IMPERIAL

NLP Reading Group

"Fantastic Prompts and where to find them"

March 5th 2024

Fantastic Prompts and where to find them

Overview:

1. History of the prompt

2. **The Power of Scale for Parameter-Efficient Prompt Tuning** (“Soft-prompts”)

3. **Towards a Unified View of Parameter-Efficient Transfer Learning**

4. **Black-Box Tuning for Language-Model-as-a-Service**

Background & (brief and recent history) - Part 1 - Classical Approaches

1. We “started*” with **static word-embeddings** (e.g. word2vec, glove, fasttext) + **Custom Model** (e.g. FFN, BiLSTM + CRF) (2013)
    a. Model Improvement happened via: “Full Training for each individual task”

2. We then had **contextualised word-embeddings** (e.g. ELMO, BERT) + Specialised Layers for different tasks. (2017)
    a. Model Improvement happened via: “Fine-tuning the full model for each individual task”

3. We then went into the area of **multi-task training** (e.g. The natural language decathlon) (2018)
    a. Model Improvement happened via: “Fine-tuning the full model for all tasks at the same time”

4. We then went into era of **efficient model adaptations** (e.g. Adapter Layers) - aka “Transfer Learning” (2019)
    a. Model Improvement happened via: “Fine-tuning specific layers (for all tasks at the same time)”

*since ~2013

Background & (brief and recent history) - Part 2 - The journey to Prompts

5. The “**extension of Multi-task learning**” entering the era of **natural language based tasks** definitions (e.g. T5) (2019)
    a. The idea was to use text input (i.e. natural language context) to provide the task context for the model.
    b. “The Birth of the Prompt” (but very few people did new tasks with these models)

6. At the same time we got very powerful “contextual word embeddings” and **large language models**. (e.g. GPT-3) (2020)
    a. Now prompt-engineering was properly born (however, it was completely unstructured), as the model would not follow commands, but complete the text.
    b. E.g. The prompt: Please give me a layout of a reading-group presentation. Would could give: “Please give me a layout of a coursework presentation”

7. We then went into “**massive multi-task fine-tuning**” (using text based tasks), (e.g. Flan 2021), which then followed with the full “**InstructGPT**” (2022).
    a. The idea was to now push the “limit” of multi-task training with text-based context.
    b. Now the current prompt engineering was born. You can now use these models for specific tasks and “tune the prompt” to solve existing or new tasks.

8. Finally we went into the era of **ChatGPT** style models, by continuing to fine-tune instruct models using additional “**alignment**” (aka fine-tuning) with **Human Feedback** (either via RL or direct optimisation).
    a. Now the models exceed the instruction dataset and responses are even better from a human perspective.

The Power of Scale for Parameter-Efficient Prompt Tuning

The Power of Scale for Parameter-Efficient Prompt Tuning

Motivation:

1. We are spending hours prompt engineering

2. Generally there are so many guides to prompt-engineering.
    a. Chain-of-Thought
    b. Chain-of-Code
    c. Structured output (e.g. JSON)

