Selective Progressive Prompts:
Improve Forward Transfer by Selectively Concatenating Prompts from Prior Tasks in Continual Learning

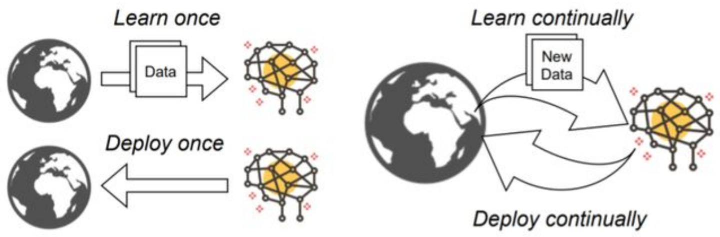
Ty Feng, Raj Kunamaneni, Srivatsan Srikanth, Henry Chou

What is continual learning?

We humans have the ability to **continuously** acquire, fine-tune, and transfer knowledge throughout our lifespan. Can machines do the same?

Unlike humans, most neural networks were trained on static datasets, and do not consider when new information becomes available over time.

Static Learning   Continual Learning



Why is it important to learn continuously/sequentially?

- Adding new skills to a robot’s repertoire in a changing environment
- Chatbots that can learn over time from conversation history
- Emulates human-level intelligence
- Enables efficient knowledge transfer between related tasks

What are the challenges for continual learning?

- Avoiding catastrophic forgetting
    - loss of previous knowledge after learning new ones
- Allowing forward transfer
    - leveraging previous knowledge to efficiently learn new tasks

Other important issues to consider when implementing continual learning:

- Increases in computation & memory usage and scalability
    - Should we add a new fine-tuned model for each subsequent task?
    - Should we update all model parameters for each subsequent task?

Fine-tuning

The most simple and straightforward approach to adapt a language model for a downstream task: updating all parameters of the model (fine-tuning)

Specifically, update gradients according to the following log-likelihood objective,

$$\max_{\Theta} \sum_{x,\, y \in T} log\, p_{\Theta}(y|x)$$

where T is a classification task with input text x and output label y. p_{Θ} is a probability distribution of output classes parameterized by the weights, Θ

Prompt Engineering

- Give the model a description of the task or a few examples
- Hand-crafted, “hard” prompts
- More formally:

  Adding additional information to condition the language model during its generation of y, thereby maximizing the likelihood of the correct y,

  $$\max p_{\Theta}(y \,|\, (P;\ x))$$

  where P is a series of tokens prepended to the input x.

- Good interpretability, but performances may vary
- How do you know if you have the best prompt?

Prompt Tuning

Learnable "soft" prompts, a series of virtual tokens

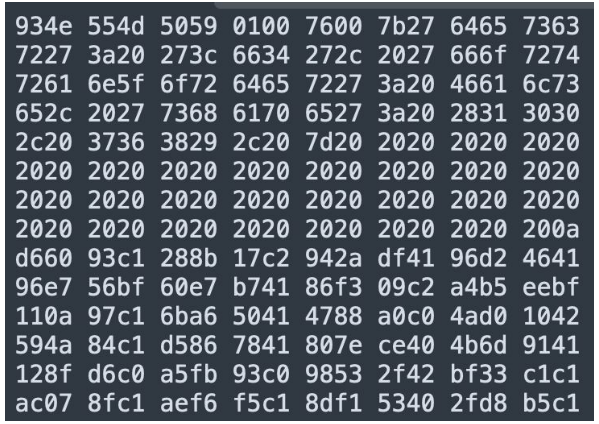
Concatenate P, the learned soft prompt, to embedded input X

Maximize likelihood of the correct output Y by updating gradients for Θ_P, the prompt parameters fine-tuned on a given task T

$$\max_{\theta_P} \sum_{x,y \in T} \log p_{\theta,\theta_P}\left(y \middle| [P;x]\right)$$

Learn prompt P via backpropagation

Keep the language model frozen



```
934e 554d 5059 0100 7600 7b27 6465 7363
7227 3a20 273c 6634 272c 2027 666f 7274
7261 6e5f 6f72 6465 7227 3a20 4661 6c73
652c 2027 7368 6170 6527 3a20 2831 3030
2c20 3736 3829 2c20 7d20 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 200a
d660 93c1 288b 17c2 942a df41 96d2 4641
96e7 56bf 60e7 b741 86f3 09c2 a4b5 eebf
110a 97c1 6ba6 5041 4788 a0c0 4ad0 1042
594a 84c1 d586 7841 807e ce40 4b6d 9141
128f d6c0 a5fb 93c0 9853 2f42 bf33 c1c1
ac07 8fc1 aef6 f5c1 8df1 5340 2fd8 b5c1
```

