CS11-711 Advanced NLP

Prompting
(+ Encoder-Decoder Pre-training)

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/anlp2021/

Most Slides by Pengfei Liu
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  
zengbaoj@cs.cmu.edu

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

Graham Neubig  
Carnegie Mellon University  
gneubig@cs.cmu.edu
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 for Text Classification
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
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:** \( x = \text{“I love this movie”} \)

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

**Prompting:** \( x’ = \text{“I love this movie. Overall it was a [z] movie.”} \)
Answer Prediction

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

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

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

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

Predicting: \(x' = \text{"I love this movie. Overall it was a } \text{fantastic} \text{ movie."}\)
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 $\Rightarrow$ Positive
Types of Prompts

- Prompt: I love this movie. Overall it was a [z] movie
- Filled Prompt: I love this movie. Overall it was a boring movie
- Answered Prompt: I love this movie. Overall it was a fantastic movie
- Prefix Prompt: I love this movie. Overall this movie is [z]
- Cloze Prompt: I love this movie. Overall it was a [z] movie
Design Considerations for Prompting

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

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

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

Popular Frameworks

- Left-to-Right LM
- Masked LM
- Prefix LM
- Encoder-decoder
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 Prompting Methods**
  - The earliest architecture chosen for prompting
  - Usually equipped with prefix prompt and the parameters of PLMs are fixed
Masked Language Model

- **Characteristics:**
  - Unidirectional -> bidirectional prediction
  - Suitable for NLU tasks

- **Example:**
  - BERT, ERNIE

- **Roles in Prompting Methods**
  - Usually combined with cloze prompt
  - Suitable for NLU tasks, which should be reformulated into a cloze task
Prefix Language Model

- **Characteristics:**
  - A combination of Masked & Left-to-right
  - Use a Transformer but two different mask mechanisms to handle text X and y separately
  - Corruption operations can be introduced when encoding X

- **Examples:**
  - UniLM 1,2, ERNIE-M
**Encoder-Decoder**

- **Characteristics:**
  - A denoised auto-encoder
  - Use two Transformers and two different mask mechanisms to handle text X and y separately
  - Corruption operations can be introduced when encoding X

- **Examples:**
  - BART, T5
Encoder-decoder Pre-training Methods

Representative Methods

- MASS
- BART (mBART)
- UniLM
- T5
MASS
(Song et al.)

- Model: Transformer-based Encoder-decoder
- Objective: *only* predict masked spans
- Data: WebText
BART
(Lewis et al.)

- Model: Transformer-based encoder-decoder model
- Objective: Re-construct (corrupted) original sentences
- Data: similar to RoBERTa (160GB): BookCorpus, CC-NEWs, WebText, Stories
**mBART** (Liu et al.)

- **Model**: Transformer-based *Multi-lingual Denoising* auto-encoder

- **Objective**: Re-construct (corrupted) *original sentences*

- **Data**: CC25 Corpus (25 languages)
UNiLM
(Dong et al.)

- Model: prefixed-LM, left-to-right LM, Masked LM
- Objective: three types of LMs, *shared* parameters
- Data: English Wikipedia and BookCorpus
T5
(Raffel et al.)

- Model: left-to-right LM, Prefixed LM, encoder-decoder
- Objective: explore different objectives respectively
- Data: C4 (750G) + Wikipedia + RealNews + WebText
T5
( Raffel et al.)

- Model: left-to-right LM, Prefix LM, encode-decoder
- Objective: explore different objectives respectively
- Data: C4 (750G) + Wikipedia + RealNews + WebText
Application of Prefix LM/Encoder-Decoders in Prompting

- **Conditional Text Generation**
  - Translation
  - Text Summarization

- **Generation-like Tasks**
  - Information Extraction
  - Question Answering
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

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

**Predicting:** $y = \text{Positive}$

---

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

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

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

**Predicting:** $x' = \text{“I love this movie. Overall it was a fantastic movie.”}$

**Mapping (answer -> label):**

fantastic $\Rightarrow$ Positive
How to define a suitable prompt template?
Prompt Template Engineering

How to define the shape of a prompt template?

How to search for appropriate prompt templates?
Prompt Shape

- **Cloze Prompt**
  - prompt with a slot [z] to fill in the middle of the text as a cloze prompt,
  - I love this movie. Overall it was a [z] movie

- **Prefix Prompt**
  - prompt where the input text comes entirely before slot [z]
  - I love this movie. Overall this movie is [z]
Design of Prompt Templates

- Hand-crafted
  - Configure the manual template based on the characteristics of the task

- Automated search
  - Search in discrete space
  - Search in continuous space
Representative Methods for Prompt Search

- Prompt Mining
- Prompt Paraphrasing
- Gradient-based Search
- Prompt/Prefix Tuning
Prompt Mining (Jiang et al. 2019)

- Mine prompts given a set of questions/answers
- **Middle-word**
  
  Barack Obama was born in Hawaii. → [X] was born in [Y].

- **Dependency-based**

  The capital of France is Paris. → capital of [X] is [Y].
Prompt Paraphrasing (Jiang et al. 2019)

- Paraphrase an existing prompt to get other candidates
- e.g. back translation with beam search

[X] shares a border with [Y].

---

[X] has a common border with [Y].
[X] adjoins [Y].

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

• Automatically optimize arbitrary prompts based on existing words

Original Input $x_{\text{inp}}$

a real joy.

Trigger Tokens $x_{\text{trig}}$

atmosphere, alot, dialogue, Clone...

Template $\lambda(x_{\text{inp}}, x_{\text{trig}})$

{sentence}[T][T][T][T][T][T][P].

AUTOPROMPT $x_{\text{prompt}}$

a real joy. atmosphere alot dialogue Clone totally [MASK].

