Prompt-tuning

and different methods of prompt-tuning

Ilya German, Feb 2024

The reason of Prompt-tuning appearance

Example:

- Two sentences, need to say whether their meanings match

Solutions:

- Learning models from scratch
- Fine-tuning generative models
- Prompt-tuning

The reason of Prompt-tuning appearance

Issue: we have small dataset

- Simple models are inefficient
- Not pre-trained large models are inefficient due to small dataset.
- The task is not extensive enough
- Fine-Tuning works great, but not always

We need to use pre-trained large models and tune it.

Prompt-tuning

Prompt-tuning is a flexible technique that enables language models to adapt to specific tasks by integrating task-specific cues or prompts

Prompt-tuning adds task-specific prompts to the input, and these prompt parameters are updated independently of the pretrained model parameters which are frozen.

Prompt-tuning

<S1> - statement 1
<S2> - statement 2

<S1> [Mask] <S2> → {‘yes’, ‘maybe’, ‘no’} → **Cloze-style task model**

<S1> yes <S2> → **similar**

Prompt-tuning

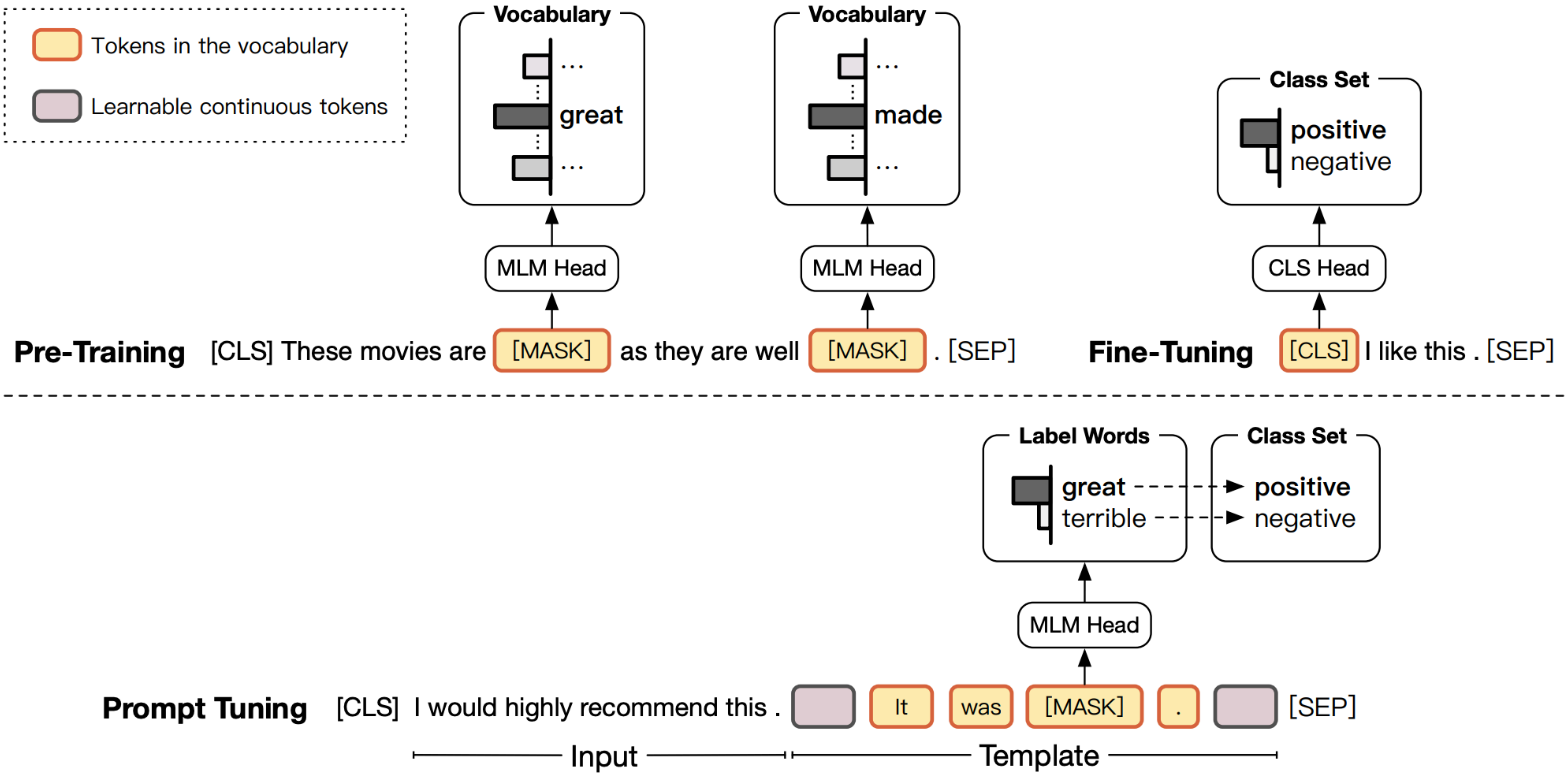
Idea

- Classification task $\mathcal{T} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X}$ is the instance set and $\mathcal{Y}$ is the class set.
- $\mathcal{M}$ - pre-trained model, solving cloze-style tasks.
- Template $T(\cdot)$ for generating prompt. $\mathcal{V}$ - set of possible words.
- $x\_prompt = T(x)$, at least one [MASK] in x_prompt