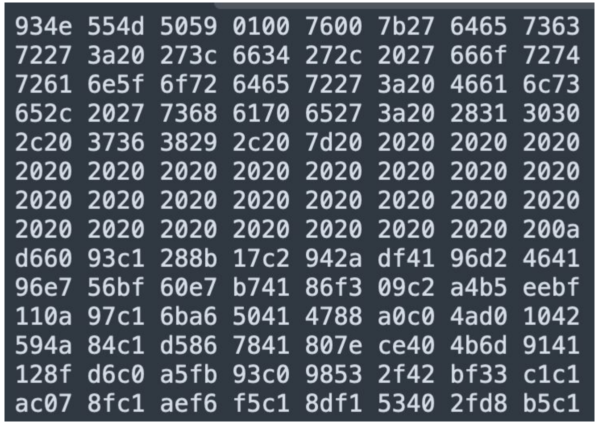
Lester et al. The Power of Scale for Parameter-Efficient Prompt Tuning
https://github.com/google-research/prompt-tuning/tree/main/prompt_tuning/pretrained_prompts/t5_1_1_lm100k_base

Prompt Tuning

Contrary to prompt engineering, where an optimal text prompt is found from searching in the text space of tokens with fixed embeddings, prompt tuning finds the best representation of embedding vector in the embedding space

Advantages:

- No need to tune billions of model params
- Only a single pre-trained model is needed
- Can use whole dataset represented as embeddings



```
934e 554d 5059 0100 7600 7b27 6465 7363
7227 3a20 273c 6634 272c 2027 666f 7274
7261 6e5f 6f72 6465 7227 3a20 4661 6c73
652c 2027 7368 6170 6527 3a20 2831 3030
2c20 3736 3829 2c20 7d20 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 2020
2020 2020 2020 2020 2020 2020 2020 200a
d660 93c1 288b 17c2 942a df41 96d2 4641
96e7 56bf 60e7 b741 86f3 09c2 a4b5 eebf
110a 97c1 6ba6 5041 4788 a0c0 4ad0 1042
594a 84c1 d586 7841 807e ce40 4b6d 9141
128f d6c0 a5fb 93c0 9853 2f42 bf33 c1c1
ac07 8fc1 aef6 f5c1 8df1 5340 2fd8 b5c1
```

Lester et al. The Power of Scale for Parameter-Efficient Prompt Tuning
https://github.com/google-research/prompt-tuning/tree/main/prompt_tuning/pretrained_prompts/t5_1_1_lm100k_base

Progressive Prompts

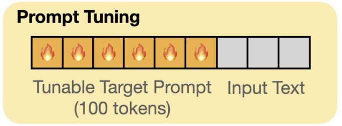
Learn a separate prompt P_k for each task in T_k and sequentially concatenate it with previously learned prompts, P_i, $i < k$, before prepending to input embeddings.

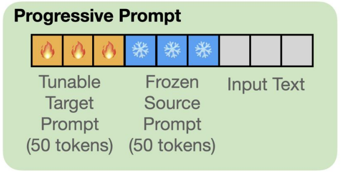
Model parameters θ are always frozen.

Prompt parameters θ_{p_k} are trainable during learning T_k and frozen afterwards.

Training objective: find θ_{p_k} that minimize negative log probability of y with previous prompts $P_k..P_1$ prepended to input x, with model parameters frozen

$$L(\theta_{P_k}) = - \sum_{x,\, y \in T_k} \log p(y | [P_k, \ldots, P_1, x], \theta, \theta_{P_1}, \ldots, \theta_{P_k})$$



