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

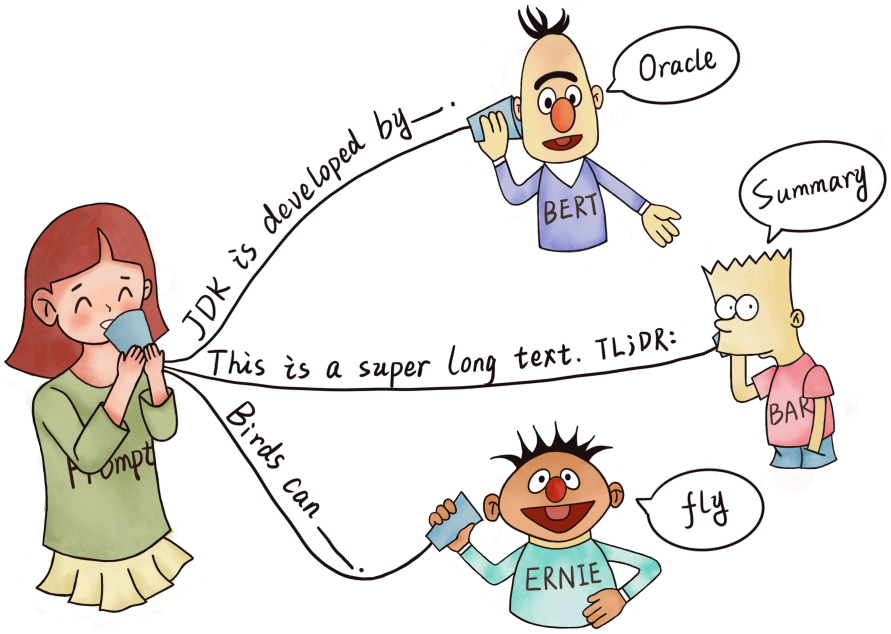
Zhengbao Jiang
Carnegie Mellon University
zhengbaj@cs.cmu.edu

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

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

Abstract

This paper surveys and organizes research works in a new paradigm in natural language processing, which we dub "prompt-based learning". Unlike traditional supervised learning, which trains a model to take in an input $\boldsymbol{x}$ and predict an output $\boldsymbol{y}$ as $P(\boldsymbol{y}|\boldsymbol{x})$, prompt-based learning is based on language models that model the probability of text directly. To use these models to perform prediction tasks, the original input $\boldsymbol{x}$ is modified using a *template* into a textual string *prompt* $\boldsymbol{x}'$ that has some unfilled slots, and then the language model is used to probabilistically fill the unfilled information to obtain a final string $\hat{\boldsymbol{x}}$, from which the final output $\boldsymbol{y}$ can be derived. This framework is powerful and attractive for a number of reasons: it allows the language model to be *pre-trained* on massive amounts of raw text, and by defining a new prompting function the model is able to perform *few-shot* or even *zero-shot* learning, adapting to new scenarios with few or no labeled data. In this paper we introduce the basics of this promising paradigm, describe a unified set of mathematical notations that can cover a wide variety of existing work, and organize existing work along several dimensions, e.g. the choice of pre-trained models, prompts, and tuning strategies. To make the field more accessible to interested beginners, we not only make a systematic review of existing works and a highly structured typology of prompt-based concepts, but also release other resources, e.g., a website NLPedia–Pretrain including constantly-updated survey, and paperlist.



Contents

1 Two Sea Changes in NLP **3**

2 A Formal Description of Prompting **4**
2.1 Supervised Learning in NLP 4
2.2 Prompting Basics 4
2.2.1 Prompt Addition 5
2.2.2 Answer Search 5
2.2.3 Answer Mapping 5
2.3 Design Considerations for Prompting . 6

3 Pre-trained Language Models **8**
3.1 Training Objectives 8
3.2 Noising Functions 8
3.3 Directionality of Representations 9
3.4 Typical Pre-training Methods 9
3.4.1 Left-to-Right Language Model . 9
3.4.2 Masked Language Models . . . 10
3.4.3 Prefix and Encoder-Decoder . . 10

4 Prompt Engineering **11**
4.1 Prompt Shape 11
4.2 Manual Template Engineering 11
4.3 Automated Template Learning 11
4.3.1 Discrete Prompts 12
4.3.2 Continuous Prompts 12

5 Answer Engineering **13**
5.1 Answer Shape 13
5.2 Answer Space Design Methods 14
5.2.1 Manual Design 14
5.2.2 Discrete Answer Search 14
5.2.3 Continuous Answer Search . . . 14

6 Multi-Prompt Learning **15**
6.1 Prompt Ensembling 15
6.2 Prompt Augmentation 16
6.3 Prompt Composition 16
6.4 Prompt Decomposition 17

7 Training Strategies for Prompting Methods **17**
7.1 Training Settings 17
7.2 Parameter Update Methods 17
7.2.1 Promptless Fine-tuning 18
7.2.2 Tuning-free Prompting 18
7.2.3 Fixed-LM Prompt Tuning . . . 18
7.2.4 Fixed-prompt LM Tuning . . . 18
7.2.5 Prompt+LM Tuning 19

8 Applications **19**
8.1 Knowledge Probing 19
8.2 Classification-based Tasks 19
8.3 Information Extraction 22
8.4 "Reasoning" in NLP 22
8.5 Question Answering 23
8.6 Text Generation 23
8.7 Automatic Evaluation of Text Generation 23
8.8 Multi-modal Learning 23
8.9 Meta-Applications 23
8.10 Resources 24

9 Prompt-relevant Topics **24**

10 Challenges **27**
10.1 Prompt Design 27
10.2 Answer Engineering 28
10.3 Selection of Tuning Strategy 28
10.4 Multiple Prompt Learning 28
10.5 Selection of Pre-trained Models 29
10.6 Theoretical and Empirical Analysis of Prompting 29
10.7 Transferability of Prompts 29
10.8 Combination of Different Paradigms . . 29
10.9 Calibration of Prompting Methods . . . 29

11 Meta Analysis **29**
11.1 Timeline 31
11.2 Trend Analysis 31

12 Conclusion **31**

A Appendix on Pre-trained LMs **44**
A.1 Evolution of Pre-trained LM Parameters 44
A.2 Auxiliary Objective 44
A.3 Pre-trained Language Model Families 45

1 Two Sea Changes in NLP

Fully supervised learning, where a task-specific model is trained solely on a dataset of input-output examples for the target task, has long played a central role in many machine learning tasks (Kotsiantis et al., 2007), and natural language processing (NLP) was no exception. Because such fully supervised datasets are ever-insufficient for learning high-quality models, early NLP models relied heavily on *feature engineering* (Tab. 1 a.; e.g. Lafferty et al. (2001); Guyon et al. (2002); Och et al. (2004); Zhang and Nivre (2011)), where NLP researchers or engineers used their domain knowledge to define and extract salient features from raw data and provide models with the appropriate inductive bias to learn from this limited data. With the advent of neural network models for NLP, salient features were learned jointly with the training of the model itself (Collobert et al., 2011; Bengio et al., 2013), and hence focus shifted to *architecture engineering*, where inductive bias was rather provided through the design of a suitable network architecture conducive to learning such features (Tab. 1 b.; e.g. Hochreiter and Schmidhuber (1997); Kalchbrenner et al. (2014); Chung et al. (2014); Kim (2014); Bahdanau et al. (2014); Vaswani et al. (2017)).[1]

However, from 2017-2019 there was a sea change in the learning of NLP models, and this fully supervised paradigm is now playing an ever-shrinking role. Specifically, the standard shifted to the *pre-train and fine-tune* paradigm (Tab. 1 c.; e.g. Radford and Narasimhan (2018); Peters et al. (2018); Dong et al. (2019); Yang et al. (2019); Lewis et al. (2020a)). In this paradigm, a model with a fixed[2] architecture is *pre-trained* as a language model (LM), predicting the probability of observed textual data. Because the raw textual data necessary to train LMs is available in abundance, these LMs can be trained on large datasets, in the process learning robust general-purpose features of the language it is modeling. The above pre-trained LM will be then adapted to different downstream tasks by introducing additional parameters and *fine-tuning* them using task-specific objective functions. Within this paradigm, the focus turned mainly to *objective engineering*, designing the training objectives used at both the pre-training and fine-tuning stages. For example, Zhang et al. (2020a) show that introducing a loss function of predicting salient sentences from a document will lead to a better pre-trained model for text summarization. Notably, the main body of the pre-trained LM is generally (but not always; Peters et al. (2019)) fine-tuned as well to make it more suitable for solving the downstream task.

Now, as of this writing in 2021, we are in the middle of a second sea change, in which the "pre-train, fine-tune" procedure is replaced by one in which we dub "*pre-train, prompt, and predict*". In this paradigm, instead of adapting pre-trained LMs to downstream tasks via objective engineering, downstream tasks are reformulated to look more like those solved during the original LM training with the help of a textual *prompt*. For example, when recognizing the emotion of a social media post, "I missed the bus today.", we may continue with a prompt "I felt so ___", and ask the LM to fill the blank with an emotion-bearing word. Or if we choose the prompt "English: I missed the bus today. French: ___"), an LM may be able to fill in the blank with a French translation. In this way, by selecting the appropriate prompts we can manipulate the model behavior so that the pre-trained LM itself can be used to *predict* the desired output, sometimes even without any additional task-specific training (Tab. 1 d.; e.g. Radford et al. (2019); Petroni et al. (2019); Brown et al. (2020); Raffel et al. (2020); Schick and Schütze (2021b); Gao et al. (2021)). The advantage of this method is that, given a suite of appropriate prompts, a single LM trained in an entirely unsupervised fashion can be used to solve a great number of tasks (Brown et al., 2020; Sun et al., 2021). However, as with most conceptually enticing prospects, there is a catch – this method introduces the necessity for *prompt engineering*, finding the most appropriate prompt to allow a LM to solve the task at hand.

This survey attempts to organize the current state of knowledge in this rapidly developing field by providing an overview and formal definition of prompting methods (§2), and an overview of the pre-trained language models that use these prompts (§3). This is followed by in-depth discussion of prompting methods, from basics such as prompt engineering (§4) and answer engineering (§5) to more advanced concepts such as multi-prompt learning methods (§6) and prompt-aware training methods (§7). We then organize the various applications to which prompt-based learning methods have been applied, and discuss how they interact with the choice of prompting method (§8). Finally, we attempt to situate the current state of prompting methods in the research ecosystem, making connections to other research fields (§9), suggesting some current challenging problems that may be ripe for further research (§10), and performing a meta-analysis of current research trends (§11).

Finally, in order to help beginners who are interested in this field learn more effectively, we highlight some systematic resources about prompt learning (as well as pre-training) provided both within this survey and on companion websites:

-  : A website of prompt-based learning that contains: frequent updates to this survey, related slides, etc.
- Fig.1: A typology of important concepts for prompt-based learning.

[1] Even during this stage, there was some use of pre-trained models exemplified by word2vec (Mikolov et al., 2013b,a) and GloVe (Pennington et al., 2014), but they were used for only a limited portion of the final model parameters.

[2] This paradigm is less conducive to architectural exploration because (i) unsupervised pre-training allows models to learn with fewer structural priors, and (ii) as pre-training of models is time-consuming, experimenting with structural variants is costly.

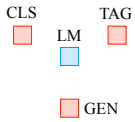
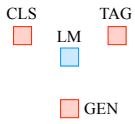
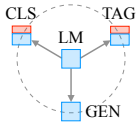
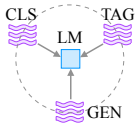
Paradigm	Engineering	Task Relation
a. Fully Supervised Learning (Non-Neural Network)	Features (e.g. word identity, part-of-speech, sentence length)	
b. Fully Supervised Learning (Neural Network)	Architecture (e.g. convolutional, recurrent, self-attentional)	
c. Pre-train, Fine-tune	Objective (e.g. masked language modeling, next sentence prediction)	
d. Pre-train, Prompt, Predict	Prompt (e.g. cloze, prefix)	

Table 1: Four paradigms in NLP. The "**engineering**" column represents the type of engineering to be done to build strong systems. The "**task relation**" column, shows the relationship between language models (LM) and other NLP tasks (CLS: classification, TAG: sequence tagging, GEN: text generation). : fully unsupervised training. : fully supervised training. : Supervised training combined with unsupervised training. indicates a textual prompt. Dashed lines suggest that different tasks can be connected by sharing parameters of pre-trained models. "LM→Task" represents *adapting LMs (objectives) to downstream tasks* while "Task→LM" denotes *adapting downstream tasks (formulations) to LMs*.

- Tab.7: A systematic and comprehensive comparison among different prompting methods.
- Tab.10: An organization of commonly-used prompts.
- Tab.12: A timeline of prompt-based research works.
- Tab.13: A systematic and comprehensive comparison among different pre-trained LMs.

2 A Formal Description of Prompting

2.1 Supervised Learning in NLP

In a traditional supervised learning system for NLP, we take an **input** $\boldsymbol{x}$, usually text, and predict an **output** $\boldsymbol{y}$ based on a model $P(\boldsymbol{y}|\boldsymbol{x};\theta)$. $\boldsymbol{y}$ could be a label, text, or other variety of output. In order to learn the parameters θ of this model, we use a dataset containing pairs of inputs and outputs, and train a model to predict this conditional probability. We will illustrate this with two stereotypical examples.

First, *text classification* takes an input text $\boldsymbol{x}$ and predicts a label y from a fixed label set $\mathcal{Y}$. To give an example, sentiment analysis (Pang et al., 2002; Socher et al., 2013) may take an input $\boldsymbol{x}$ ="I love this movie." and predict a label $y = ++$, out of a label set $\mathcal{Y} = \{++, +, \sim, -, --\}$.

Second, *conditional text generation* takes an input $\boldsymbol{x}$ and generates another text $\boldsymbol{y}$. One example is machine translation (Koehn, 2009), where the input is text in one language such as the Finnish $\boldsymbol{x}$ = "Hyvää huomenta." and the output is the English $\boldsymbol{y}$ = "Good morning"..

2.2 Prompting Basics

The main issue with supervised learning is that in order to train a model $P(\boldsymbol{y}|\boldsymbol{x};\theta)$, it is necessary to have supervised data for the task, which for many tasks cannot be found in large amounts. Prompt-based learning methods for NLP attempt to circumvent this issue by instead learning an LM that models the probability $P(\boldsymbol{x};\theta)$ of text $\boldsymbol{x}$ itself (details in §3) and using this probability to predict $\boldsymbol{y}$, reducing or obviating the need for large supervised datasets. In this section we lay out a mathematical description of the most fundamental form of prompting, which encompasses many works on prompting and can be expanded to cover others as well. Specifically, basic prompting predicts the highest-scoring $\hat{\boldsymbol{y}}$ in three steps.

Name	Notation	Example	Description
Input	$\boldsymbol{x}$	I love this movie.	One or multiple texts
Output	$\boldsymbol{y}$	++ (very positive)	Output label or text
Prompting Function	$f_{\text{prompt}}(\boldsymbol{x})$	`[X]` Overall, it was a `[Z]` movie.	A function that converts the input into a specific form by inserting the input $\boldsymbol{x}$ and adding a slot `[Z]` where answer $\boldsymbol{z}$ may be filled later.
Prompt	$\boldsymbol{x}'$	I love this movie. Overall, it was a `[Z]` movie.	A text where `[X]` is instantiated by input $\boldsymbol{x}$ but answer slot `[Z]` is not.
Filled Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$	I love this movie. Overall, it was a bad movie.	A prompt where slot `[Z]` is filled with any answer.
Answered Prompt	$f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}^*)$	I love this movie. Overall, it was a good movie.	A prompt where slot `[Z]` is filled with a true answer.
Answer	$\boldsymbol{z}$	"good", "fantastic", "boring"	A token, phrase, or sentence that fills `[Z]`

Table 2: Terminology and notation of prompting methods. $\boldsymbol{z}^*$ represents answers that correspond to true output $\boldsymbol{y}^*$.

2.2.1 Prompt Addition

In this step a *prompting function* $f_{\text{prompt}}(\cdot)$ is applied to modify the input text $\boldsymbol{x}$ into a *prompt* $\boldsymbol{x}' = f_{\text{prompt}}(\boldsymbol{x})$. In the majority of previous work (Kumar et al., 2016; McCann et al., 2018; Radford et al., 2019; Schick and Schütze, 2021a), this function consists of a two step process:

1. Apply a *template*, which is a textual string that has two slots: an *input slot* `[X]` for input $\boldsymbol{x}$ and an *answer slot* `[Z]` for an intermediate generated *answer* text $\boldsymbol{z}$ that will later be mapped into $\boldsymbol{y}$.
2. Fill slot `[X]` with the input text $\boldsymbol{x}$.

In the case of sentiment analysis where $\boldsymbol{x}$ ="I love this movie.", the template may take a form such as "`[X]` Overall, it was a `[Z]` movie.". Then, $\boldsymbol{x}'$ would become "I love this movie. Overall it was a `[Z]` movie." given the previous example. In the case of machine translation, the template may take a form such as "Finnish: `[X]` English: `[Z]`", where the text of the input and answer are connected together with headers indicating the language. We show more examples in Tab. 3

Notably, (1) the prompts above will have an empty slot to fill in for $\boldsymbol{z}$, either in the middle of the prompt or at the end. In the following text, we will refer to the first variety of prompt with a slot to fill in the middle of the text as a *cloze prompt*, and the second variety of prompt where the input text comes entirely before $\boldsymbol{z}$ as a *prefix prompt*. (2) In many cases these template words are not necessarily composed of natural language tokens; they could be virtual words (e.g. represented by numeric ids) which would be embedded in a continuous space later, and some prompting methods even generate continuous vectors directly (more in §4.3.2). (3) The number of `[X]` slots and the number of `[Z]` slots can be flexibly changed for the need of tasks at hand.

2.2.2 Answer Search

Next, we search for the highest-scoring text $\hat{\boldsymbol{z}}$ that maximizes the score of the LM. We first define $\mathcal{Z}$ as a set of permissible values for $\boldsymbol{z}$. $\mathcal{Z}$ could range from the entirety of the language in the case of generative tasks, or could be a small subset of the words in the language in the case of classification, such as defining $\mathcal{Z} = \{$"excellent", "good", "OK", "bad", "horrible"$\}$ to represent each of the classes in $\mathcal{Y} = \{$++, +, ~, -, --$\}$.

