The Fourth Paradigm of Modern Natural Language Processing Techniques

Pengfei Liu
What is the “Prompt”?  
What is the general workflow of prompt-based methods?  
What are the design considerations for prompt-based methods?  
What (unique) advantages could prompt learning bring to us?  
How does prompt-based research progress currently?
What is the “Prompt”? 
The definition of a prompt is a cue given to someone to help him remember what to say, or is something that causes another event or action to occur.

An example of prompt is when you whisper a line to an actor who forgot what to say next.

An example of prompt is an event that starts an argument.
what are the most beautiful names
what are the most beautiful places in the world
what are the most beautiful zodiac signs
what are the most beautiful flowers
What is the “prompt” in the context of NLP research?
An Intuitive Definition

- Prompt is a cue given to the **pre-trained language model** to allow it better understand **human**’s questions.
More Technical Definition

- Prompt is the technique of making better use of the knowledge from the pre-trained model by adding additional texts to the input.
Prompt is the technique of making better use of the knowledge from the pre-trained model by adding additional texts to the input.
What is the **general workflow** of prompt-based methods?
Workflow for Prompting Methods

- Prompt Construction
- Answer Construction
- Answer Prediction
- Answer-Label Mapping
Prompting for **Sentiment Classification**

- **Task Description:**
  - **Input:** sentence $x$;
  - **Output:** emotional polarity of it
    (i.e., $\smile$ v.s $\frown$).

**Input:** $x = I$ love this movie.
Step 1: Prompt Construction

- Transform $x$ into prompt $x'$ through following two steps:
  - Defining a **template** with two **slots**: $[x]$ and $[z]$;
  - Input: $x = I$ love this movie.

  **Template:** $[x]$
  Overall, it was a $[z]$ movie.
Step 1: Prompt Construction

- Transform x into prompt x’ through following two steps:
  - Defining a template with two slots: [x] and [z];

Input: \( x = \text{I love this movie.} \)

Template: \([x]\)
Overall, it was a \([z]\) movie.

Require human effort
Step 1: Prompt Construction

- Transform $x$ into prompt $x'$ through following two steps:
  - Defining a template with two slots: $[x]$ and $[z]$;
  - Instantiate slot $[x]$ with input text.

**Input:** $x = \text{I love this movie.}$

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

**Prompting:** $x' = \text{I love this movie.}$ Overall, it was a $[z]$ movie.
Step 2: Answer Construction

- Build a mapping function between answers and class labels.

<table>
<thead>
<tr>
<th>label</th>
<th>answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>😊</td>
<td>fantastic</td>
</tr>
<tr>
<td>😞</td>
<td>boring</td>
</tr>
</tbody>
</table>

**Template:** \([x]\) Overall, it was a \([z]\) movie.

**Input:** \(x = \text{I love this movie.}\)

**Prompting:** \(x' = \text{I love this movie. Overall, it was a [z] movie.}\)

**Answer:** \(\{\text{fantastic: 😊, boring: 😞}\}\)
Step 3: Answer Predicting

- Given a prompt, predict the answer \([z]\).  
- Choose a suitable pretrained language model;

Input:  
\[x = \text{I love this movie.}\]

Template:  
\([x]\)  
Overall, it was a [z] movie.

Answer:  
\{\text{fantastic}: ☺, \text{boring}: ☹\}

Prompting:  
\[x' = \text{I love this movie.}\]  
Overall, it was a [z] movie.

Which one?
Step 3: Answer Predicting

- Given a prompt, predict the answer [z]
  - Choose a suitable pretrained language model;
  - Fill in [z] as “fantastic”

**Input:** \( x = \text{I love this movie.} \)

**Template:** \( [x] \)
Overall, it was a [z] movie.

**Answer:** \{fantastic: ☺, boring: ☹\}

**Prompting:** \( x' = \text{I love this movie. Overall, it was a [z] movie.} \)

**Predicting:** \( x' = \text{I love this movie. Overall, it was a \textbf{fantastic} movie.} \)
Step 4: Answer Mapping

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

  □ fantastic => 😊

**Input:** $x = \text{I love this movie.}$

**Template:** $[x]$

Overall, it was a $[z]$ movie.

**Answer:** \{fantastic:😊, boring:🙂\}

**Prompting:** $x' = \text{I love this movie.}$

Overall, it was a $[z]$ movie.

**Predicting:** $x' = \text{I love this movie.}$

Overall, it was a **fantastic** movie.

