CS639 Deep Learning for NLP

Prompting

Junjie Hu

Slides adapted from Pengfei, Graham

https://junjiehu.github.io/cs639-spring26/

Outline

- Prompting vs other machine learning paradigms in NLP
- General Workflow of Prompting
- Key Components of Prompting
  1. Pre-trained Model Choice
  2. Prompt Engineering
  3. Answer Engineering
  4. Expanding the Paradigm
  5. Prompt-based Training Strategies

Four Paradigms of NLP Techniques

Four Paradigms of NLP Technical Development

- Feature Engineering
- Architecture Engineering
- Objective Engineering
- Prompt Engineering

Feature Engineering

- **Paradigm**: Fully Supervised Learning (Non-neural Network)
- **Time Period:** Most popular through 2015
- **Characteristics**:
  - Non-neural machine learning models mainly used
  - Require manually defined feature extraction
- **Representative Work**:
  - Manual features -> linear or kernelized support vector machine (SVM)
  - Manual features -> conditional random fields (CRF)

Architecture Engineering

- **Paradigm**: Fully Supervised Learning (Neural Networks)
- **Time Period**: About 2013-2018
- **Characteristics**:
  - Rely on neural networks
  - Do not need to manually define features, but should modify the network structure (e.g.: LSTM v.s CNN)
  - Sometimes used pre-training of LMs, but often only for shallow features such as embeddings
- **Representative Work**:
  - CNN/LSTM for Text Classification
  - Transformer for Machine Translation

Objective Engineering

- **Paradigm**: Pre-train, Fine-tune
- **Time Period**: 2017-Now
- **Characteristics**:
  - Pre-trained LMs (PLMs) used as initialization of full model - both shallow and deep features
  - Less work on architecture design, but engineer objective functions
- **Typical Work**:
  - BERT → Fine Tuning

Prompt Engineering

- **Paradigm**: Pre-train, Prompt, Predict
- **Date**: 2019-Now
- **Characteristic**:
  - NLP tasks are modeled entirely by relying on LMs
  - The tasks of shallow and deep feature extraction, and prediction of the data are all given to the LM
  - Engineering of prompts is required
- **Representative Work:**
  - GPT3, GPT4, ChatGPT

General Workflow of Prompting

Recommended Reading

Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

Pengfei Liu
Carnegie Mellon University
pliu3@cs.cmu.edu

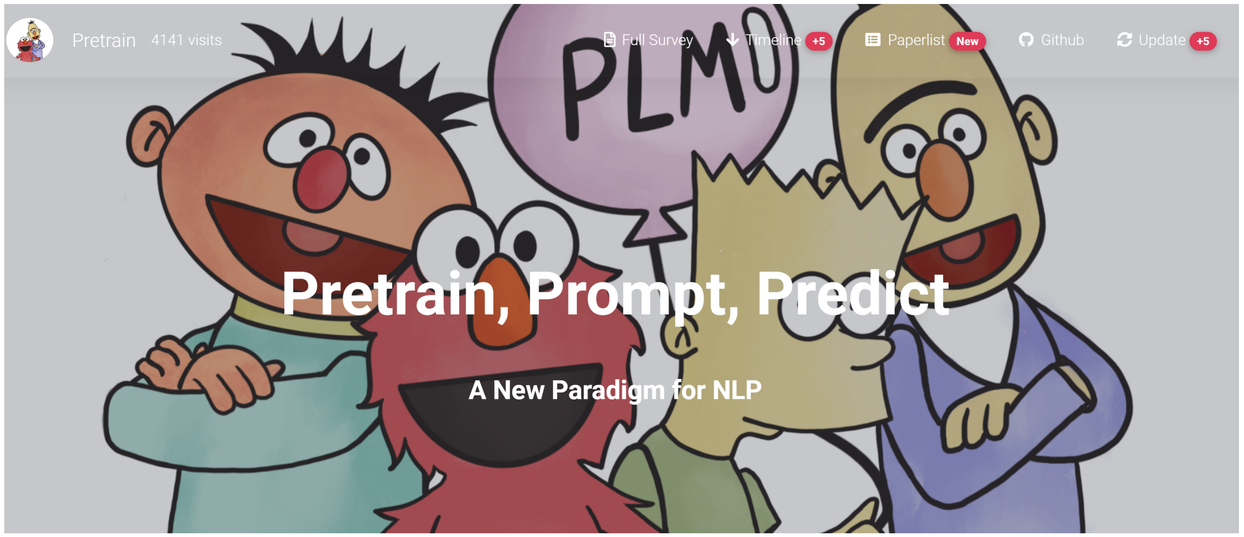
Weizhe Yuan
Carnegie Mellon University
weizhey@cs.cmu.edu

Jinlan Fu
National University of Singapore
jinlanjonna@gmail.com

Zhengbao Jiang
Carnegie Mellon University
zhengbaj@cs.cmu.edu

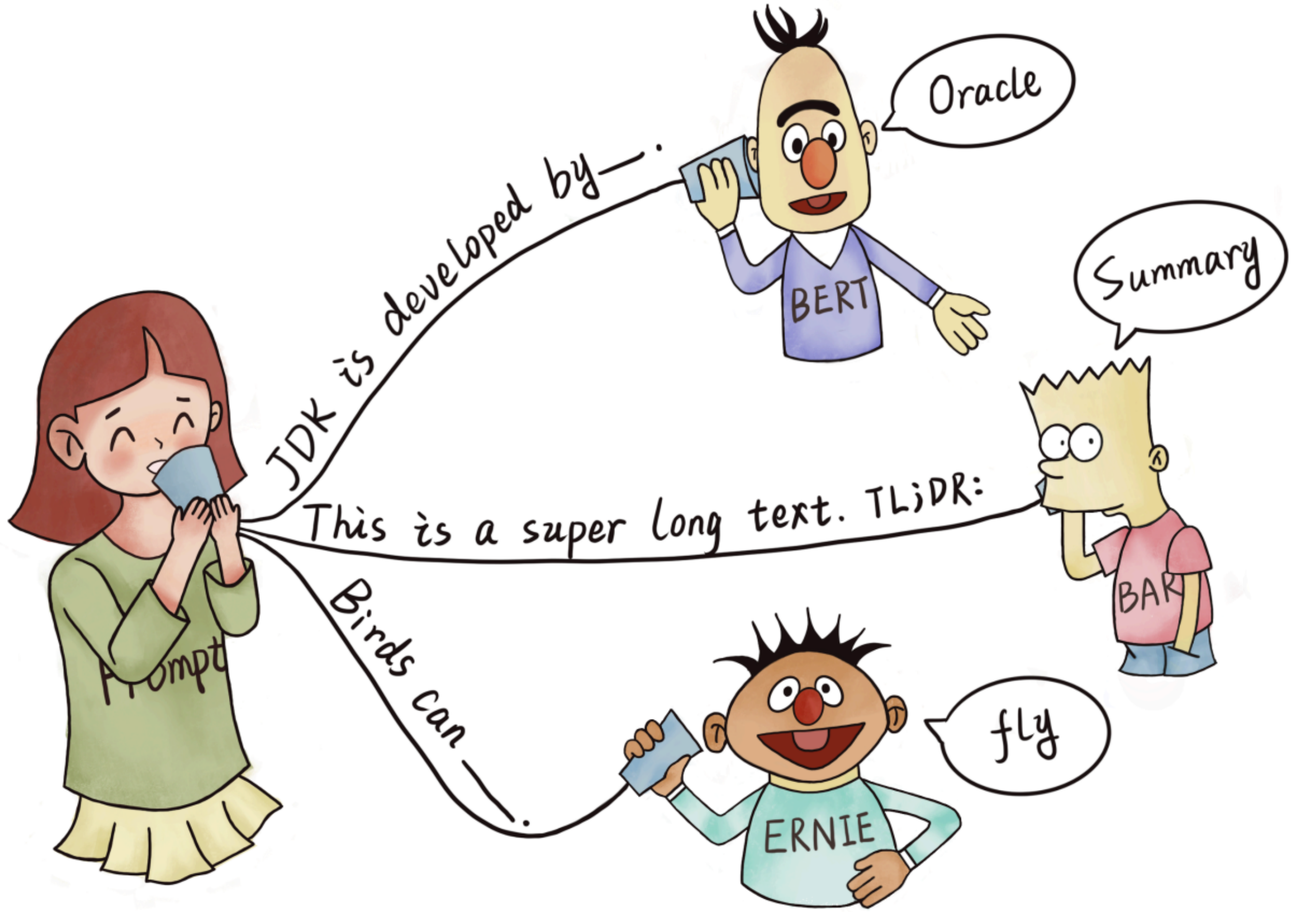
Hiroaki Hayashi
Carnegie Mellon University
hiroakih@cs.cmu.edu

Graham Neubig
Carnegie Mellon University
gneubig@cs.cmu.edu



What is Prompting?

- Encouraging a pre-trained model to make particular predictions by providing a "prompt" specifying the task to be done.



What is the general workflow of Prompting?