3. Various approaches proposed for auto-prompt generation (e.g. AutoPrompt [clever template based])

4. Also, fine-tuning is very expensive and serving multiple models with different weights can be infeasible.

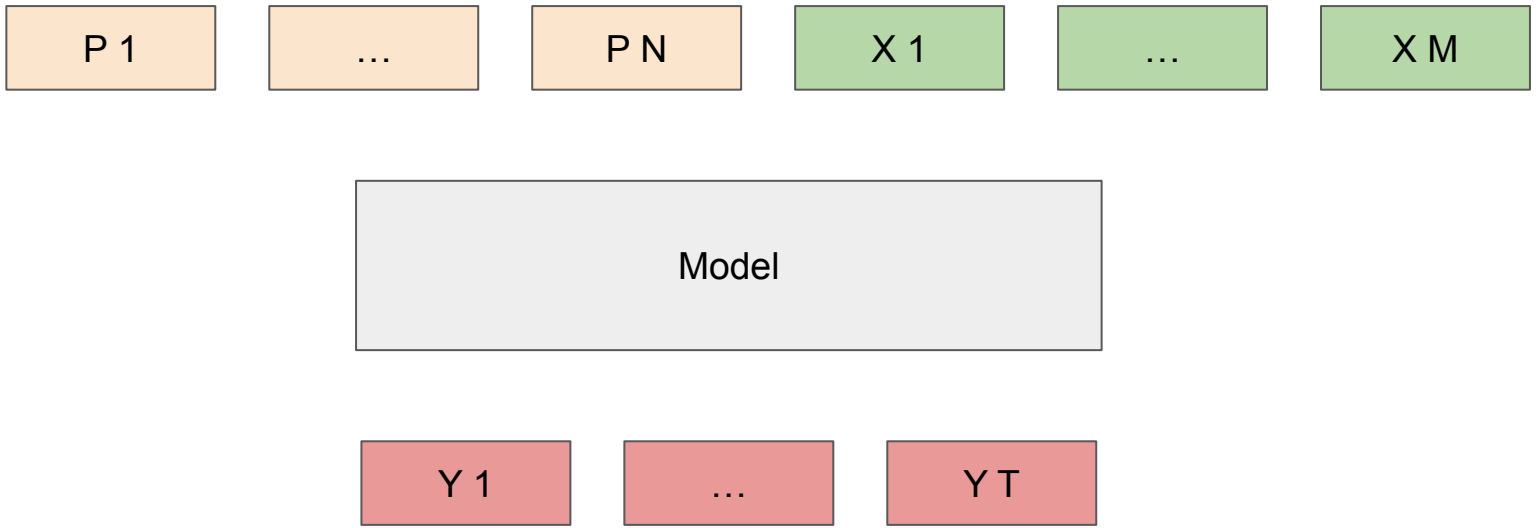
The Power of Scale for Parameter-Efficient Prompt Tuning

Key Idea: “Soft-prompts”

> Tune prompt-vectors (aka Soft-prompts) using gradients (while keeping the rest of the model fixed).

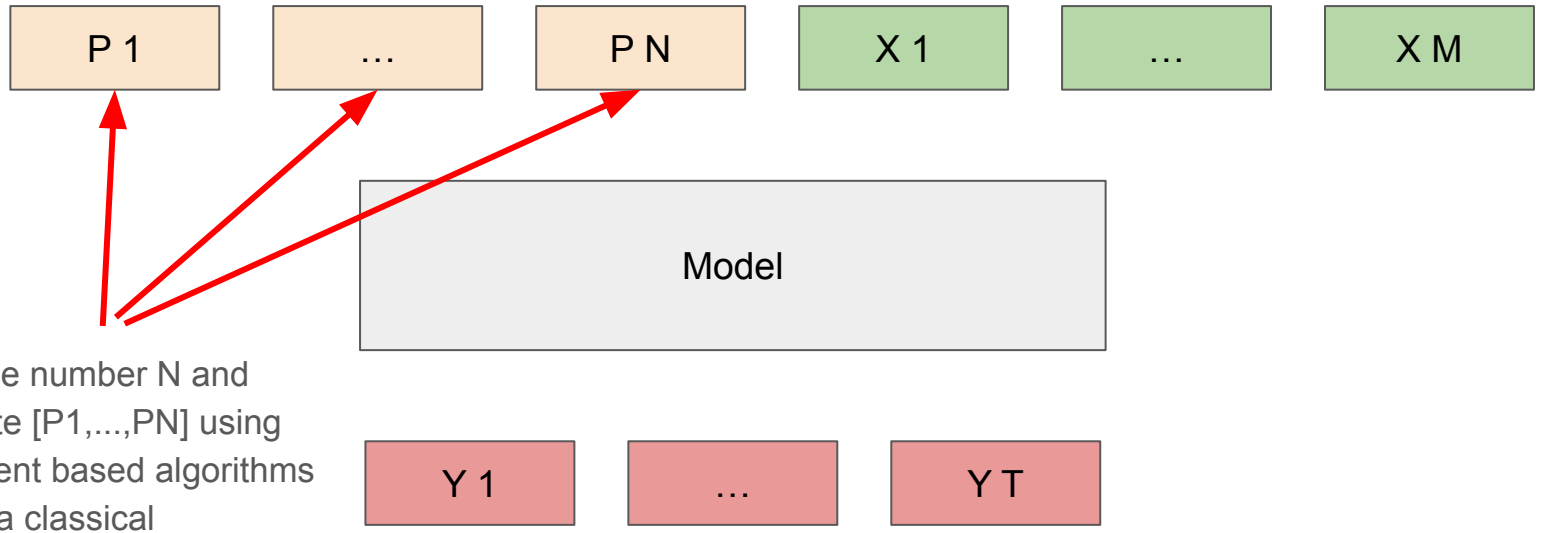
The Power of Scale for Parameter-Efficient Prompt Tuning