We then define a function $f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z})$ that fills in the location `[Z]` in prompt $\boldsymbol{x}'$ with the potential answer $\boldsymbol{z}$. We will call any prompt that has gone through this process as a *filled prompt*. Particularly, if the prompt is filled with a true answer, we will refer to it as an *answered prompt* (Tab. 2 shows an example). Finally, we search over the set of potential answers $\boldsymbol{z}$ by calculating the probability of their corresponding filled prompts using a pre-trained LM $P(\cdot; \theta)$

$$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z} \in \mathcal{Z}}{\text{search}}\, P(f_{\text{fill}}(\boldsymbol{x}', \boldsymbol{z}); \theta). \tag{1}$$

This search function could be an *argmax* search that searches for the highest-scoring output, or *sampling* that randomly generates outputs following the probability distribution of the LM.

2.2.3 Answer Mapping

Finally, we would like to go from the highest-scoring *answer* $\hat{\boldsymbol{z}}$ to the highest-scoring *output* $\hat{\boldsymbol{y}}$. This is trivial in some cases, where the answer itself is the output (as in language generation tasks such as translation), but there

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. ...

Table 3: Examples of *input*, *template*, and *answer* for different tasks. In the **Type** column, "CLS" is an abbreviation for "classification". In the **Task** column, "NLI" and "NER" are abbreviations for "natural language inference" (Bowman et al., 2015) and "named entity recognition" (Tjong Kim Sang and De Meulder, 2003) respectively.

are also other cases where multiple answers could result in the same output. For example, one may use multiple different sentiment-bearing words (e.g. "excellent", "fabulous", "wonderful") to represent a single class (e.g. "++"), in which case it is necessary to have a mapping between the searched answer and the output value.

2.3 Design Considerations for Prompting

Now that we have our basic mathematical formulation, we elaborate a few of the basic design considerations that go into a prompting method, which we will elaborate in the following sections:

- **Pre-trained Model Choice:** There are a wide variety of pre-trained LMs that could be used to calculate $P(\boldsymbol{x}; \theta)$. In §3 we give a primer on pre-trained LMs, specifically from the dimensions that are important for interpreting their utility in prompting methods.
- **Prompt Engineering:** Given that the prompt specifies the task, choosing a proper prompt has a large effect not only on the accuracy, but also on which task the model performs in the first place. In §4 we discuss methods to choose which prompt we should use as $f_{\text{prompt}}(\boldsymbol{x})$.
- **Answer Engineering:** Depending on the task, we may want to design $\mathcal{Z}$ differently, possibly along with the mapping function. In §5 we discuss different ways to do so.
- **Expanding the Paradigm:** As stated above, the above equations represent only the simplest of the various underlying frameworks that have been proposed to do this variety of prompting. In §6 we discuss ways to expand this underlying paradigm to further improve results or applicability.
- **Prompt-based Training Strategies:** There are also methods to train parameters, either of the prompt, the LM, or both. In §7, we summarize different strategies and detail their relative advantages.

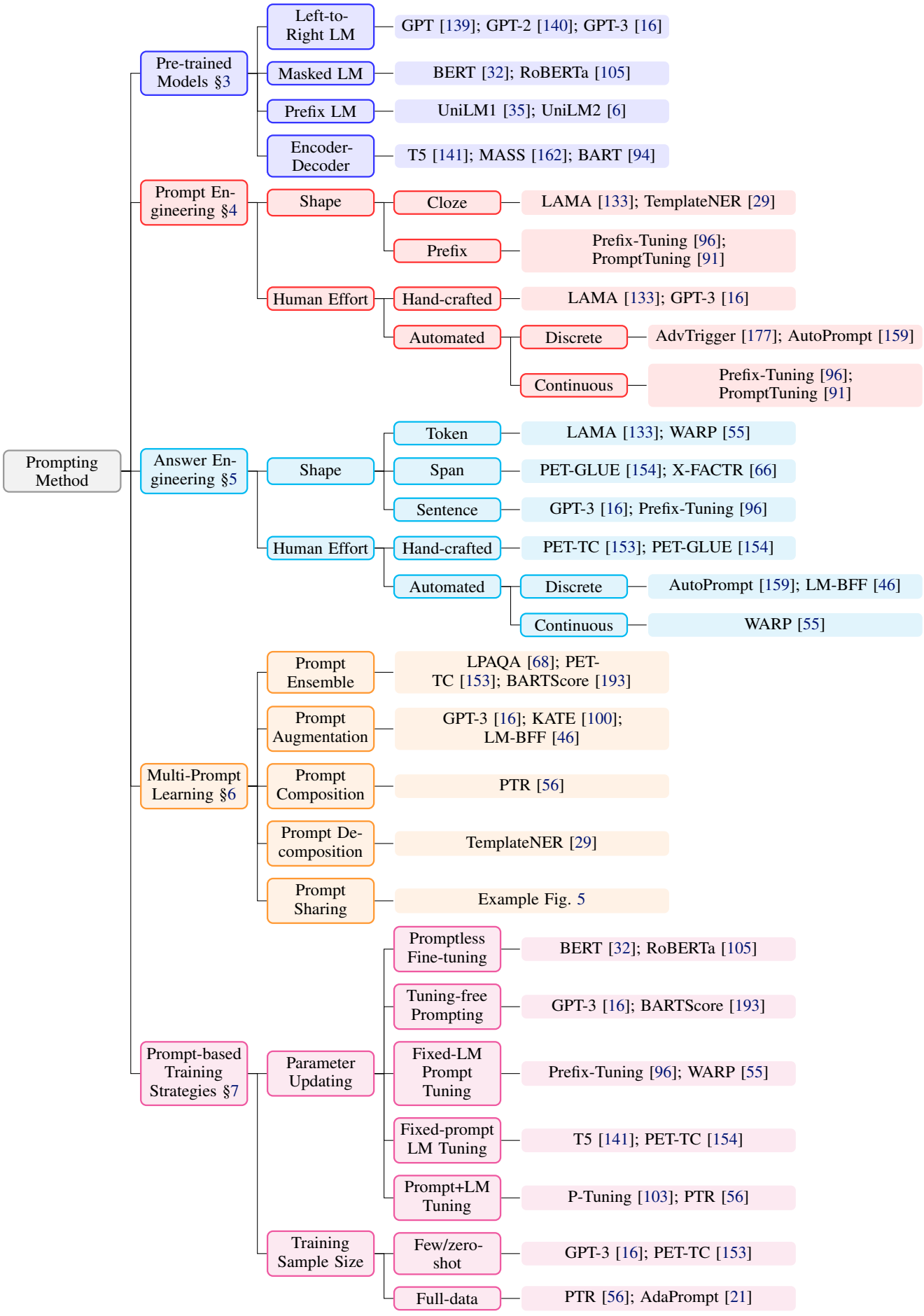


Figure 1: Typology of prompting methods.

3 Pre-trained Language Models

Given the large impact that pre-trained LMs have had on NLP in the pre-train and fine-tune paradigm, there are already a number of high-quality surveys that interested readers where interested readers can learn more (Raffel et al., 2020; Qiu et al., 2020; Xu et al., 2021; Doddapaneni et al., 2021). Nonetheless, in this chapter we present a systematic view of various pre-trained LMs which (i) organizes them along various axes in a more systematic way, (ii) particularly focuses on aspects salient to prompting methods. Below, we will detail them through the lens of *main training objective*, *type of text noising*, *auxiliary training objective*, *attention mask*, *typical architecture*, and *preferred application scenarios*. We describe each of these objectives below, and also summarize a number of pre-trained LMs along each of these axes in Tab. 13 in the appendix.

3.1 Training Objectives

The main training objective of a pre-trained LM almost invariably consists of some sort of objective predicting the probability of text $\boldsymbol{x}$.

Standard Language Model (SLM) objectives do precisely this, training the model to optimize the probability $P(\boldsymbol{x})$ of text from a training corpus (Radford et al., 2019). In these cases, the text is generally predicted in an *autoregressive* fashion, predicting the tokens in the sequence one at a time. This is usually done from left to right (as detailed below), but can be done in other orders as well.

A popular alternative to standard LM objectives are *denoising* objectives, which apply some noising function $\tilde{\boldsymbol{x}} = f_{\text{noise}}(\boldsymbol{x})$ to the input sentence (details in the following subsection), then try to predict the original input sentence given this noised text $P(\boldsymbol{x}|\tilde{\boldsymbol{x}})$. There are two common flavors of these objectives:

Corrupted Text Reconstruction (CTR) These objectives restore the processed text to its uncorrupted state by calculating loss over *only* the noised parts of the input sentence.

Full Text Reconstruction (FTR) These objectives reconstruct the text by calculating the loss over the *entirety* of the input texts whether it has been noised or not (Lewis et al., 2020a).

The main training objective of the pre-trained LMs plays an important role in determining its applicability to particular prompting tasks. For example, left-to-right autoregressive LMs may be particularly suitable for prefix prompts, whereas reconstruction objectives may be more suitable for cloze prompts. In addition, models trained with standard LM and FTR objectives may be more suitable for tasks regarding text generation, whereas other tasks such as classification can be formulated using models trained with any of these objectives.

In addition to the main training objectives above, a number of *auxiliary objectives* have been engineered to further improve models' ability to perform certain varieties of downstream tasks. We list some commonly-used auxiliary objectives in Appendix A.2.

3.2 Noising Functions

In training objectives based on reconstruction, the specific type of corruption applied to obtain the noised text $\tilde{\boldsymbol{x}}$ has an effect on the efficacy of the learning algorithm. In addition, prior knowledge can be incorporated by controlling the type of noise, e.g. the noise could focus on entities of a sentence, which allows us to learn a pre-trained model with particularly high predictive performance for entities. In the following, we introduce several types of noising functions, and give detailed examples in Tab. 4.

Operation	Element	Original Text	Corrupted Text
Mask	one token	Jane will move to New York .	Jane will `[Z]` to New York .
	two tokens	Jane will move to New York .	Jane will `[Z]` `[Z]` New York .
	one entity	Jane will move to New York .	Jane will move to `[Z]` .
Replace	one token	Jane will move to New York .	Jane will move `[X]` New York .
	two tokens	Jane will move to New York .	Jane will move `[X]` `[Y]` York .
	one entity	Jane will move to New York .	Jane will move to `[X]` .
Delete	one token	Jane ~~will~~ move to New York .	Jane move to New York .
	two token	Jane ~~will move~~ to New York .	Jane to New York .
Permute	token	Jane will move to New York .	New York . Jane will move to
Rotate	none	Jane will move to New York .	to New York . Jane will move
Concatenation	two languages	Jane will move to New York .	Jane will move to New York . `[/s]` 简将搬到纽约。

Table 4: Detailed examples for different noising operations.

Masking (e.g. Devlin et al. (2019)) The text will be masked in different levels, replacing a token or multi-token span with a special token such as `[MASK]`. Notably, masking can either be random from some distribution or specifically designed to introduce prior knowledge, such as the above-mentioned example of masking entities to encourage the model to be good at predicting entities.

Replacement (e.g. Raffel et al. (2020)) Replacement is similar to masking, except that the token or multi-token span is not replaced with a `[MASK]` but rather another token or piece of information (e.g., an image region (Su et al., 2020)).

Deletion (e.g. Lewis et al. (2020a)) Tokens or multi-token spans will be deleted from a text without the addition of `[MASK]` or any other token. This operation is usually used together with the FTR loss.

Permutation (e.g. Liu et al. (2020a)) The text is first divided into different spans (tokens, sub-sentential spans, or sentences), and then these spans are be permuted into a new text.

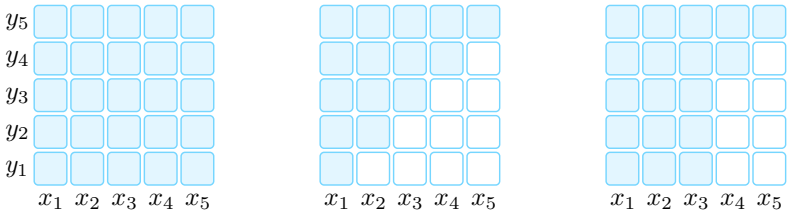
3.3 Directionality of Representations

A final important factor that should be considered in understanding pre-trained LMs and the difference between them is the directionality of the calculation of representations. In general, there are two widely used ways to calculate such representations:

Left-to-Right The representation of each word is calculated based on the word itself and all previous words in the sentence. For example, if we have a sentence "This is a good movie", the representation of the word "good" would be calculated based on previous words. This variety of factorization is particularly widely used when calculating standard LM objectives or when calculating the output side of an FTR objective, as we discuss in more detail below.

Bidirectional The representation of each word is calculated based on all words in the sentence, including words to the left of the current word. In the example above, "good" would be influenced by all words in the sentence, even the following "movie".

In addition to the two most common directionalities above, it is also possible to mix the two strategies together in a single model (Dong et al., 2019; Bao et al., 2020), or perform conditioning of the representations in a randomly permuted order (Yang et al., 2019), although these strategies are less widely used. Notably, when implementing these strategies within a neural model, this conditioning is generally implemented through *attention masking*, which masks out the values in an attentional model (Bahdanau et al., 2014), such as the popular Transformer architecture (Vaswani et al., 2017). Some examples of such attention masks are shown in Figure 2.



(a) Full. (b) Diagonal. (c) Mixture.

Figure 2: Three popular attention mask patterns, where the subscript t indicates the t-th timestep. A shaded box at (i, j) indicates that the attention mechanism is allowed to attend to the input element i at output time step j. A white box indicates that the attention mechanism is not allowed to attend to the corresponding i and j combination.

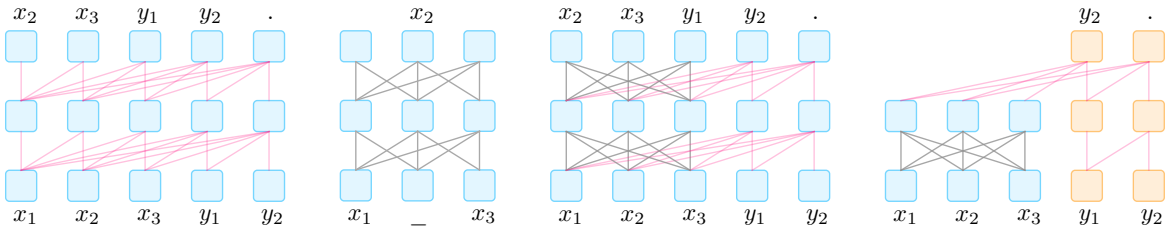
3.4 Typical Pre-training Methods

With the above concepts in mind, we introduce four popular pre-training methods, resulting from diverse combinations of objective, noising function, and directionality. These are described below, and summarized in Fig. 3 and Tab. 5.

3.4.1 Left-to-Right Language Model

Left-to-right LMs (L2R LMs), a variety of *auto-regressive LM*, predict the upcoming words or assign a probability $P(\boldsymbol{x})$ to a sequence of words $\boldsymbol{x} = x_1, \cdots, x_n$ (Jurafsky and Martin, 2021). The probability is commonly broken down using the chain rule in a left-to-right fashion: $P(\boldsymbol{x}) = P(x_1) \times \cdots P(x_n|x_1 \cdots x_{n-1})$.[3]

[3]Similarly, a right-to-left LM can predict preceding words based on the future context, such as $P(x_i|x_{i+1}, \cdots, x_n)$.



(a) Left-to-right LM. (b) Masked LM. (c) Prefix LM. (d) Encoder-Decoder.

Figure 3: Typical paradigms of pre-trained LMs.

Example & Applicable Scenario

Left-to-right LMs have been standard since their proposal by Markov in 1913 (Markov, 2006), and have been used continuously since then in both count-based (Goodman, 2001) and neural forms (Bengio et al., 2003; Mikolov et al., 2010; Radford and Narasimhan, 2018). Representative examples of modern pre-trained left-to-right LMs include GPT-3 (Brown et al., 2020), and GPT-Neo (Black et al., 2021).
L2R pre-trained LMs are also the popular backbone that many prompting methods adopt (Radford et al., 2019; Brown et al., 2020) . One practical reason for this is that many such models are large (PanGu-α (Zeng et al., 2021), Ernie-3 (Sun et al., 2021)) and ponderous to train, or not even available publicly. Thus using these models in the pre-train and fine-tune regimen is often not possible.

LMs	$\boldsymbol{x}$			$\boldsymbol{y}$			Application
	Mask	Noise	Main Obj.	Mask	Noise	Main Obj.	
L2R	Diagonal	None	SLM	-	-	-	NLU & NLG
Mask	Full	Mask	CTR	-	-	-	NLU
Prefix	Full	Any	CTR	Diagonal	None	SLM	NLU & NLG
En-De	Full	Any	None†	Diagonal	None	FTR/CRT	NLU & NLG

Table 5: Typical architectures for pre-trained LMs. $\boldsymbol{x}$ and $\boldsymbol{y}$ represent text to be encoded and decoded, respectively. **SLM**: Standard language model. **CTR**: Corrupted text reconstruction. **FTR**: Full text reconstruction. †: Encoder-decoder architectures usually apply objective functions to the decoder only.

3.4.2 Masked Language Models

While autoregressive language models provide a powerful tool for modeling the probability of text, they also have disadvantages such as requiring representations be calculated from left-to-right. When the focus is shifted to generating the optimal representations for down-stream tasks such as classification, many other options become possible, and often preferable. One popular bidirectional objective function used widely in representation learning is the *masked language model* (MLM; Devlin et al. (2019)), which aims to predict masked text pieces based on surrounded context. For example, $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represents the probability of the word x_i given the surrounding context.

Example & Applicable Scenario

Representative pre-trained models using MLMs include: BERT (Devlin et al., 2019), ERNIE (Zhang et al., 2019; Sun et al., 2019b) and many variants. In prompting methods, MLMs are generally most suitable for natural language understanding or analysis tasks (e.g., text classification, natural language inference , and extractive question answering). These tasks are often relatively easy to be reformulated into cloze problems, which are consistent with the training objectives of the MLM. Additionally, MLMs have been a pre-trained model of choice when exploring methods that combine prompting with fine-tuning, elaborated further in §7.

3.4.3 Prefix and Encoder-Decoder

For conditional text generation tasks such as translation and summarization where an input text $\boldsymbol{x} = x_1, \cdots, x_n$ is given and the goal is to generate target text $\boldsymbol{y}$, we need a pre-trained model that is both capable of encoding the input text and generating the output text. There are two popular architectures for this purpose that share a common

thread of (1) using an encoder with fully-connected mask to encode the source $\boldsymbol{x}$ first and then (2) decode the target $\boldsymbol{y}$ auto-regressively (from the left to right).

Prefix Language Model The prefix LM is a left-to-right LM that decodes $\boldsymbol{y}$ conditioned on a prefixed sequence $\boldsymbol{x}$, which is encoded by the *same* model parameters but with a fully-connected mask. Notably, to encourage the prefix LM to learn better representations of the input, a corrupted text reconstruction objective is usually applied over $\boldsymbol{x}$, in addition to a standard conditional language modeling objective over $\boldsymbol{y}$.