- Prompt Addition
- Answer Prediction
- Answer-Label Mapping

Prompt Addition

- **Prompt Addition**: Given input x, we transform it into prompt x’ through two steps:
  - Define **a template** with two slots, one for input [x], and one for the answer [z]
  - Fill in the input slot [x]

Example: Sentiment Classification

Input: x = “I love this movie”

Template: [x] Overall, it was a [z] movie

Prompting: x’ = “I love this movie. Overall it was a [z] movie.”

Answer Prediction

- Answer Prediction: Given a prompt, predict the answer [z]
  - Fill in [z]

Example

Input: x = “I love this movie”

Template: [x] Overall, it was a [z] movie

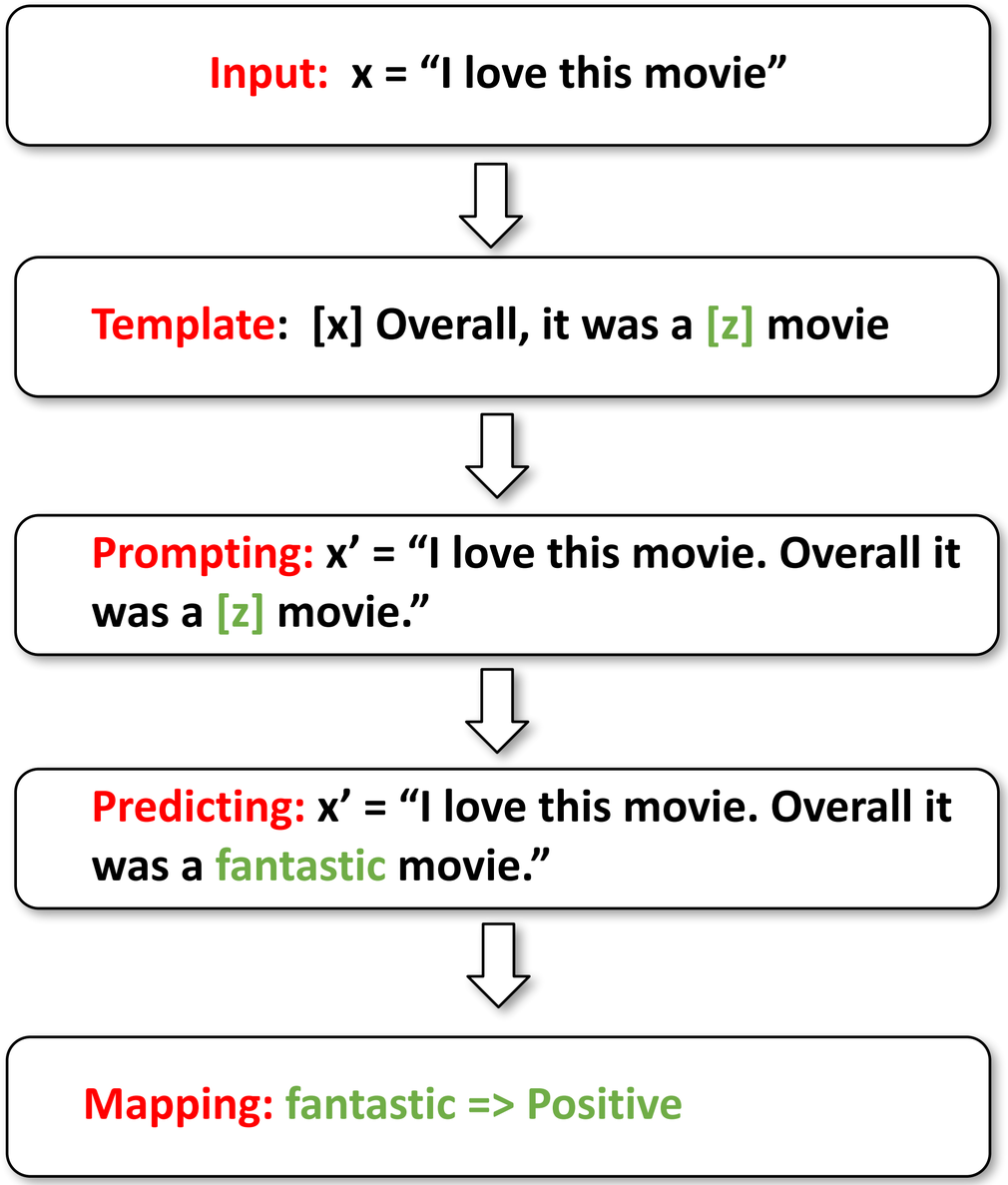
Prompting: x’ = “I love this movie. Overall it was a [z] movie.”

Predicting: x’ = “I love this movie. Overall it was a fantastic movie.”

Mapping

- Mapping: Given an answer, map it into a class label

Example


Input: x = “I love this movie”
Template: [x] Overall, it was a [z] movie
Prompting: x’ = “I love this movie. Overall it was a [z] movie.”
Predicting: x’ = “I love this movie. Overall it was a fantastic movie.”
Mapping: fantastic => Positive

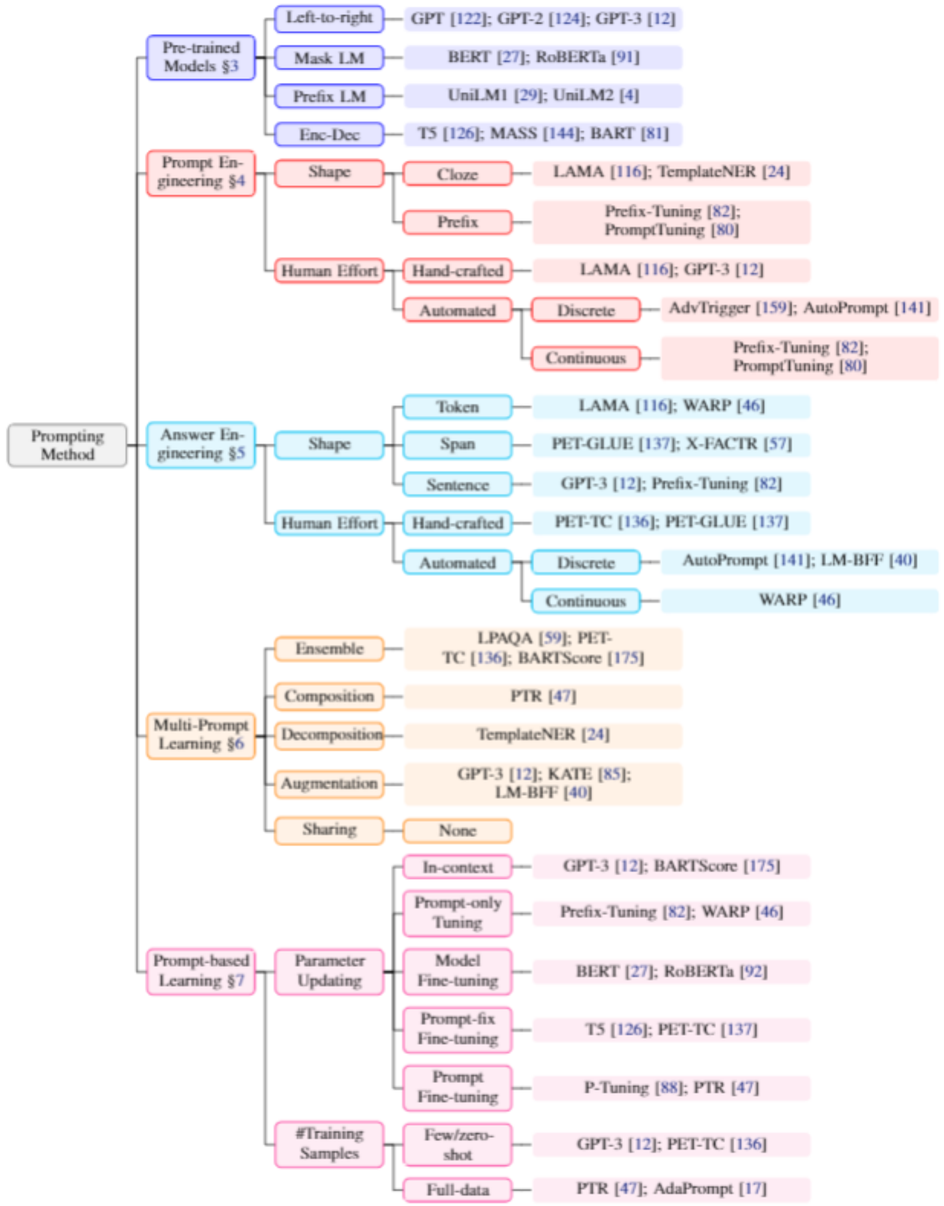
Types of Prompts

- Cloze Prompt: **I love this movie. Overall it was a [z] movie**
  Example outputs:
  - **I love this movie. Overall it was a boring movie**
  - **I love this movie. Overall it was a fantastic movie**