**Mapping:** fantastic => 😊
<table>
<thead>
<tr>
<th>Terminology</th>
<th>Notation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>x</td>
<td>I love this movie</td>
</tr>
<tr>
<td>Output (label)</td>
<td>y</td>
<td>☺ ☹</td>
</tr>
<tr>
<td>Template</td>
<td>-</td>
<td>[x] Overall, it was a [z] movie</td>
</tr>
<tr>
<td>Prompt</td>
<td>x’</td>
<td>I love this movie. Overall, it was a [z] movie</td>
</tr>
<tr>
<td>Answer</td>
<td>z</td>
<td>fantastic, boring</td>
</tr>
</tbody>
</table>
Rethinking Human Efforts in Prompt-based Methods

**Input:** \( x = I \text{ love this movie.} \)

**Template:** \([x]\)
- Overall, it was a \([z]\) movie.

**Answer:** \{fantastic: ☺, boring: ☹\}

**Prompting:** \( x' = I \text{ love this movie.} \)
- Overall, it was a \([z]\) movie.

**Predicting:** \( x' = I \text{ love this movie.} \)
- Overall, it was a fantastic movie.

**Mapping:** fantastic => ☺
Rethinking Human Efforts in Prompt-based Methods

**Input:** \( x = \text{I love this movie.} \)

**Predicting:** ☺

**Template:** \([x]\)

**Prompting:** \( x' = \text{I love this movie.} \)

**Overall, it was a [z] movie.**

**Answer:** \{fantastic:☺, boring:☺\}

**Predicting:** \( x' = \text{I love this movie.} \)

**Overall, it was a fantastic movie.**

**Mapping:** fantastic =>☺
What are the design considerations for prompt-based methods?
Design Considerations for Prompt-based Methods

- Prompt Template Engineering
- Answer Engineering
- Pre-trained Model Choice
- Expanding the Paradigm
- Prompt-based Training Strategies
Design Considerations for Prompt-based Methods

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

- Research Question:
  - how to define appropriate prompt templates

- It was a [z] movie
- The movie is [z]
- The film is [z]
Design Decision of Prompt Templates

Prompts are learnable parameters
I love this movie.

Overall it was a [z] movie.

I love this movie.

Overall the movie is [z]

I love this movie.

[e][e] [e] [e][z]
One Example

I love this movie.
Overall it was a [z] movie.

Prefix
Prefix
Prefix

Overall the movie is [z]
[e][e] [e] [e][z]
I love this movie.
Overall it was a [z] movie.

I love this movie.
Overall the movie is [z]

nonsense
token
Design Considerations for Prompt-based Methods

- Prompt Template Engineering
- **Answer Engineering**
- Pre-trained Model Choice
- Expanding the Paradigm
- Prompt-based Training Strategies
Research Question:

Given a task (or a prompt), how to define a suitable mapping function between label space and answer space?
Answer Engineering

- Research Question:

  □ Given a task (or a prompt), how to define a suitable mapping function between label space and answer space?

Label Space (Y)

- Positive
- Negative

Answer Space (Z)

- Interesting
- Fantastic
- Happy
- Boring
- 1-star
- ...

Positive

Negative
Design Decision of Prompt Answer Engineering

- Shape
  - Token: [7], [53]
  - Span: [19], [22]
  - Sent: [17], [29]

- Finite?
  - bounded
  - unbounded

- Human?
  - Manual: [2], [53]
  - Search
    - Discrete
    - Continuous: [28]

- Paraphrasing: [11]
- Prune-Search: [13]
- Label-Decomp: [44]
- Mining
<table>
<thead>
<tr>
<th>Task</th>
<th>Template</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentiment</td>
<td>[x] the movie is [z]</td>
<td>great, fantastic, boring</td>
</tr>
<tr>
<td>Classification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Template</td>
<td>Answer</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Sentiment Classification</td>
<td>[x] the movie is [z]</td>
<td>great, fantastic, boring</td>
</tr>
<tr>
<td>Summarization</td>
<td>[x] in summary, [z]</td>
<td>The news...</td>
</tr>
</tbody>
</table>
Design Considerations for Prompt-based Methods

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

■ Research Question:

☐ Given a task (or a prompt), **which pre-trained language model would be the most appropriate one?**

The story describes ...., in summary [z]
Design Decision of Pre-trained Models

PLMs

Objective Func
- Standard Language Model: GPT3
- Corrupted Text Reconstruction: BERT
- Full Text Reconstruction: BART

Data Corruption
- Mask, Deletion, Replacement: T5

Directionality
- Left-to-right: GPT
- Bidirectional: BERT
Design Decision of Pre-trained Models