Encoder-decoder The encoder-decoder model is a model that uses a left-to-right LM to decode $\boldsymbol{y}$ conditioned on a *separate* encoder for text $\boldsymbol{x}$ with a fully-connected mask; the parameters of the encoder and decoder are not shared. Similarly to the prefix LM, diverse types of noising can be applied to the input $\boldsymbol{x}$.

Example & Applicable Scenario

Prefix LMs have been explored in UniLM 1-2 (Dong et al., 2019; Bao et al., 2020) and ERNIE-M (Ouyang et al., 2020) while encoder-decoder models are widely used in pre-trained models such as T5 (Raffel et al., 2020), BART (Lewis et al., 2020a), MASS (Song et al., 2019) and their variants.
Pre-trained models with prefix LMs and encoder-decoder paradigms can be naturally used to text generation tasks with (Dou et al., 2021) or without (Yuan et al., 2021a; Liu and Liu, 2021) prompting using input texts. However, recent studies reveal that other non-generation tasks, such as information extraction (Cui et al., 2021), question answering (Khashabi et al., 2020) , and text generation evaluation (Yuan et al., 2021b) can be reformulated a generation problems by providing appropriate prompts. Therefore, prompting methods (i) broaden the applicability of these generation-oriented pre-trained models. For example, pre-trained models like BART are less used in NER while prompting methods make BART applicable, and (ii) breaks the difficulty of unified modelling among different tasks (Khashabi et al., 2020).

4 Prompt Engineering

Prompt engineering is the process of creating a prompting function $f_{\text{prompt}}(\boldsymbol{x})$ that results in the most effective performance on the downstream task. In many previous works, this has involved *prompt template engineering*, where a human engineer or algorithm searches for the best template for each task the model is expected to perform. As shown in the "Prompt Engineering" section of Fig.1, one must first consider the *prompt shape*, and then decide whether to take a *manual* or *automated* approach to create prompts of the desired shape, as detailed below.

4.1 Prompt Shape

As noted above, there are two main varieties of prompts: *cloze prompts* (Petroni et al., 2019; Cui et al., 2021), which fill in the blanks of a textual string, and *prefix prompts* (Li and Liang, 2021; Lester et al., 2021), which continue a string prefix. Which one is chosen will depend both on the task and the model that is being used to solve the task. In general, for tasks regarding generation, or tasks being solved using a standard auto-regressive LM, prefix prompts tend to be more conducive, as they mesh well with the left-to-right nature of the model. For tasks that are solved using masked LMs, cloze prompts are a good fit, as they very closely match the form of the pre-training task. Full text reconstruction models are more versatile, and can be used with either cloze or prefix prompts. Finally, for some tasks regarding multiple inputs such as *text pair classification*, prompt templates must contain space for two inputs, `[X1]` and `[X2]`, or more.

4.2 Manual Template Engineering

Perhaps the most natural way to create prompts is to manually create intuitive templates based on human introspection. For example, the seminal LAMA dataset (Petroni et al., 2019) provides manually created cloze templates to probe knowledge in LMs. Brown et al. (2020) create manually crafted prefix prompts to handle a wide variety of tasks, including question answering, translation, and probing tasks for common sense reasoning. Schick and Schütze (2020, 2021a,b) use pre-defined templates in a few-shot learning setting on text classification and conditional text generation tasks.

4.3 Automated Template Learning

While the strategy of manually crafting templates is intuitive and does allow solving various tasks with some degree of accuracy, there are also several issues with this approach: (1) creating and experimenting with these prompts is an art that takes time and experience, particularly for some complicated tasks such as semantic parsing (Shin et al., 2021); (2) even experienced prompt designers may fail to manually discover optimal prompts (Jiang et al., 2020c).

To address these problems, a number of methods have been proposed to automate the template design process. In particular, the automatically induced prompts can be further separated into *discrete prompts*, where the prompt is an

actual text string, and *continuous prompts*, where the prompt is instead described directly in the embedding space of the underlying LM.

One other orthogonal design consideration is whether the prompting function $f_{\text{prompt}}(\boldsymbol{x})$ is *static*, using essentially the same prompt template for each input, or *dynamic*, generating a custom template for each input. Both static and dynamic strategies have been used for different varieties of discrete and continuous prompts, as we will mention below.

4.3.1 Discrete Prompts

Works on discovering *discrete prompts* (a.k.a *hard prompts*) automatically search for templates described in a discrete space, usually corresponding to natural language phrases. We detail several methods that have been proposed for this below:

D1: Prompt Mining Jiang et al. (2020c)'s MINE approach is a mining-based method to automatically find templates given a set of training inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y}$. This method scrapes a large text corpus (e.g. Wikipedia) for strings containing $\boldsymbol{x}$ and $\boldsymbol{y}$, and finds either the *middle words* or *dependency paths* between the inputs and outputs. Frequent middle words or dependency paths can serve as a template as in "`[X]` middle words `[Z]`".

D2: Prompt Paraphrasing Paraphrasing-based approaches take in an existing seed prompt (e.g. manually constructed or mined), and paraphrases it into a set of other candidate prompts, then selects the one that achieves the highest training accuracy on the target task. This paraphrasing can be done in a number of ways, including using round-trip translation of the prompt into another language then back (Jiang et al., 2020c), using replacement of phrases from a thesaurus (Yuan et al., 2021b), or using a neural prompt rewriter specifically optimized to improve accuracy of systems using the prompt (Haviv et al., 2021). Notably, Haviv et al. (2021) perform paraphrasing *after* the input $\boldsymbol{x}$ is input into the prompt template, allowing a different paraphrase to be generated for each individual input.

D3: Gradient-based Search Wallace et al. (2019a) applied a gradient-based search over actual tokens to find short sequences that can trigger the underlying pre-trained LM to generate the desired target prediction. This search is done in an iterative fashion, stepping through tokens in the prompt . Built upon this method, Shin et al. (2020) automatically search for template tokens using downstream application training samples and demonstrates strong performance in prompting scenarios.

D4: Prompt Generation Other works treat the generation of prompts as a text generation task and use standard natural language generation models to perform this task. For example, Gao et al. (2021) introduce the seq2seq pre-trained model T5 into the template search process. Since T5 has been pre-trained on a task of filling in missing spans, they use T5 to generate template tokens by (1) specifying the position to insert template tokens within a template[4] (2) provide training samples for T5 to decode template tokens. Ben-David et al. (2021) propose a domain adaptation algorithm that trains T5 to generate unique domain relevant features (DRFs; a set of keywords that characterize domain information) for each input. Then those DRFs can be concatenated with the input to form a template and be further used by downstream tasks.

D5: Prompt Scoring Davison et al. (2019) investigate the task of knowledge base completion and design a template for an input (head-relation-tail triple) using LMs. They first hand-craft a set of templates as potential candidates, and fill the input and answer slots to form a filled prompt. They then use a unidirectional LM to score those filled prompts, selecting the one with the highest LM probability. This will result in custom template for each individual input.

4.3.2 Continuous Prompts

Because the purpose of prompt construction is to find a method that allows an LM to effectively perform a task, rather than being for human consumption, it is not necessary to limit the prompt to human-interpretable natural language. Because of this, there are also methods that examine *continuous prompts* (a.k.a. *soft prompts*) that perform prompting directly in the embedding space of the model. Specifically, continuous prompts remove two constraints: (1) relax the constraint that the embeddings of template words be the embeddings of natural language (e.g., English) words. (2) Remove the restriction that the template is parameterized by the pre-trained LM's parameters. Instead, templates have their own parameters that can be tuned based on training data from the downstream task. We highlight several representative methods below.

[4]The number of template tokens do not need to be pre-specified since T5 can decode multiple tokens at a masked position.

C1: Prefix Tuning Prefix Tuning (Li and Liang, 2021) is a method that prepends a sequence of continuous task-specific vectors to the input, while keeping the LM parameters frozen. Mathematically, this consists of optimizing over the following log-likelihood objective given a trainable prefix matrix M_ϕ and a fixed pre-trained LM parameterized by θ.

$$\max_{\phi} \log P(\boldsymbol{y}|\boldsymbol{x}; \theta; \phi) = \max_{\phi} \sum_{y_i} \log P(y_i|h_{<i}; \theta; \phi) \tag{2}$$

In Eq. 2, $h_{<i} = [h_{<i}^{(1)}; \cdots; h_{<i}^{(n)}]$ is the concatenation of all neural network layers at time step i. It is copied from M_ϕ directly if the corresponding time step is within the prefix (h_i is $M_\phi[i]$), otherwise it is computed using the pre-trained LM.

Experimentally, Li and Liang (2021) observe that such continuous prefix-based learning is more sensitive to different initialization in low-data settings than the use of discrete prompts with real words. Similarly, Lester et al. (2021) prepend the input sequence with special tokens to form a template and tune the embeddings of these tokens directly. Compared to Li and Liang (2021)'s method, this adds fewer parameters as it doesn't introduce additional tunable parameters within each network layer. Tsimpoukelli et al. (2021) train a vision encoder that encodes an image into a sequence of embeddings that can be used to prompt a frozen auto-regressive LM to generate the appropriate caption. They show that the resulting model can perform few-shot learning for vision-language tasks such as visual question answering etc. Different from the above two works, the prefix used in (Tsimpoukelli et al., 2021) is sample-dependent, namely a representation of input images, instead of a task embedding.

C2: Tuning Initialized with Discrete Prompts There are also methods that initialize the search for a continuous prompt using a prompt that has already been created or discovered using discrete prompt search methods. For example, Zhong et al. (2021b) first define a template using a discrete search method such as AUTOPROMPT (Shin et al., 2020)'s, initialize virtual tokens based on this discovered prompt, then fine-tune the embeddings to increase task accuracy. This work found that initializing with manual templates can provide a better starting point for the search process. Qin and Eisner (2021) propose to learn a mixture of soft templates for each input where the weights and parameters for each template are jointly learned using training samples. The initial set of templates they use are either manually crafted ones or those obtained using the "prompt mining" method. Similarly, Hambardzumyan et al. (2021) introduce the use of a continuous template whose shape follows a manual prompt template.

C3: Hard-Soft Prompt Hybrid Tuning Instead of using a purely learnable prompt template, these methods insert some tunable embeddings into a hard prompt template. Liu et al. (2021b) propose "P-tuning", where continuous prompts are learned by inserting trainable variables into the embedded input. To account for interaction between prompt tokens, they represent prompt embeddings as the output of a BiLSTM (Graves et al., 2013). P-tuning also introduces the use of task-related anchor tokens (such as "capital" in relation extraction) within the template for further improvement. These anchor tokens are not tuned during training. Han et al. (2021) propose prompt tuning with rules (PTR), which uses manually crafted sub-templates to compose a complete template using logic rules. To enhance the representation ability of the resulting template, they also insert several virtual tokens whose embeddings can be tuned together with the pre-trained LMs parameters using training samples. The template tokens in PTR contain both actual tokens and virtual tokens. Experiment results demonstrate the effectiveness of this prompt design method in relation classification tasks.

5 Answer Engineering

In contrast to prompt engineering, which designs appropriate inputs for prompting methods, *answer engineering* aims to search for an answer space $\mathcal{Z}$ and a map to the original output $\mathcal{Y}$ that results in an effective predictive model. Fig.1's "Answer Engineering" section illustrates two dimensions that must be considered when performing answer engineering: deciding the *answer shape* and choosing an *answer design method*.

5.1 Answer Shape

The shape of an answer characterizes its granularity. Some common choices include:

- **Tokens:** One of the tokens in the pre-trained LM's vocabulary, or a subset of the vocabulary.
- **Span:** A short multi-token span. These are usually used together with cloze prompts.
- **Sentence:** A sentence or document. These are commonly used with prefix prompts.

In practice, how to choose the shape of acceptable answers depends on the task we want to perform. Token or text-span answer spaces are widely used in classification tasks (e.g. sentiment classification; Yin et al. (2019)), but also other tasks such as relation extraction (Petroni et al., 2019) or named entity recognition (Cui et al., 2021). Longer phrasal or sentential answers are often used in language generation tasks (Radford et al., 2019), but also

used in other tasks such as multiple-choice question answering (where the scores of multiple phrases are compared against each-other; Khashabi et al. (2020)).

5.2 Answer Space Design Methods

The next question to answer is how to design the appropriate answer space $\mathcal{Z}$, as well as the mapping to the output space $\mathcal{Y}$ if the answers are not used as the final outputs.

5.2.1 Manual Design

In manual design, the space of potential answers $\mathcal{Z}$ and its mapping to $\mathcal{Y}$ are crafted manually by an interested system or benchmark designer. There are a number of strategies that can be taken to perform this design.

Unconstrained Spaces In many cases, the answer space $\mathcal{Z}$ is the space of all tokens (Petroni et al., 2019), fixed-length spans (Jiang et al., 2020a), or token sequences (Radford et al., 2019). In these cases, it is most common to directly map answer $\boldsymbol{z}$ to the final output $\boldsymbol{y}$ using the identity mapping.

Constrained Spaces However, there are also cases where the space of possible outputs is constrained. This is often performed for tasks with a limited label space such as text classification or entity recognition, or multiple-choice question answering. To give some examples, Yin et al. (2019) manually design lists of words relating to relevant topics ("health", "finance", "politics", "sports", etc.), emotions ("anger", "joy", "sadness", "fear", etc.), or other aspects of the input text to be classified. Cui et al. (2021) manually design lists such as "person", "location", etc. for NER tasks. In these cases, it is necessary to have a mapping between the answer $\mathcal{Z}$ and the underlying class $\mathcal{Y}$.

With regards to multiple-choice question answering, it is common to use an LM to calculate the probability of an output among multiple choices, with Zweig et al. (2012) being an early example.

5.2.2 Discrete Answer Search

As with manually created prompts, it is possible that manually created answers are sub-optimal for getting the LM to achieve ideal prediction performance. Because of this, there is some work on automatic answer search, albeit less than that on searching for ideal prompts. These work on both discrete answer spaces (this section) and continuous answer spaces (the following).

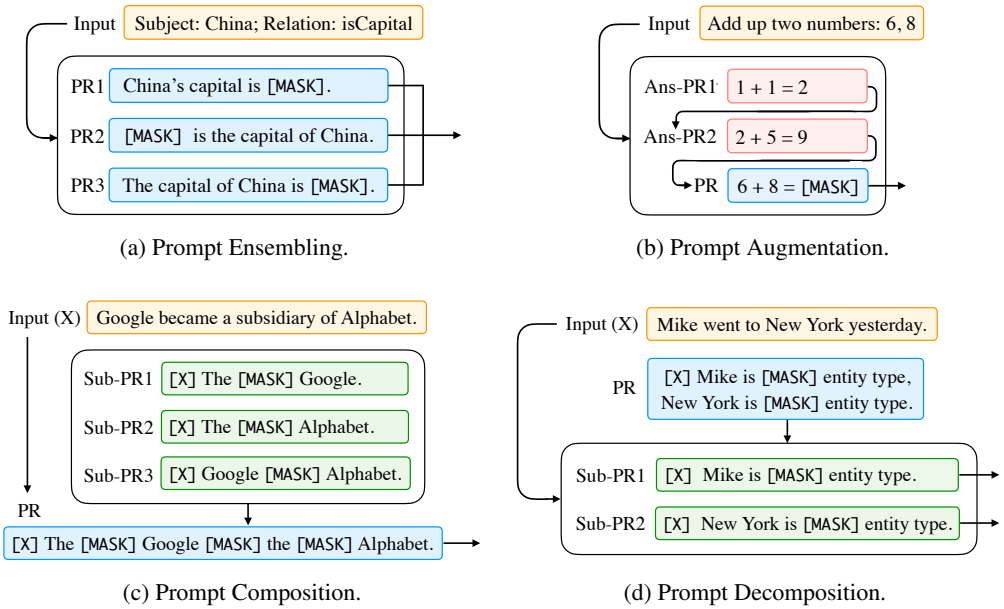
Answer Paraphrasing These methods start with an initial answer space $\mathcal{Z}'$, and then use paraphrasing to expand this answer space to broaden its coverage (Jiang et al., 2020b). Given a pair of answer and output $\langle \boldsymbol{z}', \boldsymbol{y} \rangle$, we define a function that generates a paraphrased set of answers $\text{para}(\boldsymbol{z}')$. The probability of the final output is then defined as the marginal probability *all* of the answers in this paraphrase set $P(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{z} \in \text{para}(\boldsymbol{z}')} P(\boldsymbol{z}|\boldsymbol{x})$. This paraphrasing can be performed using any method, but Jiang et al. (2020b) specifically use a back-translation method, first translating into another language then back to generate a list of multiple paraphrased answers.

Prune-then-Search In these methods, first, an initial pruned answer space of several plausible answers $\mathcal{Z}'$ is generated, and then an algorithm further searches over this pruned space to select a final set of answers. Note that in some of the papers introduced below, they define a function from label $\boldsymbol{y}$ to a single answer token $\boldsymbol{z}$, which is often called a *verbalizer* (Schick and Schütze, 2021a). Schick and Schütze (2021a); Schick et al. (2020) find tokens containing at least two alphabetic characters that are frequent in a large unlabeled dataset. In the search step, they iteratively compute a word's suitability as a representative answer $\boldsymbol{z}$ for a label $\boldsymbol{y}$ by maximizing the likelihood of the label over training data. Shin et al. (2020) learn a logistic classifier using the contextualized representation of the `[Z]` token as input. In the search step, they select the top-k tokens that achieve the highest probability score using the learned logistic classifier in the first step. Those selected tokens will form the answer. Gao et al. (2021) first construct a pruned search space $\mathcal{Z}'$ by selecting top-k vocabulary words based on their generation probability at the `[Z]` position determined by training samples. Then the search space is further pruned down by only selecting a subset of words within $\mathcal{Z}'$ based on their zero-shot accuracy on the training samples. (2) In the search step, they fine-tune the LM with fixed templates together with every answer mapping using training data and select the best label word as the answer based on the accuracy on the development set.

Label Decomposition When performing relation extraction, Chen et al. (2021b) automatically decompose each relation label into its constituent words and use them as an answer. For example, for the relation `per:city_of_death`, the decomposed label words would be {`person`, `city`, `death`}. The probability of the answer span will be calculated as the sum of each token's probability.

5.2.3 Continuous Answer Search

Very few works explore the possibility of using soft answer tokens which can be optimized through gradient descent. Hambardzumyan et al. (2021) assign a virtual token for each class label and optimize the token embedding for each



(a) Prompt Ensembling.

(b) Prompt Augmentation.

(c) Prompt Composition.

(d) Prompt Decomposition.