$$x \xrightarrow{T(\cdot)} x\_prompt \xrightarrow{\mathcal{M}} v \longrightarrow y$$

$$\mathcal{V}$$

Prompt-tuning



AutoPrompt

Reason: we want to get the best templates.

Idea: gradient search

Text → tokens → embeddings ⇢ gradient search

$\boldsymbol{x}_{\text{inp}}$

$\downarrow$

$\boldsymbol{x}_{\text{prompt}}$

AutoPrompt

Method allows us to generate a template automatically. Gradient descent guarantee the best embedding for prompt*.

$$
\begin{tikzcd}
\boldsymbol{x}_{\text{inp}} \arrow[r] & \textcolor{red}{\text{Template } T(\cdot)} \arrow[r] & \boldsymbol{x}_{\text{prompt}} \arrow[r] & p([\text{MASK}]|\boldsymbol{x}_{\text{prompt}}) \arrow[d] \\
\mathcal{V}_{\text{cand}} = \underset{w \in \mathcal{V}}{\text{top-}k} \left[\boldsymbol{w}_{\text{in}}^{T} \nabla \log p(y|\boldsymbol{x}_{\text{prompt}}) \right] \arrow[ur, dashed] & & & p(y|\boldsymbol{x}_{\text{prompt}}) = \sum_{w \in \mathcal{V}_y} p([\text{MASK}] = w|\boldsymbol{x}_{\text{prompt}}) \arrow[lll]
\end{tikzcd}
$$

Template is a learning function

*considering loss function is convex

AutoPrompt

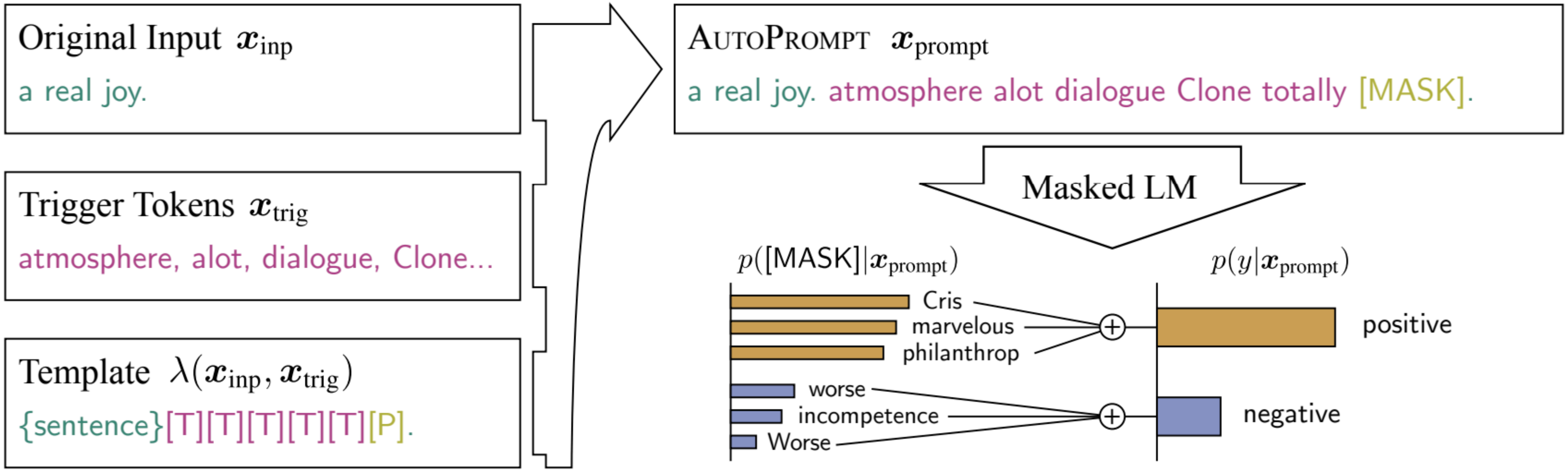


Figure 1: **Illustration of AUTOPROMPT** applied to probe a masked language model's (MLM's) ability to perform sentiment analysis. Each input, $\boldsymbol{x}_{\text{inp}}$, is placed into a natural language prompt, $\boldsymbol{x}_{\text{prompt}}$, which contains a single [MASK] token. The prompt is created using a template, λ, which combines the original input with a set of trigger tokens, $\boldsymbol{x}_{\text{trig}}$. The trigger tokens are shared across all inputs and determined using a gradient-based search (Section 2.2). Probabilities for each class label, y, are then obtained by marginalizing the MLM predictions, $p([\text{MASK}]|\boldsymbol{x}_{\text{prompt}})$, over sets of automatically detected label tokens (Section 2.3).