- Prefix Prompt: **I love this movie. Overall this movie is [z]**

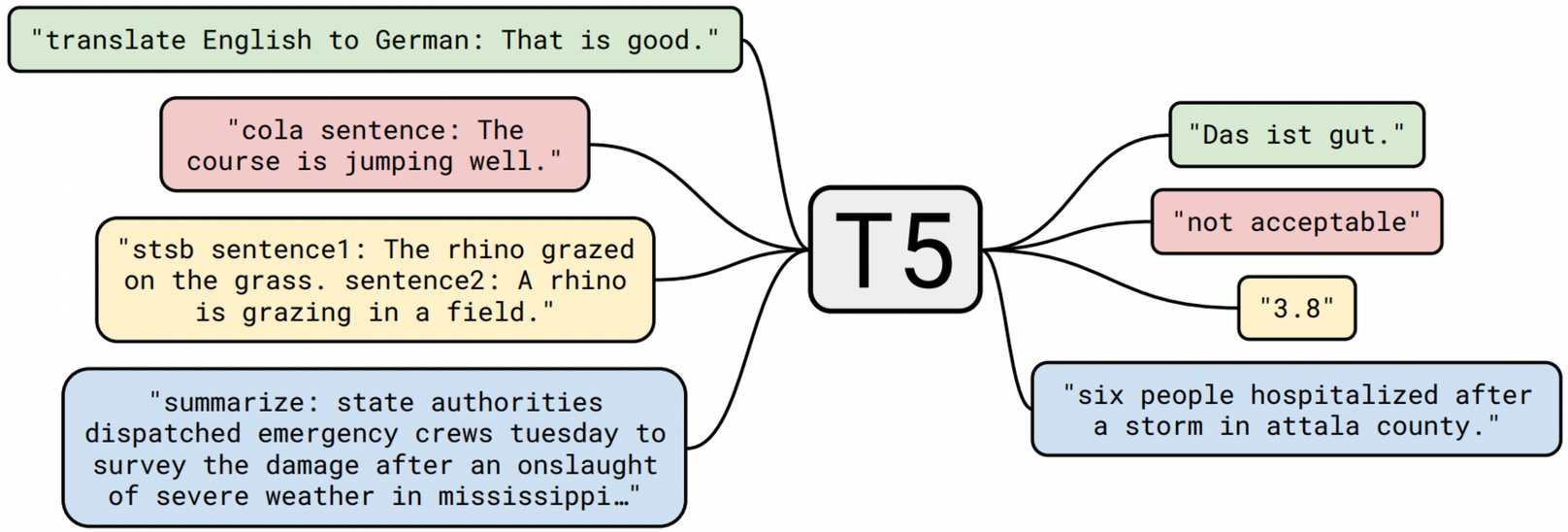
Design Considerations for Prompting

- Pre-trained Model Choice
- Prompt Template Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies



T5

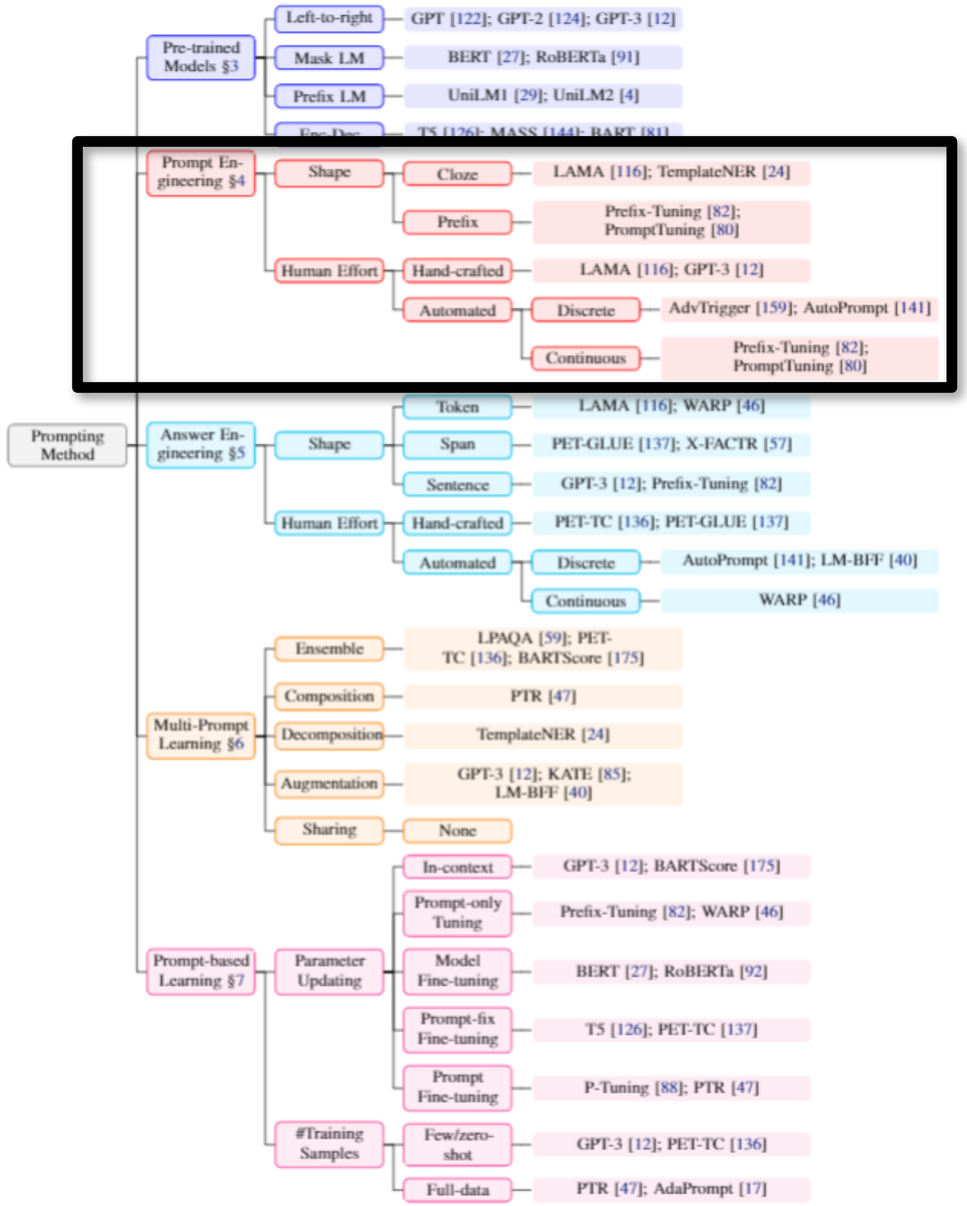
(Raffel et al. 2020)



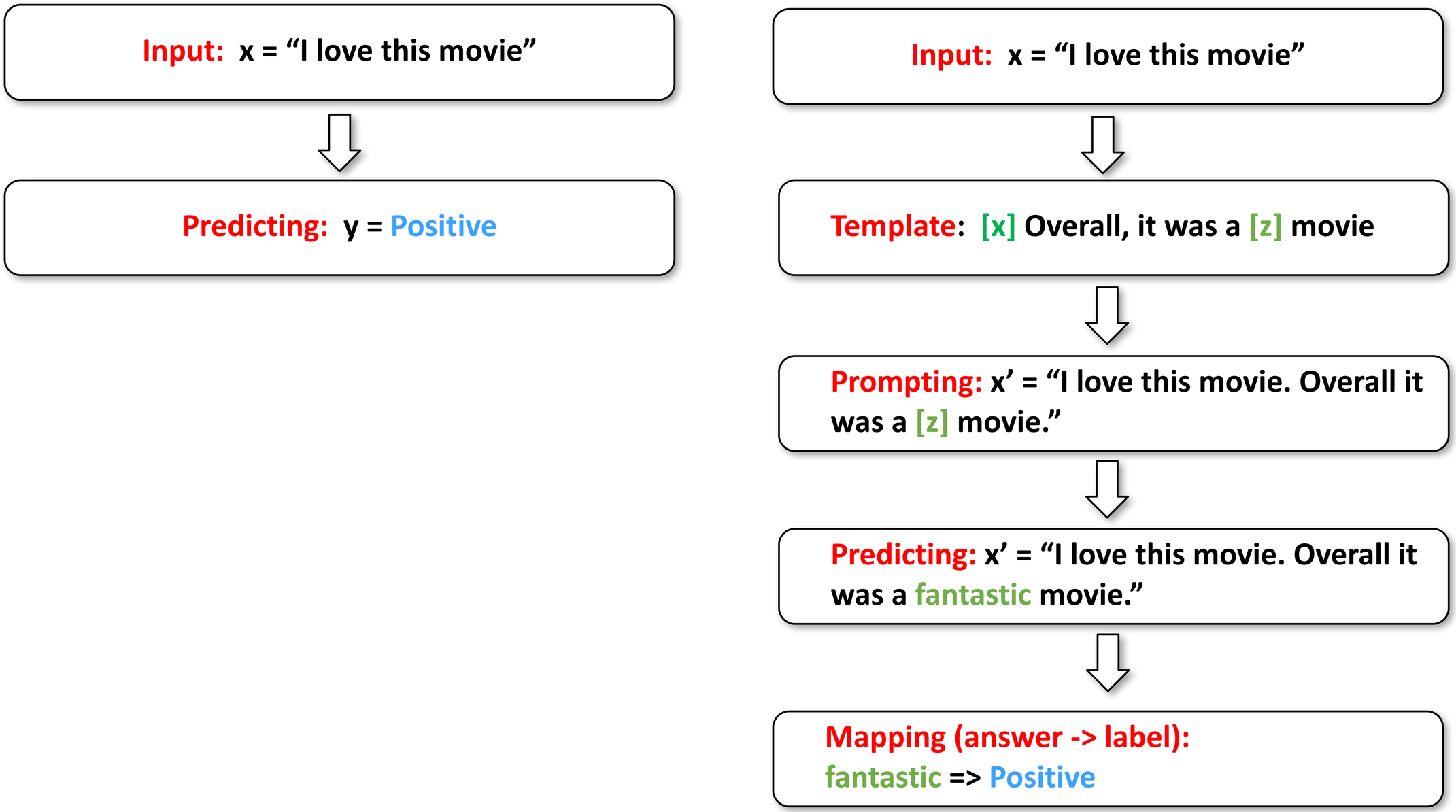
- Convert all tasks to sequence-to-sequence prediction