Figure 4: Different multi-prompt learning strategies. We use different colors to differentiate different components as follows. " " for input text, " " for prompt, " " for answered prompt. " " for sub-prompt. We use the following abbreviations. "PR" for prompt, "Ans-PR" for answered prompt, "Sub-PR" for sub-prompt.

class together with prompt token embeddings. Since the answer tokens are optimized directly in the embedding space, they do not make use of the embeddings learned by the LM and instead learn an embedding from scratch for each label.

6 Multi-Prompt Learning

The prompt engineering methods we discussed so far focused mainly on constructing a *single* prompt for an input. However, a significant body of research has demonstrated that the use of multiple prompts can further improve the efficacy of prompting methods, and we will call these methods *multi-prompt learning* methods. In practice, there are several ways to extend the single prompt learning to the use multiple prompts, which have a variety of motivations. We summarize representative methods in the "Multi-prompt Learning" section of Fig.1 as well as Fig.4.

6.1 Prompt Ensembling

Prompt ensembling is the process of using multiple *unanswered* prompts for an input at inference time to make predictions. An example is shown in Fig. 4-(a). The multiple prompts can either be discrete prompts or continuous prompts.[5] This sort of prompt ensembling can (1) leverage the complementary advantages of different prompts, (2) alleviate the cost of prompt engineering, since choosing one best-performing prompt is challenging, (3) stabilize performance on downstream tasks.

Prompt ensembling is connected to ensembling methods that are used to combine together multiple systems, which have a long history in machine learning (Ting and Witten, 1997; Zhou et al., 2002; Duh et al., 2011). Current research also borrows ideas from these works to derive effective ways for prompt ensembling, as described below.

Uniform averaging The most intuitive way to combine the predictions when using multiple prompts is to take the average of probabilities from different prompts. Concretely, this indicates that $P(\boldsymbol{z}|\boldsymbol{x}) \coloneqq \frac{1}{K}\sum_i^K P(\boldsymbol{z}|f_{\text{prompt},i}(\boldsymbol{x}))$ where $f_{\text{prompt},i}(\cdot)$ is the ith prompt in the prompt ensemble. Jiang et al. (2020c) first filter their prompts by selecting K prompts that achieve the highest accuracy on the training set, and then use the average log probabilities obtained from the top K prompts to calculate the probability for a single token at `[Z]` position when performing factual probing tasks. Schick and Schütze (2021a) also try a simple average when using an ensemble model to annotate an unlabeled dataset. When performing text generation evaluation, Yuan et al. (2021b) formulates this task as a text generation problem and take the average of the final generation scores obtained using different prompts.

Weighted averaging Simple uniform averaging of results from multiple prompts is easy to implement, but can also be suboptimal given that some prompts are more performant than others. To account for this, some works also

[5]Multiple continuous prompts are typically learned by using different initializations or different random seeds.

explore to use of weighted averages for prompt ensembling where each prompt is associated with a weight. The weights are typically pre-specified based on prompt performance or optimized using a training set. For example, Jiang et al. (2020c) learn the weight for each prompt by maximizing the probability of the target output over training data. Qin and Eisner (2021) use the same approach except that the weight for each prompt is optimized together with soft prompt parameters. Besides, Qin and Eisner (2021) also introduce a data-dependent weighting strategy where the probability of the input appearing in that prompt is considered in weighting different prompts as well. Schick and Schütze (2021a,b) set the weight for each prompt proportional to the accuracy on the training set before training.

Majority voting For classification tasks, majority voting can also be used to combine the results from different prompts (Lester et al., 2021; Hambardzumyan et al., 2021).

Knowledge distillation An ensemble of deep learning models can typically improve the performance, and this superior performance can be distilled into a single model using knowledge distillation (Allen-Zhu and Li, 2020). To incorporate this idea, Schick and Schütze (2021a,b, 2020) train a separate model for each manually-created template-answer pair, and use the ensemble of them to annotate an unlabeled dataset. Then the final model is trained to distill the knowledge from the annotated dataset. Gao et al. (2021) use a similar ensemble method on their automatically generated templates.

Prompt ensembling for text generation There is relatively little work on prompt ensembling for generation tasks (i.e. tasks where the answers is a string of tokens instead of a single one). A simple way to perform ensembling in this case is to use standard methods that generate the output based on the ensembled probability of the next word in the answer sequence $P(z_t|\boldsymbol{x}, z_{<t}) \coloneqq \frac{1}{K}\sum_i^K P(z_t|f_{\text{prompt},i}(\boldsymbol{x}), z_{<t})$. In contrast, Schick and Schütze (2020) train a separate model for each prompt $f_{\text{prompt},i}(\boldsymbol{x})$, and thus storing each of these fine-tuned LMs in memory is infeasible. Instead, they first decode generations using each model and then score each generation by averaging their generation probability across all models.

6.2 Prompt Augmentation

Prompt augmentation, also sometimes called *demonstration learning* (Gao et al., 2021), provides a few additional *answered prompts* that can be used to demonstrate how the LM should provide the answer to the actual prompt instantiated with the input $\boldsymbol{x}$. For example, instead of just providing a prompt of "China's capital is `[Z]` .", the prompt can be prefaced by a few examples such as "Great Britain's capital is London . Japan's capital is Tokyo . China's capital is `[Z]` ." Another example of performing addition of two numbers can be found in Fig. 4-(b). These few-shot demonstrations take advantage of the ability of strong language models to learn repetitive patterns (Brown et al., 2020).

Although the idea of prompt augmentation is simple, there are several aspects that make it challenging: (1) *Sample Selection:* how to choose the most effective examples? (2) *Sample Ordering:* How to order the chosen examples with the prompt?

Sample Selection Researchers have found that the choice of examples used in this few-shot scenario can result in very different performance, ranging from near state-of-the-art accuracy on some tasks to near random guess (Lu et al., 2021). To address this issue, Gao et al. (2021); Liu et al. (2021a) utilize sentence embeddings to sample examples that are close to the input in this embedding space. To measure the generalization capability of pre-trained LMs to perform new tasks based on instructions, Mishra et al. (2021) provide both positive samples and negative samples that highlight things to avoid.

Sample Ordering Lu et al. (2021) found that the order of answered prompts provided to the model plays an important role in model performance, and propose entropy-based methods to score different candidate permutations. Kumar and Talukdar (2021) search for a good permutation of training examples as augmented prompts and learn a separator token between the prompts for further gains in performance.

Prompt augmentation is closely related to retrieval-based methods that provide more textual context to the model to improve performance (Guu et al., 2018), a method which has also been shown to be effective in prompt-based learning (Petroni et al., 2020). However, the key difference lies in the fact that prompt augmentation also leverages the template and answer, while larger context learning does not.

6.3 Prompt Composition

For those composable tasks, which can be composed based on more fundamental subtasks, we can also perform *prompt composition*, using multiple sub-prompts, each for one subtask, and then defining a composite prompt based on those sub-prompts. This process is illustrated in Fig. 4-(c). For example, in the relation extraction task, which aims to extract the relation of two entities, we can break down the task into several subtasks including identifying the characteristics of entities and classifying the relationships between entities. Based on this intuition, Han et al.

(2021) first use multiple manually created sub-prompts for entity recognition and relation classification and then compose them into a complete prompt based on logic rules for relation extraction.

6.4 Prompt Decomposition

For tasks where multiple predictions should be performed for one sample (e.g., sequence labeling), directly defining a holistic prompt with regards to the entire input text $\boldsymbol{x}$ is challenging. One intuitive method to address this problem is to break down the holistic prompt into different sub-prompts, and then answer each sub-prompt separately. Fig.4-(d) illustrates this idea with an example from the named entity recognition task, which aims to identify all named entities in an input sentence. In this case, the input will first be converted into a set of text spans, and the model can then be prompted to predict the entity type (including "Not an Entity") for each span. It is not easy to predict all the span types at the same time due to the large number of spans, so different prompts for each span can be created and predicted separately. This sort of *prompt decomposition* for named entity recognition has been explored by Cui et al. (2021) where they apply the approach we discussed here.

7 Training Strategies for Prompting Methods

With the methods in the above sections, it is now clear how to obtain an appropriate prompt (or prompts) and corresponding answers. Now we discuss about methods that explicitly train models in concert with prompting methods, as outlined in the "Training Strategies" section of Fig.1.

7.1 Training Settings

In many cases, prompting methods can be used without *any* explicit training of the LM for the down-stream task, simply taking an LM that has been trained to predict the probability of text $P(\boldsymbol{x})$ and applying it as-is to fill the cloze or prefix prompts defined to specify the task. This is traditionally called the *zero-shot* setting, as there is zero training data for the task of interest.

However, there are also methods that use training data to train the model in concert with prompting methods. These consist of either *full-data learning*, where a reasonably large number of training examples are used to train the model, or *few-shot learning* where a very small number of examples are used to train the model. Prompting methods are particularly useful in the latter case, as there are generally not enough training examples to fully specify the desired behavior, and thus using a prompt to push the model in the right direction is particularly effective.

One thing to note is that for many of the prompt engineering methods described in §4, although annotated training samples are not explicitly used in the training of the downstream task model, they *are* often used in the construction or validation of the prompts that the downstream task will use. As noted by Perez et al. (2021), this is arguably not true zero-shot learning with respect to the downstream task.

7.2 Parameter Update Methods

In prompt-based downstream task learning, there are usually two types of parameters, namely those from (1) pre-trained models and (2) prompts. Which part of parameters should be updated is one important design decision, which can lead to different levels of applicability in different scenarios. We summarize five tuning strategies (as shown in Tab. 6) based on (i) whether the parameters of the underlying LM are tuned, (ii) whether there are additional prompt-related parameters, (iii) if there are additional prompt-related parameters, whether those parameters are tuned.

Strategy	LM Params	Prompt Params		Example
		Additional	Tuned	
Promptless Fine-tuning	Tuned	-		ELMo [130], BERT [32], BART [94]
Tuning-free Prompting	Frozen	✗	✗	GPT-3 [16], AutoPrompt [159], LAMA [133]
Fixed-LM Prompt Tuning	Frozen	✓	Tuned	Prefix-Tuning [96], Prompt-Tuning [91]
Fixed-prompt LM Tuning	Tuned	✗	✗	PET-TC [153], PET-Gen [152], LM-BFF [46]
Prompt+LM Fine-tuning	Tuned	✓	Tuned	PADA [8], P-Tuning [103], PTR [56]

Table 6: Characteristics of different tuning strategies. "Additional" represents if there are additional parameters beyond LM parameters while "Tuned" denotes if parameters are updated.

7.2.1 Promptless Fine-tuning

As mentioned in the introduction, the *pre-train and fine-tune* strategy has been widely used in NLP since before the popularization of prompting methods. Here we refer to pre-training and fine-tuning *without* prompts as *promptless fine-tuning*, to contrast with the prompt-based learning methods introduced in the following sections. In this strategy, given a dataset of a task, all (or some (Howard and Ruder, 2018; Peters et al., 2019)) of the parameters of the pre-trained LM will be updated via gradients induced from downstream training samples. Typical examples of pre-trained models tuned in this way include BERT [32] and RoBERTa [105]. This is a simple, powerful, and widely-used method, but it may overfit or not learn stably on small datasets (Dodge et al., 2020). Models are also prone to *catastrophic forgetting*, where the LM loses its ability to do things that it was able to do before fine-tuning (McCloskey and Cohen, 1989).

- **Advantages**: Simplicity, no need for prompt design. Tuning all the LM parameters allows the model to fit to larger training datasets.
- **Disadvantages**: LMs may overfit or not learn stably on smaller datasets.

7.2.2 Tuning-free Prompting

Tuning-free prompting directly generates the answers without changing the parameters of the pre-trained LMs based only on a prompt, as described in the simplest incarnation of prompting in §2. These can be optionally augmenting input with answered prompts as described in §6.2, and this combination of tuning-free prompting and prompt augmentation is also referred to as *in-context learning* (Brown et al., 2020). Typical examples of tuning-free prompting include LAMA [133] and GPT-3 [16].

- **Advantages**: Efficiency, there is no parameter update process. No catastrophic forgetting, as LM parameters remain fixed. Applicable in zero-shot settings.
- **Disadvantages**: Because prompts are the only method that provide the task specification, heavy engineering is necessary to achieve high accuracy. In particular in the in-context learning setting, providing many answered prompts can be slow at test time, and thus cannot easily use large training datasets.

7.2.3 Fixed-LM Prompt Tuning

In the scenario where additional prompt-relevant parameters are introduced besides parameters of the pre-trained model, *fixed-LM prompt tuning* updates only the prompts' parameters using the supervision signal obtained from the downstream training samples, while keeping the entire pre-trained LM unchanged. Typical examples are Prefix-Tuning [96] and WARP [55].

- **Advantages**: Similarly to tuning-free prompting, it can retain knowledge in LMs and is suitable in few-shot scenarios. Often superior accuracy to tuning-free prompting.
- **Disadvantages**: Not applicable in zero-shot scenarios. While effective in few-shot scenarios, representation power is limited in large-data settings. Prompt engineering through choice of hyperparameters or seed prompts is necessary. Prompts are usually not human-interpretable or manipulable.

7.2.4 Fixed-prompt LM Tuning

Fixed-prompt LM tuning tunes the parameters of the LM, as in the standard pre-train and fine-tune paradigm, but additionally uses prompts with fixed parameters to specify the model behavior. This potentially leads to improvements, particularly in few-shot scenarios.

The most natural way to do so is to provide a discrete textual template that is applied to every training and test example. Typical examples include PET-TC [153], PET-Gen [152], LM-BFF [46]. Logan IV et al. (2021) more recently observe that the prompt engineering can be reduced by allowing for a combination of answer engineering and partial LM fine-tuning. For example, they define a very simple template, *null prompt*, where the input and mask are directly concatenated "`[X][Z]`" without any template words, and find this achieves competitive accuracy.

- **Advantages**: Prompt or answer engineering more completely specify the task, allowing for more efficient learning, particularly in few-shot scenarios.
- **Disadvantages**: Prompt or answer engineering are still required, although perhaps not as much as without prompting. LMs fine-tuned on one downstream task may not be effective on another one.

7.2.5 Prompt+LM Tuning

In this setting, there are prompt-relevant parameters, which can be fine-tuned together with the all or some of the parameters of the pre-trained models. Representative examples include PADA [8], P-Tuning [103]. Notably, this setting is very similar to the standard pre-train and fine-tune paradigm, but the addition of the prompt can provide additional bootstrapping at the start of model training.

- **Advantages**: This is the most expressive method, likely suitable for high-data settings.
- **Disadvantages**: Requires training and storing all parameters of the models. May overfit to small datasets.

8 Applications

In previous sections, we examined prompting methods from the point of view of the mechanism of the method itself. In this section, we rather organize prompting methods from the point of view of which applications they have been applied to. We list these applications in Tab. 7-8 and summarize them in the following sections.

8.1 Knowledge Probing

Factual Probing *Factual probing* (a.k.a. fact retrieval) is one of the earliest scenarios with respect to which prompting methods were applied. The motivation of exploring this task is to quantify how much factual knowledge the pre-trained LM's internal representations bear. In this task, parameters of pre-trained models are usually fixed, and knowledge is retrieved by transforming the original input into a cloze prompt as defined in §2.2, which can be manually crafted or automatically discovered. Relevant datasets including `LAMA` (Petroni et al., 2019) and `X-FACTR` (Jiang et al., 2020a). Since the answers are pre-defined, fact retrieval only focuses on finding effective templates and analyzing the results of different models using these templates. Both discrete template search (Petroni et al., 2019, 2020; Jiang et al., 2020c,a; Haviv et al., 2021; Shin et al., 2020; Perez et al., 2021) and continuous template learning (Qin and Eisner, 2021; Liu et al., 2021b; Zhong et al., 2021b) have been explored within this context, as well as prompt ensemble learning (Jiang et al., 2020c; Qin and Eisner, 2021).

Linguistic Probing Besides factual knowledge, large-scale pre-training also allows LMs to handle linguistic phenomena such as analogies (Brown et al., 2020), negations (Ettinger, 2020), semantic role sensitivity (Ettinger, 2020), semantic similarity (Sun et al., 2021), cant understanding (Sun et al., 2021), and rare word understanding (Schick and Schütze, 2020). The above knowledge can also be elicited by presenting *linguistic probing* tasks in the form of natural language sentences that are to be completed by the LM.

8.2 Classification-based Tasks

Prompt-based learning has been widely explored in classification-based tasks where prompt templates can be constructed relatively easily, such as text classification (Yin et al., 2019) and natural language inference (Schick and Schütze, 2021a). The key to prompting for classification-based tasks is reformulating it as an appropriate prompt. For example, Yin et al. (2019) use a prompt such as "the topic of this document is `[Z]`.", which is then fed into mask pre-trained LMs for slot filling.

Text Classification For *text classification* tasks, most previous work has used cloze prompts, and both prompt engineering (Gao et al., 2021; Hambardzumyan et al., 2021; Lester et al., 2021) and answer engineering (Schick and Schütze, 2021a; Schick et al., 2020; Gao et al., 2021) have been explored extensively. Most existing works explore the efficacy of prompt learning for text classification in the context of *few-shot* setting with "*fixed-prompt LM Tuning*" strategies (defined in §7.2.4).

Natural Language Inference (NLI) NLI aims to predict the relationship (e.g., `entailment`) of two given sentences. Similar to text classification tasks, for *natural language inference* tasks, cloze prompts are commonly used (Schick and Schütze, 2021a). Regarding prompt engineering, researchers mainly focus on the template search in the few-shot learning setting and the answer space $\mathcal{Z}$ is usually manually pre-selected from the vocabulary.