AutoPrompt

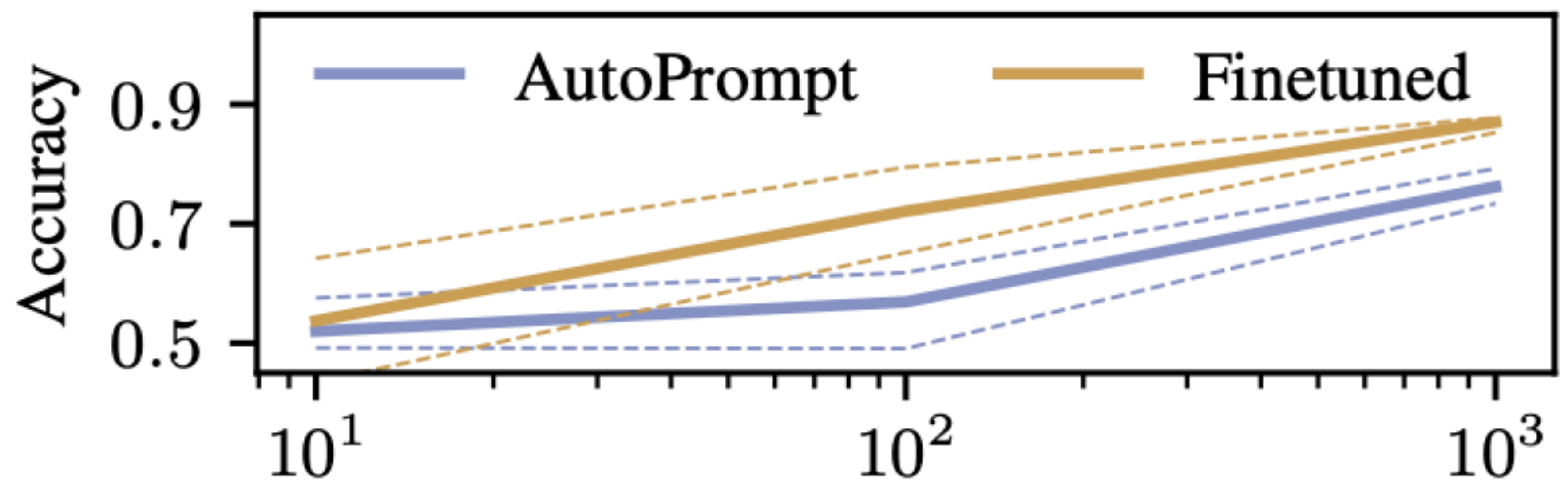
Advantage: we can generate better prompts and do not waste human resources.

Still not magnificent. We can solve only simple tasks.

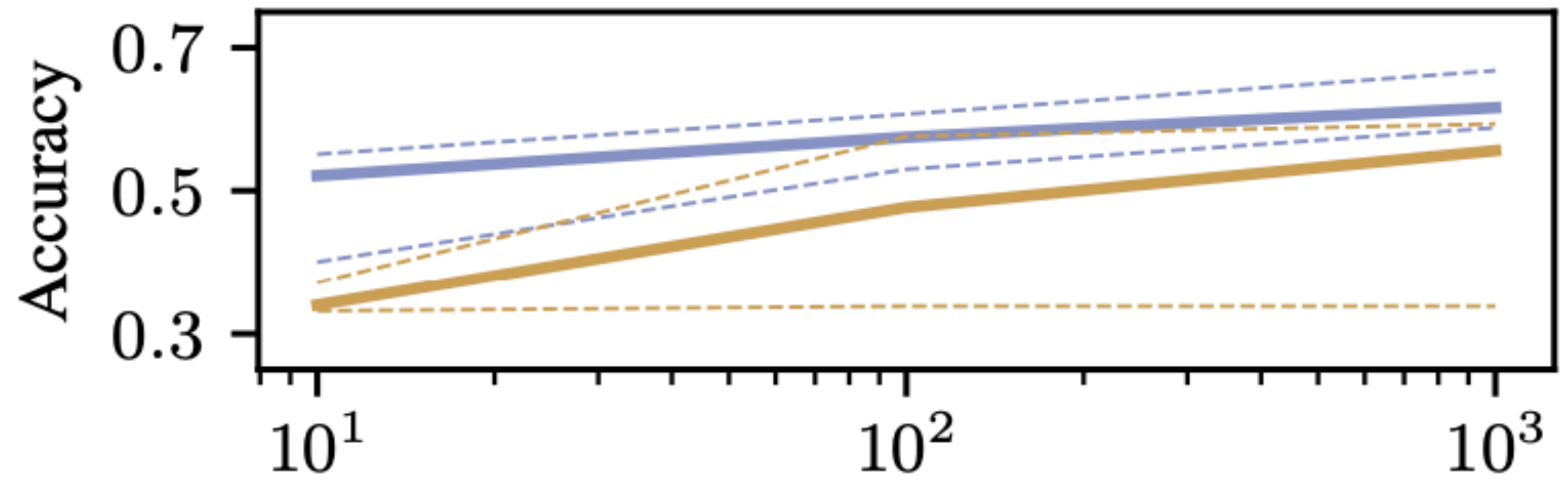
Model	**Dev**	**Test**
BiLSTM	-	82.8†
BiLSTM + ELMo	-	89.3†
BERT (linear probing)	85.2	83.4
BERT (finetuned)	-	93.5†
RoBERTa (linear probing)	87.9	88.8
RoBERTa (finetuned)	-	96.7†
BERT (manual)	63.2	63.2
BERT (AUTOPROMPT)	80.9	82.3
RoBERTa (manual)	85.3	85.2
RoBERTa (AUTOPROMPT)	91.2	91.4

