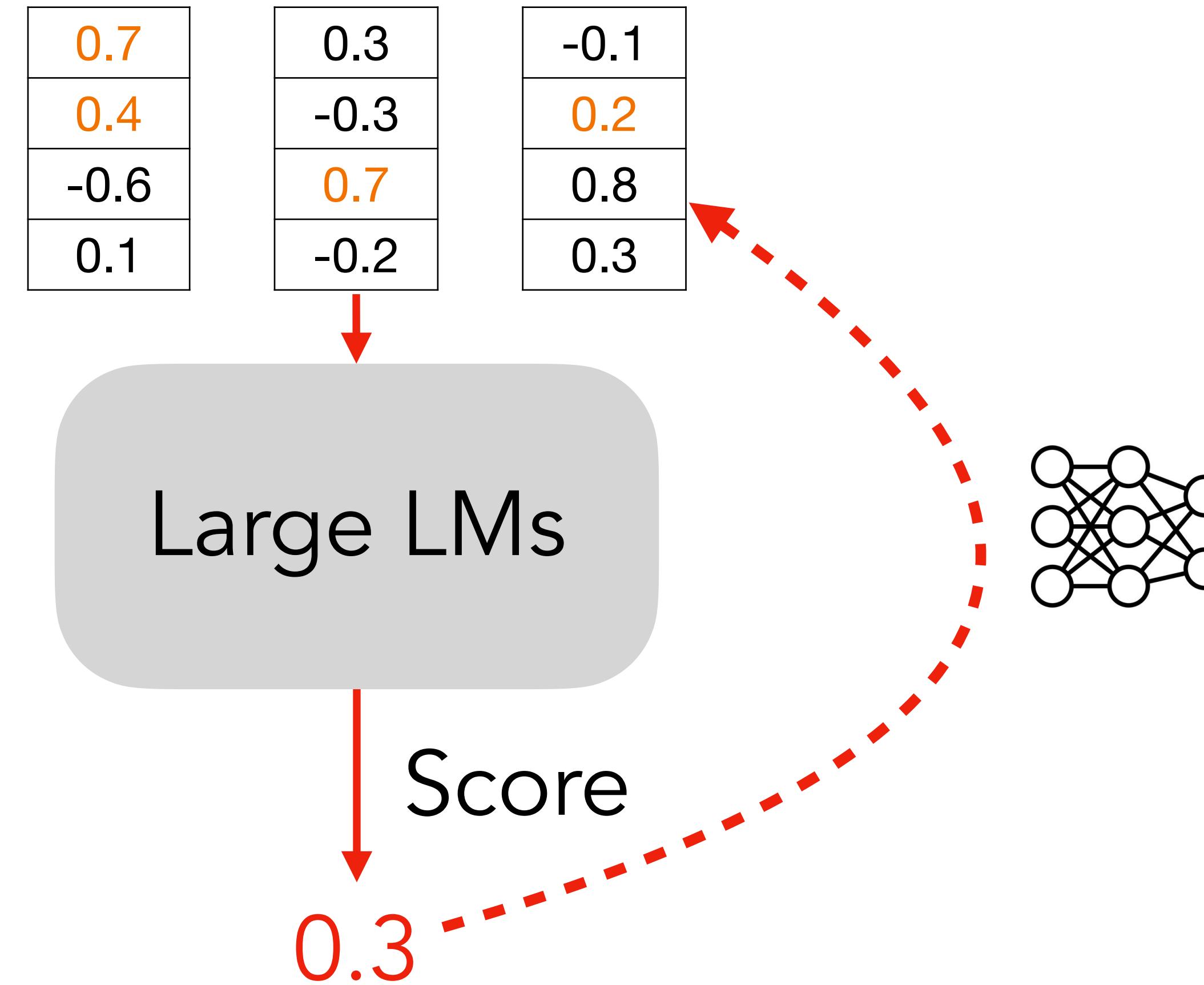
- Manual Design
- Enumerate and Select
- Numerical Tuning



Summarization

# Finding the Best Prompt Can Be Challenging

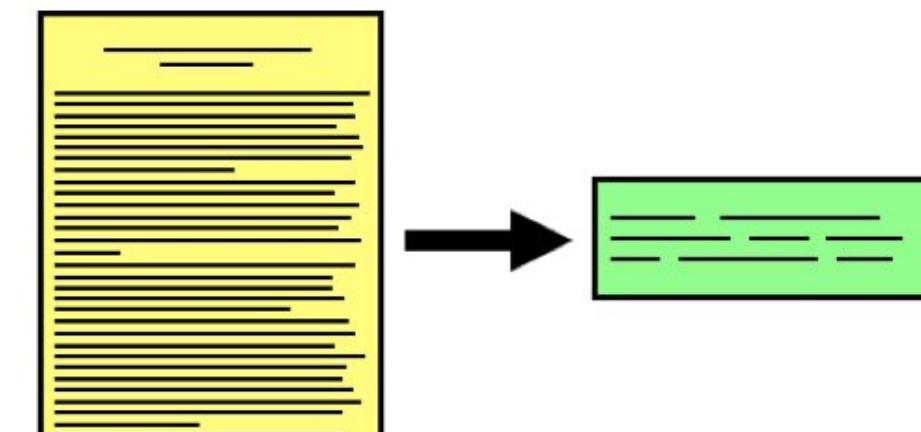
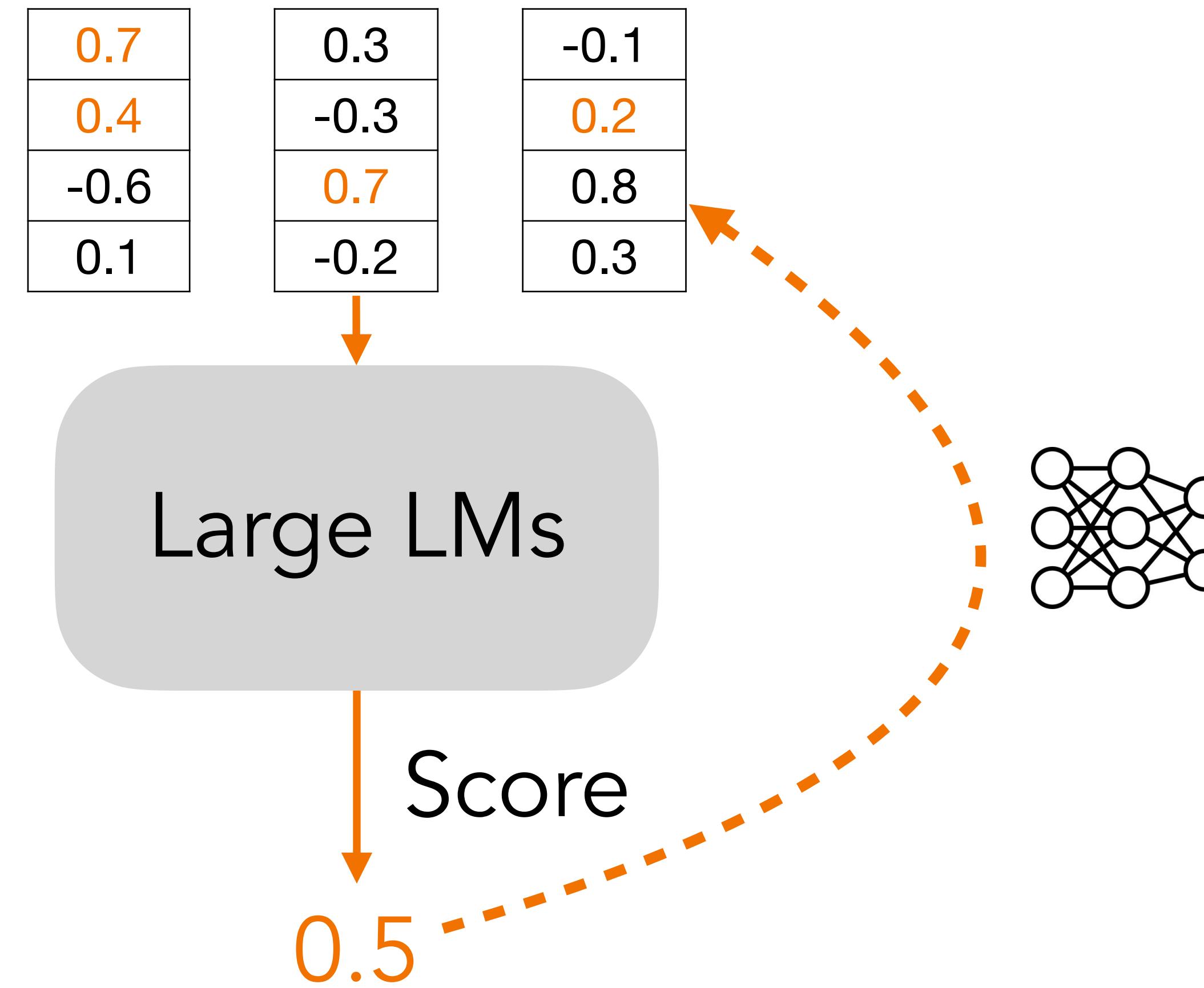
- Manual Design
- Enumerate and Select
- Numerical Tuning