PLMs
- Left-to-Right LM: GPT, GPT3
- Masked LM: BERT, RoBERT
- Encoder-decoder: BART, T5
Left-to-right Language Model

- **Characteristics**
  - First proposed by Markov (1913)
  - Count-based -> Neural network-based
  - Specifically suitable to highly larger-scale LMs

- **Example**
  - GPT-1, GPT-2, GPT-3

- **Roles in Promoting Methods**
  - The earliest architecture chosen for prompting
  - Usually equipped with prefix prompt and the parameters of PLMs are fixed
Masked Language Model

- **Characteristics**
  - An extension of left-to-right architecture
  - Unidirection -> bidirection prediction
  - Suitable for NLU tasks

- **Example**
  - BERT, ERNIE

- **Roles in Prompting Methods**
  - Usually combined with cloze prompt
  - Suitable for NLU tasks
Encoder-Decoder

- **Characteristics**
  - A denoised auto-encoder
  - Use two Transformers and two different mask mechanisms to handle text X and Y separately

- **Examples**
  - BART, T5

- **Roles in Prompting methods**
  - Text generation tasks or some tasks that can be formulated into a text generation problem
Design Considerations for Prompt-based Methods

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

Research Questions

- How to extend the current prompting framework to support more NLP tasks?
Design Decision of Multiple Prompt Learning

Multi-Prompt

- Prompt Ensemble [11], [57]
- Prompt Augmentation [47], [48]
- Prompt Composition [52]
- Prompt Decomposition [53]
- Prompt Sharing
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 Augmentation

- **Definition**
  - Help the model answer the prompt with additional answered prompts

- **Advantage**
  - make use of the small amount of information that has been annotated

- **Core step**
  - Selection of answered prompts
  - Ordering of answered prompts
Prompt Composition

■ Definition

☐ Prompts for a composable task can be designed with multiple sub-prompts, which can then be combined to complete the task

■ Advantage

☐ It provides a method of prompt learning for complex tasks
Prompt Decomposition

■ Definition
  □ For tasks where multiple predictions should be performed for one sample, handle it individually

■ Advantages
  □ Break-down a complicated task into multiple separate ones
Prompt Sharing

- **Definition**
  - When prompting method is applied to multiple tasks, domains or languages, prompts can be shared across different tasks.

- **Advantage**
  - Task- or language invariant information can be captured through prompting.

![Diagram showing the relationship between prompt sharing and prompting in different tasks and domains](image)
Design Considerations for Prompt-based Methods

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

Data Perspective

- Zero-shot: without any explicit training of the LM for the down-stream task
- Few-shot: few training (e.g., 100) samples of downstream tasks
- Full-data: lots of training samples (e.g., 10K) of downstream tasks
Parameter Perspective

Pre-trained LMs
- tuned
- frozen

Prompts
- no
- without
- frozen
- Tuned

No prompts
No parameters (discrete prompts)
Cases of Parameter Updating

- **Pre-trained LMs**
  - tuned
  - frozen

- **Prompts**
  - no
  - without
  - frozen
  - Tuned

**Promptless Fine-tuning**

Example: BERT for text classification
Cases of Parameter Updating

Pre-trained LMs

- tuned
- frozen

Prompts

- no
- without
- frozen
- Tuned

Fixed-prompt Tuning

Example: BERT + Discrete Prompt for text classification
Cases of Parameter Updating

Pre-trained LMs
- tuned
- frozen

Prompts
- no
- without
- frozen
- Tuned

Fixed-prompt Tuning

Example: BERT + Transferred Continuous Prompt for text classification
Cases of Parameter Updating

Pre-trained LMs
- tuned
- frozen

Prompts
- no
- without
- frozen
- Tuned

Prompt+LM Fine-tuning
Example: BERT + Continuous Prompt for text classification
Cases of Parameter Updating

Pre-trained LMs

- tuned
- frozen

Prompts

- no
- without
- frozen
- Tuned

Example: BERT + Adapter for text classification
Cases of Parameter Updating

Pre-trained LMs
- tuned
- frozen

Prompts
- no
- without
- frozen
- Tuned

Tuning-free Prompting

Example: GPT3 + Discrete Prompts for Machine Translation
Cases of Parameter Updating

Pre-trained LMs
- tuned
- frozen

Prompts
- no
- without
- frozen
- Tuned

Tuning-free Prompting

Example: GPT3 + Continuous Prompts for Machine Translation
Cases of Parameter Updating

Pre-trained LMs
- tuned
- frozen

Prompts
- no
- without
- frozen
- Tuned

Fixed-LM Prompt Tuning

Example: BART + Continuous Prompts for Machine Translation
Too many, difficult to select?