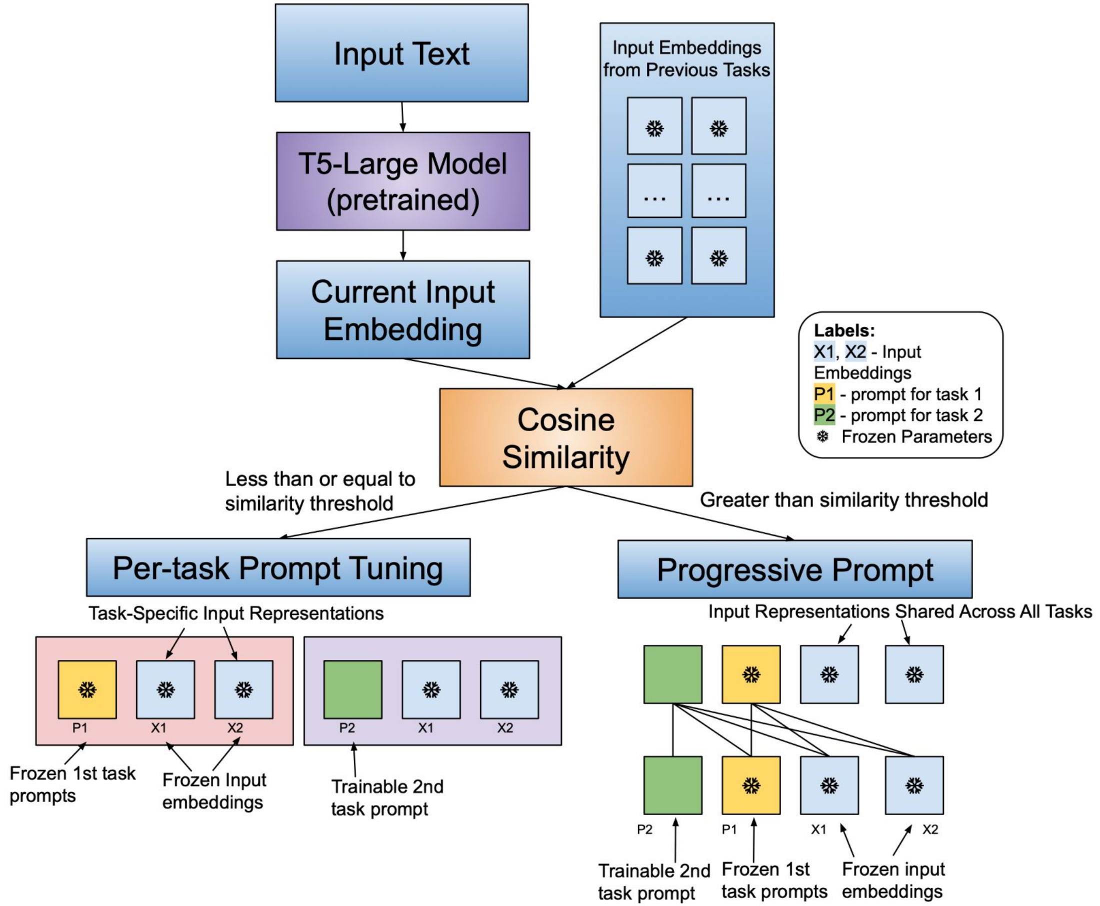

Progressive Prompt
Tunable Target Prompt (50 tokens)
Frozen Source Prompt (50 tokens)
Input Text

What is the Problem we are trying to Solve?

- Optimal integration of knowledge from multiple tasks in a task sequence is still a challenge in continual learning
- Learning a separate prompt for each task neglects the benefits of forward transfer
- Concatenating all previous prompts can lead to negative interference on forward transfer when tasks are dissimilar
- In the real world, there is no guarantee that tasks are similar in a sequence for continual learning

We need to selectively choose the knowledge (learned prompts from prior tasks) that would be the most beneficial to learning the current task

Our Method: Selective Progressive Prompts


Input Text
T5-Large Model (pretrained)
Current Input Embedding
Input Embeddings from Previous Tasks
Labels:
X1, X2 - Input Embeddings
P1 - prompt for task 1
P2 - prompt for task 2
Frozen Parameters
Cosine Similarity
Less than or equal to similarity threshold
Greater than similarity threshold
Per-task Prompt Tuning
Progressive Prompt
Task-Specific Input Representations
Input Representations Shared Across All Tasks
P1
X1
X2
P2
X1
X2
Frozen 1st task prompts
Frozen Input embeddings
Trainable 2nd task prompt
P2
P1
X1
X2
Trainable 2nd task prompt
Frozen 1st task prompts
Frozen input embeddings