Summarization

# Finding the Best Prompt Can Be Challenging

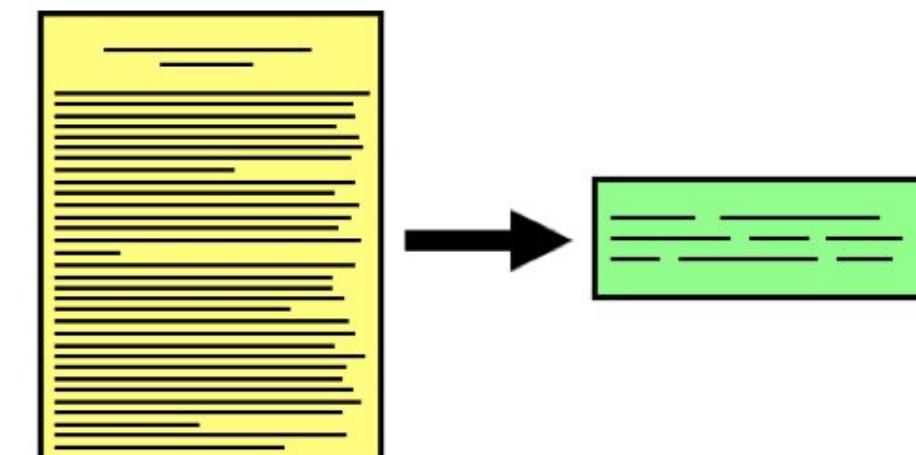
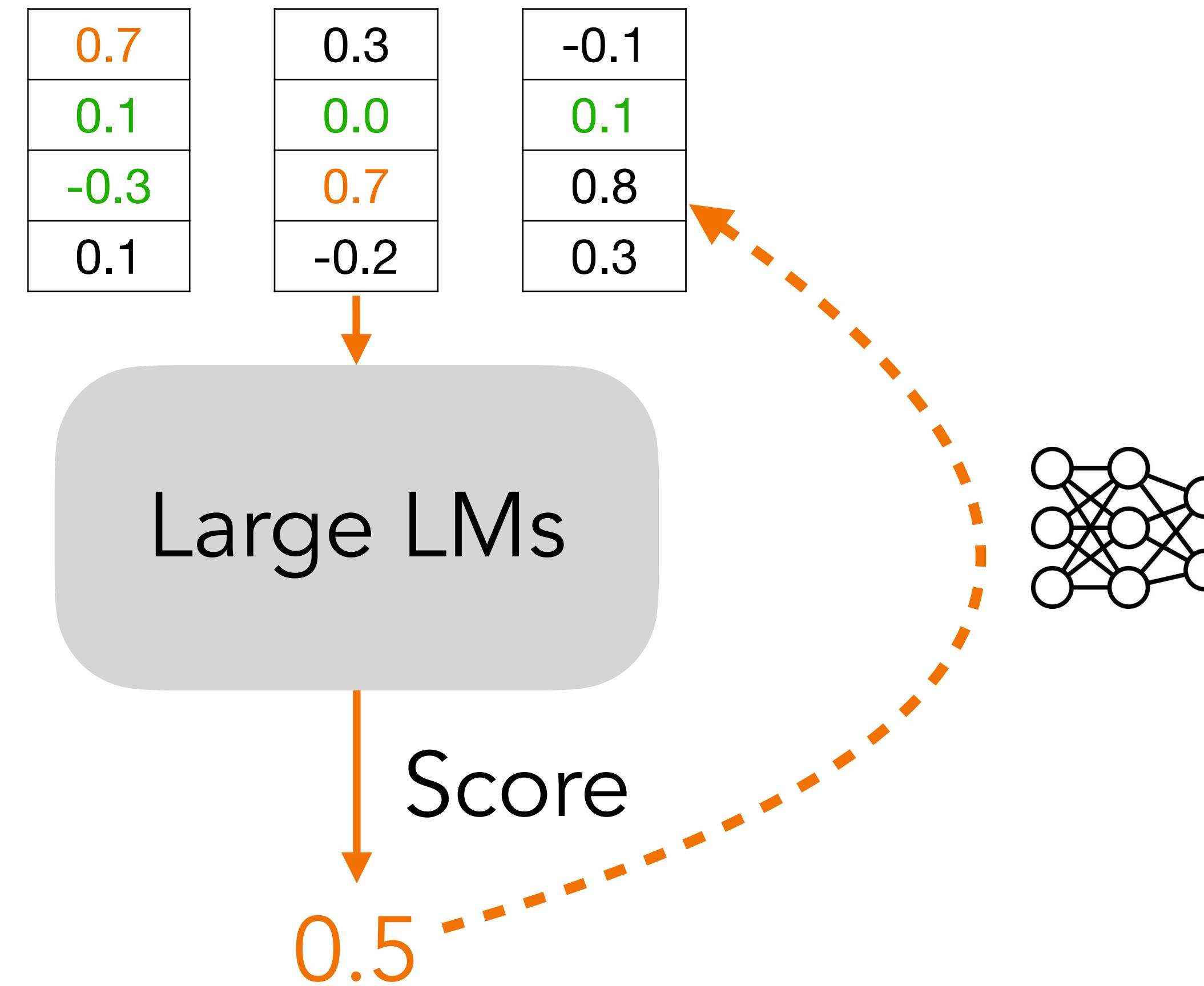
- Manual Design
- Enumerate and Select
- Numerical Tuning