Work	Task	PLM	Setting	Prompt Engineering			Answer Engineering			Tuning	Mul-Pr
				Shape	Man	Auto	Shape	Man	Auto		
LMComm [173]	CR	L2R	Zero	Clo	✓	-	Sp	✓	-	TFP	-
GPT-2 [140]	CR,QA SUM,MT	GPT-2	Zero,Few	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
WNLaMPro [150]	LCP	BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
LMDiagnose [39]	CR,LCP	BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
AdvTrigger [177]	GCG	GPT-2	Full	Pre	-	Disc	Sen	✓	-	TFP	-
CohRank [31]	CKM	BERT	Zero	Clo	✓	-	Tok,Sp	✓	-	TFP	-
LAMA [133]	FP	Conv,Trans ELMo,BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
CTRL [75]	GCG	CTRL	Full	Pre	✓	-	Sen	✓	-	LMT	-
T5 [141]	TC,SUM QA,MT	T5	Full	Pre	✓	-	Tok,Sp,Sen	✓	-	LMT	-
Neg & Mis [74]	FP	Trans,ELMo BERT	Zero	Clo	✓	-	Tok	✓	-	TFP	-
LPAQA [68]	FP	BERT,ERNIE	Full	Clo	✓	Disc	Tok	✓	-	TFP	PE
ZSC [135]	TC	GPT-2	Full	Pre	✓	-	Tok,Sp	✓	-	LMT	-
PET-TC [153]	TC	RoBERTa,XLM-R	Few	Pre	✓	-	Tok	✓	Disc	LMT	PE
ContxFP [132]	FP	BERT,RoBERTa	Zero	Clo	✓	Disc	Tok	✓	-	TFP	-
UnifiedQA [76]	QA	T5,BART	Full	Prefix	✓	-	Tok,Sp,Sen	✓	-	LMT	-
RAG [95]	QA,GCG,TC	BART	Full	Pre	-	Disc	Tok,Sp,Sen	✓	-	LMPT	PE
GPT-3 [16]	QA,MT,GCG CR,TC,LCP MR,SR,AR	GPT-3	Zero,Few	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
CommS2S [187]	CR	T5	Full	Pre	✓	-	Tok	✓	-	LMT	-
PET-SGLUE [154]	TC	ALBERT	Few	Clo	✓	-	Tok,Sp	✓	-	LMT	PE
ToxicityPrompts [47]	GCG	GPT-1,GPT-2 GPT-3,CTRL	Zero	Pre	✓	-	N/A			TFP	-
WhyLM [147]	Theory	GPT-2	Full	Pre	✓	-	Tok	✓	-	PT	-
X-FACTR [66]	FP	mBERT,BERT XLM,XLM-R	Zero	Clo	✓	-	Tok,Sp	✓	-	TFP	-
Petal [149]	TC	RoBERTa	Few	Clo	✓	-	Tok	-	Disc	LMT	PE
AutoPrompt [159]	TC,FP,IE	BERT,RoBERTa	Full	Clo	-	Disc	Tok	-	Disc	TFP	-
CTRLsum [59]	SUM	BART	Full	Pre	✓	-	Sen	✓	-	LMT	-
PET-Gen [152]	SUM	PEGASUS	Few	Pre	✓	-	Sen	✓	-	LMT	PE
LM-BFF [46]	TC	RoBERTa	Few	Clo	-	Disc	Tok	-	Disc	LMT	PE,PA
WARP [55]	TC	RoBERTa	Few,Full	Clo,Pre	✓	Cont	Tok	✓	Cont	PT	PE
Prefix-Tuning [96]	D2T,SUM	GPT-2,BART	Full	Pre	-	Cont	Sen	✓	-	PT	-
KATE [100]	TC,D2T,QA	GPT-3	Few	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
PromptProg [145]	MT,MR AR,QA	GPT-3	Zero,Few	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
ContxCalibrate [201]	TC,FP,IE	GPT-2,GPT-3	Few	Pre	✓	-	Tok,Sp	✓	-	TFP	PA
PADA [8]	TC,TAG	T5	Full	Pre	-	Disc	N/A			LMPT	-
SD [155]	GCG	GPT-2	Zero	Pre	✓	-	N/A			TFP	-
BERTese [58]	FP	BERT	Full	Clo	✓	Disc	Tok	✓	-	TFP	-
Prompt2Data [148]	TC	RoBERTa	Full	Clo	✓	-	Tok,Sp	✓	-	LMT	-
P-Tuning [103]	FP,TC	GPT-2,BERT ALBERT	Few,Full	Clo,Pre	✓	Cont	Tok,Sp	✓	-	TFP,LMPT	-
GLM [37]	TC	GLM	Full	Clo	✓	-	Tok,Sp	✓	-	LMT	-

Table 7: An organization of works on prompting (Part 1). See the caption of Tab. 8 for a detailed description for all the abbreviations used in this table.

Work	Task	PLM	Setting	Prompt Engineering			Answer Engineering			Tuning	Mul-Pr
				Shape	Man	Auto	Shape	Man	Auto		
ADAPET [170]	TC	ALBERT	Few	Clo	✓	-	Tok,Sp	✓	-	LMT	-
Meta [202]	TC	T5	Full	Pre	✓	-	Tok	✓	-	LMT	-
OptiPrompt [203]	FP	BERT	Full	Clo	✓	Cont	Tok	✓	-	PT	-
Soft [137]	FP	BERT,BART RoBERTa	Full	Clo	✓	Cont	Tok	✓	-	PT	PE
DINO [151]	GCG	GPT-2	Zero	Pre	✓	-	N/A			TFP	-
AdaPrompt [21]	IE	BERT	Few,Full	Clo	✓	-	Tok	-	Disc	LMT	-
PMI_{DC} [62]	GCG,QA,TC	GPT-2,GPT-3	Zero	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	-
Prompt-Tuning [91]	TC	T5	Full	Pre	-	Cont	Tok,Sp	✓	-	PT	PE
Natural-Instr [120]	GCG	GPT-3,BART	Few,Full	Pre	✓	-	Tok,Sp,Sen	✓	-	TFP,LMT	PA
OrderEntropy [111]	TC	GPT-2,GPT-3	Few	Pre	✓	-	Tok	✓	-	TFP	PA
FewshotSemp [158]	SEMP	GPT-3	Few	Pre	✓	-	Sen	✓	-	TFP	PA
PanGu-α [194]	QA,CR,TC SUM,GCG	PanGu-α	Zero,Few	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	PA
TrueFewshot [129]	TC,FP	GPT-2,GPT-3 ALBERT	Few	Clo,Pre	✓	Disc	Tok,Sp	✓	-	TFP,LMT	-
PTR [56]	IE	RoBERTa	Full	Clo	✓	Cont	Tok,Sp	✓	-	LMPT	PC
TemplateNER [29]	TAG	BART	Few,Full	Clo,Pre	✓	-	Tok	✓	-	LMT	PD
PERO [83]	TC,FP	BERT,RoBERTa	Few	Pre	✓	-	Tok	✓	-	TFP	PA
PromptAnalysis [181]	Theory	BERT	Full	Clo	-	Cont	N/A			PT	-
CPM-2 [198]	QA,MR,SUM TC,GCG,MT	CPM-2	Full	Pre	-	Cont	Tok,Sp,Sent	✓	-	PT,LMPT	-
BARTScore [193]	EVALG	BART	Zero	Pre	✓	Disc	Sen	✓	-	TFP	PE
NullPrompt [109]	TC	RoBERTa,ALBERT	Few	Pre	✓	-	Tok	✓	-	LMPT	-
Frozen [174]	VQA,VFP,MG	GPT-like	Full	Pre	-	Cont	Sp (Visual)	✓	-	PT	PA
ERNIE-B3 [167]	TC,LCP,NLI CR,QA,SUM GCG	ERNIE-B3	Zero	Clo,Pre	✓	-	Tok,Sp,Sen	✓	-	TFP	-
Codex [20]	CodeGen	GPT	Zero,Few Full	Pre	✓	-	Span	✓	Disc	TFP,LMT	PA
HTLM [1]	TC,SUM	BART	Zero,Few Full	Clo	✓	Disc	Tok,Sp,Sen	✓	-	LMT	PA
FLEX [15]	TC	T5	Zero,Few	Pre	✓	-	Tok,Sp	✓	-	LMT	-

Table 8: An organization of works on prompting (Part 2). The **Task** column lists the tasks that are performed in corresponding papers. We use the following abbreviations. **CR**: Commonsense Reasoning. **QA**: Question Answering. **SUM**: Summarization. **MT**: Machine Translation. **LCP**: Linguistic Capacity Probing. **GCG**: General Conditional Generation. **CKM**: Commonsense Knowledge Mining. **FP**: Fact Probing. **TC**: Text Classification. **MR**: Mathematical Reasoning. **SR**: Symbolic Reasoning. **AR**: Analogical Reasoning. **Theory**: Theoretical Analysis. **IE**: Information Extraction. **D2T**: Data-to-text. **TAG**: Sequence Tagging. **SEMP**: Semantic Parsing. **EVALG**: Evaluation of Text Generation. **VQA**: Visual Question Answering. **VFP**: Visual Fact Probing. **MG**: Multimodal Grounding. **CodeGen**: Code generation. The **PLM** column lists all the pre-trained LMs that have been used in corresponding papers for downstream tasks. **GPT-like** is an autoregressive language model which makes small modifications to the original GPT-2 architecture. For other pre-trained LMs, please refer to §3 for more information. **Setting** column lists the settings for prompt-based learning, can be zero-shot learning (**Zero**), few-shot learning (**Few**), fully supervised learning (**Full**). Under **Prompt Engineering**, **Shape** denotes the shape of the template (**Clo** for cloze and **Pre** for prefix), **Man** denotes whether human effort is needed, **Auto** denotes data-driven search methods (**Disc** for discrete search, **Cont** for continuous search). Under **Answer Engineering**, **Shape** indicates the shape of the answer (**Tok** for token-level, **Sp** for span-level, **Sen** for sentence- or document-level), and **Man** and **Auto** are the same as above. The **Tuning** column lists tuning strategies (§7). **TFP**: Tuning-free Prompting. **LMT**: Fixed-prompt LM Tuning. **PT**: Fixed-LM Prompt Tuning. **LMPT**: LM+Prompt Tuning. The **Mul-Pr** column lists multi-prompt learning methods. **PA**: Prompt Augmentation. **PE**: Prompt Ensembling. **PC**: Prompt Composition. **PD**: Prompt Decomposition.

8.3 Information Extraction

Unlike classification tasks where cloze questions can often be intuitively constructed, for *information extraction* tasks constructing prompts often requires more finesse.

Relation Extraction *Relation extraction* is a task of predicting the relation between two entities in a sentence. Chen et al. (2021b) first explored the application of *fixed-prompt LM Tuning* in relation extraction and discuss two major challenges that hinder the direct inheritance of prompting methodology from classification tasks: (1) The larger label space (e.g. 80 in relation extraction v.s 2 in binary sentiment classification) results in more difficulty in answer engineering. (2) In relation extraction, different tokens in the input sentence may be more or less important (e.g. entity mentions are more likely to participate in a relation), which, however, can not be easily reflected in the prompt templates for classification since the original prompt template regards each word equally. To address the above problems, Chen et al. (2021b) propose an adaptive answer selection method to address the issue (1) and task-oriented prompt template construction for the issue (2), where they use special markers (e.g. `[E]`) to highlight the entity mentions in the template. Similarly, Han et al. (2021) incorporate entity type information via multiple prompt composition techniques (illustrated in Fig. 4).

Semantic Parsing *Semantic parsing* is a task of generating a structured meaning representation given a natural language input. Shin et al. (2021) explore the task of few-shot semantic parsing using LMs by (1) framing the semantic parsing task as a paraphrasing task (Berant and Liang, 2014) and (2) constraining the decoding process by only allowing output valid according to a grammar. They experiment with the *in-context learning* setting described in §7.2.2, choosing answered prompts that are semantically close to a given test example (determined by the conditional generation probability of generating a test sample given another training example). The results demonstrate the effectiveness of the paraphrasing reformulation for semantic parsing tasks using pre-trained LMs.

Named Entity Recognition *Named entity recognition* (NER) is a task of identifying named entities (e.g., person name, location) in a given sentence. The difficulty of prompt-based learning's application to tagging tasks, exemplified as NER, is that, unlike classification, (1) each unit to be predicted is a token or span instead of the whole input text, (2) there is a latent relationship between the token labels in the sample context. Overall, the application of prompt-based learning in tagging task has not been fully explored. Cui et al. (2021) recently propose a template-based NER model using BART, which enumerates text spans and considers the generation probability of each type within manually crafted templates. For example, given an input "Mike went to New York yesterday", to determine what type of entity "Mike" is, they use the template "Mike is a `[Z]` entity", and the answer space $\mathcal{Z}$ consists of values such as "person" or "organization".

8.4 "Reasoning" in NLP

There is still a debate[6] about if deep neural networks are capable of performing "reasoning" or just memorizing patterns based on large training data (Arpit et al., 2017; Niven and Kao, 2019). As such, there have been a number of attempts to probe models' reasoning ability by defining benchmark tasks that span different scenarios. We detail below how prompting methods have been used in these tasks.

Commonsense Reasoning There are a number of benchmark datasets testing commonsense reasoning in NLP systems (Huang et al., 2019; Rajani et al., 2019; Lin et al., 2020; Ponti et al., 2020). Some commonly attempted tasks involve solving Winograd Schemas (Levesque et al., 2012), which require the model to identify the antecedent of an ambiguous pronoun within context, or involve completing a sentence given multiple choices. For the former, an example could be "The trophy doesn't fit into the brown suitcase because it is too large." And the task for the model is to infer whether "it" refers to the trophy or the "suitcase". By replacing "it" with its potential candidates in the original sentences and calculating the probability of the different choices, pre-trained LMs can perform quite well by choosing the choice that achieves the highest probability (Trinh and Le, 2018). For the latter, an example could be "Eleanor offered to fix her visitor some coffee. Then she realized she didn't have a clean `[Z]`.". The candidate choices are "cup", "bowl" and "spoon". The task for the pre-trained LM is to choose the one from the three candidates that most conforms to common sense. For these kinds of tasks, we can also score the generation probability of each candidate and choose the one with the highest probability (Ettinger, 2020).

Mathematical Reasoning Mathematical reasoning is the ability to solve mathematical problems, e.g. arithmetic addition, function evaluation. Within the context of pre-trained LMs, researchers have found that pre-trained embeddings and LMs can perform simple operations such as addition and subtraction when the number of digits is small, but fail when the numbers are larger (Naik et al., 2019; Wallace et al., 2019b; Brown et al., 2020). Reynolds and McDonell (2021) explore more complex mathematical (e.g. $f(\boldsymbol{x}) = \boldsymbol{x} * \boldsymbol{x}$, what is $f(f(3))$?) reasoning problems and improve LM performance through serializing reasoning for the question.

[6] e.g. https://medium.com/reconstruct-inc/the-golden-age-of-computer-vision-338da3e471d1

8.5 Question Answering

Question answering (QA) aims to answer a given input question, often based on a context document. QA can take a variety of formats, such as extractive QA (which identifies content from the context document containing the answer; e.g. SQuAD (Rajpurkar et al., 2016)), multiple-choice QA (where the model has to pick from several choices; e.g. RACE (Lai et al., 2017)), and free-form QA (where the model can return an arbitrary textual string as a response; e.g. NarrativeQA (Kočiský et al., 2018)). Generally, these different formats have been handled using different modeling frameworks. One benefit of solving QA problems with LMs, potentially using prompting methods, is that different formats of QA tasks can be solved within the same framework. For example, Khashabi et al. (2020) reformulate many QA tasks as a text generation problem by fine-tuning seq2seq-based pre-trained models (e.g. T5) and appropriate prompts from the context and questions. Jiang et al. (2020b) take a closer look at such prompt-based QA systems using sequence to sequence pre-trained models (T5, BART, GPT2) and observe that probabilities from these pre-trained models on QA tasks are not very predictive of whether the model is correct or not.

8.6 Text Generation

Text generation is a family of tasks that involve generating text, usually conditioned on some other piece of information. Prompting methods can be easily applied to these tasks by using *prefix prompts* together with autoregressive pre-trained LMs. Radford et al. (2019) demonstrated impressive ability of such models to perform generation tasks such as text summarization and machine translation using prompts such as "translate to french, `[X]`, `[Z]`". Brown et al. (2020) perform *in-context learning* (§7.2.2) for text generation, creating a prompt with manual templates and augmenting the input with multiple *answered prompts*. Schick and Schütze (2020) explore *fixed-prompt LM tuning* (§7.2.4) for few-shot text summarization with manually crafted templates. (Li and Liang, 2021) investigate *fixed-LM prompt tuning* (§7.2.3) for text summarization and data-to-text generation in few-shot settings, where learnable prefix tokens are prepended to the input while parameters in pre-trained models are kept frozen. Dou et al. (2021) explored the *prompt+LM tuning* strategy (§7.2.5) on text summarization task, where learnable prefix prompts are used and initialized by different types of guidance signals, which can then be updated together with parameters of pre-trained LMs.

8.7 Automatic Evaluation of Text Generation

Yuan et al. (2021b) have demonstrated that prompt learning can be used for automated evaluation of generated texts. Specifically, they conceptualize the evaluation of generated text as a text generation problem, modeled using a pre-trained sequence-to-sequence, and then use *prefix prompts* that bring the evaluation task closer to the pre-training task. They experimentally find that simply adding the phrase "such as" to the translated text when using pre-trained models can lead to a significant improvement in correlation on German-English machine translation (MT) evaluation.

8.8 Multi-modal Learning

Tsimpoukelli et al. (2021) shift the application of prompt learning from text-based NLP to the *multi-modal* setting (vision and language). Generally, they adopt the *fixed-LM prompt tuning* strategy together with *prompt augmentation* techniques. They specifically represent each image as a sequence of continuous embeddings, and a pre-trained LM whose parameters are frozen is prompted with this prefix to generate texts such as image captions. Empirical results show few-shot learning ability: with the help of a few demonstrations (answered prompts), system can rapidly learn words for new objects and novel visual categories.

8.9 Meta-Applications

There are also a number of applications of prompting techniques that are not NLP tasks in and of themselves, but are useful elements of training strong models for any application.

Domain Adaptation Domain adaptation is the practice of adapting a model from one domain (e.g. news text) to another (e.g. social media text). Ben-David et al. (2021) use self-generated *domain related features* (DRFs) to augment the original text input and perform sequence tagging as a sequence-to-sequence problem using a seq2seq pre-trained model.

Debiasing Schick et al. (2021) found that LMs can perform self-diagnosis and self-debiasing based on biased or debiased instructions. For example, to self-diagnosis whether the generated text contains violent information, we can use the following template "The following text contains violence. `[X][Z]`". Then we fill `[X]` with the input text and look at the generation probability at `[Z]`, if the probability of "Yes" is greater than "No", then we would assume the given text contains violence, and vice versa. To perform debiasing when generating text, we first compute the probability of the next word $P(x_t|\boldsymbol{x}_{<t};\theta)$ given the original input. Then we compute the probability

of next word $P(x_t|[\boldsymbol{x}_{<t}; \boldsymbol{x}_{\text{diagnosis}}]; \theta)$ by appending self-diagnosis textual input to the original input as mentioned above. These two probability distributions for the next token can be combined to suppress the undesired attribute.

Dataset Construction Schick and Schütze (2021) propose to use pre-trained LMs to generate datasets given certain instructions. As an example, suppose we have an unlabeled dataset in which each sample is a sentence. If we want to construct a dataset containing pairs of semantically similar sentences, then we can use the following template for each input sentence: "Write two sentences that mean the same thing. `[X] [Z]`" and attempt to generate a sentence that shares the same meaning as the input sentence.