Our Method: Selective Progressive Prompts

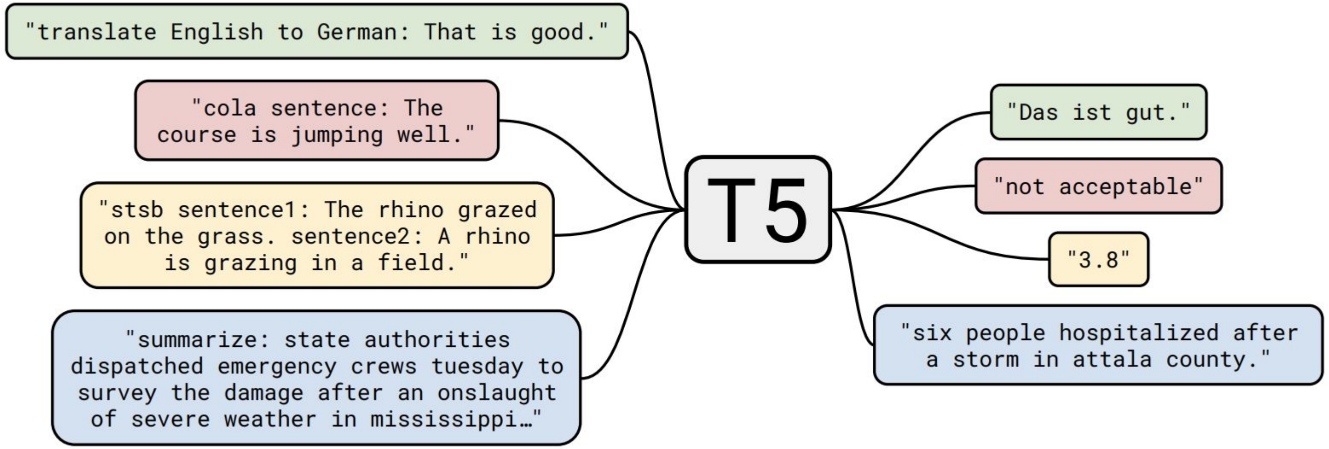
Training objective is to find the prompt parameters θ_P that minimize negative log likelihood of *y* given the text input, *x*, concatenated with the previous soft prompts $[P_k*...P_1*]$ for previous tasks, T_k, while model parameters, θ, are frozen:

$$L(\theta_{P_k}) = - \sum_{\substack{x,y \in T_k \\ sim(x, X_{prev}) \geq \text{similarity\_threshold}}} \log p(y | [P_k^*, \ldots, P_1^*, x], \theta, \theta_{P_1}, \ldots, \theta_{P_k})$$

where P_i* are the selectively chosen prompts that meet the input text similarity threshold out of all the available prior prompts P_i, and sim(x, X_{prev}) is the cosine similarity function comparing the current input embedding x to the set of previous input embeddings X_{prev}.

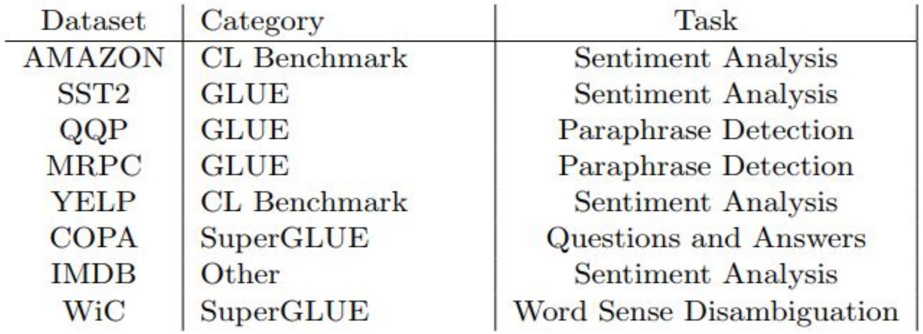
Experimental Setup

- Implementation
  - Encoder-decoder T5 model (Raffel et al. 2020) that is pretrained
  - Modified Progressive Prompts implementation on T5 codebase
- Trained via 3 Nvidia V100 GPUs from Google Cloud
  - 10 epochs per task. We used same training parameters as Progressive Prompts
  - 3 days to finish training


"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Experimental Setup

- Datasets
  - GLUE, SuperGLUE, and CL Benchmark measure language model performance