Summarization

# Finding the Best Prompt Can Be Challenging

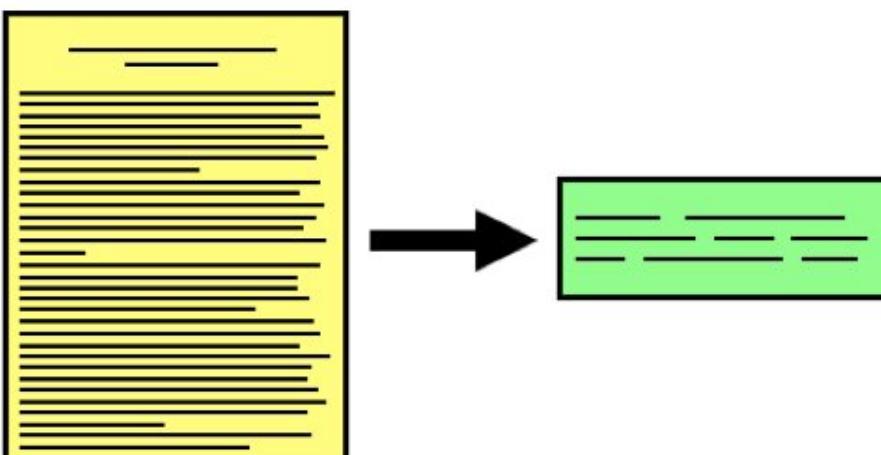
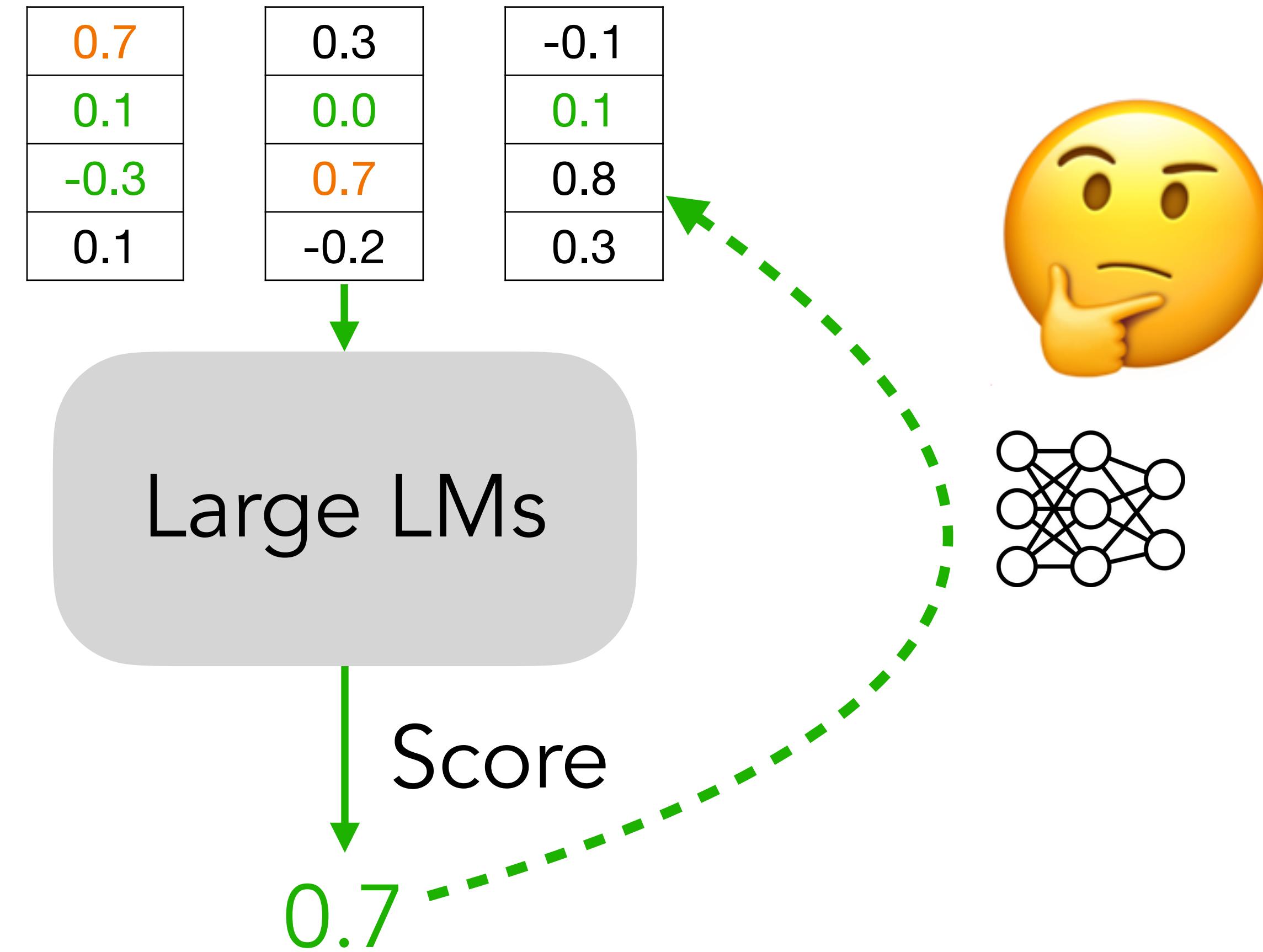
- Manual Design
- Enumerate and Select
- Numerical Tuning



Summarization

# Finding the Best Prompt Can Be Challenging

- Manual Design
- Enumerate and Select
- Numerical Tuning



Summarization

# Our Solution



## RLPROMPT: Optimizing Discrete Text Prompts with Reinforcement Learning

Mingkai Deng<sup>1\*</sup>, Jianyu Wang<sup>2\*</sup>, Cheng-Ping Hsieh<sup>2\*</sup>, Yihan Wang<sup>2</sup>, Han Guo<sup>1</sup>,  
Tianmin Shu<sup>3</sup>, Meng Song<sup>2</sup>, Eric P. Xing<sup>1,4,5</sup>, Zhiting Hu<sup>2</sup>

<sup>1</sup>Carnegie Mellon University, <sup>2</sup>UC San Diego,

<sup>3</sup>MIT, <sup>4</sup>Mohamed bin Zayed University of Artificial Intelligence, <sup>5</sup>Petuum Inc.