Table 1: **Sentiment Analysis** performance on the SST-2 test set of supervised classifiers (top) and fill-in-the-blank MLMs (bottom). Scores marked with † are from the GLUE leaderboard: http://gluebenchmark.com/leaderboard.

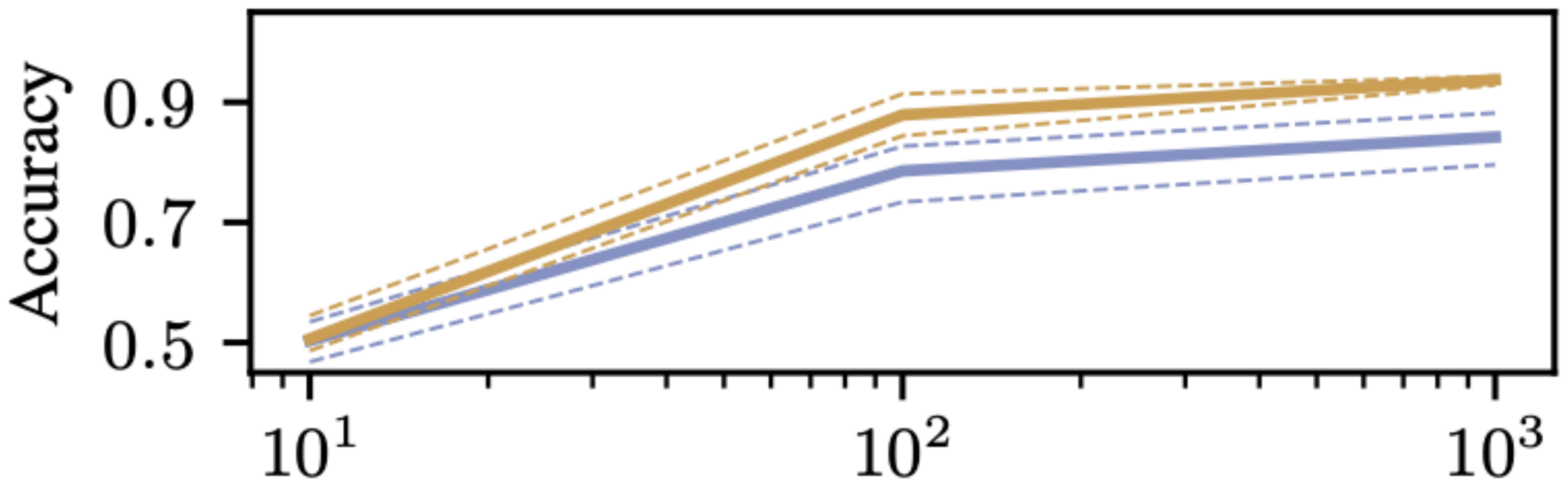
AutoPrompt



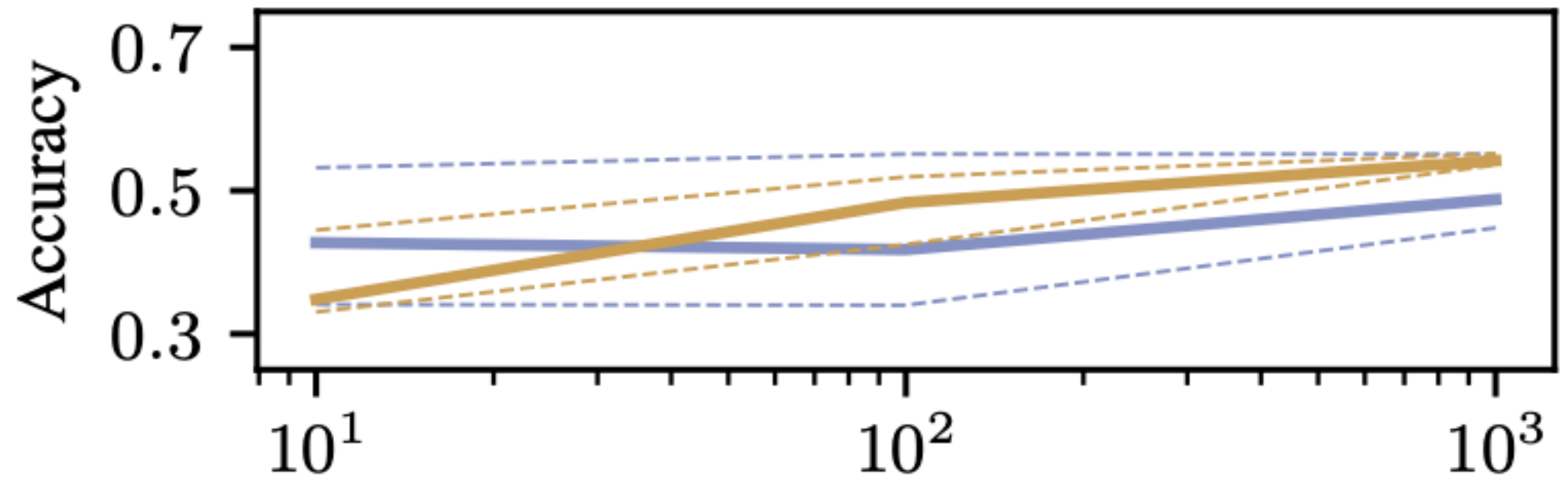
(a) BERT on SST-2



(d) RoBERTa on SICK-E



(b) RoBERTa on SST-2



(c) BERT on SICK-E

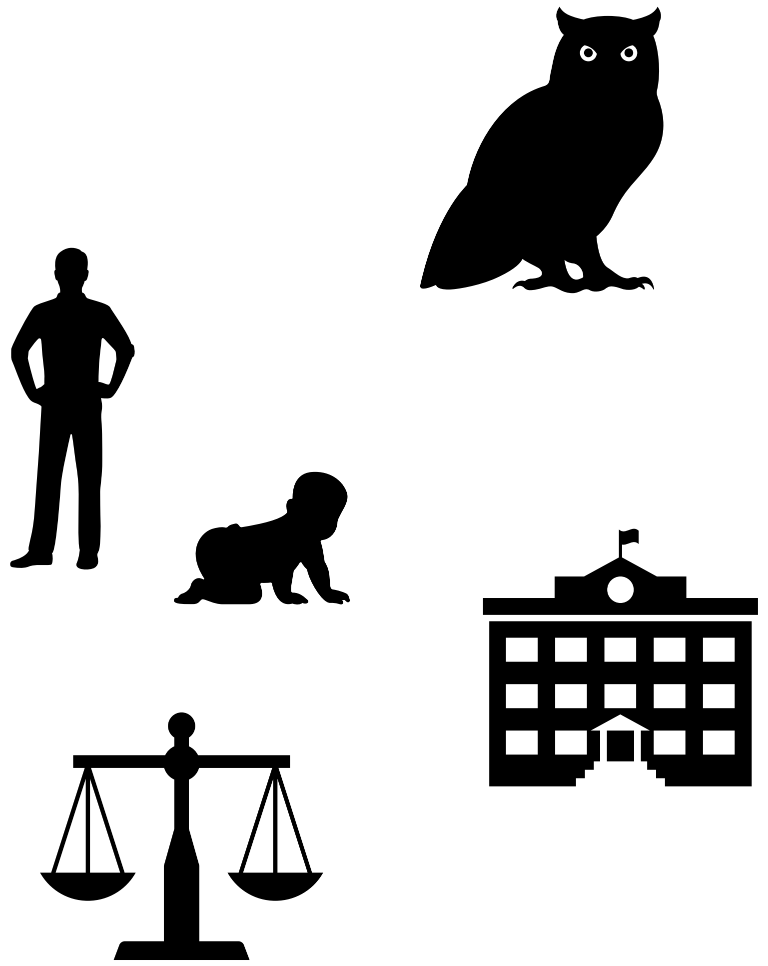
Figure 2: **Effect of Training Data** on sentiment analysis and NLI for AUTOPROMPT vs. finetuning. X-axis is the number of data points used during training. Error bars plot the max. and min. accuracies observed over 10 independent runs. *(revised since EMNLP version)*.

Prompt-Tuning with Rules (PTR)

AutoPrompt is good. Hence, it is still challenging for prompt tuning to address many-class classification tasks.
AutoPrompt becomes useless when we cannot get minimum of loss function.

Let’s split one task to many!