LLM output [Y1, …,YT] given N prompt tokens and M input tokens.



The Power of Scale for Parameter-Efficient Prompt Tuning

A fixed number of Soft-prompt “tokens” are tuned using gradient descent.



Fix the number N and update [P1,...,PN] using gradient based algorithms (and a classical log-likelihood objective).

The Power of Scale for Parameter-Efficient Prompt Tuning

Key Results:

As the model size increases prompt-tuning “closes the gap” with fine-tuning, while hand-crafted prompts (albeit with GPT-3) underperforms any fine-tuning.

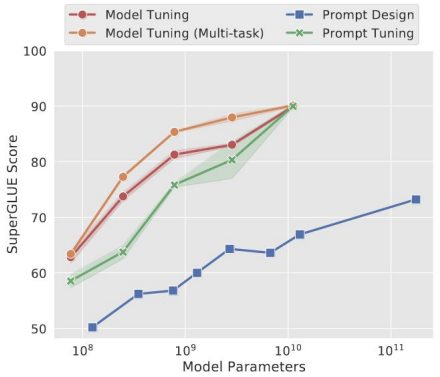
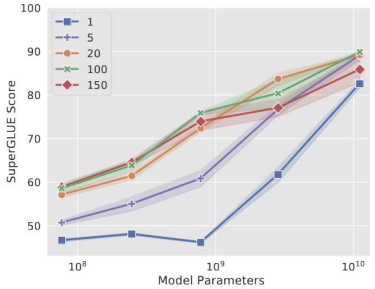


Figure 1: Standard **model tuning** of T5 achieves strong performance, but requires storing separate copies of the model for each end task. Our **prompt tuning** of T5 matches the quality of model tuning as size increases, while enabling the reuse of a single frozen model for all tasks. Our approach significantly outperforms few-shot **prompt design** using GPT-3. We show mean and standard deviation across 3 runs for tuning methods.

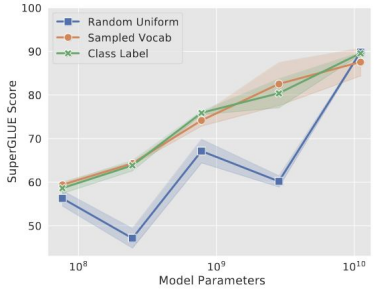
The Power of Scale for Parameter-Efficient Prompt Tuning

Important Technical Considerations:

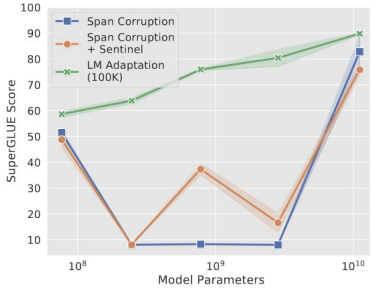
1. Initialisation (random, random token embeddings, output classes, …)
2. Length of Prompt
3. Training Steps [they do 30K, LR=0.3, Batchsize=32, Adafactor(1e-5,0.8)]
4. [In their case] LM-adaptation of T5.