{mingkaid,hanguo}@cs.cmu.edu, {jiw102,c2hsieh,yiw007,zhh019}@ucsd.edu

Automatic ✓

Flexible ✓

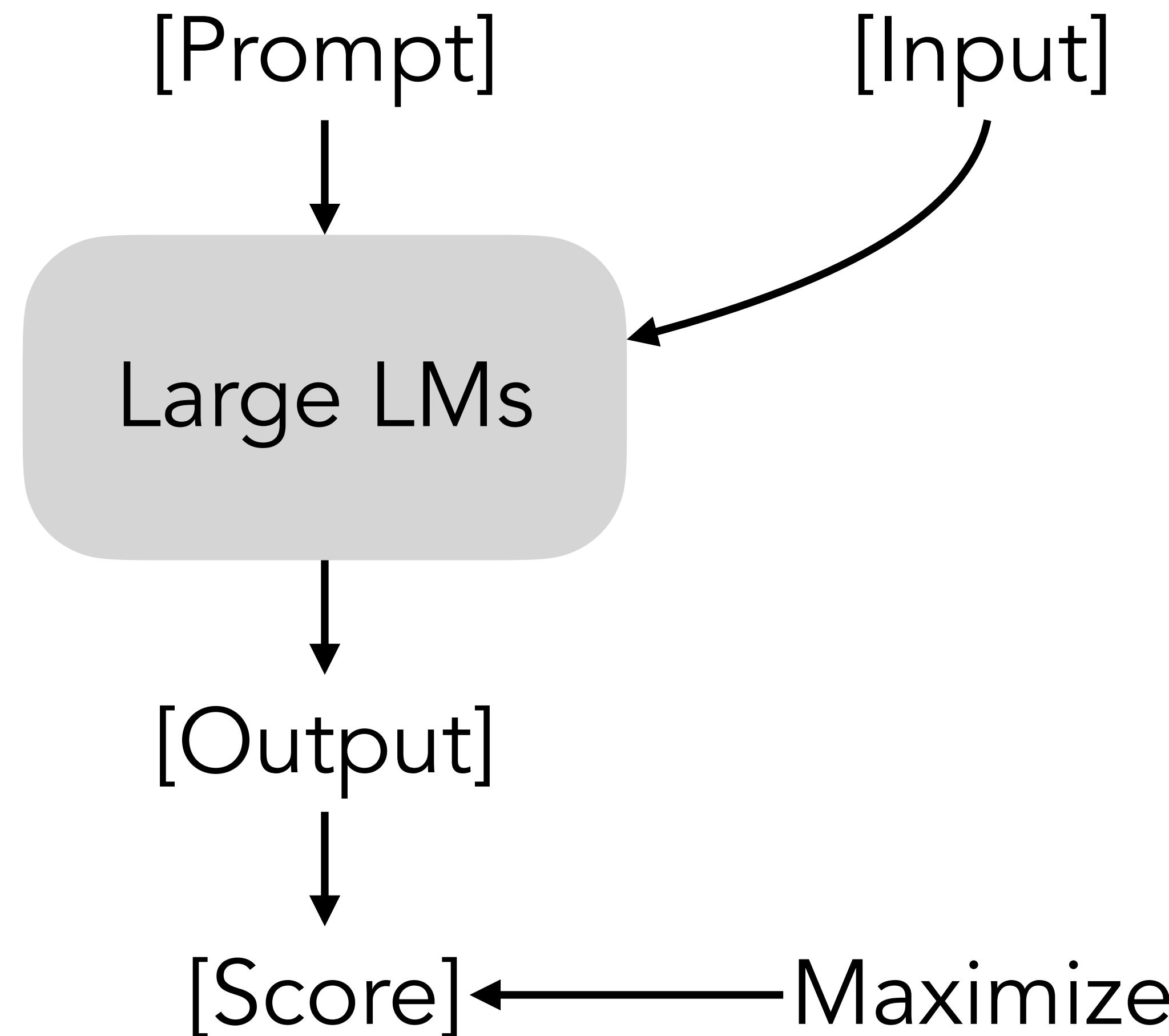
Systematic ✓

Transferrable ✓

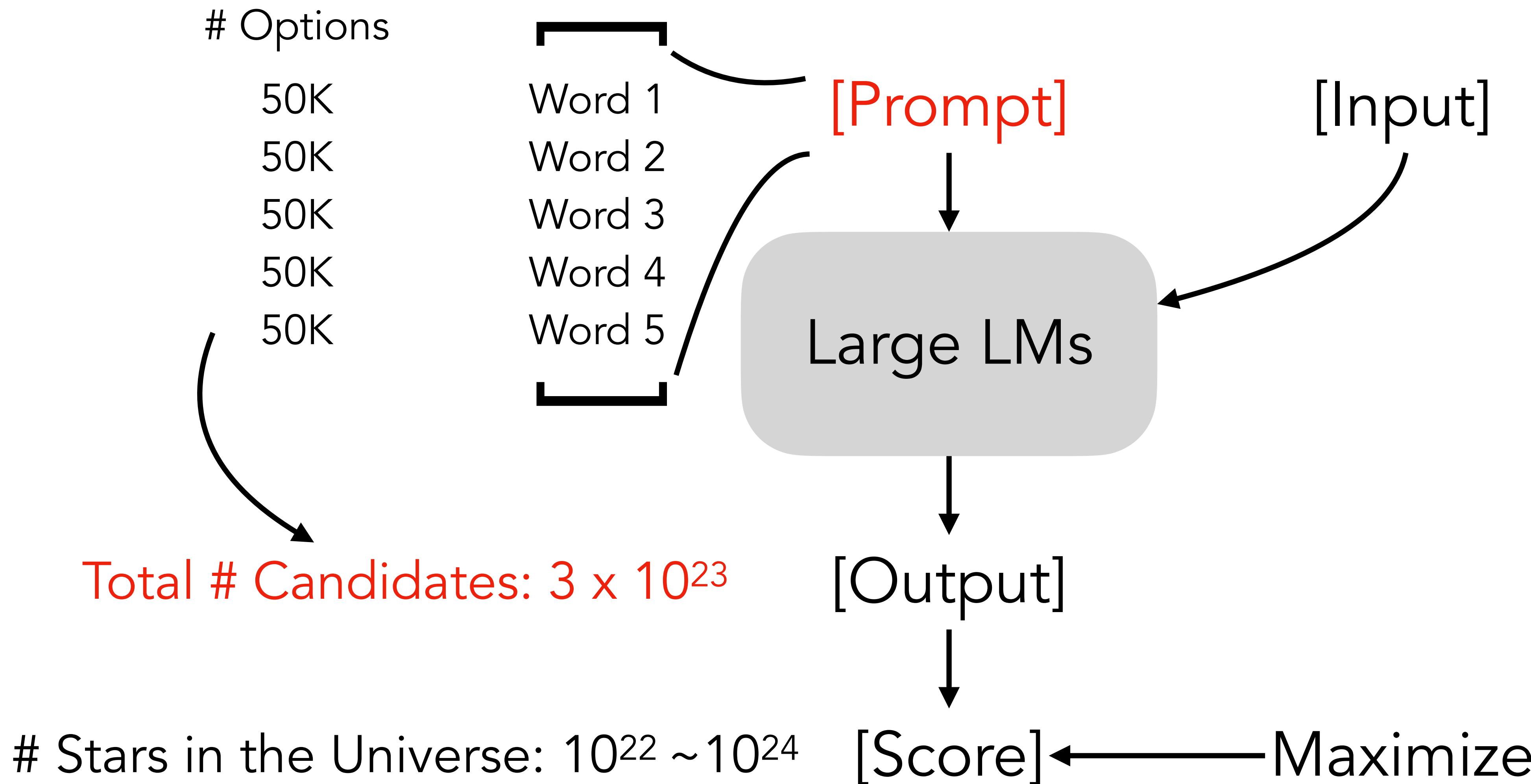
Accessible ✓

Transparent ✓

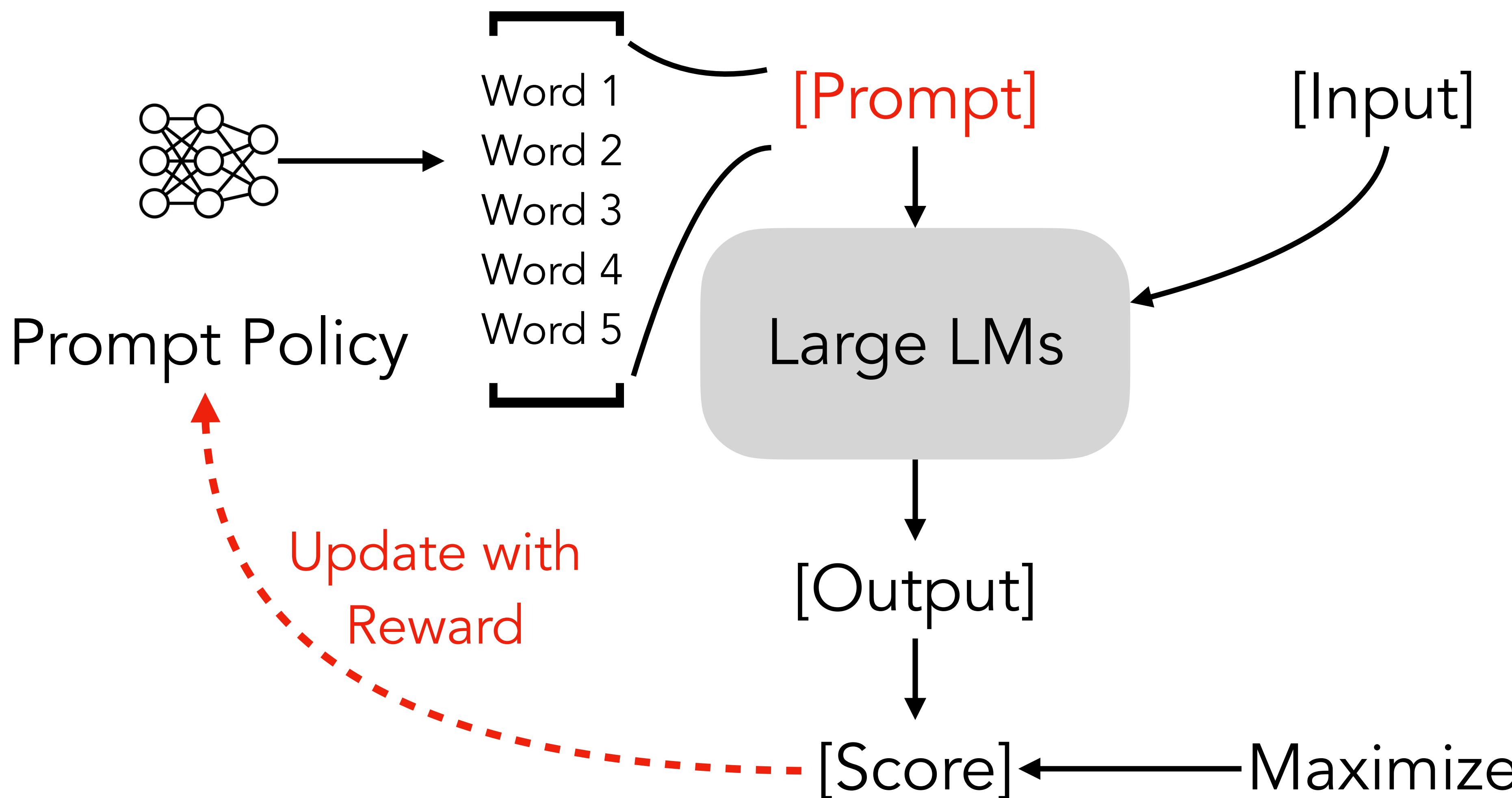
# The Prompt Optimization Problem



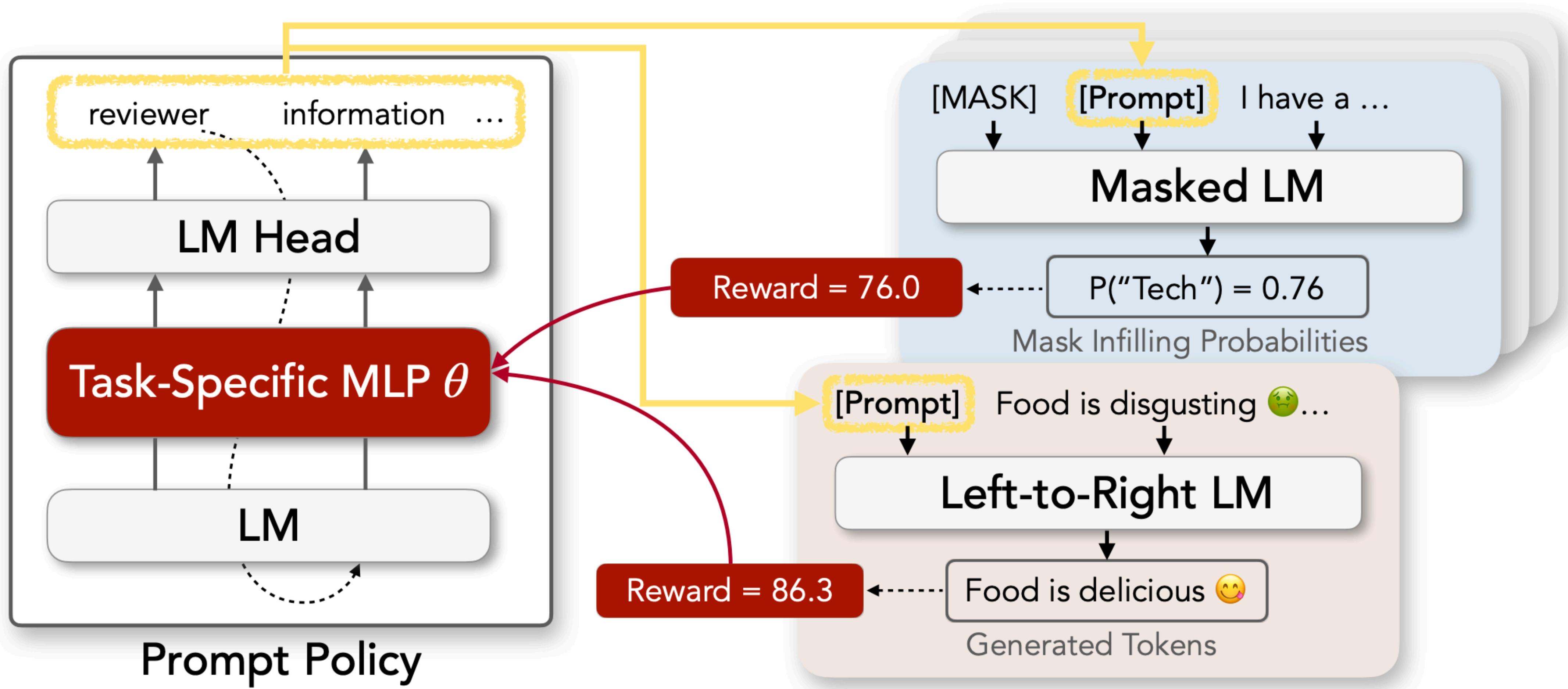