8.10 Resources

We also collect some useful resources for different prompt-based applications.

Dataset Some datasets specifically designed for few-shot and zero-shot learning are shown in Tab. 9.

Task	Dataset	Setting	URL
Commonsense Reasoning	Pronoun Disambiguation Problems [93]	Zero	https://cs.nyu.edu/ davise/papers/...
	Winograd Schema Challenge [93]	Zero	https://cs.nyu.edu/ davise/papers/...
	CPRAG-102 [39]	Zero	https://github.com/aetting/lm-diagnostics
Linguistic Capacity Probing	WNLaMPro [150]	Zero	https://github.com/timoschick/...
	ROLE-88 [39]	Zero	https://github.com/aetting/lm-diagnostics
	NEG-136 [39]	Zero	https://github.com/aetting/lm-diagnostics
Fact Probing	LAMA [133]	Zero	https://dl.fbaipublicfiles.com/LAMA/...
	Negated LAMA [74]	Zero	https://github.com/norakassner/LAMA...
	Misprimed LAMA [74]	Zero	https://github.com/norakassner/LAMA...
	X-FACTR [66]	Zero	https://x-factr.github.io/
	LAMA-TREx-easy-hard [203]	Zero	https://github.com/princeton-nlp/...
Text Classification	FLEX [15]	Zero,Few	https://github.com/allenai/flex
	FewGLUE [154]	Few	https://github.com/timoschick/fewglue
General Conditional Gen.	REALTOXICITYPROMPTS [47]	Zero	https://allenai.org/data/...
	Natural-Instructions [120]	Few,Full	https://instructions.apps.allenai.org/

Table 9: Few-shot and zero-shot datasets for prompt-based learning.

Prompts As shown in Tab. 10, we collect existing commonly-used prompts designed manually, which can be regarded as off-the-shelf resource for future research and applications.

9 Prompt-relevant Topics

What is the essence of prompt-based learning and how does it relate to other learning methods? In this section, we connect prompt learning with other similar learning methods.

Ensemble Learning *Ensemble learning* (Ting and Witten, 1997; Zhou et al., 2002) is a technique that aims to improve the performance of a task by taking advantage of the complementarity of multiple systems. Generally, the different systems used in an ensemble result from different choices of architectures, training strategies, data ordering, and/or random initialization. In prompt ensembling (§6.1), the choice of prompt templates becomes another way to generate multiple results to be combined. This has the clear advantage that this does not necessarily require training the model multiple times. For example, when using discrete prompts, these prompts can simply be changed during the inference stage (Jiang et al., 2020c).

Few-shot Learning *Few-shot learning* aims to learn a machine learning system in the data-scarce scenarios with few training samples. There are a wide variety of methods to achieve few-shot learning including model agnostic meta-learning (Finn et al., 2017b) (learning features rapidly adaptable to new tasks), embedding learning (Bertinetto et al., 2016) (embedding each sample in a lower-dimensional space where similar samples are close together), memory-based learning (Kaiser et al., 2017) (representing each sample by a weighted average of contents from the memory) etc. (Wang et al., 2020). Prompt augmentation can be regarded as another way to achieve few-shot learning (a.k.a. priming-based few-shot learning (Kumar and Talukdar, 2021)). Compared to previous methods, prompt augmentation directly prepends several labeled samples to the currently-processed sample elicit knowledge from pre-trained LMs even without any parameter tuning.

Task	Example Prompt-Answer	Resource
Fact Probing	**Prompt** Adolphe Adam died in `[Z]`. **Answer** $\mathcal{V}$ **Prompt** iPod Touch is produced by `[Z]`. **Answer** $\mathcal{V}$ **Prompt** The official language of Mauritius is `[Z]`. **Answer** $\mathcal{V}$	LAMA dataset LPAQA dataset X-FACTR dataset
Text Classificatin	**Prompt** Which of these choices best describes the following document? "`[Class A]`", "`[Class B]`", "`[Class C]`". `[X][Z]` **Answer** `[Class A]`, `[Class B]`, `[Class C]` **Prompt** How is the text best described? : "`[Class A]`", "`[Class B]`" , or "`[Class C]`". `[X][Z]` **Answer** `[Class A]`, `[Class B]`, `[Class C]` **Prompt** This passage is about `[Z]`: `[X]` **Answer** `[Class A]`, `[Class B]`, `[Class C]` **Prompt** `[X]`. Is this review positive? `[Z]` **Answer** Yes, No **Prompt** `[X]` It was `[Z]`. **Answer** great, terrible	Meta [202]
Natural Language Inference	**Prompt** `[X1]`? `[Z]`, `[X2]` **Answer** Yes, No, Maybe **Prompt** `[X1]` `[Z]`, `[X2]` **Answer** Yes, No, Maybe	
Commonsense Reasoning	**Prompt** The trophy doesn't fit into the brown suitcase because `[Z]` is too large. **Answer** trophy, suitcase **Prompt** Ann asked Mary what time the library closes, because `[Z]` had forgotten. **Answer** Ann, Mary	PDP dataset WSC dataset CPRAG-102 dataset
Linguistic Knowledge Probing	**Prompt** A robin is a `[Z]`. **Answer** bird, tree **Prompt** A robin is not a `[Z]`. **Answer** bird, tree **Prompt** New is the opposite of `[Z]`. **Answer** old, young, current	WNLaMPro dataset ROLE-88 dataset NEG-136 dataset
Named Entity Recognition	**Prompt-Pos** `[X]` `[Span]` is a `[Z]` entity. **Prompt-Neg** `[X]` `[Span]` is not a named entity. **Answer** person, location, organization, miscellaneous **Prompt-Pos** The entity type of `Span` is `[Z]`. **Prompt-Neg** `[X]` The entity type of `[Span]` is none entity. Answer person, location, organization, miscellaneous	TemplateNER [29]
Question Answering	**Prompt** `[Question]` `[Passage]` `[Z]` **Prompt** `[Passage]` According to the passage, `[Question]` `[Z]` **Prompt** Based on the following passage, `[Question]` `[Z]`. `[Passage]`	
Summarization	**Prompt** Text: `[X]` Summary: `[Z]` **Prompt** `[X]` TL;DR: `[Z]` **Prompt** `[X]` In summary, `[Z]`	BARTScore [193]
Machine Translation	**Prompt** French: `[French sentence]` English: **Prompt** A French sentence is provided: `[French sentence]` The French translator translates the sentence into English: `[Z]` **Prompt** `[French sentence]` = `[Z]`	

Table 10: Commonly used prompts and answers for different tasks. `[X]` and `[Z]` denote slots for input and answer respectively. $\mathcal{V}$ denotes the vocabulary of the LM. More prompts for each task can be found using the **Resource** column.

Prompt Concept	Relevant Topic	Commonality	Peculiarity
Prompt Ensembling [68; 153]	Ensemble Learning [171; 204]	Combine results of multiple systems to get better performance	In prompt ensembling, multiple predictions result from different prompt variants. This contrasts with architecture or feature variations, each of which requires separate training.
Prompt Augmentation [16; 46]	Few-shot Learning [160; 42]	Use few examples to learn generalized rules	Prompt augmentation is a specific subset of few-shot learning.
	Larger-context Learning [18; 53]	Introduce larger context to aid the learning process	Additional information introduced in larger-context learning is not necessarily the labeled data.
Discrete Prompt Search [68; 159]	Query reformulation [123; 123]	Reformulate the input into a query form	Query reformulation commonly focuses on information extraction and question answering tasks, while prompt learning can be applied to a variety of NLP tasks
Discrete Prompt Fine-tuning [46]	QA-based multi-task learning [115; 97]	Reformulate many tasks into an QA form	QA-based formulations aim to solve different tasks through question answering, while prompting additionally targets full use of pre-trained models.
Continuous Prompt Fine-tuning [103; 36]	Controlled Text Generation [191; 77; 156]	Input is augmented with additional inputs to control the generation process	Controlled generation targets generation of a particular type of text while prompt learning uses prompts to specify the task itself.
Prompt-based downstream task learning [153; 193]	Supervised Attention [101; 165]	Require external hint to remind the model of which part information should be focused on	Research works on supervised attention usually target at salient information from an image or text, while prompt learning aims to utilize relevant knowledge from the pre-trained model.
	Data augmentation [40; 144]	Improving downstream tasks' performance by introducing additional samples	Data augmentation introduce additional training samples in an explicit way while prompts can be regarded as highly-condensed training samples [88].

Table 11: Other research topics relevant to prompting methods.

Larger-context Learning *Larger-context learning* aims to improve the system's performance by augmenting the input with additional contextual information, e.g. retrieved from the training set (Cao et al., 2018) or external data sources (Guu et al., 2020). Prompt augmentation can be regarded as adding relevant labeled samples into the input, but a minor difference is in larger-context learning, the introduced context is not necessarily labeled data.

Query Reformulation *Query reformulation* (Mathieu and Sabatier, 1986; Daumé III and Brill, 2004) is commonly used in information retrieval (Nogueira and Cho, 2017) and question answering tasks (Buck et al., 2017; Vakulenko et al., 2020), which aim to elicit more relevant texts (documents or answers) by expanding the input query with related query terms (Hassan, 2013) or generating paraphrases. There are several commonalities between prompt-based learning and query reformulation, for example (1) both aim to make better use of some existing knowledge bases by asking a right questions (2) the knowledge bases are usually a black-box, not available to the users, so researchers must learn how to probe it optimally based on solely questions.

There are also differences: the knowledge base in traditional query reformulation problems is usually a search engine (Nogueira and Cho, 2017), or QA system (Buck et al., 2017). By contrast, for prompt-based learning, we usually define this knowledge base as an LM, and need to find the appropriate query to elicit an appropriate answer from it. The input reformulation in prompt learning has changed the form of tasks. For example, an original text classification task has been converted into a cloze question problem, therefore bringing additional complexity regarding how to (1) make an appropriate task formulation, and (2) change the modeling framework accordingly. These steps are not required in traditional query formulation. Despite these discrepancies, some methodologies from query reformulation research still can be borrowed for prompt learning, such as decomposing input query into multiple sub-queries (Nogueira et al., 2019), similar to prompt decomposition.

QA-based Task Formulation *QA-based task formulation* aims to conceptualize different NLP tasks as a question-answering problem. (Kumar et al., 2016; McCann et al., 2018) are earlier works that attempt to unify multiple NLP tasks into a QA framework. Later, this idea has been further explored in information extraction (Li et al., 2020; Wu

et al., 2020) and text classification (Chai et al., 2020). These methods are very similar to the prompting methods introduced here in that they use textual questions to specify which task is to be performed. However, one of the key points of prompting methods is how to better use the knowledge in pre-trained LMs, and these were not covered extensively on previous works advocating for QA formulations.

Controlled Generation *Controlled generation* aims to incorporate various types of guidance beyond the input text into the generation model (Yu et al., 2020). Specifically, the guidance signals could be *style tokens* (Sennrich et al., 2016b; Fan et al., 2018), *length specifications* (Kikuchi et al., 2016), *domain tags* (Chu et al., 2017), or any variety of other pieces of information used to control of the generated text. It could also be *keywords* (Saito et al., 2020), *relation triples* (Zhu et al., 2020) or even *highlighted phrases or sentences* (Grangier and Auli, 2018; Liu et al., 2021c) to plan the content of generated texts. In a way, many of the prompting methods described here are a type of controllable generation, where the prompt is usually used to specify the *task itself*. Thus, it is relatively easy to find commonalities between the two genres: (1) both add extra information to the input text for better generation, and these additional signals are (often) learnable parameters. (2) If "controlled generation" is equipped with seq2seq-based pre-trained models (e.g., BART), then it is can be regarded as prompt learning with input-dependent prompts and the *prompt+LM fine-tuning* strategy (§7.2.5), e.g. *GSum* (Dou et al., 2021), where both the prompt's and pre-trained LM's parameters can be tuned.

Also, some clear discrepancies between controlled generation and prompt-based text generation are: (1) In controlled generation work, the control is generally performed over the style or content of the generations (Fan et al., 2018; Dou et al., 2021) while the underlying task remains the same. They don't necessarily require a pre-trained model. In contrast, the main motivation for using prompts for text generation is to specify the task itself and better utilize the pre-trained model. (2) Moreover, most of the current work on prompt learning in text generation shares a dataset- or task-level prompt (Li and Liang, 2021). Only very few works have explored input-dependent ones (Tsimpoukelli et al., 2021). However, this is a common setting and effective in the controlled text generation, which may provide valuable direction for the future work on prompt learning.

Supervised Attention Knowing to pay attention to the important information is a key step when extracting useful information from objects such as long text sequences (Liu et al., 2016; Sood et al., 2020), images (Sugano and Bulling, 2016; Zhang et al., 2020b), or knowledge bases (Yu et al., 2020; Dou et al., 2021)). *Supervised attention* (Liu et al., 2017b) aims to provide explicit supervision over the attention of models based on the fact that completely data-driven attention can overfit to some artifacts (Liu et al., 2017a). In this respect, prompt learning and supervised attention share ideas that both aim to extract salient information with some clues, which need to be provided separately. To solve this problem, supervised attention methods tried to use additional loss functions to learn to predict gold attention on a manually labeled corpus (Jiang et al., 2015; Qiao et al., 2018; Gan et al., 2017). Research on prompt learning may also borrow ideas from this literature.

Data Augmentation Data augmentation is a technique that targets increasing the amount of data that can be used for training by making modifications to existing data (Fadaee et al., 2017; Ratner et al., 2017). As recently observed by (Scao and Rush, 2021), adding prompts can achieve a similar accuracy improvement to the addition of 100s of data points on average across classification tasks, which suggests that using prompts for a downstream task is similar to conducting data augmentation implicitly.

10 Challenges

Although prompt-based learning has shown significant potential among different tasks and scenarios, several challenges remain, some of which we detail below.

10.1 Prompt Design

Tasks beyond Classification and Generation Most existing works about prompt-based learning revolve around either text classification or generation-based tasks. Applications to information extraction and text analysis tasks have been discussed less, largely because the design of prompts is less straightforward. We expect that applying prompting methods to these tasks in the future it will require either reformulating these tasks so that they can be solved using classification or text generation-based methods, or performing effective answer engineering that expresses structured outputs in an appropriate textual format.

Prompting with Structured Information In many NLP tasks, the inputs are imbued with some variety of structure, such as tree, graph, table, or relational structures. How to best express these structures in prompt or answer engineering is a major challenge. Existing works (Chen et al., 2021b) make a step by making prompts with additional marks to encode lexical information, such as entity markings. Aghajanyan et al. (2021) present structured prompts based on hyper text markup language for more fine-grained web text generation. However, moving beyond this to more complicated varieties of structure is largely unexplored, and a potentially interesting research area.

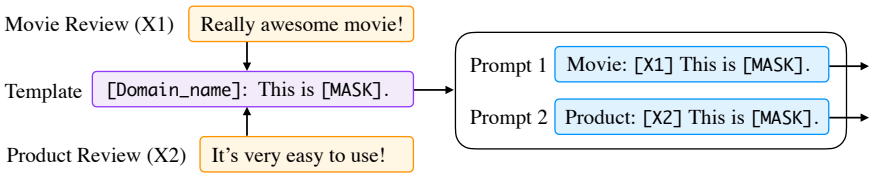


Figure 5: Multi-prompt learning for multi-task, multi-domain or multi-lingual learning. We use different colors to differentiate different components as follows. " " for input text, " " for template, " " for prompt.

Entanglement of Template and Answer The performance of a model will depend on *both* the templates being used and the answer being considered. How to simultaneously search or learn for the best combination of template and answer remains a challenging question. Current works typically select answers before select template (Gao et al., 2021; Shin et al., 2020), but Hambardzumyan et al. (2021) have demonstrated the initial potential of simultaneously learning both.

10.2 Answer Engineering

Many-class and Long-answer Classification Tasks For classification-based tasks, there are two main challenges for answer engineering: (a) When there are too many classes, how to select an appropriate answer space becomes a difficult combinatorial optimization problem. (b) When using multi-token answers, how to best decode multiple tokens using LMs remains unknown, although some multi-token decoding methods have been proposed (Jiang et al., 2020a).

Multiple Answers for Generation Tasks For text generation tasks, qualified answers can be semantically equivalent but syntactically diverse. So far, almost all works use prompt learning for text generation relying solely on a single answer, with only a few exceptions (Jiang et al., 2020c). How to better guide the learning process with multiple references remains a largely open research problem.

10.3 Selection of Tuning Strategy

As discussed in §7, there are a fairly wide variety of methods for tuning parameters of prompts, LMs, or both. However, given the nascent stage of this research field, we still lack a systematic understanding of the tradeoffs between these methods. The field could benefit from systematic explorations such as those performed in the pre-train and fine-tune paradigm regarding the tradeoffs between these different strategies (Peters et al., 2019).

10.4 Multiple Prompt Learning

Prompt Ensembling In prompt ensembling methods, the space and time complexity increase as we consider more prompts. How to distill the knowledge from different prompts remains underexplored. Schick and Schütze (2020, 2021a,b) use an ensemble model to annotate a large dataset to distill the knowledge from multiple prompts.

In addition, how to select ensemble-worthy prompts is also under-explored. For text generation tasks, the study of prompt ensemble learning has not been performed so far, probably because ensemble learning in text generation itself is relatively complicated. To remedy this problem, some recently proposed neural ensembling methods such as *Refactor* (Liu et al., 2021c) could be considered as a method for prompt ensembling in text generation tasks.

Prompt Composition and Decomposition Both prompt composition and decomposition aim to break down the difficulty of a complicated task input by introducing multiple sub-prompts. In practice, how to make a good choice between them is a crucial step. Empirically, for those token (Ma and Hovy, 2016) or span (Fu et al., 2021) prediction tasks (e.g., NER), prompt decomposition can be considered, while for those span relation prediction (Lee et al., 2017) tasks (e.g., entity coreference), prompts composition would be a better choice. In the future, the general idea of de-/composing can be explored in more scenarios.

Prompt Augmentation Existing prompt augmentation methods are limited by the input length, i.e., feeding too many demonstrations to input is infeasible. Therefore, how to select informative demonstrations, and order them in an appropriate is an interesting but challenging problem (Kumar and Talukdar, 2021).

Prompt Sharing All the above considerations refer to the application of prompt in a single task, domain or language. We may also consider *prompt sharing*, where prompt learning is applied to multiple tasks, domains, or languages. Some key issues that may arise include how to design individual prompts for different tasks, and how to modulate their interaction with each other. So far this field has not been explored. Fig.5 illustrates a simple multiple prompt learning strategy for multiple tasks, where prompt templates are partially shared.