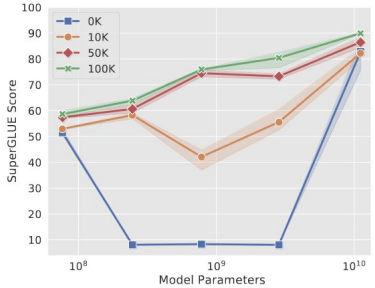
(a) Prompt length



(b) Prompt initialization



(c) Pre-training method



(d) LM adaptation steps

IMPERIAL

The Power of Scale for Parameter-Efficient Prompt Tuning

Additional nice thing:

1. Seems to generalise better to out-of-domain (but similar tasks) compared to model fine-tuning

Dataset	Domain	Model	Prompt	Δ
SQuAD	Wiki	94.9 ±0.2	94.8 ±0.1	−0.1
TextbookQA	Book	54.3 ±3.7	**66.8** ±2.9	+12.5
BioASQ	Bio	77.9 ±0.4	**79.1** ±0.3	+1.2
RACE	Exam	59.8 ±0.6	**60.7** ±0.5	+0.9
RE	Wiki	88.4 ±0.1	**88.8** ±0.2	+0.4
DuoRC	Movie	**68.9** ±0.7	67.7 ±1.1	−1.2
DROP	Wiki	**68.9** ±1.7	67.1 ±1.9	−1.8

Table 1: F1 mean and stddev for models trained on SQuAD and evaluated on out-of-domain datasets from the MRQA 2019 shared task. Prompt tuning tends to give stronger zero-shot performance than model tuning, especially on datasets with large domain shifts like TextbookQA.

The Power of Scale for Parameter-Efficient Prompt Tuning

Conclusion:

1. Very easy and effective method.

2. Reaches Model Fine-Tuning performance.

3. More robust when out-of-domain data!

Towards a Unified View of Parameter-Efficient Transfer Learning

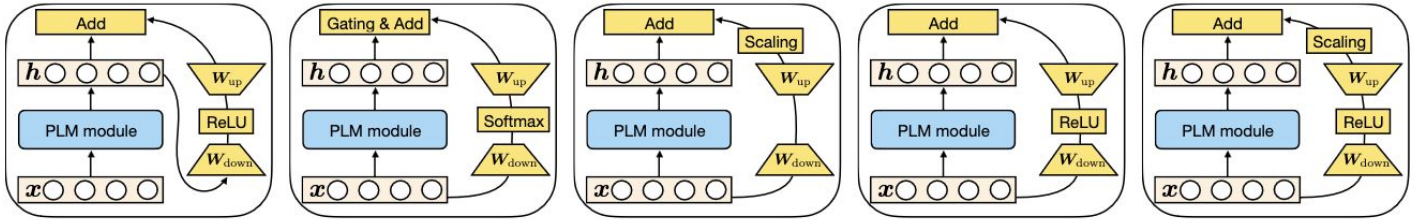
IMPERIAL

Towards a Unified View of Parameter-Efficient Transfer Learning

Key Idea: Unifying various approaches:

1. Prefix tuning
2. Adapters
3. Lora into a single framework.

Towards a Unified View of Parameter-Efficient Transfer Learning



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

Figure 3: Graphical illustration of existing methods and the proposed variants. "PLM module" represents a certain sublayer of the PLM (e.g. attention or FFN) that is frozen. "Scaled PA" denotes scaled parallel adapter. We do not include multi-head parallel adapter here to save space.

Towards a Unified View of Parameter-Efficient Transfer Learning

Table 1: Parameter-efficient tuning methods decomposed along the defined design dimensions. Here, for clarity, we directly write the adapter nonlinear function as ReLU which is commonly used. The bottom part of the table exemplifies new variants by transferring design choices of existing approaches.