# The Prompt Optimization Problem



# The Reinforcement Learning Formulation

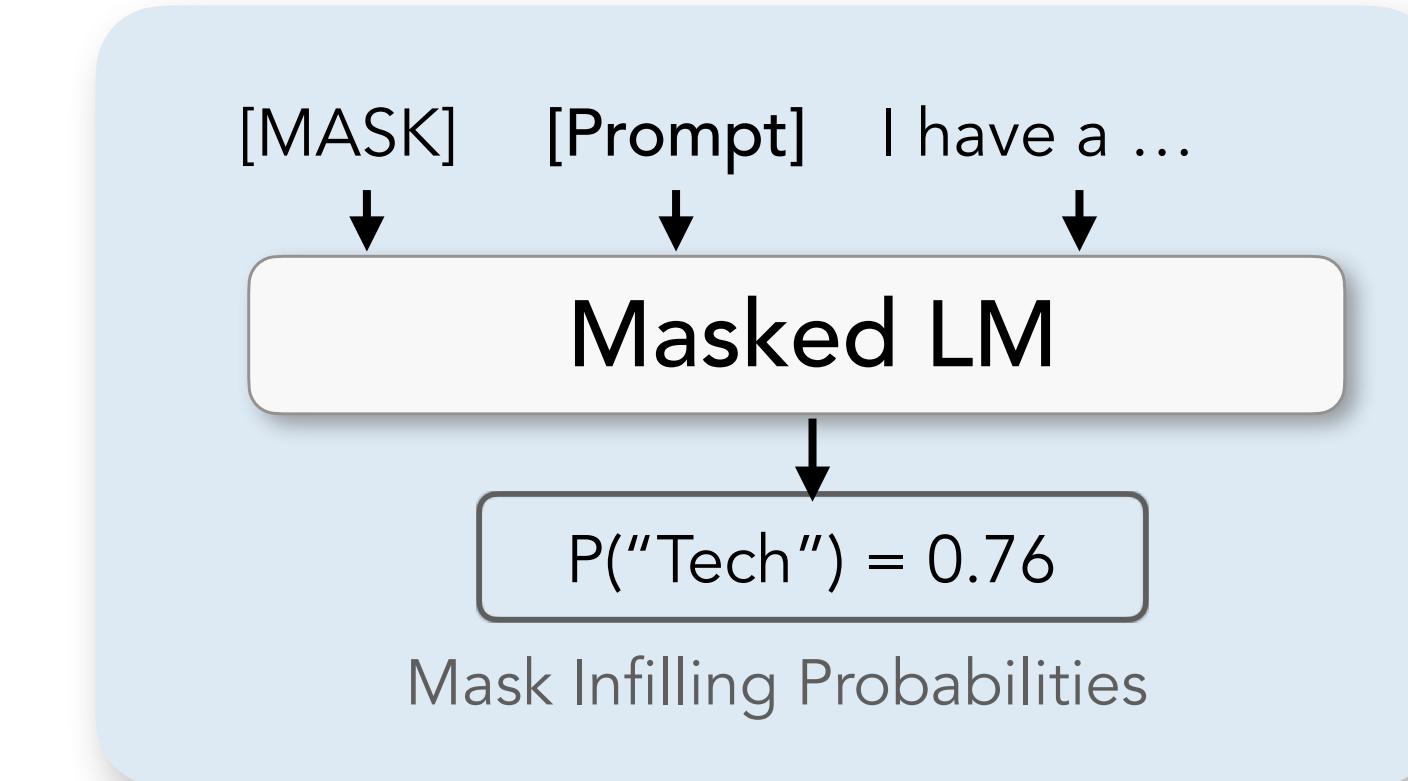


# Concrete Overview of the Process

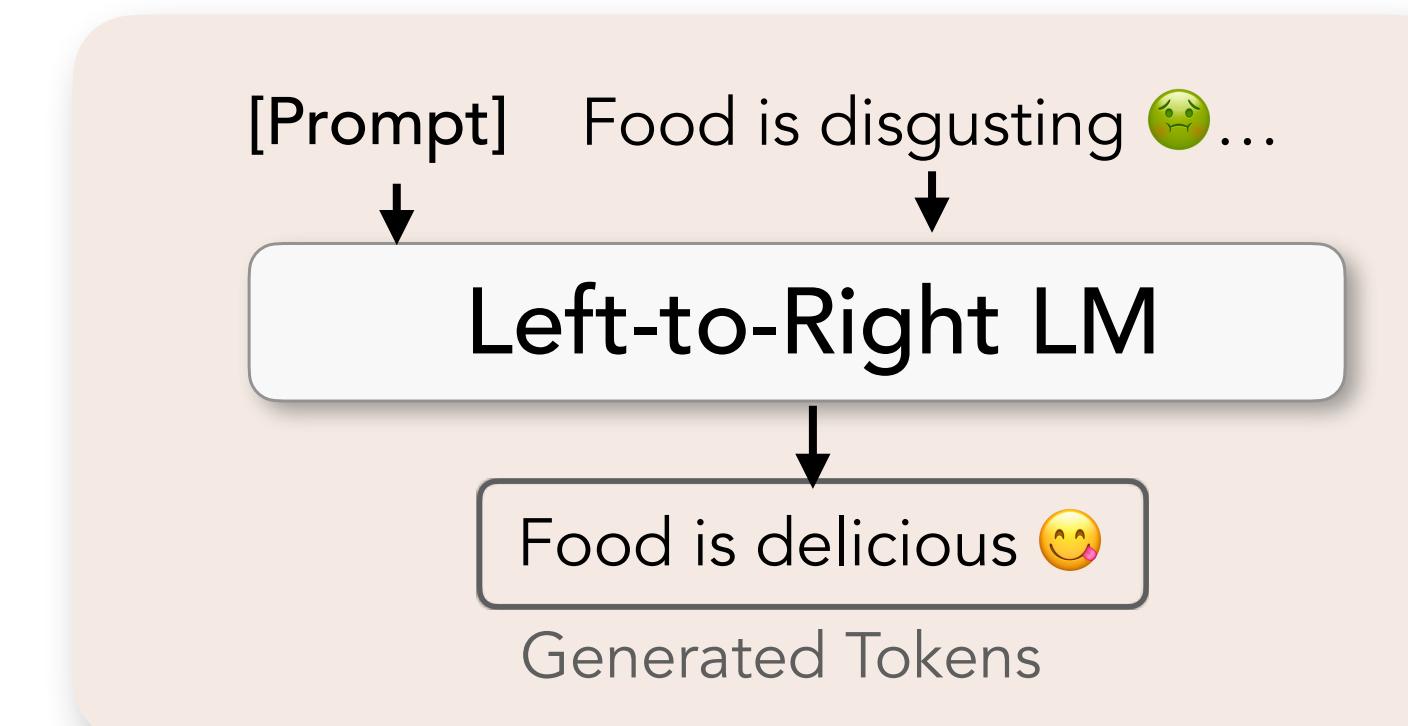


# Experiments / Example Use Cases

- Few-Shot Text Classification  
(e.g., sentiment and topic analysis)