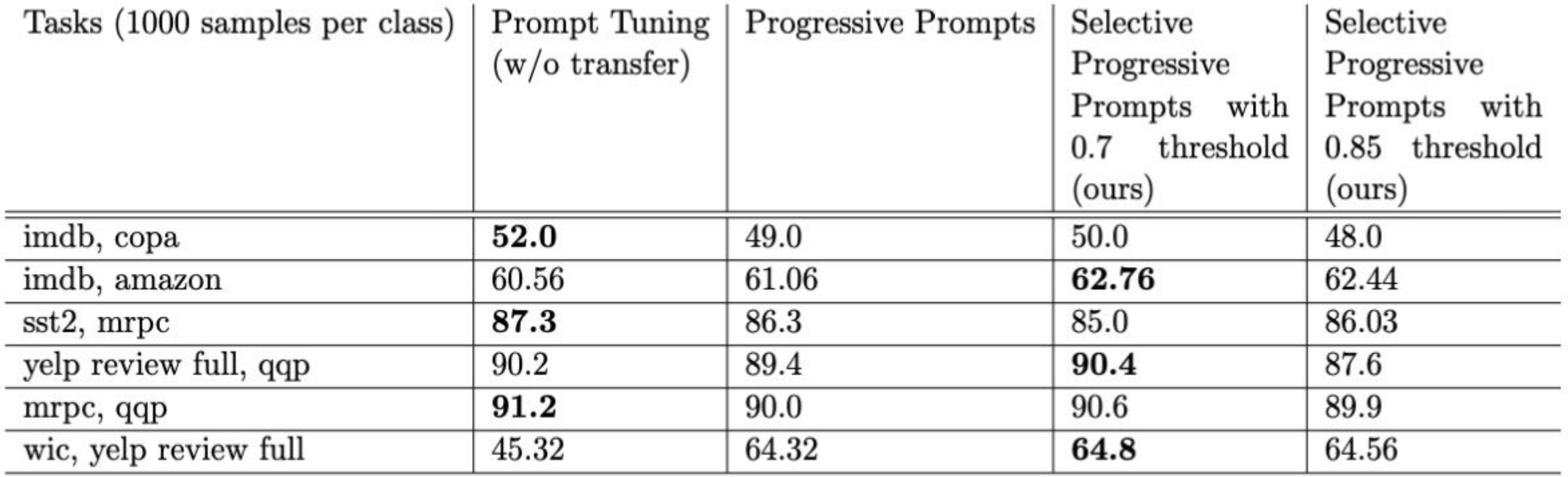
Dataset	Category	Task
AMAZON	CL Benchmark	Sentiment Analysis
SST2	GLUE	Sentiment Analysis
QQP	GLUE	Paraphrase Detection
MRPC	GLUE	Paraphrase Detection
YELP	CL Benchmark	Sentiment Analysis
COPA	SuperGLUE	Questions and Answers
IMDB	Other	Sentiment Analysis
WiC	SuperGLUE	Word Sense Disambiguation

- Compared our approach to 2 baselines methods:
  - Per-task prompt tuning (Lester et al. 2019)
  - Progressive Prompts (Razdaibiedina et al. 2023)
- 6 experiments on 6 different pairs of NLP tasks to study forward transfer
  - Two similarity thresholds: 0.7 and 0.85

Experimental Results: Forward Transfer

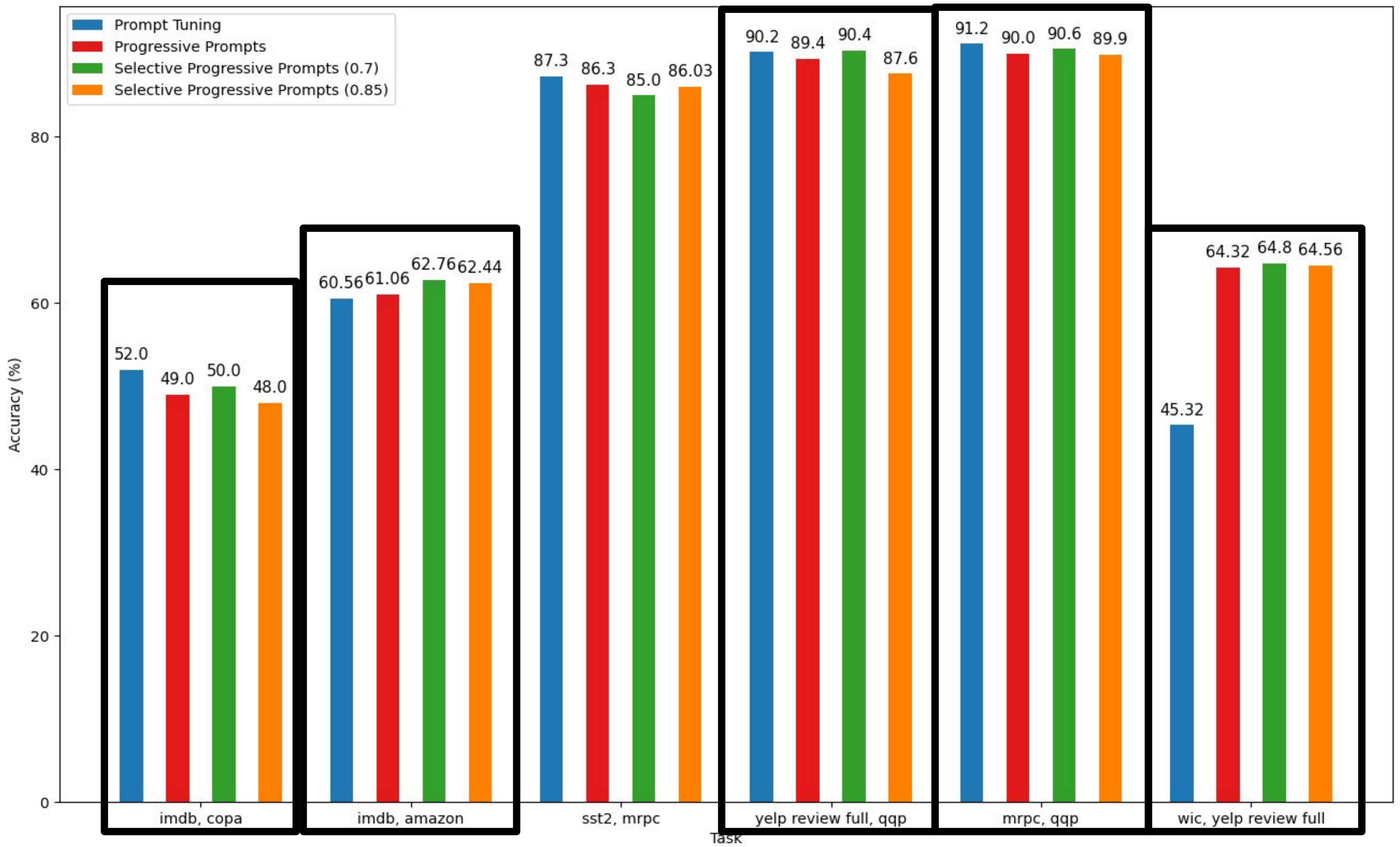
Transfer Learning on pairs of dissimilar tasks from different domains