Method	$\Delta \boldsymbol{h}$ functional form	insertion form	modified representation	composition function
		Existing Methods		
Prefix Tuning	$\text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v$	parallel	head attn	$\boldsymbol{h} \leftarrow (1-\lambda)\boldsymbol{h} + \lambda\Delta\boldsymbol{h}$
Adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	sequential	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
LoRA	$\boldsymbol{x}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$	parallel	attn key/val	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$
		Proposed Variants		
Parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Muti-head parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	head attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Scaled parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$

Towards a Unified View of Parameter-Efficient Transfer Learning

Results of previous approaches:

- Missing (soft-prompt approach)

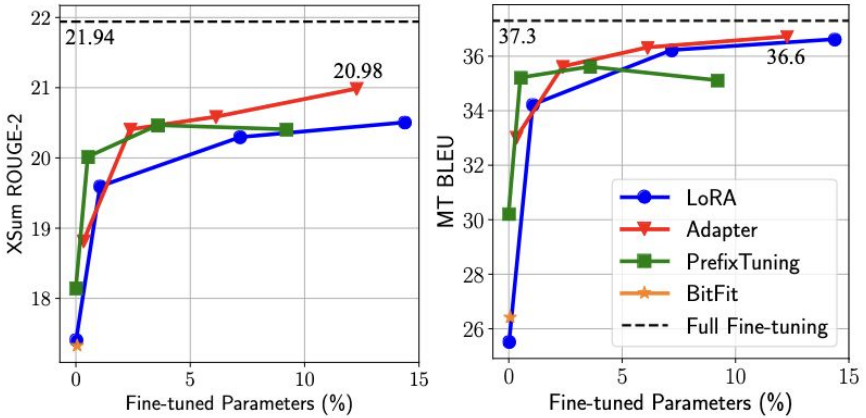


Figure 4: Performance of previous state-of-the-art parameter-efficient tuning methods on XSum (left) and en-ro (right).

Towards a Unified View of Parameter-Efficient Transfer Learning

Insightful Results:

Adapting FFN layers is more powerful than adapting the attention layers.

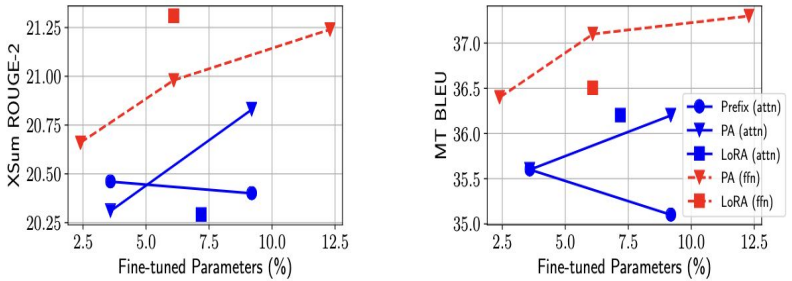


Figure 5: Results on XSum (left) and en-ro (right). PA represents parallel adapter. Blue and red markers apply modifications at attention and FFN sub-layers respectively (best viewed in color).

IMPERIAL

Towards a Unified View of Parameter-Efficient Transfer Learning

Conclusion:

1. Various approaches are related.

2. Parallel Adapters are proposed (Question: outperform prompt-tuning?)

3. FFNs is a better place to fine-tune than attention heads.

Black-Box Tuning for Language-Model-as-a-Service

IMPERIAL

Black-Box Tuning for Language-Model-as-a-Service

Key Idea: “Black-box tuning of prompts”

1. Use Derivative Free Optimisation (DFO) to find good prompts

2. Use intrinsically low “task dimensionality” of LLMs to do that (as otherwise it is infeasible).

Black-Box Tuning for Language-Model-as-a-Service

Various Derivative Free Optimisers:

1. “Gradient Descent Approximation” (sample around x_t and see if the new direction x_t+1 was better)