- Unsupervised, Controlled Text Generation  
(e.g., text style transfer - rewriting Shakespeare into modern English)

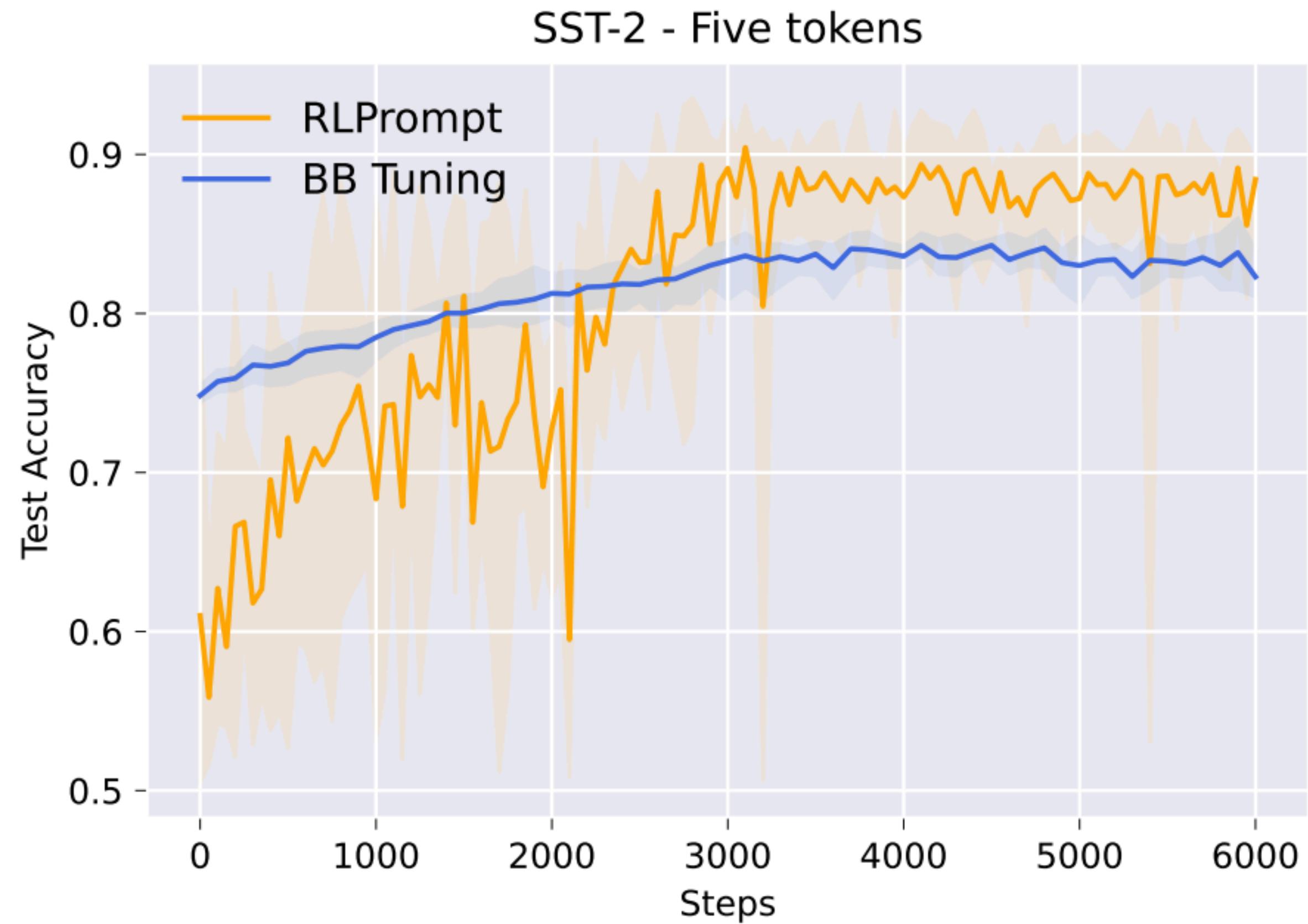


# Experiments - Few-Shot Classification

Setting: 16 training examples per category

Reward Function: Gap between probabilities of correct and incorrect categories

	Avg. Acc.
Fine-Tuning	69.5
Manual Prompt	68.6
Instructions	58.5
In-Context Demo.	72.2
Prompt Tuning ( <i>Soft Prompt Tuning</i> )	69.2
Black-Box Tuning (2 soft tokens)	69.2
Black-Box Tuning (5 soft tokens)	67.1
Black-Box Tuning ( <i>Mixed, 50 soft tokens</i> )	74.7
GrIPS ( <i>Discrete Prompt Enumeration</i> )	69.4
AutoPrompt	56.7
RLPrompt (Ours, 2 discrete tokens)	74.4
RLPrompt (Ours, 5 discrete tokens)	<b>75.8</b>