We outperformed Progressive Prompts on 5/6 experiments. Best performance on 3/6 experiments compared to Prompt Tuning and Progressive Prompts

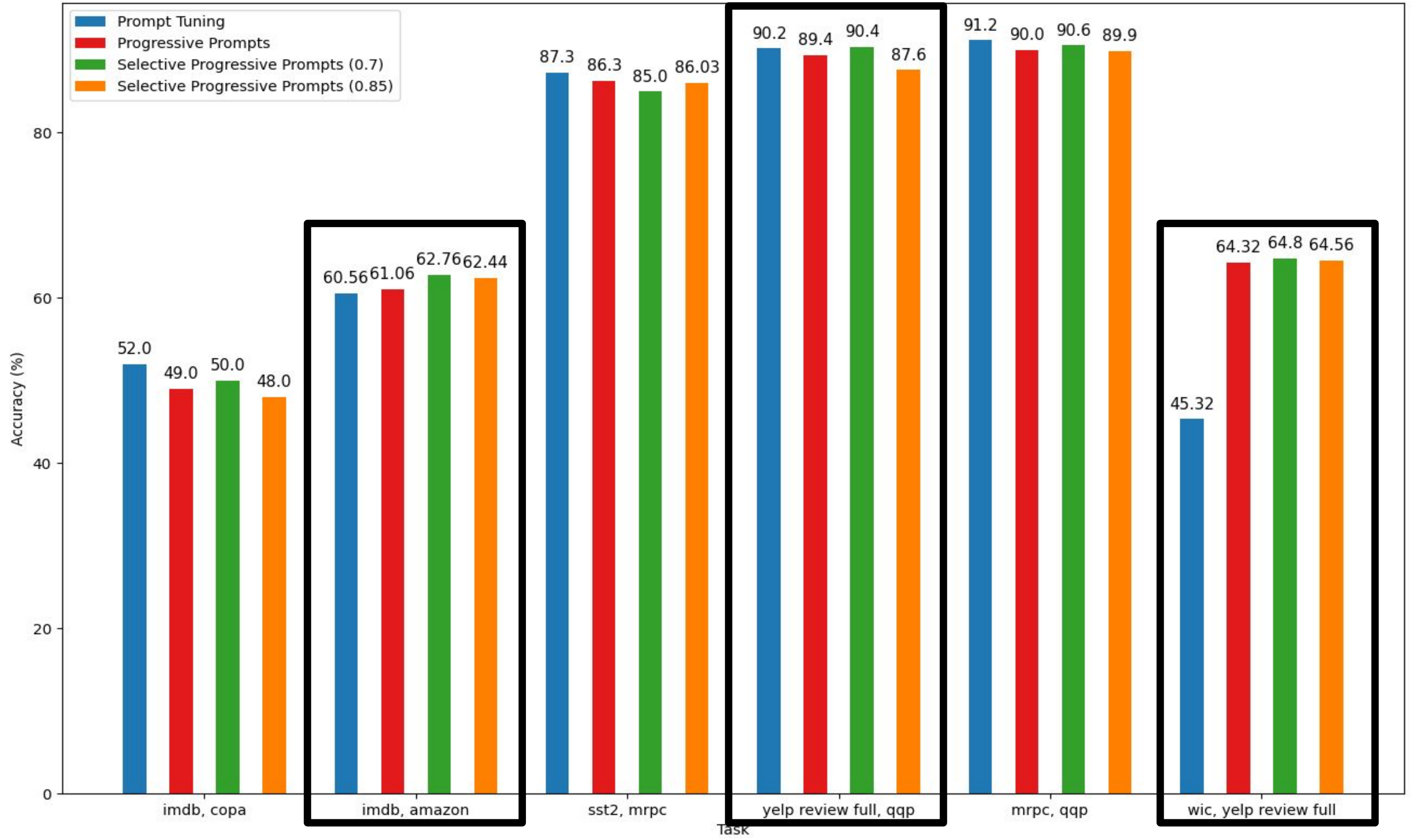


Tasks (1000 samples per class)	Prompt Tuning (w/o transfer)	Progressive Prompts	Selective Progressive Prompts with 0.7 threshold (ours)	Selective Progressive Prompts with 0.85 threshold (ours)
imdb, copa	**52.0**	49.0	50.0	48.0
imdb, amazon	60.56	61.06	**62.76**	62.44
sst2, mrpc	**87.3**	86.3	85.0	86.03
yelp review full, qqp	90.2	89.4	**90.4**	87.6
mrpc, qqp	**91.2**	90.0	90.6	89.9
wic, yelp review full	45.32	64.32	**64.8**	64.56