Masked LM

$p([\text{MASK}]|x_{\text{prompt}})$

positive

$p(y|x_{\text{prompt}})$

negative

Cris marvelous philanthrop

worse incompetence

Worse
Prefix/Prompt Tuning (Li and Liang 2021, Lester et al. 2021)

- Optimize the embeddings of a prompt, instead of the words.
- "Prompt Tuning" optimizes only the embedding layer, "Prefix Tuning" optimizes prefix of all layers
Design Considerations for Prompting

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

- Why do we need answer engineering?
  - We have reformulated the task! We also should re-define the “ground truth labels”
Traditional Formulation V.S Prompt Formulation

Input: $x = \text{“I love this movie”}$

Predicting: $y = \text{Positive}$

Input: $x = \text{“I love this movie”}$

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

Prompting: $x’ = \text{“I love this movie. Overall it was a [z] movie.”}$

Predicting: $x’ = \text{“I love this movie. Overall it was a fantastic movie.”}$

Mapping (answer -> label): fantastic => Positive
Traditional Formulation V.S Prompt Formulation

Label Space (Y)

Positive

Negative

Answer Space (Z)

Interesting
Fantastic
Happy
Boring
1-star
...

47
Why do we need answer engineering?

- We have reformulated 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
Design of Prompt Answer

How to define the shape of an answer?

How to search for appropriate answers?
Answer Shape

- **Token**: Answers can be one or more tokens in the pre-trained language model vocabulary
- **Chunk**: Answers can be chunks of words made up of more than one tokens
  - Usually used with cloze prompt
- **Sentence**: Answers can be a sentence of arbitrary length
  - Usually used with prefix prompt
<table>
<thead>
<tr>
<th>Type</th>
<th>Task</th>
<th>Input ([X])</th>
<th>Template</th>
<th>Answer ([Z])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text CLS</td>
<td>Sentiment</td>
<td>I love this movie.</td>
<td>[X] The movie is [Z].</td>
<td>great</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fantastic</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Topics</td>
<td>He prompted the LM.</td>
<td>[X] The text is about [Z].</td>
<td>sports</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>science</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Intention</td>
<td>What is taxi fare to Denver?</td>
<td>[X] The question is about [Z].</td>
<td>quantity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>city</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Text-span CLS</td>
<td>Aspect</td>
<td>Poor service but good food.</td>
<td>[X] What about service? [Z].</td>
<td>Bad</td>
</tr>
<tr>
<td></td>
<td>Sentiment</td>
<td></td>
<td></td>
<td>Terrible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Text-pair CLS</td>
<td>NLI</td>
<td>[X1]: An old man with ...</td>
<td>[X1]? [Z], [X2]</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[X2]: A man walks ...</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Tagging</td>
<td>NER</td>
<td>[X1]: Mike went to Paris.</td>
<td>[X1] [X2] is a [Z] entity.</td>
<td>organization</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[X2]: Paris</td>
<td></td>
<td>location</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Text Generation</td>
<td>Summarization</td>
<td>Las Vegas police ...</td>
<td>[X] TL;DR: [Z]</td>
<td>The victim ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A woman ...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>I fancy you.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Answer Search

- **Hand-crafted**
  - Infinite answer space
  - Finite answer space

- **Automated Search**
  - Discrete Space
  - Continuous Space
Discrete Search Space

- **Answer Paraphrasing**
  - start with an initial answer space,
  - then use paraphrasing to expand this answer space

- **Prune-then-Search**
  - an initial pruned answer space of several plausible answers is generated
  - an algorithm further searches over this pruned space to select a final set of answers

- **Label Decomposition**
  - decompose each relation label into its constituent words and use them as an answer
    - city_of_death => {person, city, death}
Continuous Search Space

- Core idea: assign a virtual token for each class label and optimize the token embedding for each label
Design Considerations for Prompting

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

Single Prompt ➔ Multiple Prompts
Multi-Prompt Learning

Single Prompt ➔ Multiple Prompts

- Prompt Ensemble
- Prompt Augmentation
- Prompt Composition
- Prompt Decomposition
- 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 Ensembling

- **Typical Methods**
  - Uniform Averaging
  - Weighted Averaging
  - Majority Voting
Prompt Augmentation

- **Definition**
  - Help the model answer the prompt that is currently being answered by 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
Design Considerations for Prompting

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

- **Data Perspective**
  - How many training samples are used?

- **Parameter Perspective**
  - Whether/How are parameters updated?
Prompt-based Training: Data Perspective

- **Zero-shot**: without any explicit training of the LM for the downstream task
- **Few-shot**: few training samples (e.g., 1-100) of downstream tasks
- **Full-data**: lots of training samples (e.g., 10K) of downstream tasks
Prompt-based Training: Parameter Perspective

<table>
<thead>
<tr>
<th>Strategy</th>
<th>LM Params Tuned</th>
<th>Additional Prompt Params</th>
<th>Prompt Params Tuned</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Promptless Fine-Tuning</td>
<td>Yes</td>
<td>N/A</td>
<td>N/A</td>
<td>BERT Fine-tuning</td>
</tr>
<tr>
<td>Tuning-free Prompting</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
<td>GPT-3</td>
</tr>
<tr>
<td>Fixed-LM Prompt Tuning</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Prefix Tuning</td>
</tr>
<tr>
<td>Fixed-prompt LM Tuning</td>
<td>Yes</td>
<td>No</td>
<td>N/A</td>
<td>PET</td>
</tr>
<tr>
<td>Prompt+LM Fine-tuning</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>PADA</td>
</tr>
</tbody>
</table>
Too many, difficult to select?

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

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?