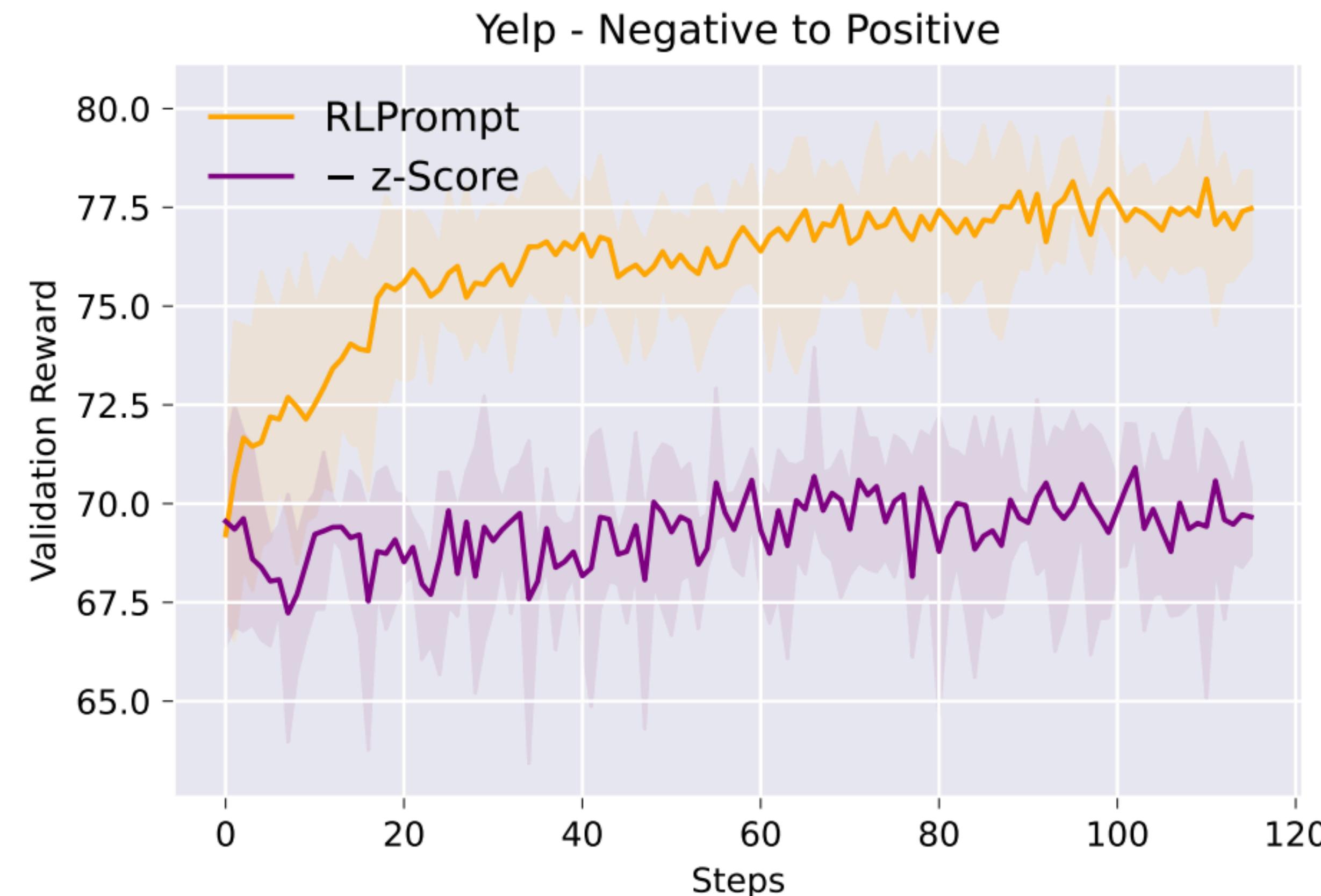


# Experiments - Unsupervised Text Style Transfer

Setting: No training example, just two corpora representing each style

Reward Function: Metrics for style correspondence and content preservation

Model	$J(C, S, F)$
<i>Training Baselines</i>	
Style Transformer	46.1
DiRR	59.6
<i>Prompting Baselines (GPT-2-xl)</i>	
Null Prompt	33.6
Random Prompt	34.7
Manual Prompt	53.4 (7.9)
<b>RLPROMPT (Ours)</b>	
distilGPT-2	46.0 (0.9)
GPT-2-small	50.7 (1.3)
GPT-2-medium	56.1 (1.0)
GPT-2-large	56.5 (1.3)
GPT-2-xl	<b>61.4 (2.2)</b>



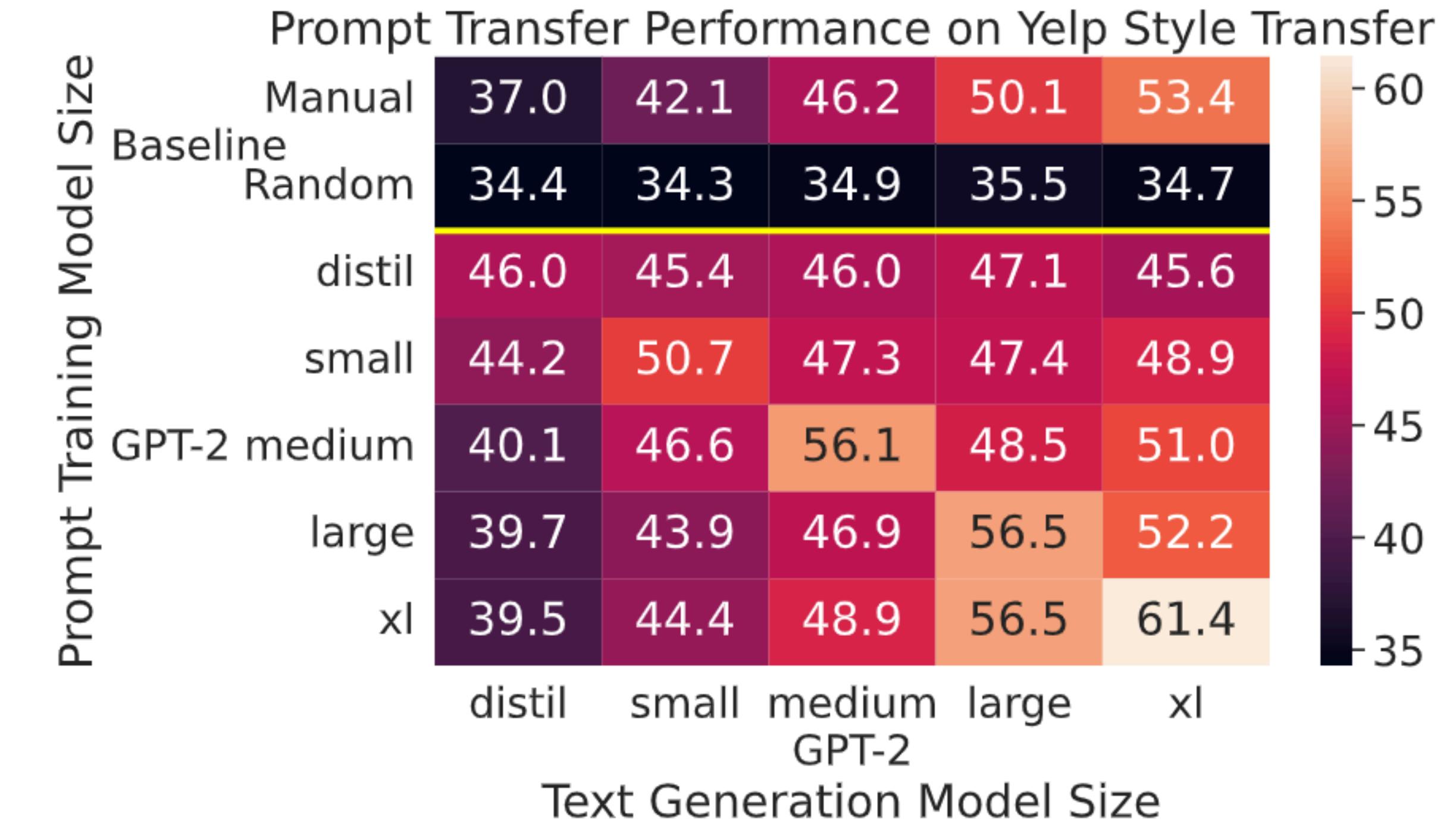
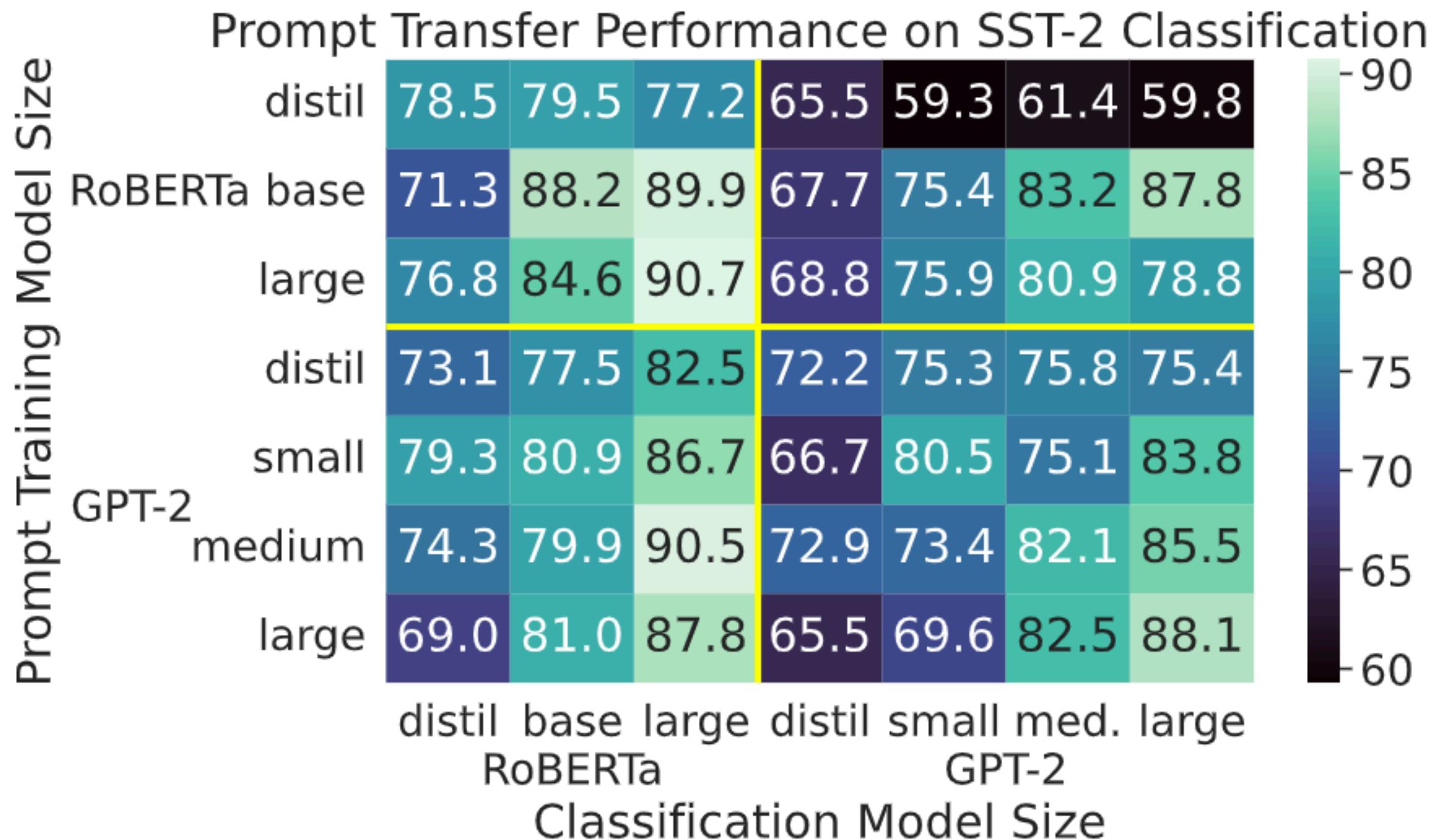
# Analysis Insights