2. Evolutionary Methods (similar method, sampling and population are different)

Black-Box Tuning for Language-Model-as-a-Service

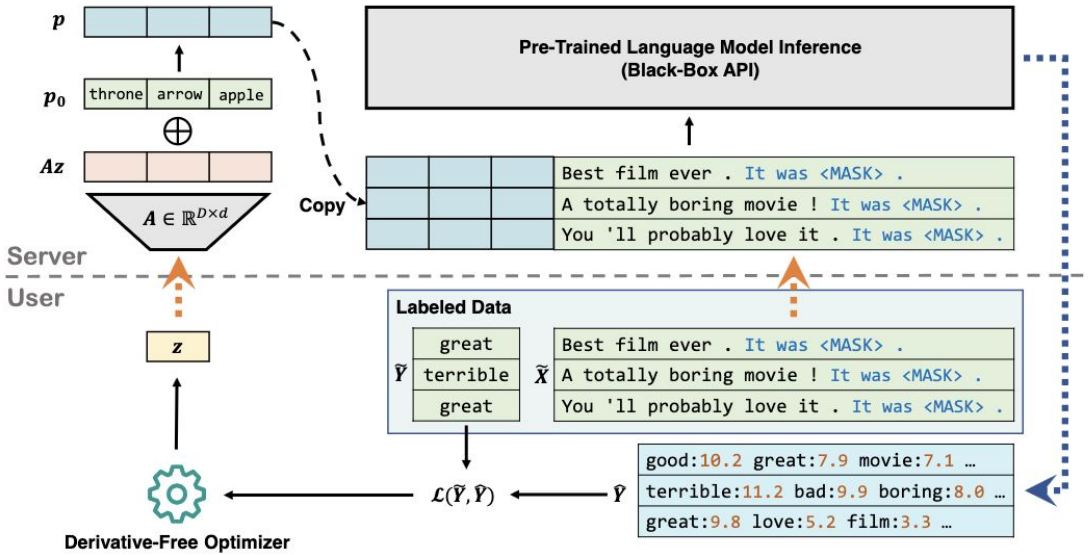


Figure 2. A single iteration of the optimization. Given $\mathbf{z} \in \mathbb{R}^d$ provided by the derivative-free optimizer, we project it to the prompt space by a random matrix $\mathbf{A} \in \mathbb{R}^{D \times d}$. By adding the projected prompt embeddings $\mathbf{Az}$ with some initial prompt embeddings $\mathbf{p}_0$ (in this illustration are the embeddings of tokens randomly sampled from the PTM's vocabulary), we obtain the final prompt embeddings that are then concatenated with the input texts $\tilde{X}$. By calling the black-box API f, which implements the forward computation of the PTM, the predictions on the masked positions are obtained, i.e., $\hat{\mathbf{Y}} = f(\mathbf{p}; \tilde{X})$. With the prediction $\hat{\mathbf{Y}}$ and the golden labels $\tilde{Y}$ at hand, we can calculate the loss that is used by the derivative-free optimizer to suggest a new $\mathbf{z}$.

IMPERIAL

Black-Box Tuning for Language-Model-as-a-Service

Table 3. Overall comparison on various language understanding tasks. We report mean and standard deviation of performance over 3 different splits (§ 4.1). All of the results are obtained with pre-trained $\text{RoBERTa}_{\text{LARGE}}$ in 16-shot (per class) setting.