Design Considerations for Prompting

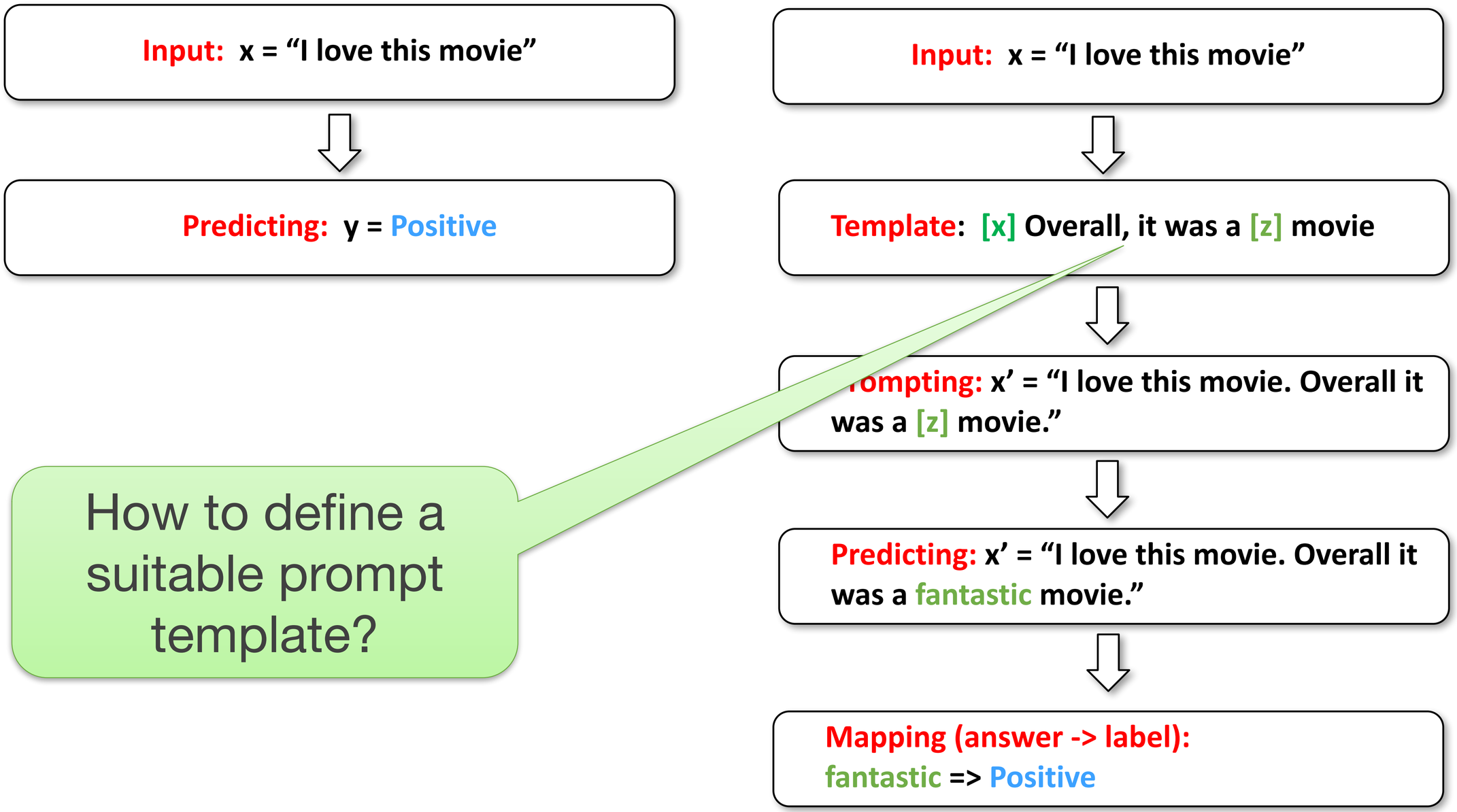
- Pre-trained Model Choice
- Prompt Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies



Traditional Formulation V.S Prompt Formulation

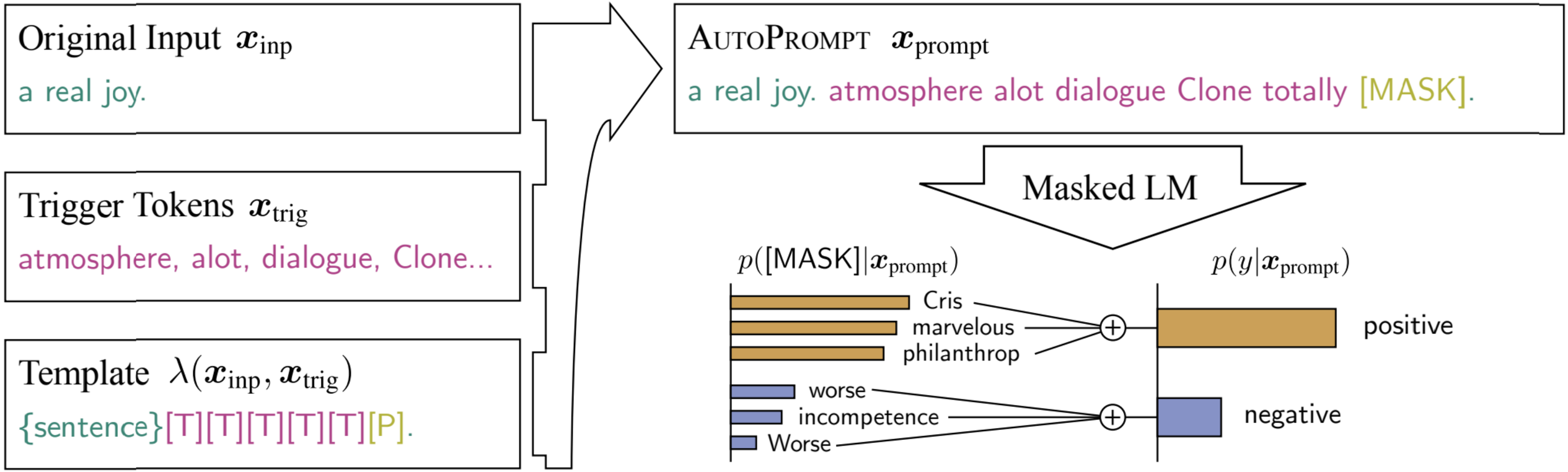


Traditional Formulation V.S Prompt Formulation

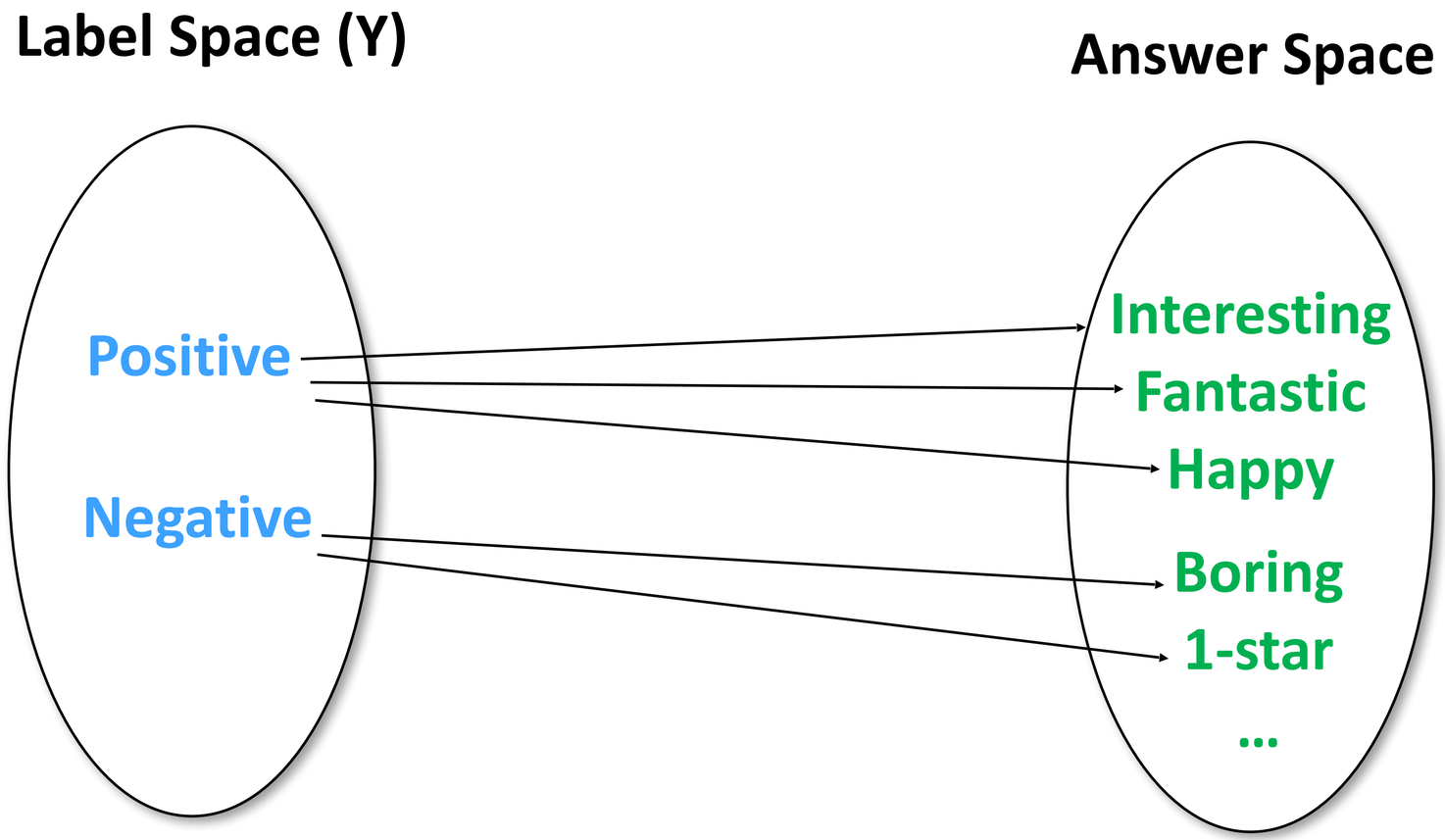


Gradient-based Search —AutoPrompt (Shin et al. 2020)

- Automatically optimize arbitrary prompts based on existing words



Traditional Formulation V.S Prompt Formulation



Answer Engineering

- Why do we need answer engineering?
  - We have reformulate the task! We also should re-define the “ground truth labels”
- Definition:
  - aims to search for an answer space and a map to the original output Y that results in an effective predictive model

Answer Shape

- **Token**: Answers can be one token in the pre-trained language model vocabulary
- **Chunk**: Answers can be chunks of words made up of more than one tokens
  - Usually used with the Cloze prompt
- **Sentence**: Answers can be a sentence of arbitrary length
  - Usually used with prefix prompt (seq2seq LM for generative tasks)

Answer Shape

Type	Task	Input ([X])	Template	Answer ([Z])
Text CLS	Sentiment	I love this movie.	[X] The movie is [Z].	great fantastic ...
	Topics	He prompted the LM.	[X] The text is about [Z].	sports science ...
	Intention	What is taxi fare to Denver?	[X] The question is about [Z]	quantity city ...
Text-span CLS	Aspect Sentiment	Poor service but good food.	[X] What about service? [Z].	Bad Terrible ...
Text-pair CLS	NLI	[X1]: An old man with ... [X2]: A man walks ...	[X1]? [Z], [X2]	Yes No ...
Tagging	NER	[X1]: Mike went to Paris. [X2]: Paris	[X1] [X2] is a [Z] entity.	organization location ...
Text Generation	Summarization	Las Vegas police ...	[X] TL;DR: [Z]	The victim ... A woman ... ...
	Translation	Je vous aime.	French: [X] English: [Z]	I love you. I fancy you. ...

token

Token or span

sentences

Advanced Prompting and Searching Answers

Chain-of-Thought Prompting

- Instead of searching for the answer directly, and manually add some intermediate reasoning steps in the prompt to guide the model derive the answer

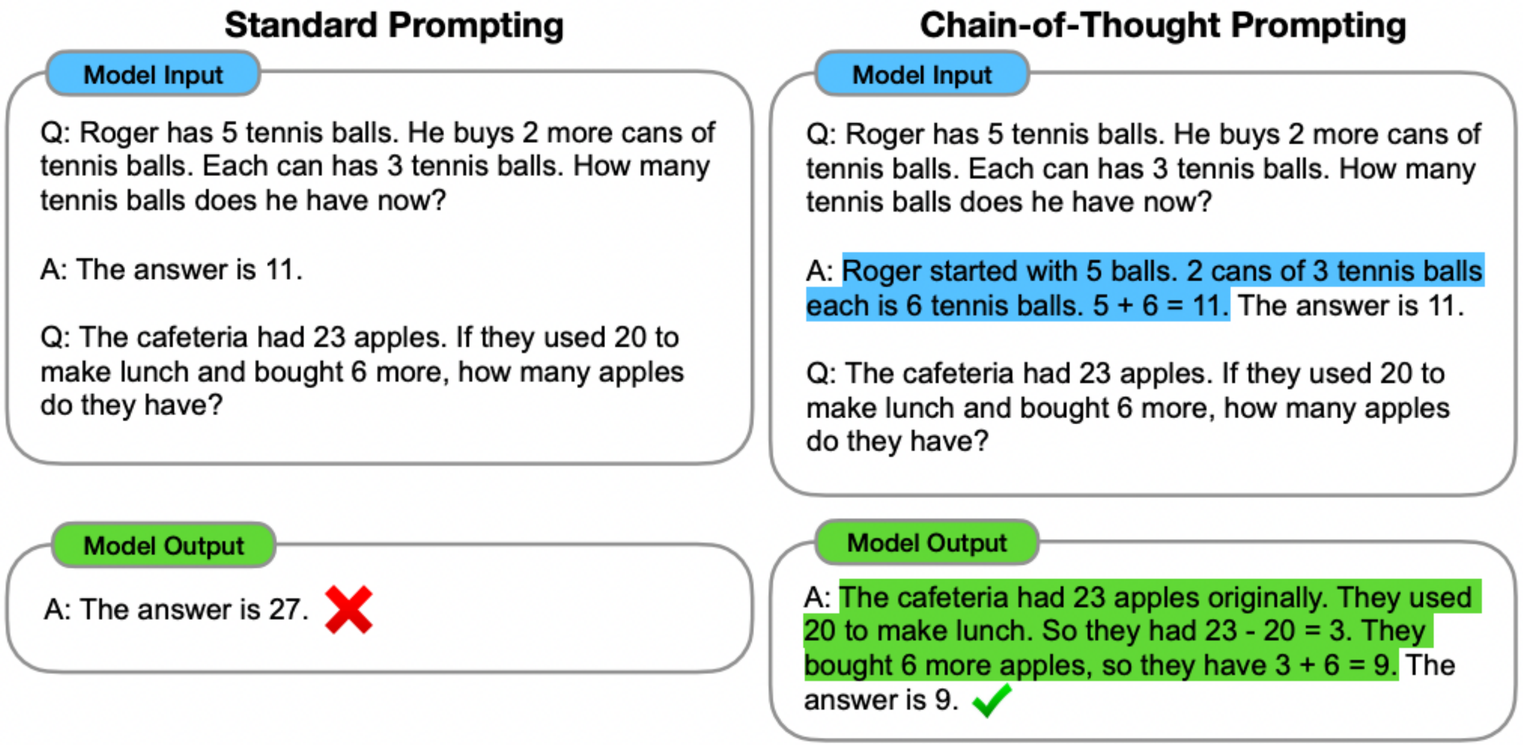
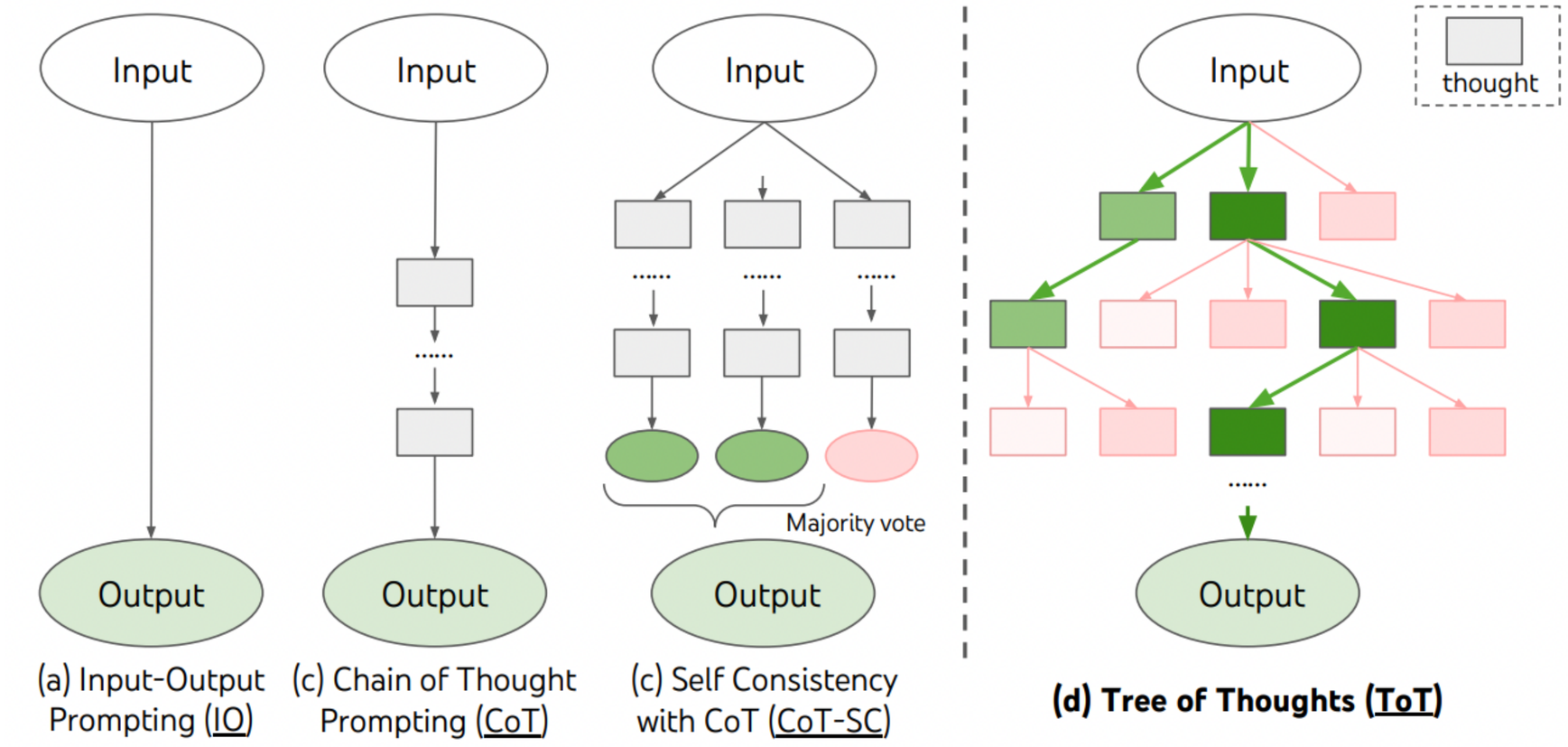