- Promptless Fine-tuning
- Fixed-prompt Tuning
- Prompt+LM Fine-tuning
- Adapter Tuning
- Tuning-free Prompting
- Fixed-LM Prompt Tuning

If you have a highly large left-to-right pre-trained language model (e.g., GPT3)

If you have few training samples?

If you have lots of training samples?
What (unique) advantages could prompt learning bring to us?
Four levels of vision

Level 1: Prompt learning

Level 2: Prompt learning v.s Fine-tuning

Level 3: Modern NLP history

Level 4: Beyond NLP
Level-1: Make All NLP Tasks as a Language Modeling Problem
Level-1: Make All NLP Tasks as a Language Modeling Problem

- Pretrained Language models can be better utilized
Level-1: Make All NLP Tasks as a Language Modeling Problem

- Pretrained Language models can be fully utilized
- (Almost) all NLP tasks can be handled zero-shotly

![Sentiment Classification Diagram]

- **Input & Template**
  - Input: \(x = \text{"I love this movie"}\)
  - Template: \([x] \text{ Overall, it was a } [z] \text{ movie}\)

- **Prompting**
  - \(x' = \text{"I love this movie. Overall it was a } [z] \text{ movie."}\)

- **Predicting**
  - \(x' = \text{"I love this movie. Overall it was a fantastic movie."}\)

- **Mapping (answer->label)**
  - \(y = \text{fantastic } \Rightarrow \text{Positive}\)
Pretrained Language models can be fully utilized

(Almost) all NLP tasks can be handled zero-shotly

Level-1: Make All NLP Tasks as a Language Modeling Problem

- Input
  - Input & Template
    - $x_1 = \text{“I will go to New York”}$
    - $x_2: \text{“New York”}$
  - Template
    - $[x_1], [x_2] \text{ is } [z]$

- Prompting
  - $x' = \text{I will go to New York, New York is } [z]$
  - Model

- Scoring
  - $y_1 = \text{New York is location name}$
  - $y_2 = \text{New York is person name}$
  - Score $= 5$
  - Score $= -3$

- Selecting & Mapping
  - $y(\text{New York}) = \text{location name}$ => LOC

Named Entity Recognition
Pretrained Language models can be fully utilized

(Almost) all NLP tasks can be handled zero-shotly
Level-1: Make All NLP Tasks as a Language Modeling Problem

- Pretrained Language models can be fully utilized
- (Almost) all NLP tasks can be handled zero-shotly
- Better few-shot performance
Level-1: (Almost) All NLP Tasks as Language Modeling

- Pretrained Language models can be fully utilized
- (Almost) all NLP tasks can be handled zero-shotly
- Better few-shot performance
- Make different tasks methodologically-connected available

Unified QA (Daniel et al 2020)
Different QA tasks are trained using one model
Level-2: Reverse Thinking

Objective modification

Task Reformulation

Fine-tuning

Prompting
Level-3: Reveal a “secret” about NLP development

- Prompting methods let out a secret how the technique of modern NLP progress
Four Paradigms in Modern NLP

- Feature Engineering
- Architecture Engineering
- Objective Engineering
- Prompt Engineering
Four Paradigms in Modern NLP

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

- **Paradigm**：Fully Supervised Learning (Non-neural Network)
- **Date**：Before 2013
- **Characteristic**：Traditional machine learning model is mainly used, which requires manual feature definition of input text
- **Typical Work**：
  - CRF (Conditional Random Field)
Four Paradigms in Modern NLP

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

- **Paradigm**: Fully Supervised Learning (Neural Network)
- **Date**: 2013 - 2018
- **Characteristic**:
  - Rely on neural networks
  - Do not need to manually define features, but should explore the network structure (e.g.: LSTM v.s CNN)

- **Typical Work**:
  - CNN for Text Classification
Four Paradigms in Modern NLP

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

- **Paradigm**: Pre-train, Fine-tune
- **Date**: 2018-Now
- **Characteristic**:
  - context-dependent PLMs
  - Need to pay attention to the definition and selection of objective functions
- **Typical Work**: BERT
Four Paradigms in Modern NLP

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

- **Paradigm**: Pre-train, Prompt, Predict
- **Date**: 2019-Now
- **Characteristic**:
  - NLP tasks are modeled entirely by relying on PLMs
  - More efforts on prompt design
- **Typical Work**: GPT3
PLMs and Downstream Tasks are Getting Closer and Closer