Prompt-Tuning with Rules (PTR)

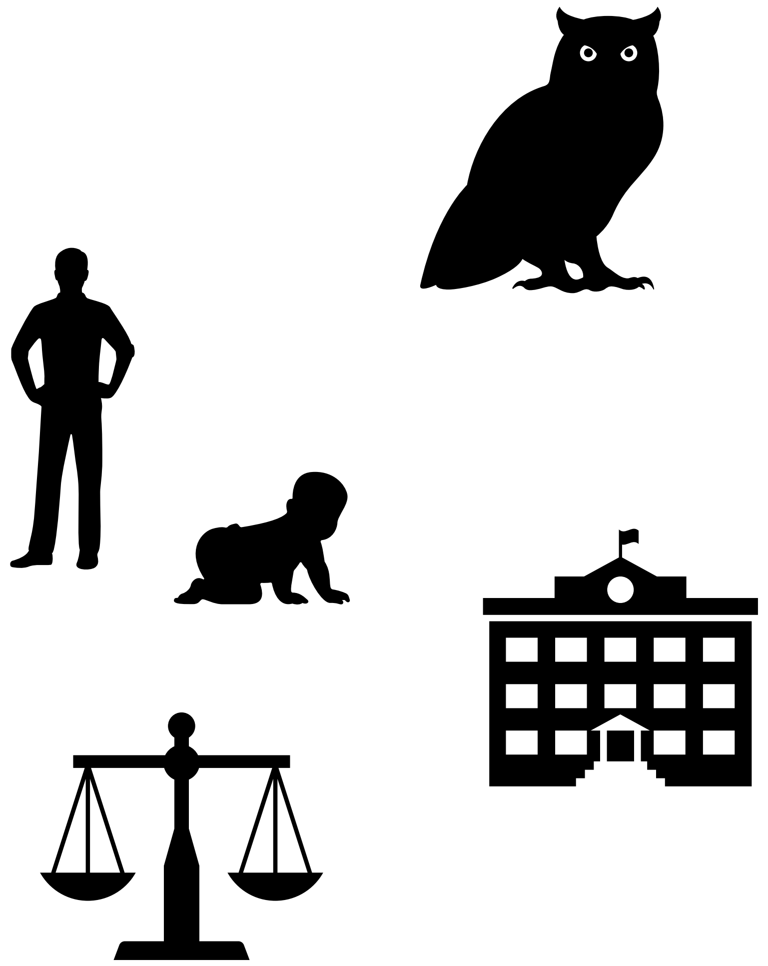


Take the RE task.
For any classification task $\mathcal{T} = \{\mathcal{X}, \mathcal{Y}\}$, we design a conditional function set $\mathcal{F}$. Each conditional function $f \in \mathcal{F}$ determines whether the function input meets certain conditions.

$$f(x, \texttt{person})$$

$$f(x, \texttt{'s parent was}, y)$$

Prompt-Tuning with Rules (PTR)



Take the RE task.
For any classification task $\mathcal{T} = \{\mathcal{X}, \mathcal{Y}\}$, we design a conditional function set $\mathcal{F}$. Each conditional function $f \in \mathcal{F}$ determines whether the function input meets certain conditions.

$$f_{e_s}(x, \texttt{person}) \wedge f_{e_s,e_o}(x, \texttt{'s parent was}, y) \wedge f_{e_o}(y, \texttt{person}) \rightarrow \text{“person:parent”},$$

$$f_{e_s}(x, \texttt{organization}) \wedge f_{e_s,e_o}(x, \texttt{'s parent was}, y) \wedge f_{e_o}(y, \texttt{organization}) \rightarrow \text{“organization:parent”}.$$