Figure 1: Chain-of-thought prompting enables large language models to tackle complex arithmetic, commonsense, and symbolic reasoning tasks. Chain-of-thought reasoning processes are highlighted.

Tree-of-Thought

- Instead of search the answer using a linear chain structure, prompt the output sequence to follow a tree structure



Tree of Thought: Example

- Game of 24 is a mathematical reasoning challenge, where the goal is to use 4 numbers and basic arithmetic operations (+-*/) to obtain 24. For example, given input “4 9 10 13”, a solution output could be “(10 - 4) * (13 - 9) = 24”.

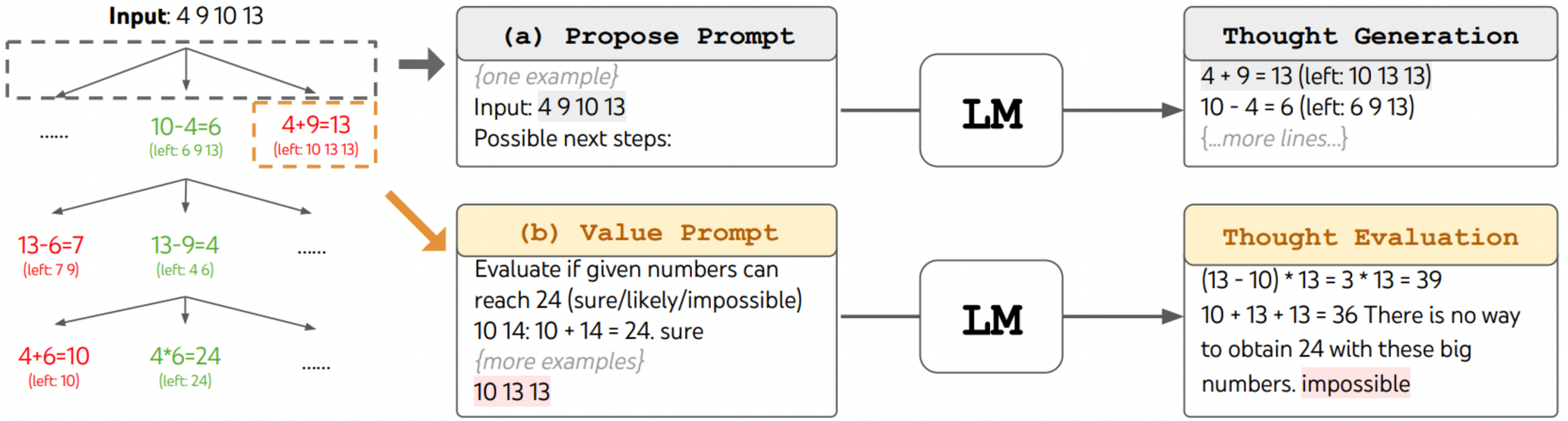
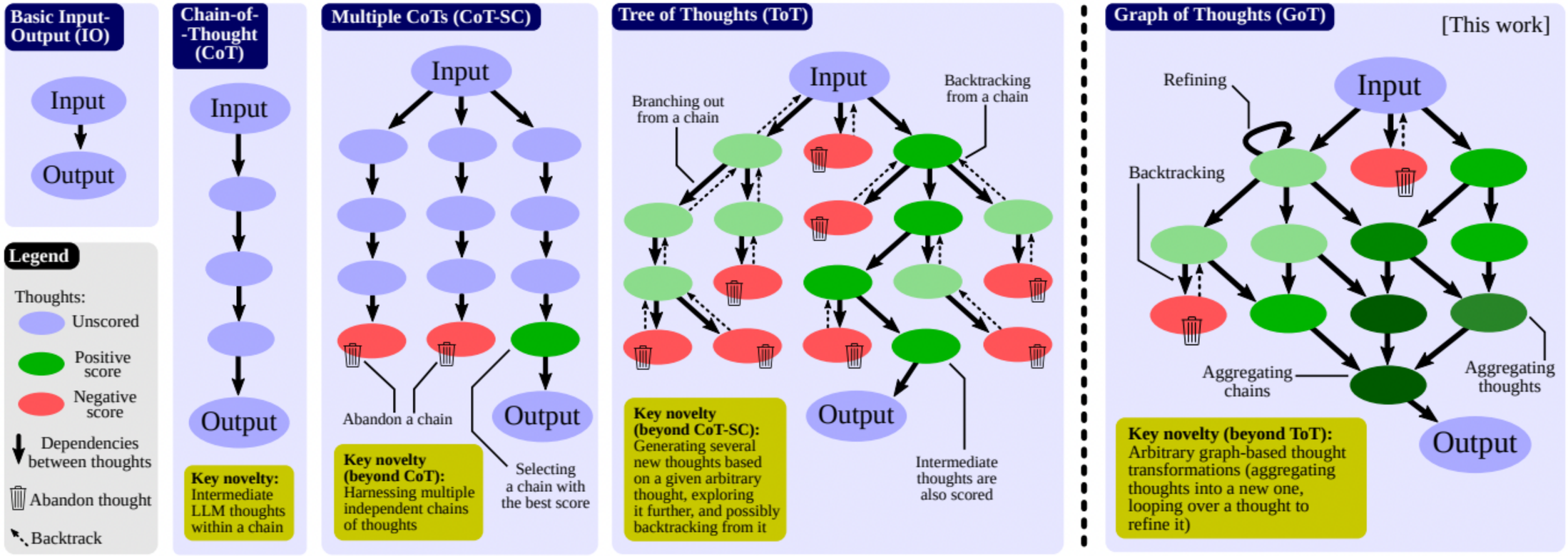


Figure 2: ToT in a game of 24. The LM is prompted for (a) thought generation and (b) valuation.