**Stages**

- Traditional machine learning
- Neural network methods enhanced by word2vec
- The fine-tune method represented by BERT
- The prompt approach represented by GPT3

**Downstream Tasks**

- No pre-training language model

**Pre-trained LMs**

- The pre-trained language model plays the role of initializing the input text signal
- The pre-trained language model is responsible for extracting high-level features from the input text
- Pre-training language models take on more responsibilities: feature extraction, result prediction

**Reasons**

- No pre-training language model
- The pre-trained language model plays the role of initializing the input text signal
- The pre-trained language model is responsible for extracting high-level features from the input text
- Pre-training language models take on more responsibilities: feature extraction, result prediction
The history of modern natural language processing is essentially (probably) a history of changes in the relationship between downstream tasks and pre-trained language models (PLMs).

1. use pre-trained language models
2. use a better pre-trained language model
3. better use a pre-trained language model
Prompting methods make

- more modalities of signals (e.g. image) connected using natural language as relay node

New view for human to **interact** with data in the world
How does prompt-based research progress currently?
Website Resource for Prompt-based Research

- Timeline
- Paperlist
Website Resource for Prompt-based Research

- **Timeline**
- **Paperlist**

---

**ExplainaBoard - Prompt-based Learning**

<table>
<thead>
<tr>
<th>Year</th>
<th>Task</th>
<th>Pretrained LMs</th>
<th>Settings</th>
<th>Prompt Mining</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018</td>
<td>AR</td>
<td>ALBERT</td>
<td>Few</td>
<td>automated</td>
</tr>
<tr>
<td>2019</td>
<td>CM</td>
<td>BART</td>
<td>Full</td>
<td>hand-crafted</td>
</tr>
<tr>
<td>2020</td>
<td>CR</td>
<td>BERT</td>
<td>Zero</td>
<td>+ more</td>
</tr>
<tr>
<td>2021</td>
<td>OCM</td>
<td>CTRL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODEG</td>
<td>DOTT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DOTT</td>
<td>EVALG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FP</td>
<td>+ more</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table of Papers**

<table>
<thead>
<tr>
<th>Year</th>
<th>Conf.</th>
<th>Title</th>
<th>Task</th>
<th>PLMs</th>
<th>Citations</th>
<th>Bib</th>
</tr>
</thead>
<tbody>
<tr>
<td>2021</td>
<td>NAACL</td>
<td>It's Not Just That Matters: Small Language Models Are Also Fine-Grained Similarity Detectors</td>
<td>TC</td>
<td>ALBERT</td>
<td>58</td>
<td>B.b.</td>
</tr>
</tbody>
</table>

**The number of papers over different tasks**

- TC
- QCG
- SUM
- NT
- IE
- AR
- D2T
- CPM
- SEMP
- UOA
- MG
Summary of Prompt-based Research

- How to apply prompting methods to diverse NLP tasks
  - More on classification/generation tasks + few-shot
  - Prediction-extensive tasks are under-explored
Summary of Prompt-based Research

- How to apply prompting methods to diverse NLP tasks
- Tuning Strategy
  - Head tuning
  - Adaptor tuning
  - Prompt tuning
Summary of Prompt-based Research

- How to apply prompting methods to diverse NLP tasks
- Tuning Strategy
- Non-NLP Tasks
  - Multi-modal/Computer vision
  - Biomedical
Summary of Prompt-based Research

- How to apply prompting methods to diverse NLP tasks
- Tuning Strategy
- Non-NLP Tasks
- Annotation
  - Generate training samples
  - Annotate data
Summary of Prompt-based Research

- How to apply prompting methods to diverse NLP tasks
- Tuning Strategy
- Non-NLP Tasks
- Annotation
- Pre-training
  - New pretraining framework
Summary of this talk

**What is the “Prompt”?**
- tool for human – PLM communication
- technique of making better use of pre-trained model by task reformulation

**What is the general workflow?**
- Prompt Construction
- Answer Construction
- Answer Prediction
- Answer-Label Mapping

**What are the design considerations?**
- Prompt Template Engineering
- Answer Engineering
- Pre-trained Model Choice
- Expanding the Paradigm
- Prompt-based Training Strategies

**What (unique) advantages?**
- Level-1: Within Prompt Learning
- Level-2: Prompt Learning v.s. Fine-tuning
- Level-3: Modern NLP History
- Level-4: Beyond NLP

**How does prompt-based research progress?**
- More diverse NLP tasks
- Tuning Strategy
- Non-NLP Tasks
- Annotation
- Pre-training