Our approach outperformed Progressive Prompts for 5/6 task pairs

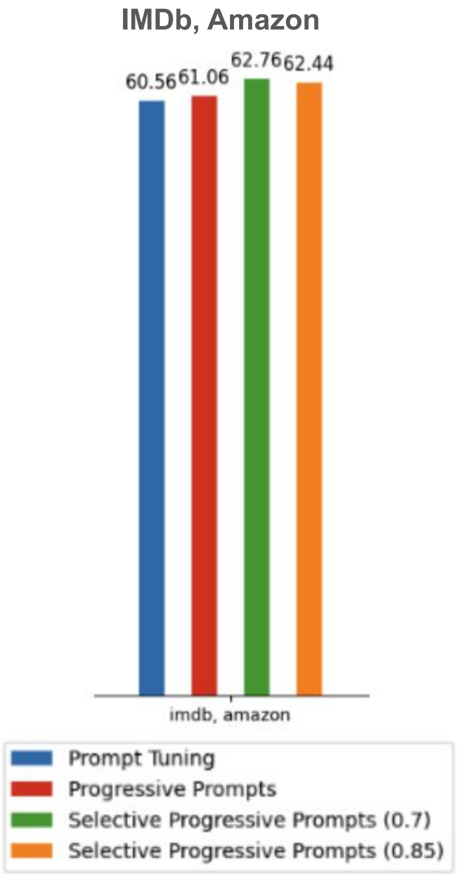


Our approach outperformed *both* Progressive Prompt *and* Prompt Tuning for 3/6 task pairs



Results (IMDb, Amazon)

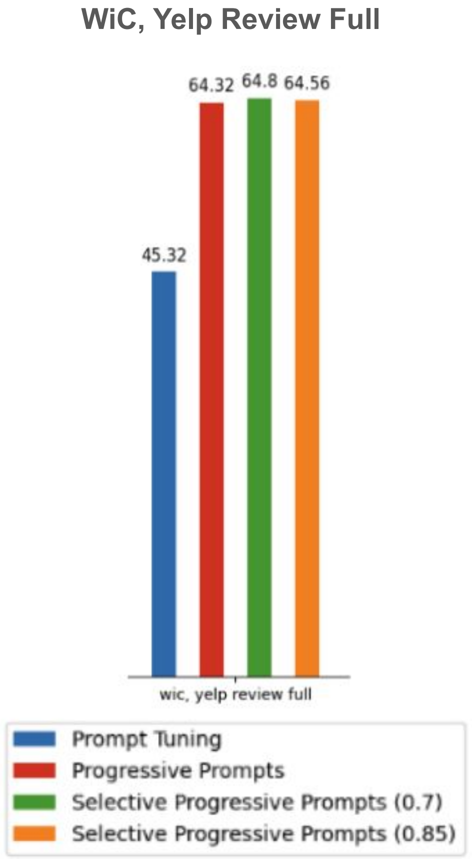
- Our selective approach helps the model to select relevant prior task prompts when learning a given task, thereby improving the accuracy and performance of transfer learning when learning the current task
- The strongest improvement we have seen is with IMDB-Amazon with a 1.7% improvement increase from progressive prompt and a 2.2% improvement from Prompt Tuning.