10.5 Selection of Pre-trained Models

With plenty of pre-trained LMs to select from (see §3), how to choose them to better leverage prompt-based learning is an interesting and difficult problem. Although we have conceptually introduced (§3.4) how different paradigms of pre-trained models are selected for diverse NLP tasks, there are few to no systematic comparisons of the benefits brought by prompt-based learning for different pre-trained LMs.

10.6 Theoretical and Empirical Analysis of Prompting

Despite their success in many scenarios, theoretical analysis and guarantees for prompt-based learning are scarce. Wei et al. (2021) showed that soft-prompt tuning can relax the non-degeneracy assumptions (the generation probability of each token is linearly independent) needed for downstream recovery (i.e. recover the ground-truth labels of the downstream task.), making it easier to extract task-specific information. Saunshi et al. (2021) verified that text classification tasks can be reformulated as sentence completion tasks, thus making language modeling a meaningful pre-training task. Scao and Rush (2021) empirically show that prompting is often worth 100s of data points on average across classification tasks.

10.7 Transferability of Prompts

Understanding the extent to which prompts are specific to the model and improving the transferability of prompts are also important topics. (Perez et al., 2021) show that prompts selected under tuned few-shot learning scenario (where one has a larger validation set to choose prompts) generalize well across models of similar sizes while prompts selected under true few-shot learning scenario (where one only has a few training samples) do not generalize as effectively as the former setting among models with similar sizes. The transferability is poor when the model sizes are quite different in both scenarios.

10.8 Combination of Different Paradigms

Notably, much of the success of the prompting paradigm is built on top of pre-trained models that were developed for the pre-train and fine-tune paradigm, such as BERT. However, are the pre-training methods that are effective for the latter applicable as-is to the former, or can we entirely re-think our pre-training methods to further improve accuracy or ease of applicability to prompting-based learning? This is an important research question that has not been covered extensively by the literature.

10.9 Calibration of Prompting Methods

Calibration (Gleser, 1996) refers to the ability of a model to make good probabilistic predictions. When using the generation probability of the pre-trained LMs (e.g., BART) to predict the answer, we need to be careful since the probability distribution is typically not well calibrated. Jiang et al. (2020b) observed the probabilities of pre-trained models (e.g., BART, T5, GPT-2) on QA tasks are well calibrated. Zhao et al. (2021) identify three pitfalls (majority label bias, recency bias and common token bias) that lead the pre-trained LMs to be biased toward certain answers when provided answered prompts. For example, if the final answered prompt has a positive label, then this will bias the model towards predicting positive words. To overcome those pitfalls, Zhao et al. (2021) first use context-free input (e.g. the prompt would be "Input: Subpar acting. Sentiment: Negative\n Input: Beautiful film. Sentiment: Positive\n Input: N/A. Sentiment:") to get the initial probability distribution P_0, then they use the real input (e.g. the prompt would be "Input: Subpar acting. Sentiment: Negative\n Input: Beautiful film. Sentiment: Positive\n Input: Amazing. Sentiment:") to get the probability distribution P_1. Finally, these two distributions can be used to get a calibrated generation probability distribution. However, this method has two drawbacks: (1) it comes with the overhead of finding proper context-free input (e.g. whether to use "N/A" or "None") and (2) the probability distribution of the underlying pre-trained LM is still not calibrated.

Even though we have a calibrated probability distribution, we also need to be careful when we assume a single gold answer for an input. This is because that all surface forms of a same object will compete for finite probability mass (Holtzman et al., 2021). For example, if we consider the gold answer to be "Whirlpool bath", the generation probability of it will typically be low since the word "Bathtub" shares the same meaning and it will take over a large probability mass. To address this issue, we could either (i) perform answer engineering to construct a comprehensive gold answer set using paraphrasing methods (§5.2.2) or (ii) calibrate the probability of a word based on its prior likelihood within the context (Holtzman et al., 2021).

11 Meta Analysis

In this section, we aim to give a quantitative birds-eye view of existing research on prompting methods by performing a meta analysis over existing research works along different dimensions.

TABLE 12 Timeline of prompt-based learning. The time for each paper is based on its first arXiv version (if exists) or estimated submission time. A web-version can refer to NLPedia-Pretrain. Works in red consider natural language understanding (NLU) tasks; works in blue consider natural language generation (NLG) tasks; works in green consider both NLU tasks and NLG tasks.

2018.06.07	LMComm (Trinh and Le, 2018)
2019.02.14	GPT-2 (Radford et al., 2019)
2019.04.14	WNLaMPro (Schick and Schütze, 2020)
2019.07.31	LMDiagnose (Ettinger, 2020)
2019.08.20	AdvTrigger (Wallace et al., 2019a)
2019.09.02	CohRank (Davison et al., 2019)
2019.09.03	LAMA (Petroni et al., 2019)
2019.09.11	CTRL (Keskar et al., 2019)
2019.10.23	T5 (Raffel et al., 2020)
2019.11.08	Neg & Misprim (Kassner and Schütze, 2020)
2019.11.28	LPAQA (Jiang et al., 2020c)
2019.12.10	ZSC (Puri and Catanzaro, 2019)
2020.01.21	PET-TC (Schick and Schütze, 2021a)
2020.03.10	ContxFP (Petroni et al., 2020)
2020.05.02	UnifiedQA (Khashabi et al., 2020)
2020.05.22	RAG (Lewis et al., 2020b)
2020.05.28	GPT-3 (Brown et al., 2020)
2020.09.08	CommS2S(Yang et al., 2020)
2020.09.15	PET-SGLUE (Schick and Schütze, 2021b)
2020.09.24	ToxicityPrompts (Gehman et al., 2020)
2020.10.07	WhyLM (Saunshi et al., 2021)
2020.10.13	X-FACTR (Jiang et al., 2020a)
2020.10.26	Petal (Schick et al., 2020)
2020.10.29	AutoPrompt (Shin et al., 2020)
2020.12.08	CTRLsum (He et al., 2020a)
2020.12.22	PET-Gen (Schick and Schütze, 2020)
2020.12.31	LM-BFF (Gao et al., 2021)
2021.01.01	WARP (Hambardzumyan et al., 2021)
2021.01.01	Prefix-Tuning (Li and Liang, 2021)
2021.01.17	KATE (Liu et al., 2021a)
2021.02.15	PromptProg (Reynolds and McDonell, 2021)
2021.02.19	ContxCalibrate (Zhao et al., 2021)
2021.02.24	PADA (Ben-David et al., 2021)
2021.02.27	SD (Schick et al., 2021)
2021.03.09	BERTese (Haviv et al., 2021)
2021.03.15	Prompt2Data (Scao and Rush, 2021)
2021.03.18	P-Tuning (Liu et al., 2021b)
2021.03.18	GLM (Du et al., 2021)
2021.03.22	ADAPET (Tam et al., 2021)
2021.04.10	Meta (Zhong et al., 2021a)
2021.04.12	OptiPrompt (Zhong et al., 2021b)
2021.04.14	Soft (Qin and Eisner, 2021)
2021.04.15	DINO (Schick and Schütze, 2021)
2021.04.15	AdaPrompt (Chen et al., 2021b)
2021.04.16	PMI_{DC} (Holtzman et al., 2021)
2021.04.18	Prompt-Tuning (Lester et al., 2021)
2021.04.18	Natural-Instr (Mishra et al., 2021)
2021.04.18	OrderEntropy (Lu et al., 2021)
2021.04.18	FewshotSemp (Shin et al., 2021)
2021.04.26	PanGu-α (Zeng et al., 2021)
2021.05.24	TrueFewshot (Perez et al., 2021)
2021.05.24	PTR (Han et al., 2021)
2021.06.03	TemplateNER (Cui et al., 2021)
2021.06.03	PERO (Kumar and Talukdar, 2021)
2021.06.16	PromptAnalysis (Wei et al., 2021)
2021.06.20	CPM-2 (Zhang et al., 2021)
2021.06.21	BARTScore (Yuan et al., 2021b)
2021.06.24	NullPrompt (Logan IV et al., 2021)
2021.06.25	Frozen (Tsimpoukelli et al., 2021)
2021.07.05	ERNIE-B3 (Sun et al., 2021)
2021.07.07	Codex (Chen et al., 2021a)
2021.07.14	HTLM (Aghajanyan et al., 2021)
2021.07.15	FLEX (Bragg et al., 2021)

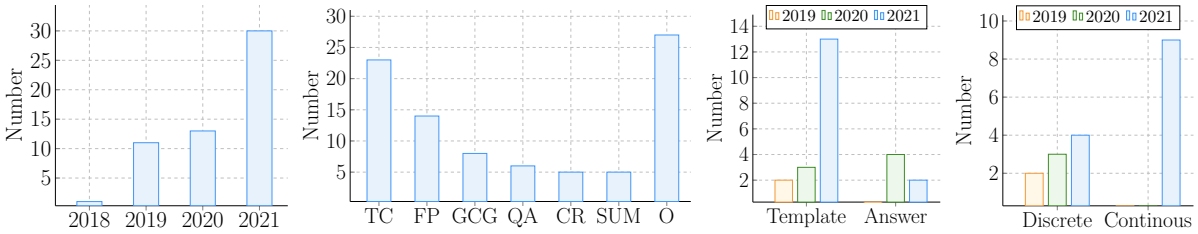
11.1 Timeline

We first summarize a number of existing research papers in a chronological order with in the form of a *timeline*, which hopefully, help researchers who are new to this topic understand the evolution of the field.

11.2 Trend Analysis

We also calculate the number of prompt-based papers with respect to different dimensions.

Year With the emergence of different kinds of pre-trained LMs, prompt-based learning has become a more and more active research field, as can be seen in Fig. 6-(a). We can see a huge surge in 2021, which is perhaps due to the prevalence of GPT-3 (Brown et al., 2020), which greatly increased the popularity of prompting in the few-shot multi-task setting.



(a) Year. (b) Task. (c) Automatic Search. (d) Search Space.

Figure 6: Meta-analyses over different dimensions. The statistics are based on the works in Tab. 7 and Tab. 8. In (d), we use the following abbreviations. TC: text classification, FP: factual probing, GCG: general conditional generation, QA: question answering, CR: commonsense reasoning, SUM: summarization, O: others.

Tasks We plot the number of works that investigate various tasks in Fig. 6-(b). For a task that has fewer than 5 relevant works, we group it into "Others". As the bar chart indicates, most tasks regarding prompt-based learning revolve around text classification and factual probing. We conjecture that this is because that for these tasks, both template engineering and answer engineering are relatively easy to conduct, and experiments are relatively computationally inexpensive.

Prompt vs. Answer Search As noted in previous sections, both prompt and answer search are important tools to take advantage of pre-trained language models for many tasks. Current research mainly focuses on template search instead of answer search, as shown in Fig. 6-(c).

Likely reasons are: (1) For conditional generation tasks (e.g. summarization or translation), the gold references can be directly used as answer. Although there are many sequences that may share the same semantics, how to effectively conduct multi-reference learning in conditional text generation problems is non-trivial. (2) For classification tasks, most of the time, label words are relative easy to select using domain knowledge.

Discrete Search vs. Continuous Search Since there are only a few works focus on automatic answer search, we analyze the automatic template search. As time goes by, there has been a shift from discrete search to continuous search for prompt engineering, as shown in Fig. 6-(d). Likely reasons are: (1) discrete search is harder to optimize compared to continuous search, (2) soft prompts have greater representation ability.

12 Conclusion

In this paper, we have summarized and analyzed several paradigms in the development of statistical natural language processing techniques, and have argued that *prompt-based learning* is a promising new paradigm that may represent another major change in the way we look at NLP. First and foremost, we hope this survey will help researchers more effectively and comprehensively understand the paradigm of prompt-based learning, and grasp its core challenges so that more scientifically meaningful advances can be made in this field. In addition, looking all the way back to the summary of the four paradigms of NLP research presented in §1, we hope to highlight the commonalities and differences between them, making research on any of these paradigms more full-fledged, and potentially providing a catalyst to inspire work towards the next paradigm shift as well.

Acknowledgements

We would like to thank Chunting Zhou for her constructive comments on this work.

References

[1] Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021. Htlm: Hyper-text pre-training and prompting of language models. *arXiv preprint arXiv:2107.06955*.

[2] Zeyuan Allen-Zhu and Yuanzhi Li. 2020. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *CoRR*, abs/2012.09816.

[3] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. 2017. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242. PMLR.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[5] Hangbo Bao, Li Dong, and Furu Wei. 2021. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*.

[6] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 642–652. PMLR.

[7] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

[8] Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. Pada: A prompt-based autoregressive approach for adaptation to unseen domains.

[9] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[10] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

[11] Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

[12] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. 2016. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531.

[13] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow.

[14] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

[15] Jonathan Bragg, Arman Cohan, Kyle Lo, and Iz Beltagy. 2021. FLEX: unifying evaluation for few-shot NLP. *CoRR*, abs/2107.07170.

[16] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

[17] Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2017. Ask the right questions: Active question reformulation with reinforcement learning. *arXiv preprint arXiv:1705.07830*.

[18] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.

[19] Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. In *International Conference on Machine Learning*, pages 1371–1382. PMLR.

[20] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. 2021a. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

[21] Xiang Chen, Xin Xie, Ningyu Zhang, Jiahuan Yan, Shumin Deng, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021b. Adaprompt: Adaptive prompt-based finetuning for relation extraction. *CoRR*, abs/2104.07650.

[22] Zewen Chi, Li Dong, Shuming Ma, Shaohan Huang, Xian-Ling Mao, Heyan Huang, and Furu Wei. 2021a. mt6: Multilingual pretrained text-to-text transformer with translation pairs. *CoRR*, abs/2104.08692.

[23] Zewen Chi, Shaohan Huang, Li Dong, Shuming Ma, Saksham Singhal, Payal Bajaj, Xia Song, and Furu Wei. 2021b. XLM-E: cross-lingual language model pre-training via ELECTRA. *CoRR*, abs/2106.16138.

[24] Chenhui Chu, Raj Dabre, and Sadao Kurohashi. 2017. An empirical comparison of domain adaptation methods for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–391, Vancouver, Canada. Association for Computational Linguistics.

[25] J. Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv*, abs/1412.3555.

[26] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[27] Ronan Collobert, J. Weston, L. Bottou, Michael Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

[28] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.

[29] Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart.

[30] Hal Daumé III and Eric Brill. 2004. Web search intent induction via automatic query reformulation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 49–52, Boston, Massachusetts, USA. Association for Computational Linguistics.

[31] Joe Davison, Joshua Feldman, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1173–1178. Association for Computational Linguistics.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[33] Sumanth Doddapaneni, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M Khapra. 2021. A primer on pretrained multilingual language models. *arXiv preprint arXiv:2107.00676*.

[34] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

[35] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054.

[36] Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. GSum: A general framework for guided neural abstractive summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.

[37] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All nlp tasks are generation tasks: A general pretraining framework.

[38] Kevin Duh, Katsuhito Sudoh, Xianchao Wu, Hajime Tsukada, and Masaaki Nagata. 2011. Generalized minimum bayes risk system combination. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1356–1360.

[39] Allyson Ettinger. 2020. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Trans. Assoc. Comput. Linguistics*, 8:34–48.

[40] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. Association for Computational Linguistics.

[41] Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.

[42] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017a. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

[43] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017b. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

[44] Jinlan Fu, Xuanjing Huang, and Pengfei Liu. 2021. Spanner: Named entity re-/recognition as span prediction. *arXiv preprint arXiv:2106.00641*.

[45] Chuang Gan, Yandong Li, Haoxiang Li, Chen Sun, and Boqing Gong. 2017. Vqs: Linking segmentations to questions and answers for supervised attention in vqa and question-focused semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1811–1820.

[46] Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

[47] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3356–3369.

[48] Leon Jay Gleser. 1996. Measurement, regression, and calibration.

[49] Joshua T Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

[50] David Grangier and Michael Auli. 2018. QuickEdit: Editing text & translations by crossing words out. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 272–282, New Orleans, Louisiana. Association for Computational Linguistics.

[51] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649.

[52] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

[53] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

[54] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. 2002. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1):389–422.

[55] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. Warp: Word-level adversarial reprogramming. *ArXiv*, abs/2101.00121.

[56] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification.

[57] Ahmed Hassan. 2013. Identifying web search query reformulation using concept based matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1000–1010, Seattle, Washington, USA. Association for Computational Linguistics.

[58] Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. BERTese: Learning to speak to BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3618–3623, Online. Association for Computational Linguistics.

[59] Junxian He, Wojciech Kryscinski, Bryan McCann, Nazneen Fatema Rajani, and Caiming Xiong. 2020a. Ctrlsum: Towards generic controllable text summarization. *CoRR*, abs/2012.04281.

[60] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020b. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

[61] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[62] Ari Holtzman, Peter West, Vered Schwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right.

[63] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339.

[64] Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401, Hong Kong, China. Association for Computational Linguistics.

[65] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. 2015. Salicon: Saliency in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1072–1080.

[66] Zhengbao Jiang, Antonios Anastasopoulos, Jun Araki, Haibo Ding, and Graham Neubig. 2020a. X-FACTR: Multilingual factual knowledge retrieval from pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5943–5959, Online. Association for Computational Linguistics.

[67] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2020b. How can we know when language models know? *CoRR*, abs/2012.00955.

[68] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020c. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

[69] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4163–4174. Association for Computational Linguistics.

[70] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

[71] Daniel Jurafsky and James H Martin. 2021. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.

[72] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. *arXiv preprint arXiv:1703.03129*.

[73] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland. Association for Computational Linguistics.

[74] Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7811–7818. Association for Computational Linguistics.

[75] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

[76] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

[77] Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas. Association for Computational Linguistics.

[78] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

[79] Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

[80] Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

[81] Sotiris B Kotsiantis, I Zaharakis, P Pintelas, et al. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24.

[82] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR.

[83] Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot learning.

[84] J. Lafferty, A. McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

[85] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

[86] Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

[87] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[88] Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

[89] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

[90] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

[91] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

[92] Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

[93] Hector J. Levesque. 2011. The winograd schema challenge. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

[94] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

[95] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[96] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

[97] Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified MRC framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859, Online. Association for Computational Linguistics.

[98] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.

[99] Chenxi Liu, Junhua Mao, Fei Sha, and Alan Yuille. 2017a. Attention correctness in neural image captioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[100] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3?

[101] Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102, Osaka, Japan. The COLING 2016 Organizing Committee.

[102] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017b. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798.

[103] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *CoRR*, abs/2103.10385.

[104] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020a. Multilingual denoising pre-training for neural machine translation. *Trans. Assoc. Comput. Linguistics*, 8:726–742.

[105] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

[106] Yixin Liu, Zi-Yi Dou, and Pengfei Liu. 2021c. RefSum: Refactoring neural summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1437–1448, Online. Association for Computational Linguistics.