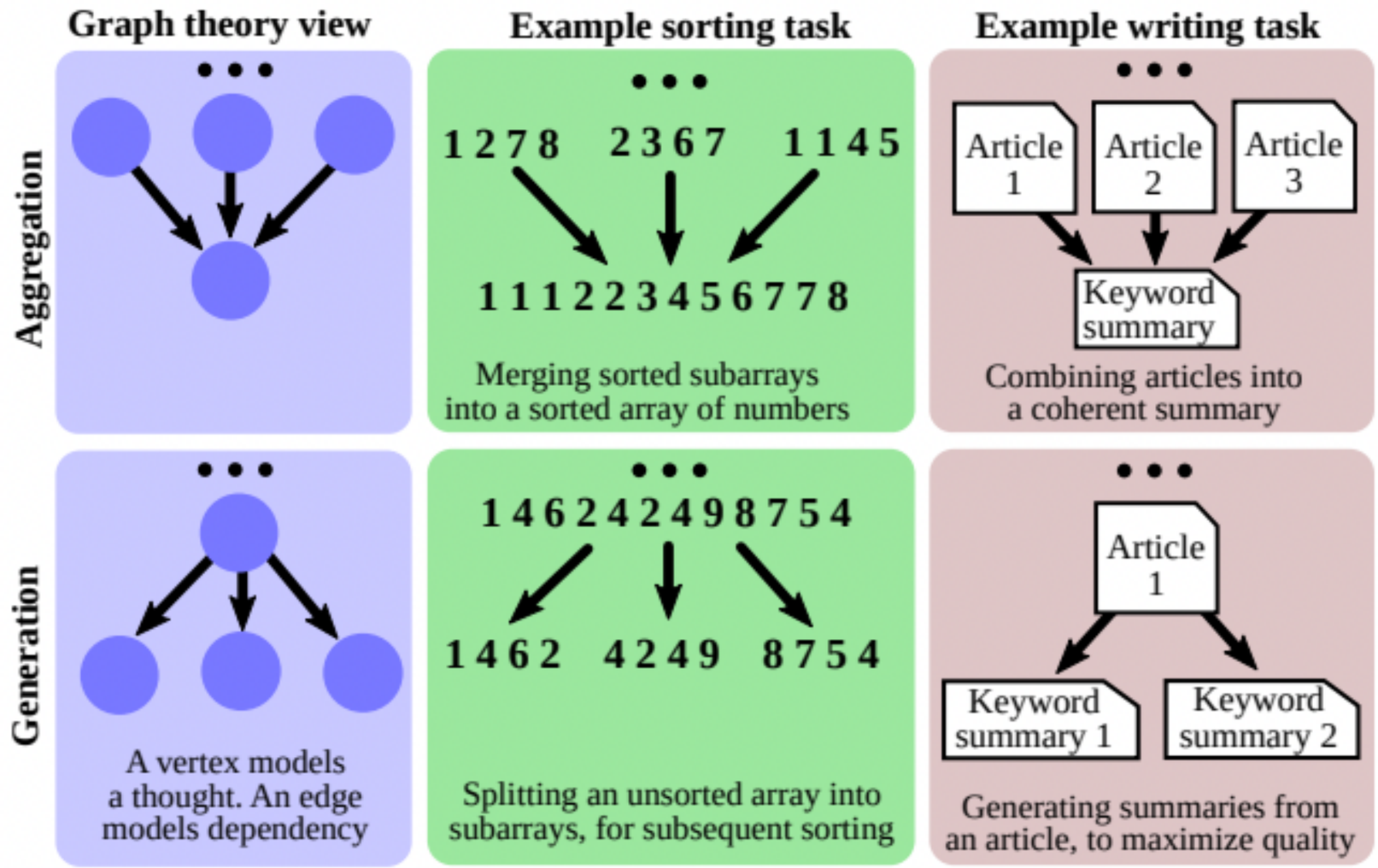
Graph-of-Thought

- Use a graph structure instead
  - Refining: allow self-loop over a single node
  - Aggregating: allow merging of multiple nodes

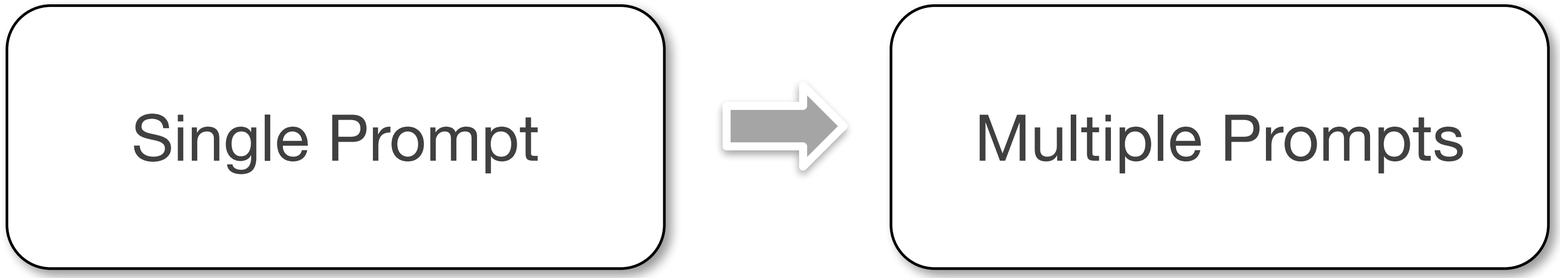


Graph-of-Thought: Example

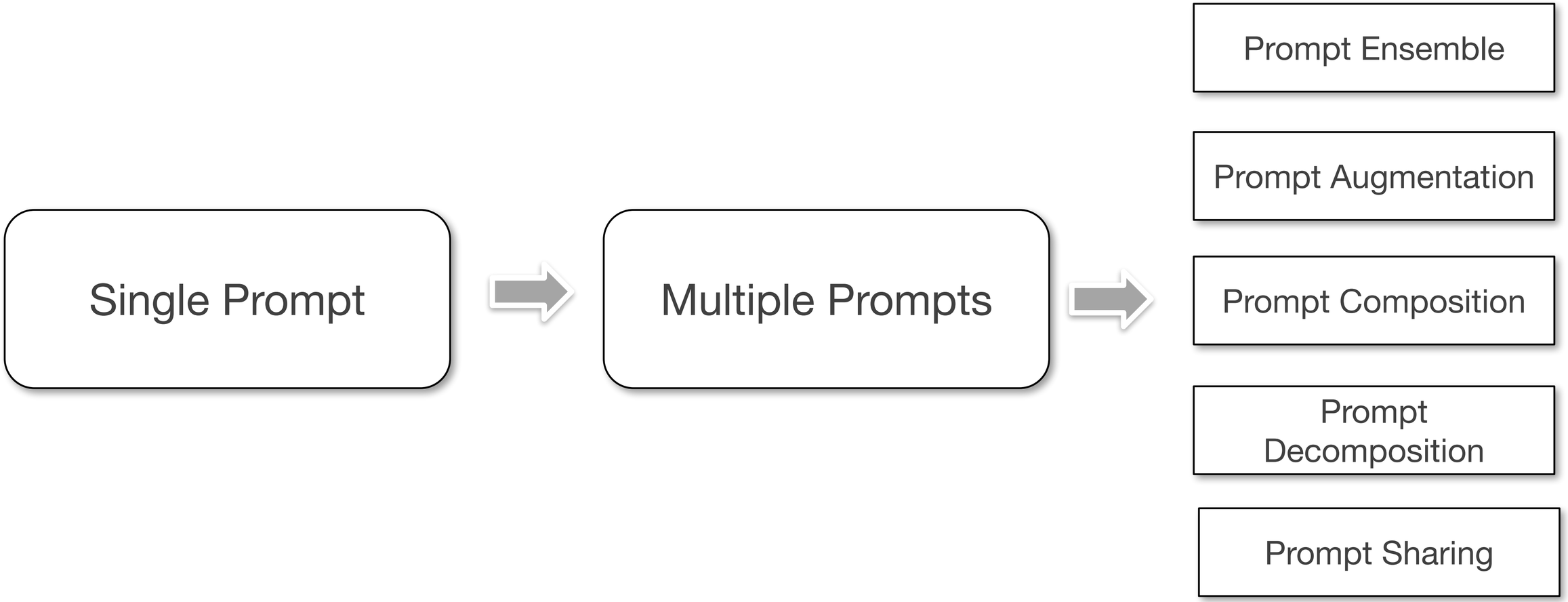
- Useful for some divide-and-conquer tasks: sorting, etc.