Selective Progressive Prompts provides best accuracy

Results (WiC, Yelp Review Full)

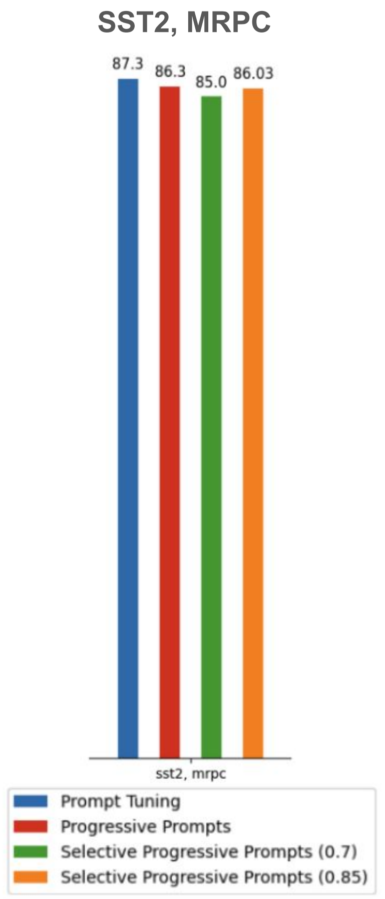
- Our selective approach helps the model to select relevant prior task prompts when learning a given task, thereby improving the accuracy and performance of transfer learning when learning the current task
- We also see a 19.48% improvement in WiC-Yelp Review full when compared to Prompt Tuning and a 0.48% improvement when compared to Progressive Prompts.



Selective Progressive Prompts provides best accuracy

Results (SST2, MRPC)

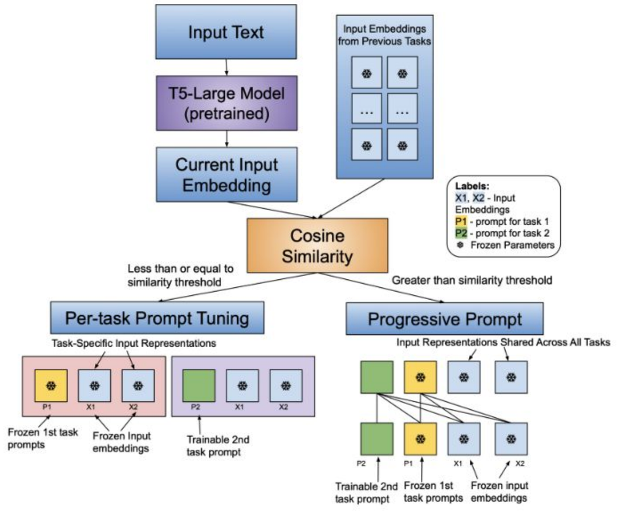
- For tasks with low similarity, it is better to train a per-task prompt using Prompt Tuning
- For SST2 - MRPC, we observed a lower accuracy score than Prompt Tuning and Progressive Prompts
- Our approach with 0.85 threshold performed better than 0.7 threshold
- Need to adaptively choose the similarity threshold for future work



Better to use Prompt Tuning for dissimilar tasks

CONCLUSION

- By introducing Selective Progressive Prompts, we aim to improve forward transfer on a sequence of diverse tasks in continual learning.
- Our selective approach allows us to learn prompts only using prompts from relevant tasks, thereby improving model accuracy when learning a task sequence with varying task similarities.


Input Text
T5-Large Model (pretrained)
Current Input Embedding
Input Embeddings from Previous Tasks
Cosine Similarity
Labels:
X1, X2 - Input Embeddings
P1 - prompt for task 1
P2 - prompt for task 2
Frozen Parameters
Less than or equal to similarity threshold
Greater than similarity threshold
Per-task Prompt Tuning
Task-Specific Input Representations
Progressive Prompt
Input Representations Shared Across All Tasks
P1
X1
X2
P2
Frozen 1st task prompts
Frozen Input embeddings
Trainable 2nd task prompt
Trainable 2nd task prompt
Frozen 1st task prompts
Frozen input embeddings

Future work

- We would like to try with different samples sizes per class to see the effect of limited data setting (few shot learning) on accuracy
- Compare with different similarity thresholds and have an adaptive threshold based on the task and model performance
- Conduct more experiments with larger sequence of tasks to verify the performance of Selective Progressive Prompts

References

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In International Conference on Learning Representations, 2023.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. ArXiv, abs/1606.04671, 2016.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21:140:1–140:67, 2019.

Questions