[107] Yixin Liu and Pengfei Liu. 2021. Simcls: A simple framework for contrastive learning of abstractive summarization. *arXiv preprint arXiv:2106.01890*.

[108] Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. 2020b. Finbert: A pre-trained financial language representation model for financial text mining. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4513–4519. ijcai.org.

[109] Robert L. Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models.

[110] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13–23.

[111] Yao Lu, Max Bartolo, A. Moore, S. Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *ArXiv*, abs/2104.08786.

[112] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

[113] Andreĭ Andreevich Markov. 2006. An example of statistical investigation of the text eugene onegin concerning the connection of samples in chains. *Science in Context*, 19(4):591–600.

[114] Yvette Mathieu and Paul Sabatier. 1986. INTERFACILE: Linguistic coverage and query reformulation. In *Coling 1986 Volume 1: The 11th International Conference on Computational Linguistics*.

[115] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

[116] Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

[117] Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR*.

[118] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

[119] Tomas Mikolov, Ilya Sutskever, Kai Chen, G. Corrado, and J. Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[120] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2021. Natural instructions: Benchmarking generalization to new tasks from natural language instructions. *CoRR*, abs/2104.08773.

[121] Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. 2019. Exploring numeracy in word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3374–3380, Florence, Italy. Association for Computational Linguistics.

[122] Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy. Association for Computational Linguistics.

[123] Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572*.

[124] Rodrigo Frassetto Nogueira, Jannis Bulian, and Massimiliano Ciaramita. 2019. Multi-agent query reformulation: Challenges and the role of diversity. *ICLR Workshop on Deep Reinforcement Learning for Structured Prediction*.

[125] Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 161–168, Boston, Massachusetts, USA. Association for Computational Linguistics.

[126] Xuan Ouyang, Shuohuan Wang, Chao Pang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE-M: enhanced multilingual representation by aligning cross-lingual semantics with monolingual corpora. *CoRR*, abs/2012.15674.

[127] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

[128] Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

[129] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models.

[130] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

[131] Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

[132] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *ArXiv*, abs/2005.04611.

[133] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

[134] Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. 2020. XCOPA: A multilingual dataset for causal commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2362–2376, Online. Association for Computational Linguistics.

[135] Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *CoRR*, abs/1912.10165.

[136] Tingting Qiao, Jianfeng Dong, and Duanqing Xu. 2018. Exploring human-like attention supervision in visual question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[137] Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

[138] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

[139] Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training. In *arXiv*.

[140] Alec Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *arXiv*.

[141] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

[142] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

[143] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

[144] Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3236–3246.

[145] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

[146] Itsumi Saito, Kyosuke Nishida, Kosuke Nishida, and Junji Tomita. 2020. Abstractive summarization with combination of pre-trained sequence-to-sequence and saliency models. *arXiv preprint arXiv:2003.13028*.

[147] Nikunj Saunshi, Sadhika Malladi, and Sanjeev Arora. 2021. A mathematical exploration of why language models help solve downstream tasks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[148] Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? *arXiv preprint arXiv:2103.08493*.

[149] Timo Schick, Helmut Schmid, and Hinrich Schütze. 2020. Automatically identifying words that can serve as labels for few-shot text classification. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 5569–5578. International Committee on Computational Linguistics.

[150] Timo Schick and Hinrich Schütze. 2020. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8766–8774. AAAI Press.

[151] Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. *arXiv preprint arXiv:2104.07540*.

[152] Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training.

[153] Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze questions for few shot text classification and natural language inference.

[154] Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners.

[155] Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp.

[156] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40, San Diego, California. Association for Computational Linguistics.

[157] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

[158] Richard Shin, C. H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, D. Klein, J. Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. *ArXiv*, abs/2104.08768.

[159] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing (EMNLP)*.

[160] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4077–4087.

[161] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

[162] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 5926–5936. PMLR.

[163] Ekta Sood, Simon Tannert, Philipp Mueller, and Andreas Bulling. 2020. Improving natural language processing tasks with human gaze-guided neural attention. *Advances in Neural Information Processing Systems*, 33:6327–6341.

[164] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. VL-BERT: pre-training of generic visual-linguistic representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

[165] Yusuke Sugano and Andreas Bulling. 2016. Seeing with humans: Gaze-assisted neural image captioning. *arXiv preprint arXiv:1608.05203*.

[166] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019a. Videobert: A joint model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 7463–7472. IEEE.

[167] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *CoRR*, abs/2107.02137.

[168] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. ERNIE 2.0: A continual pre-training framework for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8968–8975. AAAI Press.

[169] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

[170] Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training.

[171] Kai Ming Ting and Ian H. Witten. 1997. Stacked generalizations: When does it work? In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 866–873. Morgan Kaufmann.

[172] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

[173] Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning. *CoRR*, abs/1806.02847.

[174] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, S. M. Ali Eslami, Oriol Vinyals, and Felix Hill. 2021. Multimodal few-shot learning with frozen language models. *CoRR*, abs/2106.13884.

[175] Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2020. A wrong answer or a wrong question? an intricate relationship between question reformulation and answer selection in conversational question answering. In *Proceedings of the 5th International Workshop on Search-Oriented Conversational AI (SCAI)*, pages 7–16, Online. Association for Computational Linguistics.

[176] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[177] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019a. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2153–2162. Association for Computational Linguistics.

[178] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019b. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

[179] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.

[180] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34.

[181] Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning.

[182] Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

[183] Dongling Xiao, Yu-Kun Li, Han Zhang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Ernie-gram: Pre-training with explicitly n-gram masked language modeling for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1702–1715. Association for Computational Linguistics.

[184] Han Xu, Zhang Zhengyan, Ding Ning, Gu Yuxian, Liu Xiao, Huo Yuqi, Qiu Jiezhong, Zhang Liang, Han Wentao, Huang Minlie, et al. 2021. Pre-trained models: Past, present and future. *arXiv preprint arXiv:2106.07139*.

[185] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2021a. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *CoRR*, abs/2105.13626.

[186] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021b. mt5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 483–498. Association for Computational Linguistics.

[187] Jheng-Hong Yang, Sheng-Chieh Lin, Rodrigo Nogueira, Ming-Feng Tsai, Chuan-Ju Wang, and Jimmy Lin. 2020. Designing templates for eliciting commonsense knowledge from pretrained sequence-to-sequence models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3449–3453, Barcelona, Spain (Online). International Committee on Computational Linguistics.

[188] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

[189] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics.

[190] Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3912–3921. Association for Computational Linguistics.

[191] Wenhao Yu, Chenguang Zhu, Zaitang Li, Zhiting Hu, Qingyun Wang, Heng Ji, and Meng Jiang. 2020. A survey of knowledge-enhanced text generation. *arXiv preprint arXiv:2010.04389*.

[192] Weizhe Yuan, Pengfei Liu, and Graham Neubig. 2021a. Can we automate scientific reviewing? *arXiv preprint arXiv:2102.00176*.

[193] Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021b. Bartscore: Evaluating generated text as text generation.

[194] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. 2021. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation.

[195] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

[196] Ruohan Zhang, Akanksha Saran, Bo Liu, Yifeng Zhu, Sihang Guo, Scott Niekum, Dana Ballard, and Mary Hayhoe. 2020b. Human gaze assisted artificial intelligence: a review. In *IJCAI: Proceedings of the Conference*, volume 2020, page 4951. NIH Public Access.

[197] Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA. Association for Computational Linguistics.

[198] Zhengyan Zhang, Yuxian Gu, Xu Han, Shengqi Chen, Chaojun Xiao, Zhenbo Sun, Yuan Yao, Fanchao Qi, Jian Guan, Pei Ke, Yanzheng Cai, Guoyang Zeng, Zhixing Tan, Zhiyuan Liu, Minlie Huang, Wentao Han, Yang Liu, Xiaoyan Zhu, and Maosong Sun. 2021. CPM-2: large-scale cost-effective pre-trained language models. *CoRR*, abs/2106.10715.

[199] Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics.

[200] Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, YuSheng Su, Haozhe Ji, Jian Guan, Fanchao Qi, Xiaozhi Wang, Yanan Zheng, Guoyang Zeng, Huanqi Cao, Shengqi Chen, Daixuan Li, Zhenbo Sun, Zhiyuan Liu, Minlie Huang, Wentao Han, Jie Tang, Juanzi Li, Xiaoyan Zhu, and Maosong Sun. 2020c. CPM: A large-scale generative chinese pre-trained language model. *CoRR*, abs/2012.00413.

[201] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models.

[202] Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021a. Meta-tuning language models to answer prompts better. *arXiv preprint arXiv:2104.04670*.

[203] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021b. Factual probing is [MASK]: learning vs. learning to recall. *CoRR*, abs/2104.05240.

[204] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. 2002. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263.

[205] Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2020. Enhancing factual consistency of abstractive summarization. *arXiv preprint arXiv:2003.08612*.

[206] Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J.C. Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 601–610, Jeju Island, Korea. Association for Computational Linguistics.

A Appendix on Pre-trained LMs

In this appendix we present some auxiliary information on pre-trained LMs that may be useful to the readers to better understand the current lay of the land with respect to this dynamic research area.

A.1 Evolution of Pre-trained LM Parameters

Fig. 7 lists several popular pre-trained models' statistics of parameters, ranging from 0 to 200 billion. GPT3, CPM2, and PanGu-α are the top three largest models with parameters greater than 150 billion.

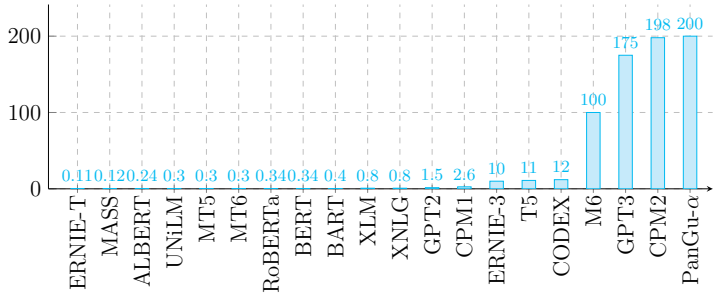


Figure 7: Comparison of the size of existing popular pre-trained language models.

A.2 Auxiliary Objective

In this subsection, more auxiliary objectives for pre-training language models have been listed.

- **Next Sentence Prediction (NSP)** (Devlin et al., 2019): A binary classification loss predicting whether two segments appear consecutively within a larger document, or are random unrelated sentences.
- **Sentence Order Prediction (SOP)** (Lan et al., 2020): A binary classification loss for predicting whether two sentences are in a natural or swapped order.
- **Capital Word Prediction (CWP)** (Liu et al., 2020b): A binary classification objective calculated over each word, predicting whether whether each word is capitalized or not.
- **Sentence Deshuffling (SDS)** (Liu et al., 2020b): A multi-class classification task to reorganize permuted segments.
- **Sentence distance prediction (SDP)** (Liu et al., 2020b) : A three-class classification task, predicting the positional relationship between two sentences (adjacent in the same document, not adjacent but in the same document, in different documents).
- **Masked Column Prediction (MCP)** (Yin et al., 2020): Given a table, recover the names and data types of masked columns.
- **Linguistic-Visual Alignment (LVA)** (Lu et al., 2019): A binary classification to Predict whether the text content can be aligned to visual content.
- **Image Region prediction (IRP)** (Su et al., 2020): Given an image whose partial features are masked (zeroed out), predict the masked regions.
- **Replaced Token Detection (RTD)** (Xiao et al., 2021): A binary classification loss predicting whether each token in corrupted input was replaced by a generative sample or not.
- **Discourse Relation Prediction (DRP)** (Sun et al., 2020): Predict the semantic or rhetorical relation between two sentences.
- **Translation Language Modeling (TLM)** (Lample and Conneau, 2019): Consider parallel sentences and mask words randomly in both source and target sentences.
- **Information Retrieval Relevance (IRR)** (Sun et al., 2020): Predict the information retrieval relevance of two sentences.
- **Token-Passage Prediction (TPP)** (Liu et al., 2020b): Identify the keywords of a passage appearing in the segment.
- **Universal Knowledge-Text Prediction (UKTP)** (Sun et al., 2021): Incorporate knowledge into one pre-trained language model.
- **Machine Translation (MT)** (Chi et al., 2021a) : Translate a sentence from the source language into the target language.
- **Translation Pair Span Corruption (TPSC)** (Chi et al., 2021a) : Predict the masked spans from a translation pair.
- **Translation Span Corruption (TSC)** (Chi et al., 2021a) : Unlike TPSC, TSC only masks and predicts the spans in one language.

- **Multilingual Replaced Token Detection (MRTD)** (Chi et al., 2021b): Distinguish real input tokens from corrupted multilingual sentences by a Generative Adversarial Network, where both the generator and the discriminator are shared across languages.
- **Translation Replaced Token Detection (TRTD)** (Chi et al., 2021b): Distinguish the real tokens and masked tokens in the translation pair by the Generative Adversarial Network.
- **Knowledge Embedding (KE)** (Wang et al., 2021): Encode entities and relations in knowledge graphs (KGs) as distributed representations
- **Image-to-text transfer (ITT)** (Wang et al., 2021): Is similar to the image caption that generates a corresponding description for the input image.
- **Multimodality-to-text transfer (MTT)** (Wang et al., 2021): Generate the target text based on both the visual information and the noised linguistic information.

A.3 Pre-trained Language Model Families

The increasing number of models makes it difficult for people to clearly grasp the differences between them. Based on this, we cluster the current mainstream pre-training models and characterize them from diverse dimensions.

Family	Models	LM	Pre-training Tasks			Corruption				Application
			Main	Auxiliary	Parallel	Mask	Replace	Delete	Permute	
GPT	GPT [139]	L2R	SLM	-	✗	-	-	-	-	NLG
	GPT-2 [140]	L2R	SLM	-	✗	-	-	-	-	NLG
	GPT-3 [16]	L2R	SLM	-	✗	-	-	-	-	NLG
	Codex [20]	L2R	SLM	-	✗	-	-	-	-	NLG
ELMo	ELMo [130]	L2R	SLM	-	✗	-	-	-	-	NLU, NLG
BERT	BERT [32]	Mask	CTR	NSP	✗	Tok	-	-	-	NLU
	RoBERTa [105]	Mask	CTR	-	✗	Tok	-	-	-	NLU
	SpanBERT [70]	Mask	CTR	-	✗	Span	-	-	-	NLU
	DeBERTa [60]	Mask	CTR	-	✗	Tok	-	-	-	NLU
	SciBERT [7]	Mask	CTR	NSP	✗	Tok	-	-	-	Sci-NLU
	BioBERT [89]	Mask	CTR	NSP	✗	Tok	-	-	-	Bio-NLU
	ALBERT [87]	Mask	CTR	SOP	✗	Tok	-	-	-	mSent
	FinBERT [108]	Mask	CTR	CWP, SDS, SDP, TPP	✗	Span	-	-	Sent	Fin-NLU
	VLBERT [164]	Mask	CTR	IRP	✓	Tok, Region	-	-	-	VLU
	ViLBERT [110]	Mask	CTR	IRP, LVA	✓	Tok, Region	-	-	-	VLU
	BEIT [5]	Mask	CTR,FTR	-	✗	Visual "Tok"[7]	-	-	-	VLU
	VideoBERT [166]	Mask	CTR	LVA	✓	Tok, Frame	-	-	-	VLU
	TaBERT [189]	Mask	CTR	MCP	✓	Tok, Column	-	-	-	Tab2Text
	mBERT [32]	Mask	CTR	NSP	✗	Tok	-	-	-	XLU
	TinyBERT [69]	Mask	CTR	NSP	✗	Tok	-	-	-	XLU
ERNIE	ERNIE-T [199]	Mask	CTR	NSP	✗	Tok, Entity	-	-	-	NLU
	ERNIE-B [169]	Mask	CTR	-	✗	Tok,Entity, Phrase	-	-	-	NLU
	ERNIE-NG [183]	Mask	CTR	RTD	✗	N-gram	Tok	-	-	NLU
	ERNIE-B2 [168]	Mask	CTR	CWP,SDS,SOP, SDP,DRP,IRR	✗	Entity, Phrase	-	-	Sent	NLU
	ERNIE-M [126]	LPM	CTR	-	✓	Tok	-	-		XLU, XLG
	ERNIE-B3 [167]	Mask	CTR	SOP,SDP,UKTP	✗	Entity, Phrase	-	-	-	NLU
BART	BART [94]	En-De	FTR	-	✗	Tok	Span	Tok	Sent,Doc	NLU, NLG
	mBART [104]	En-De	FTR	-	✗	Span	-	-	Sent	NLG
UniLM	UniLM1 [35]	LPM	SLM,CTR	NSP	✗	Tok	-	-	-	NLU, NLG
	UniLM2 [6]	LPM	SLM,CTR	-	✗	Tok	-	-	Tok	NLU, NLG
T5	T5 [141]	En-De	CTR	-	✗	-	Span	-	-	NLU, NLG
	mT5 [186]	En-De	CTR	-	✗	-	Span	-	-	XLU, XLG
	mT6 [22]	En-De	CTR	MT,TPSC,TSC	✓	-	Span	-	-	XLU, XLG
	ByT5 [185]	En-De	CTR	-	✗	-	byte-span	-	-	XLU, XLG
XLM	XLM [86]	LPM	CTR	TLM	✓	Tok	-	-	-	XLU, XLG
	XLM-R [28]	Mask	CTR	-	✗	Tok	-	-	-	XLU
	XLM-E [23]	Mask	CTR	MRTD,TRTD	✗	-	Tok	-	-	XLU, XLG
CPM	CPM [200]	L2R	SLM	-	✗	-	-	-	-	NLG
	CPM-2 [198]	En-De	CTR	-	✗	Span	-	-	-	NLU,NLG
Other	XLNet [188]	L2R	SLM	-	✗	-	-	-	Tok	NLU
	PanGu-α [194]	L2R	SLM	-	✗	-	-	-	-	NLG
	ELECTRA [26]	Mask	CTR	RTD	✗	Tok	Tok	-	-	NLU,NLG
	MASS [162]	En-De	CTR	-	✗	Span	-	-	-	NLG
	PEGASUS [195]	En-De	CTR	-	✗	Tok, Sent	-	-	-	Summarization
	M6 [179]	En-De	CTR	ITT,MTT	✗	Span	-	-	-	NLG

Table 13: A detailed illustration of different pre-trained models characterized by the four aspects. "**Parallel**" represents if parallel data have been used for pre-training. `Sci`, `Bio`, `Fin`, `K` represent scientific, biomedical, financial, and knowledge, respectively. `Tok`, `Sent`, `Doc` denote token, sentence and document, respectively. `Region`, `Frame` denote basic units of images and video respectively.