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