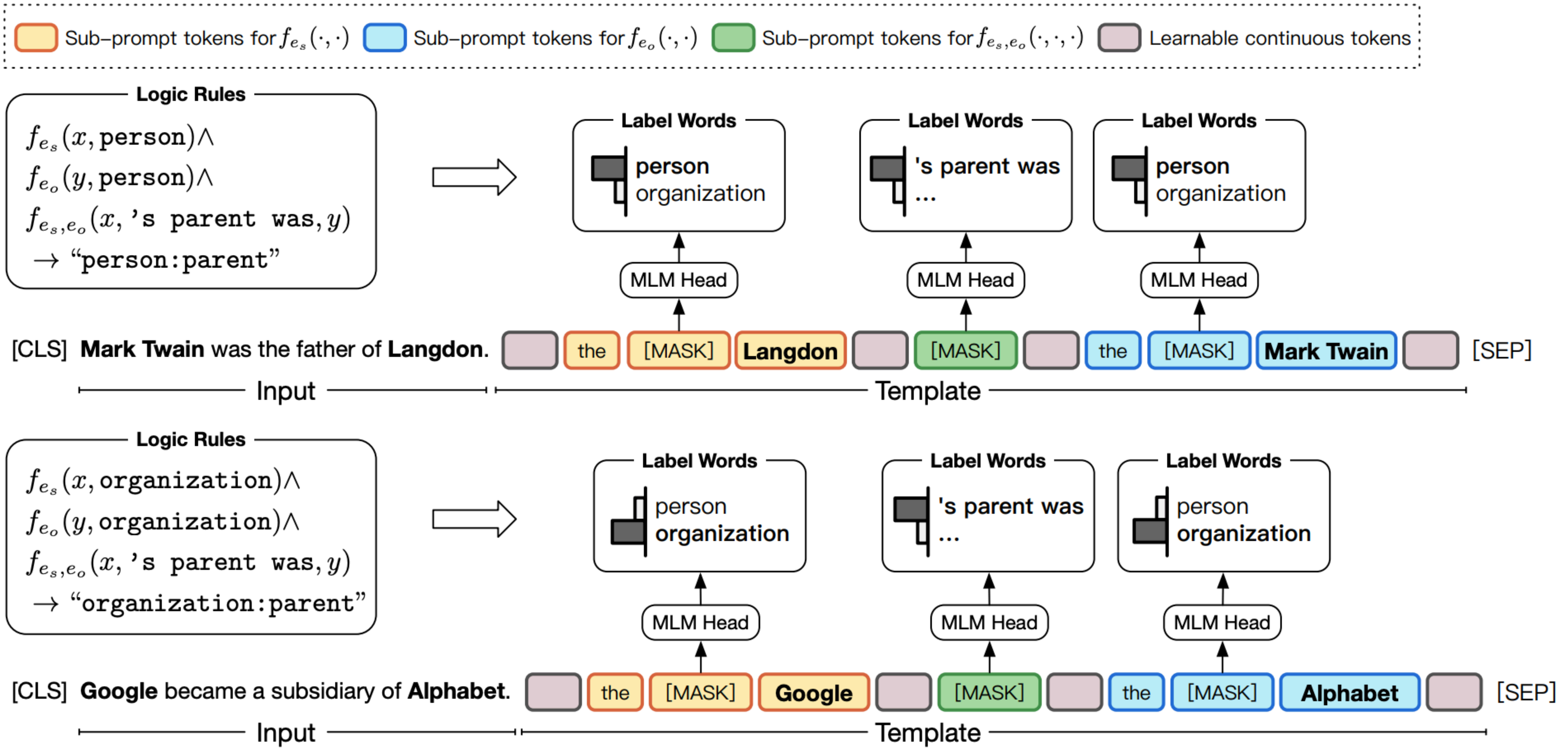
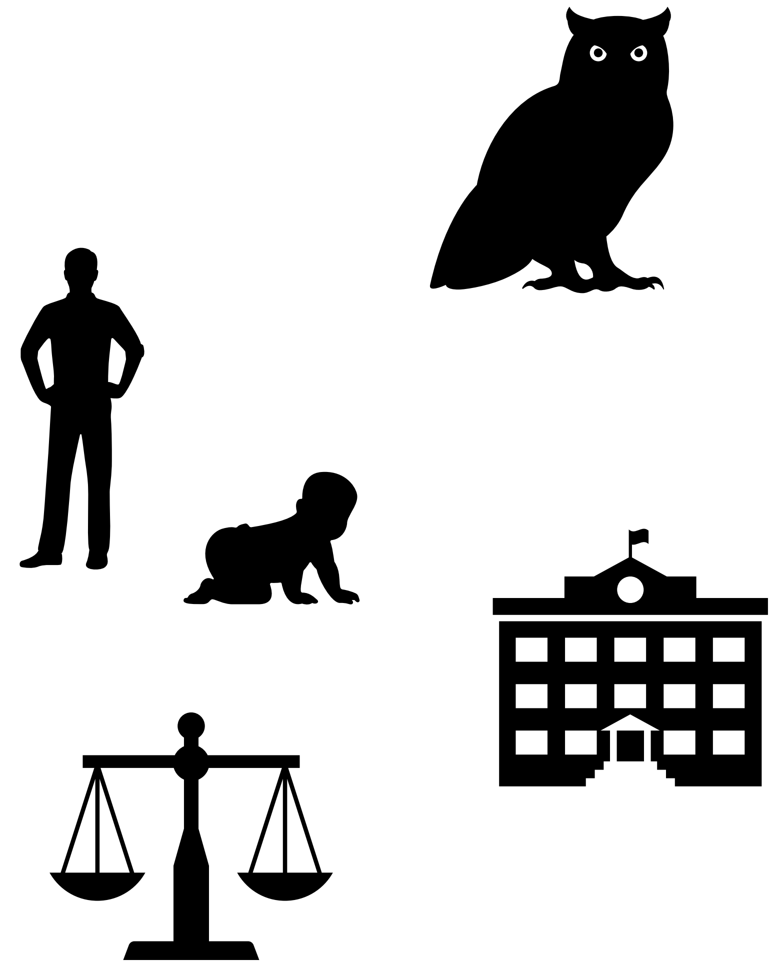


Figure 2: Illustration of PTR. By composing the sub-prompts of the conditional functions $f_{e_s}(\cdot,\cdot)$, $f_{e_o}(\cdot,\cdot)$ and $f_{e_s,e_o}(\cdot,\cdot,\cdot)$, we could easily get an effective prompt to distinguish "person:parent" and "organization:parent".