Method	**SST-2** acc	**Yelp P.** acc	**AG's News** acc	**DBPedia** acc	**MRPC** F1	**SNLI** acc	**RTE** acc	**Avg.**
				Gradient-Based Methods				
Prompt Tuning	68.23 ±3.78	61.02 ±6.65	84.81 ±0.66	87.75 ±1.48	51.61 ±8.67	36.13 ±1.51	54.69 ±3.79	63.46
+ Pre-trained prompt	/	/	/	/	77.48 ±4.85	64.55 ±2.43	77.13 ±0.83	74.42
P-Tuning v2	64.33 ±3.05	92.63 ±1.39	83.46 ±1.01	97.05 ±0.41	68.14 ±3.89	36.89 ±0.79	50.78 ±2.28	70.47
Model Tuning	85.39 ±2.84	91.82 ±0.79	86.36 ±1.85	97.98 ±0.14	77.35 ±5.70	54.64 ±5.29	58.60 ±6.21	78.88
				Gradient-Free Methods				
Manual Prompt	79.82	89.65	76.96	41.33	67.40	31.11	51.62	62.56
In-Context Learning	79.79 ±3.06	85.38 ±3.92	62.21 ±13.46	34.83 ±7.59	45.81 ±6.67	47.11 ±0.63	60.36 ±1.56	59.36
Feature-MLP	64.80 ±1.78	79.20 ±2.26	70.77 ±0.67	87.78 ±0.61	68.40 ±0.86	42.01 ±0.33	53.43 ±1.57	66.63
Feature-BiLSTM	65.95 ±0.99	74.68 ±0.10	77.28 ±2.83	90.37 ±3.10	71.55 ±7.10	46.02 ±0.38	52.17 ±0.25	68.29
Black-Box Tuning	89.56 ±0.25	91.50 ±0.16	81.51 ±0.79	87.80 ±1.53	61.56 ±4.34	46.58 ±1.33	52.59 ±2.21	73.01
+ Pre-trained prompt	/	/	/	/	75.51 ±5.54	83.83 ±0.21	77.62 ±1.30	**83.90**

Black-Box Tuning for Language-Model-as-a-Service

Conclusion:

1. Blackbox approach can be more powerful than manual prompts.

2. Works with API based methods.

Takeaways & Other ideas

Takeaways & Other Ideas

1. “Demystifying Prompts via Perplexity”
    a. Really interesting idea where they are able to predict whether a specific prompt behaves well by calculating the perplexity of a prompt. If a prompt has low perplexity (i.e. is well understood by the model) then the performance is often good on this task.

2. “Attempt” Paper learns “soft-prompts” for different tasks and then combines them as an attention layer into the final “soft-prompt” given a specific task.

3. Soft-Prompts can be very powerful (if one has access to the model)
    a. Better / Faster / Easier than manual prompt-engineering?

4. Black-box Tuning (and follow-on ideas) might be very powerful for API based models? [An area to investigate?]
    a. Does it work in practice for our tasks?

Bibliography

- Word2vec: https://arxiv.org/abs/1301.3781
- BiLSTM-CRF https://arxiv.org/abs/1508.01991
- ELMO https://arxiv.org/abs/1802.05365
- Decathlon https://arxiv.org/pdf/1806.08730.pdf
- T5 https://arxiv.org/pdf/1910.10683.pdf
- GPT3 https://arxiv.org/abs/2005.14165
- Flan https://arxiv.org/pdf/2109.01652.pdf
- InstructGPT https://arxiv.org/abs/2203.02155
- The Power of Scale for Parameter-Efficient Prompt Tuning https://arxiv.org/abs/2104.08691
- Towards a Unified View of Parameter-Efficient Transfer Learning https://arxiv.org/abs/2110.04366
- Black-Box Tuning for Language-Model-as-a-Service https://www.semanticscholar.org/reader/002c58077a1f1b296468b117230a1199e91f35c2
- Demystifying Prompts in Language Models via Perplexity Estimation https://aclanthology.org/2023.findings-emnlp.679.pdf
- AUTOPROMPT: Eliciting Knowledge from Language Models with Automatically Generated Prompts https://arxiv.org/pdf/2010.15980.pdf
- ATTEMPT: Parameter-Efficient Multi-task Tuning via Attentional Mixtures of Soft Prompts https://www.semanticscholar.org/reader/55a250868627de2d202d06e7cb3f6cbcd3a66f88