- Optimal prompts don't follow human language
- Optimized prompts transfer trivially across models

# Analysis - Optimal prompts don't follow human language

Prompt Template [to negative   to positive]	$J(c, s, f)$
<b>Manual Prompt</b>	
Here is some text: "{input}". Here is a rewrite of the text, which is more [negative   positive]: "	62.3
Change the following sentence from [positive   negative] sentiment to [negative   positive] sentiment but keep its semantics. "{input}" "	50.5
"{input}". Rewrite the sentence to be [sadder   happier] but have the same meaning. "	47.4
<b>RLPROMPT (Ours)</b>	
[Fixed (- contrasts   Dutch English excellent Correct (>) " {input} " )	62.8
[Fixed RemovedChanged Prevent outcomes   Parameters Comparison )=( Compare either] " {input} " )	58.9
[Affect differed judgments (- analysis   Difference experiences (- contrasting experience] " {input} " )	62.6

# Analysis - Learned prompts transfer trivially across LMs



# Codebase and Library on GitHub

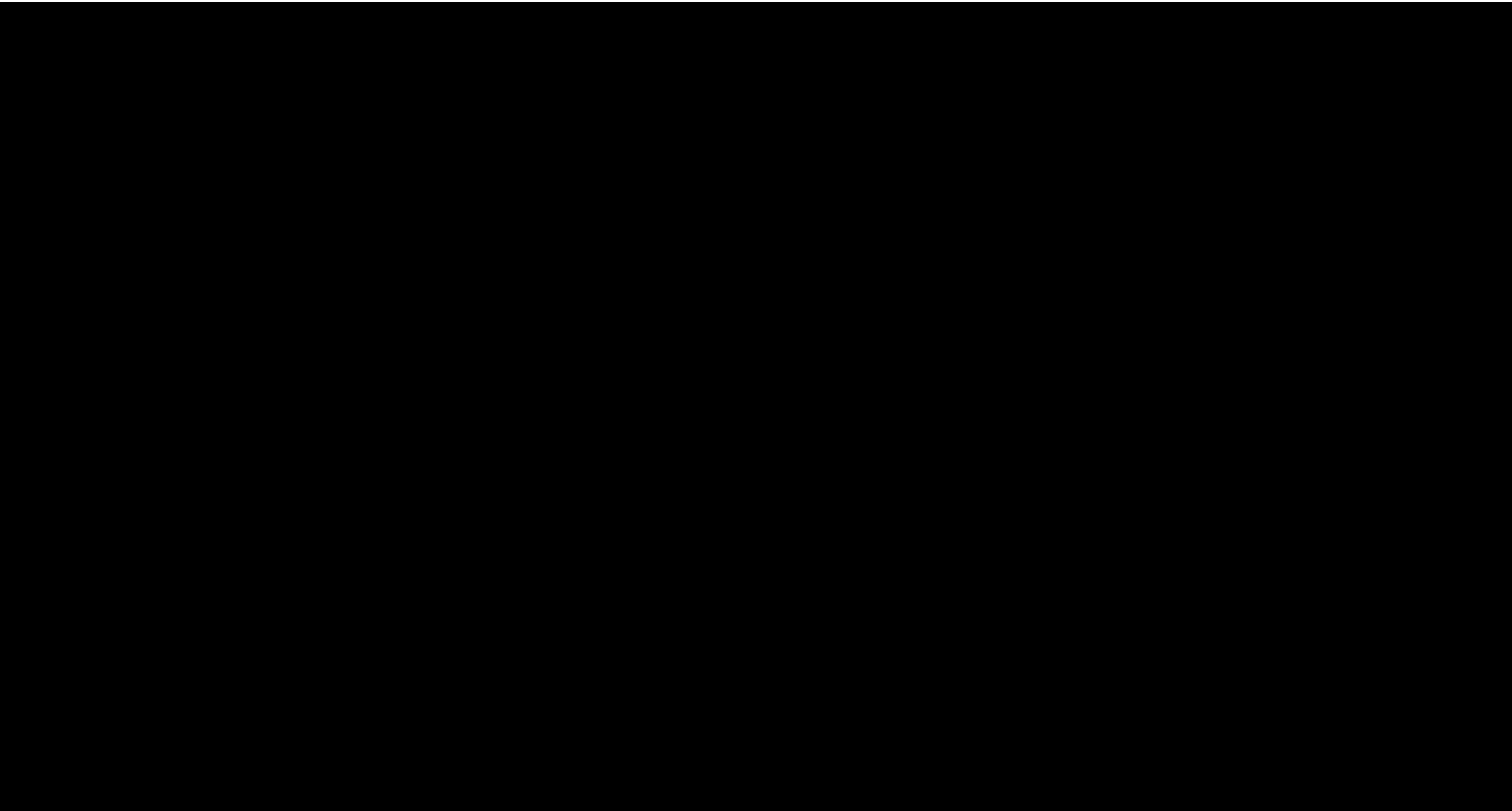
The screenshot shows a GitHub repository page for 'mingkaid / rl-prompt'. The repository is public and has 25 forks and 139 stars. It contains 3 branches and 0 tags. The main branch is selected. The repository is described as an accompanying repo for the RL Prompt. The codebase includes files like README.md, examples, rlprompt, .gitattributes, .gitignore, and LICENSE. The README.md file contains Python code for setting up and running the RL Prompt library.

```
27 @hydra.main(version_base=None, config_path="./", config_name="tst_config")
28 def main(config: "DictConfig"):
29     colorful_print(OmegaConf.to_yaml(config), fg='red')
30     output_dir = get_hydra_output_dir()
31
32     train_dataset, val_dataset, test_dataset = \
33         make_text_style_transfer_datasets(config)
34     print('Train Size:', len(train_dataset))
35     print('Examples:', train_dataset[:5])
36     print('Val Size', len(val_dataset))
37     print('Examples:', val_dataset[:5])
38
39     policy_model = make_lm_adaptor_model(config)
40     prompt_model = make_single_prompt_model(policy_model, config)
41     config.style_classifier = get_style_classifier('train', config)
42     reward = make_prompted_text_style_transfer_reward(config)
43     algo_module = make_sql_module(prompt_model, reward, config)
44
45     config.save_dir = os.path.join(output_dir, config.save_dir)
46     trainer = make_trainer(algo_module, train_dataset, val_dataset, config)
47     trainer.train(config=config)
48
49
50 if __name__ == "__main__":
51     main()
```

**bout**  
ccompanying repo for the RL Prompt  
aper

- ] Readme
- MIT license
- 139 stars
- 8 watching
- 25 forks

# Demo



# Takeaways

- Optimizing prompts for LLMs is challenging should be done carefully
- The best way to talk to LLMs is not necessarily human language
- RL-based optimization allows us to find out the best prompts effectively

# Thank You!



Blog Post



Paper



Code & API



Email: [mingkaid@cs.cmu.edu](mailto:mingkaid@cs.cmu.edu)

Twitter: [twitter.com/mdeng34](https://twitter.com/mdeng34)

Website: [mingkaid.github.io](https://mingkaid.github.io)