Multi-Prompt Learning

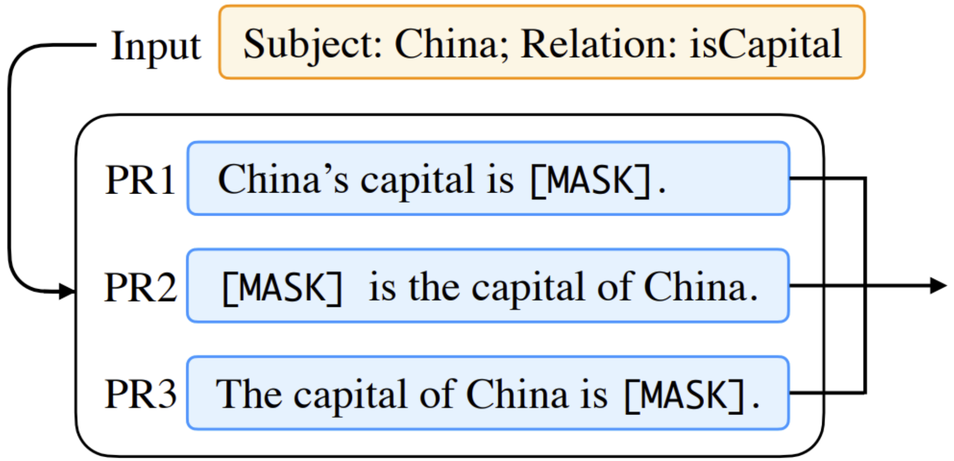


Multi-Prompt Learning



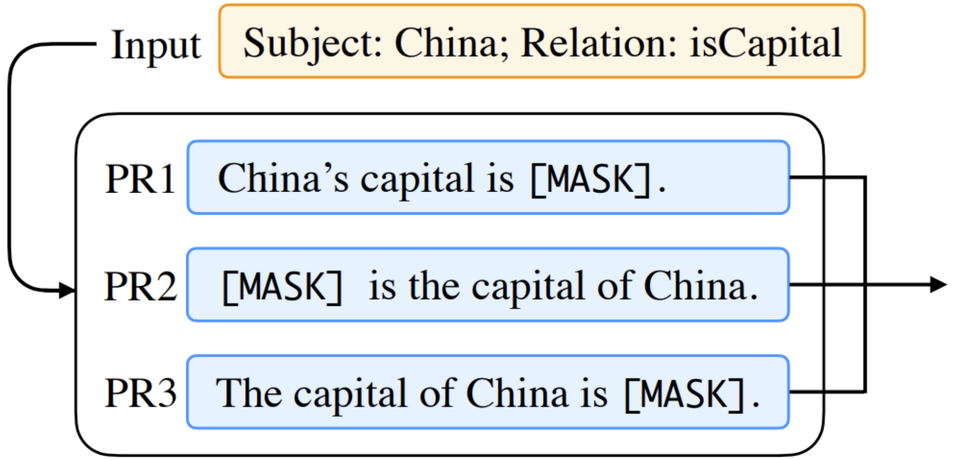
Prompt Ensembling

- Definition
  - using multiple unanswered prompts for an input at inference time to make predictions
- Advantages
  - Utilize complementary advantages
  - Alleviate the cost of prompt engineering
  - Stabilize performance on downstream tasks

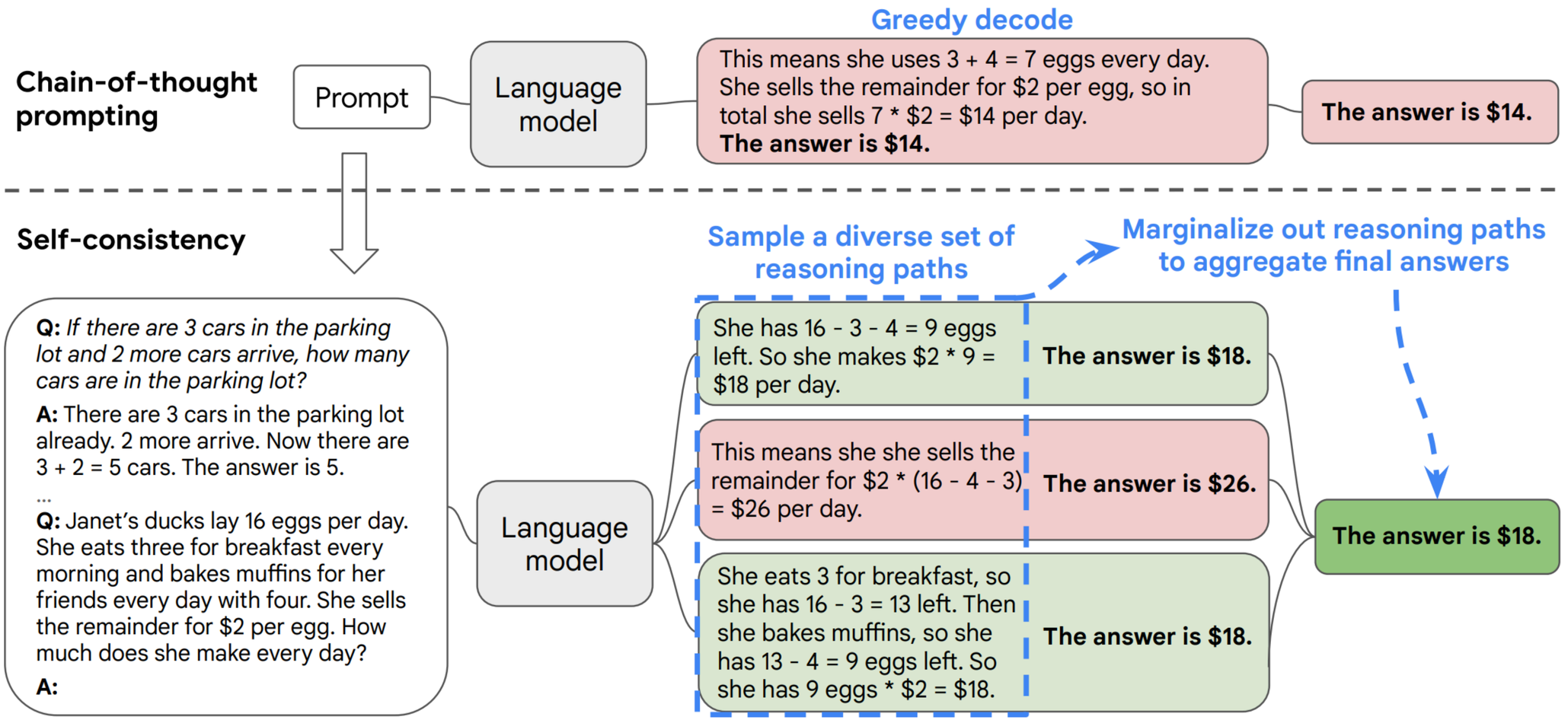


Prompt Ensembling

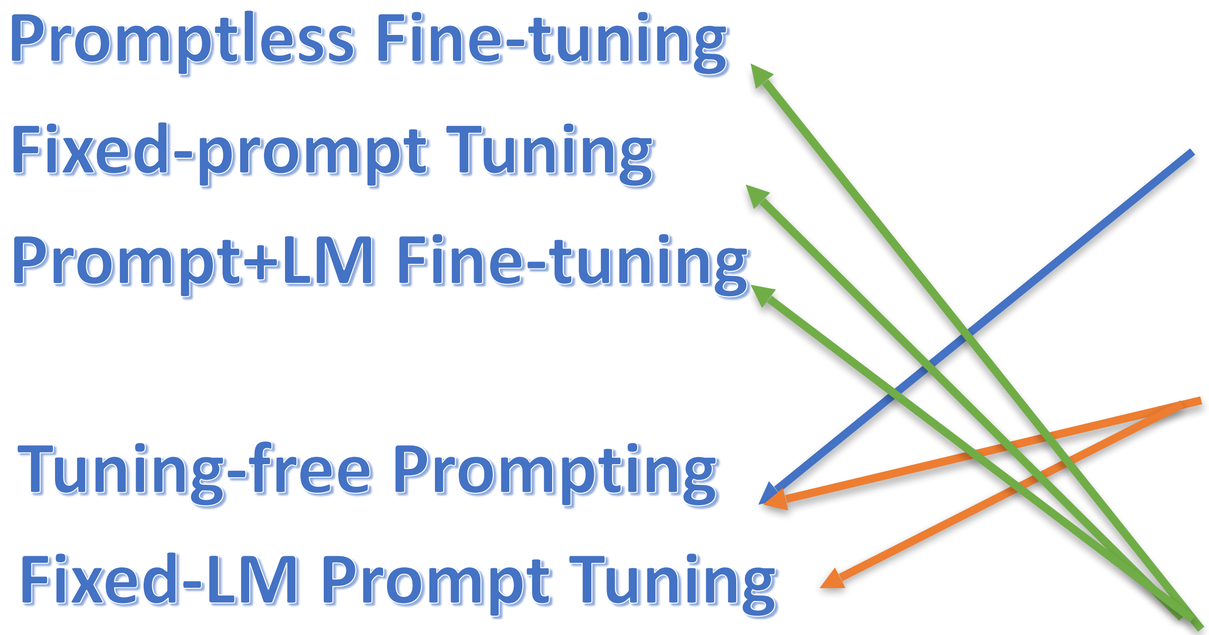
- Typical Methods
  - Uniform Averaging
  - Weighted Averaging
  - Majority Voting



Self-Consistency Prompting (Wang et al. 2022)



Too many, difficult to select?



If you have a huge pre-trained language model (e.g., GPT3)

If you have few training samples?

If you have lots of training samples?

Questions?