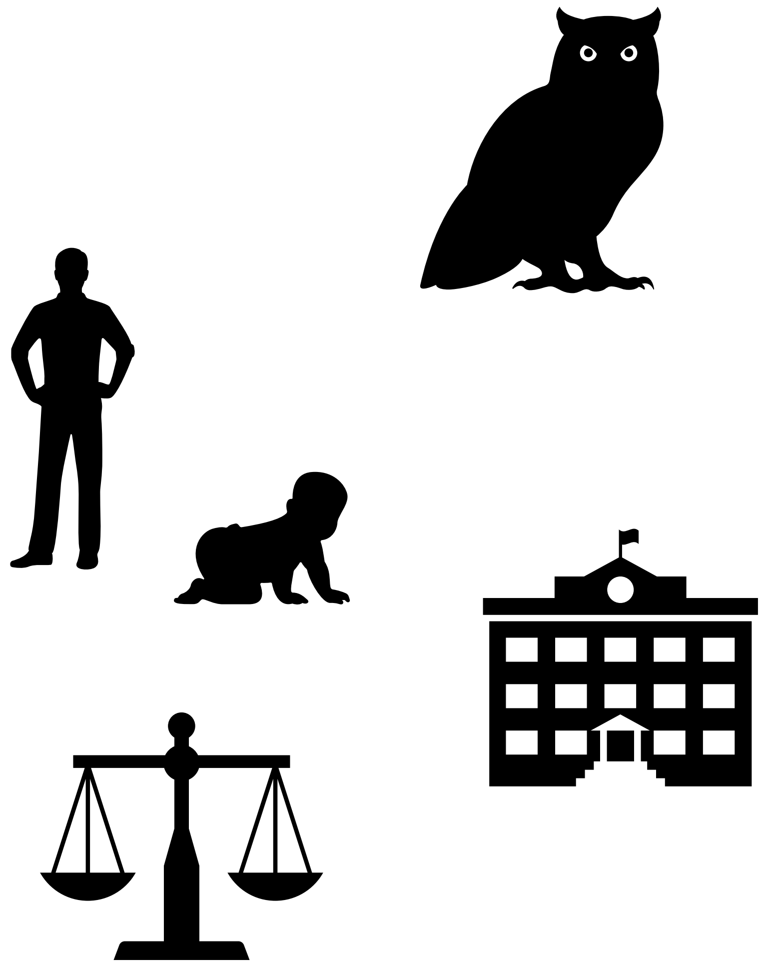
Prompt-Tuning with Rules (PTR)



$$p(y|x) = \prod_{j=1}^{n} p(\texttt{[MASK]}_j = \phi_j(y)|T(x)),$$

$$\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \log \prod_{j=1}^{n} p\big(\texttt{[MASK]}_j = \phi_j(y)|T(x)\big).$$

PTR and KnowPrompt



Standard Supervised Setting						
Methods	Extra Data	SemEval	DialogRE†	TACRED	TACRED-Revisit	Re-TACRED
Fine-tuning pre-trained models						
Fine-tuning-[RoBERTa]	w/o	87.6	57.3	68.7	76.0	84.9
SpanBERT [30]	w/	-	-	70.8	78.0	85.3
KnowBERT [38]	w/	89.1	-	71.5	79.3	89.1
LUKE [52]	w/	-	-	**72.7**	80.6	-
MTB [3]	w/	89.5	-	70.1	-	-
GDPNet [51]	w/o	-	64.9	71.5	79.3	-
Dual [2]	w/o	-	67.3	-	-	-
Prompt-tuning pre-trained models						
PTR-[RoBERTa] [22]	w/o	89.9	63.2	72.4	81.4	90.9
KnowPrompt-[RoBERTa]	w/o	**90.2** (+0.3)	**68.6** (+5.4)	72.4 (-0.3)	**82.4** (+1.0)	**91.3** (+0.4)

Table 3: Standard RE performance of F_1 scores (%) on different test sets. "w/o" means that no additional data is used for pre-training and fine-tuning, yet "w/" means that the model uses extra data for tasks. It is worth noting that "†" indicates we exceptionally rerun the code of KnowPrompt and PTR with RoBERT_base for a fair comparison with current SOTA models on DialogRE. Subscript in red represents advantages of KnowPrompt over the best results of baselines. Best results are bold.

References

- AutoPrompt: Epiciting Knowledge from Language Models with Automatically Generated Prompts
- PTR: Prompt Tuning with Rules for Text Classification
- KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction