Task-Adaptation Curriculum Learning

Anonymous ACL submission

Abstract

Despite the prevailing applications of foundation models, when adapted to downstream tasks, their performance sensitively varies with the distribution shift/gap between the pertaining task and the target task. Moreover, direct fine-tuning might be overfitting to limited target task data. In the realm of NLP, tasks are semantically related with shared skills and those general purposed ones usually have more available data than the highly specific and user-defined ones. In this paper, we mitigate the distribution shift in task adaptation by developing a smooth transfer learning curriculum, which, by fine-tuning the model along a path of intermediate tasks on a graph, progressively bridges the gap between the pretrained model and a target task with limited data. To this end, we formulate the curriculum learning as a graph search problem and address its efficiency by a deep dive into accelerating the transferability estimation between tasks and two classical search algorithm applied to our problem, i.e., greedy best first search and Monte Carlo tree search. We evaluate our approach, i.e., "task-adaptation curriculum learning (TACL)" on two benchmark settings with tasks drawn from GLUE. Extensive experiments on different target tasks demonstrate the effectiveness and advantages of TACL on more specific and data-deficient downstream tasks.

1 Introduction

Foundation models pretrained on large-scale corpora have shown substantial potential to generalize to downstream tasks with promising performance (Devlin et al., 2019). While simple fine-tuning these models on the target task data usually suffices to obtain a transfer learning from the pretrained task to the target task, the final performance heavily depends on the distribution shift between the two tasks and the amount of fine-tuning data available for the target task, e.g., transfer learning may perform poorly under large distribution shift and limited target task data.

Fortunately, many NLP tasks are semantically related and their structures are shared so we may fine-tune on an intermediate task before transitioning to the target task. This approach is valuable because the intermediate task may encapsulate pertinent information for solving the target task, facilitating smoother training and aiding in the retention of knowledge acquired from pretrained tasks. Recognizing the efficacy of utilizing relevant tasks, our objective is to devise a method that guides the model through a sequence of intermediate tasks. This approach aims to establish a seamless transfer pathway from pretraining tasks to the target task, addressing issues related to overfitting and task distribution shift. However, the search for the optimal transfer curriculum presents a formidable challenge, characterized by a combinatorial optimization problem. The impracticality of a brute-force solution becomes evident as the sequence length increases, leading to an exponential growth in the number of possible task combinations. Furthermore, discerning the relative importance of each task in the sequence to the target task is challenging. Additionally, the dynamic nature of model parameters, altered after training on each task, makes it hard to determine a sequence in an a priori manner.

To mitigate these challenges, we address the problem by formulating it as searching for a path on a graph of tasks, effectively connecting the pretrained task to the target task. This graph-based approach offers several advantages in tackling these issues. Firstly, leveraging existing graph search algorithms allows us to confine the search space, thereby circumventing the need for a computationally intensive brute-force solution. Secondly, the flexibility of employing heuristic or non-heuristic methods facilitates the estimation of the priority of specific states within the graph. Lastly, the dynamic nature of graph search takes into account the

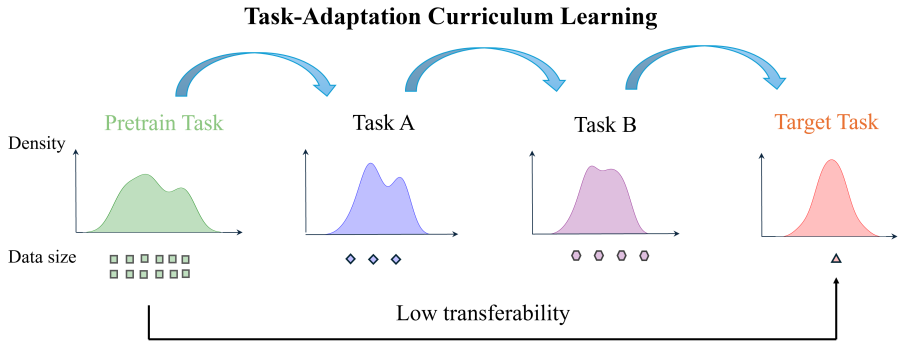

Figure 1: Comparison of direct transfer learning (bottom) vs. task-adaptation curriculum learning (top).

evolving model parameters.

To this end, we proposed the framework of task-adaptation curriculum learning, which involves finding a sequence of adaptation tasks that progressively bridges the gap between the pretrained model and the target task by searching a transfer learning path on a graph of tasks. Specifically, we employ two classic search algorithms within this framework: greedy best-first search and Monte-Carlo tree search. We employ some approximation methods to avoid intensive computation. Our approach is examined on two sets of NLP tasks. Through a meticulous analysis of the experimental results, we find that task-adaptation curriculum learning emerges as a beneficial approach, particularly in scenarios characterized by limited data availability. Furthermore, our findings underscore the scalability and flexibility of this framework, showcasing its adaptability to diverse task settings.

2 Related Work

The method that we propose in this paper addresses the general problem of task adaptation, which generally refers to adapting a pre-trained model to a downstream task. Commonly employed practices include fine-tuning directly and linear probing. Others, such as task/domain-adaptive methods, consider the issue of catastrophic forgetting (Kirkpatrick et al., 2017), wherein models may forget knowledge from previous tasks after training on a new one, leading to negative transfer. DAPT (Gururangan et al., 2020) tackles this by first tuning the pre-trained model on data related to the target domain or the target task itself, and then fine-tuning the adaptive-tuned model on the target task. Similarly, Dery et al. (2021) propose a multi-task framework to bridge the gap between pre-trained tasks and the end task by adaptively updating weights of auxiliary tasks. However, our method differs in that it seeks to design an algorithm capable of automatically determining intermediate training task sequences between pre-trained tasks and the target task, eschewing a multi-task approach.

The concept of intermediate training is also pertinent to our work. In this paradigm, practitioners typically designate one task as an intermediate step between pre-trained tasks and the target task. Previous works in this domain leverage transferability or similarity to identify intermediate tasks (Vu et al., 2020). For instance, task embeddings for transfer learning (Achille et al., 2019) consider the Fisher information matrix of a model fine-tuned on a task as the "task embedding," predicting inter-task transferability by computing the cosine similarity between the task embeddings of the source and target tasks. Notably, our approach diverges in that we seek not just one intermediate task but a sequence of adaptation tasks.

Our method also intersects with the concept of curriculum learning, which involves ranking the difficulty or priority of learning examples and then proceeding with learning in such a sequence. While traditional curriculum learning operates at the data level, our focus in the realm of task adaptation learning is on task-level curriculum learning. Noteworthy work by Pentina et al. (2015) employs curriculum learning to sequentially solve multiple tasks, demonstrating its superiority over joint task-solving. Their aim, however, was to enhance the average performance across multiple tasks, whereas our method specifically targets the performance improvement of the target task.

3 Problem Formulation

The task is a pair of an objective function and a dataset: $\mathcal{T} := \{\mathcal{L}, \mathcal{D}\}$, where $\mathcal{D}$ consists of n sam-

ples. Given a specific end-task $\mathcal{T}^*$, our aim is to improve the performance on $\mathcal{T}^*$ by leveraging a set of auxiliary tasks $\{\mathcal{T}_n\} := \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \cdots \mathcal{T}_n\}$. A task graph, denoted as $\mathcal{G}_n$, is a graph wherein the nodes represent individual tasks, and the edges symbolize the connections between these tasks. Typically, we assume $\mathcal{G}_n$ to be a complete graph, signifying that each task is directly connected to every other task in the graph. Our objective is to find an optimal intermediate training sequence denoted as

$$s := \text{Pretrain} \rightarrow \mathcal{T}_a \rightarrow \mathcal{T}_b \rightarrow \cdots \rightarrow \mathcal{T}_i \rightarrow \mathcal{T}^*$$

This sequence is path in $\mathcal{G}_n$ and connects the pretrained task to the target task, aiming to maximize the performance of $\mathcal{T}^*$.

For each task, we add a task-specific output layer ϕ to the pretrained model during training. This presents a discrete bi-level optimization problem, formulated as follows:

$$s^* = \arg\min_s \mathcal{L}_{\text{out}}[f_{\theta(s)}; \phi^*] \quad (1)$$

$$\phi^* = \arg\min_\phi \mathcal{L}_{\text{in}}[f_{\theta(s),\phi}]. \quad (2)$$

Here $f_{\theta(s)}$ represents the function parameterized by θ, determined by sequence s, $\mathcal{L}_{\text{out}} : \mathcal{F} \rightarrow \mathbb{R}$ is a functional of the encoder functions $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$, and $\mathcal{L}_{\text{in}} : \mathcal{F} \rightarrow \mathbb{R}$ is a functional of the functions representing the entire model $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

To address this optimization problem, we would explore the discrete space consisting of every possible sequence s of tasks. However, considering the infinite nature of this space, a significant number of states are not worth investigating. Therefore, the strategic pruning of unhelpful branches becomes imperative. To achieve this, we adopt the approach of searching on a graph of tasks, dynamically evaluating the value of each sequence during the search examining the current state of the model, specifically its parameters. This process can be conceptualized as utilizing search algorithms to approximate the outer level of the original optimization problem. In essence, we seek to find the optimal sequence of tasks s^* through a search algorithm, operating on the graph $\mathcal{G}_n$, and simultaneously determine the optimal ϕ^* that minimizes the inner loss function $\mathcal{L}_{\text{in}}[f_{\theta,\phi}]$. This dynamic and iterative exploration allows us to efficiently prune the solution space, leading to a more effective and targeted approach

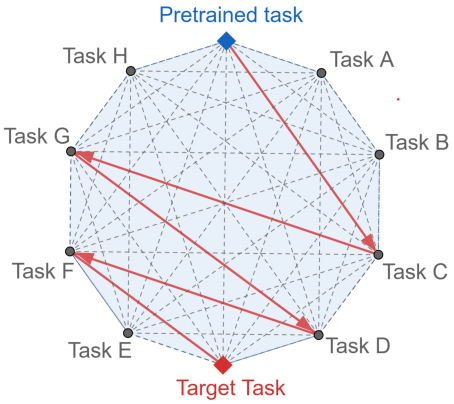

Figure 2: Example of a task-adaptation curriculum (path) on the task graph, which bridges the pretrained and target tasks by a sequence of intermediate tasks.

to solving the optimization challenge.

$$s^* \approx \text{SearchAlg}(f_{\theta,\phi}; \mathcal{G}_n) \quad (3)$$

$$\phi^* = \arg\min_\phi \mathcal{L}_{\text{in}}[f_{\theta,\phi}] \quad (4)$$

4 Task-Adaptation Curriculum Learning

In the realm of task-adaptation curriculum learning, our aim is to determine a sequence of adaptation tasks that bridge the gap between the pretrained task and the target task, with the ultimate goal of enhancing the performance on the target task. Framed as a search problem, we introduce two straightforward yet effective methods: the greedy best first search (GBFS) and Monte-Carlo tree search (MCTS), both geared towards identifying the optimal adaptation sequence.

4.1 Greedy Search of Task Curriculum

The concept of greedy search, a prevalent technique in the field of search algorithms, involves making the best possible decision at each step. This approach entails examining only the immediate future and selecting the most favorable action. When a problem exhibits an optimal substructure property, the greedy algorithm tends to yield optimal results. Due to its simplicity and efficiency, greedy algorithms are frequently employed to solve optimization problems.

In task-adaptation curriculum learning, the challenge is to select the subsequent adaptation task after training on a given task. The objective is to make decisions that collectively enhance the overall performance on the target task. In the case of

greedy best first search, we adopt a methodical approach by selecting the most promising task at each step. This involves fine-tuning the model on each auxiliary task, followed by training on the target task. The validation accuracy on the target task serves as a reward metric, representing the efficacy of each task in aiding the target task. Other potential metrics include validation loss, geometric distance, or task embedding similarity. The chosen task is the one that maximizes the estimation of the target task performance. This process is elucidated in detail in algorithm 1.

Algorithm 1 Greedy Best First Search

Require: l: Length of sequence
Require: $\{\mathcal{T}_n\}$: A set of n auxiliary tasks
Require: f_θ: Pretrained model
$k \leftarrow 0$
while $k < l$ **do**
  for $\mathcal{T}_i \in \{\mathcal{T}_n\}$ **do**
    Train f_θ on $\mathcal{T}_i$: $\theta_i = \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{T}_i)$
    Compute heuristics on target task $\mathcal{T}^*$
  end for
  $\mathcal{T}' \leftarrow$ task with the lowest heuristic
  Train f_θ on $\mathcal{T}'$: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{T}')$
  $k \leftarrow k + 1$
end while

4.2 Monte Carlo Tree Search of Task Curriculum

Monte Carlo Tree Search (MCTS) proposed by Coulom (2006) is a heuristic search algorithm designed for decision processes, particularly in applications involving playing board games. In such scenarios, MCTS is employed to solve the intricate game tree by approximating the true game-theoretic value of potential actions from the current state. The algorithm achieves this by iteratively constructing a partial search tree.

A notable advantage of MCTS lies in its independence from domain-specific knowledge, rendering it applicable to a wide range of domains that can be modeled using a tree structure. In the realm of task-adaptation curriculum learning, the process of determining the next task inherently involves decision-making, akin to a growing tree structure. Consequently, MCTS seamlessly aligns with our framework for task-adaptation curriculum learning, offering a versatile and domain-agnostic approach to solving the intricate decision processes involved in the selection of intermediate tasks. In this context, the state represents the current model, a node corresponds to a specific task, an action involves training on the chosen task, and the reward is determined by the performance of the target task after completing the adaptation sequence. A simulation entails training the model on a sequence of tasks of a specified length.

How the tree is built depends on how nodes in the tree are selected. By framing the choice of a child node as a multiarmed-bandit problem, we employ the Upper Confidence Bound (UCB1) algorithm to estimate the value of each child node. The UCB1 algorithm considers the expected reward as approximated by Monte Carlo simulations, treating these rewards as random variables with unknown distributions. This approach ensures simplicity, efficiency, and a guaranteed closeness to the best possible bound on the growth of regret. The selection of a child node is determined by the following formula:

$$v' := \underset{v' \in \text{children of } v}{\arg\max} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2\log N(v)}{N(v')}}. \quad (5)$$

Here, $N(v)$ is the number of times the current (parent) node has been visited, $N(v')$ is the number of times the child has been visited, and $c > 0$ is a constant.

As a result, we employ UCB1 for the selection process and implement a random policy for rollout. The performance of the target task, such as validation accuracy or loss, is utilized to compute the reward associated with a given sequence. As the tree grows, we iteratively refine our estimates of the value of choosing the next task. The entire process is encapsulated in algorithm 2.

5 Experiments

In our experimental investigations, we aim to address the following questions pivotal to the efficacy of our proposed task-adaptation curriculum learning (TACL) methodology: (1) Can models gain significant benefits from the adoption of task-adaptation curriculum learning? (2) What are some similarities and differences in the results produced by GBFS and MCTS? (3) What are some possible factors that could potentially influence the performance of TACL?

To systematically tackle these questions, we design and execute experiments on two graphs: a smaller graph comprising six tasks and a more extensive graph encompassing nine tasks. This exper-

imental setup allows us to evaluate the robustness and scalability of our proposed approach under varying parameter settings.

5.1 Experimental Setting

Throughout our experiments, we consistently employ the BERT model (Devlin et al., 2019), a powerful language representation model. The experimentation is conducted on the nine datasets from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018), which spans various linguistic tasks. These tasks include sentiment analysis (SST-2; Socher et al., 2013), Quora Question Pairs (QQP; Iyer et al., 2017), paraphrase identification (MRPC; Dolan and Brockett, 2005), semantic similarity (STS-B; Cer et al., 2017), grammatical acceptability judgments (CoLA, Warstadt et al., 2019), natural language inference (NLI) with Multi-Genre NLI (MNLI; Williams et al., 2018), SQuAD (Rajpurkar et al., 2016) converted into Question-answering NLI (QNLI; Wang et al., 2018), Recognizing Textual Entailment (RTE; Dagan et al., 2005), and the Winograd Schema Challenge (Levesque et al., 2012) recast as Winograd NLI (WNLI). The diverse nature of these datasets allows us to comprehensively evaluate the adaptability of our method across various language understanding tasks.

5.2 Baselines

Fine-tune: One of our baseline comparisons involves the direct fine-tuning of the model, as this serves as a standard approach and aligns with our primary goal of enhancing the performance of fine-tuning on the target task. Linear probing, another common method, is not adopted as a baseline in our study. The rationale behind this decision is our pursuit of identifying better pretrained parameters, whereas linear probing freezes the pretrained parameters during training.

Random: In addition to direct fine-tuning, we include a random sequence of the same length as the paths searched by our method as an additional baseline. This comparison aims to evaluate whether our method can effectively discover valuable information regarding task transferability within the graph, as opposed to a random exploration. This baseline also provides insights into whether our method achieves more significant improvements on the target task, showcasing its efficacy in leveraging the structure of the task graph to enhance transfer learning.

Task	Size	Domain
SST-2	128	movie reviews
MRPC	128	news
MNLI	1024	misc.
QNLI	1024	Wikipedia
QQP	1024	social QA
RTE	2048	news, Wikipedia

Table 1: Tasks used in the small graph

5.3 Main results and analysis

Given that the test sets of GLUE datasets are not publicly available, our reported performance metrics are based on the validation sets. We split some samples from the training set to serve as a validation set during the course of our experiments. Regarding performance metrics, we report F1 scores for QQP and MRPC, and accuracy scores for the other tasks.

In terms of the training methodology, we use a fresh optimizer for each phase of training. For each task, we add only a single task-specific, randomly initialized output layer to the pretrained Transformer model. For all experiments, the loss function is the cross-entropy error between the predicted and true class. The implementation is carried out using Hugging Face's transformers library and PyTorch. While we follow the recommended hyperparameters by Devlin et al. (2019), we adjust the batch size to suit our experimental requirements.

TACL **on a graph of six tasks** In this experimental setup, we use six tasks from the GLUE benchmark. Additionally, every task included in this graph is considered a potential target task, allowing for comprehensive exploration and evaluation of the model's adaptability across various tasks. The core aim of our experiments is to evaluate the efficacy of Task-Adaptation Curriculum Learning (TACL) in addressing challenges associated with fine-tuning, particularly in situations marked by limited training data. To achieve this, we explore varying levels of data scarcity across different tasks. In particular, we deliberately impose an extremely scarce data regime in the SST-2 and MRPC tasks, where the size of the training set is severely restricted to only 128 samples. In the case of MNLI, QNLI, and QQP, we adopt a moderately limited regime with a training set size of 1024

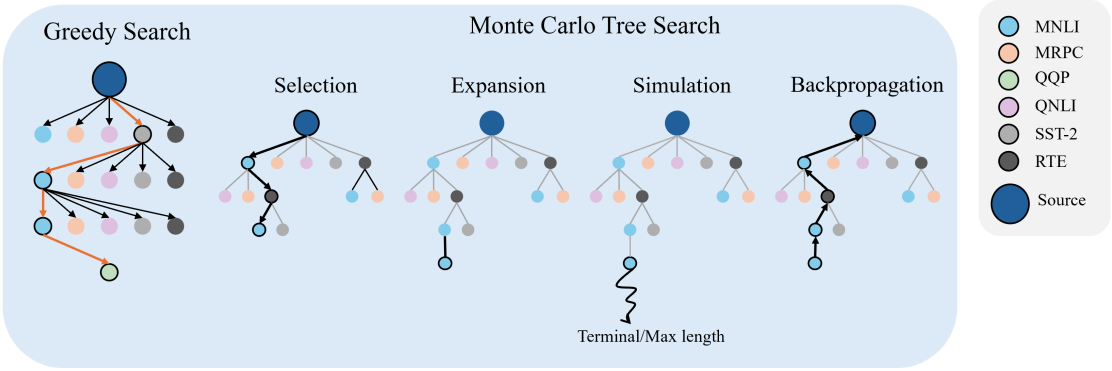

Figure 3: Illustration of searching a curriculum path to QQP by greedy search vs. Monte Carlo tree search

samples. On the other hand, for the RTE task, the dataset is characterized by a relatively larger size, with 2048 samples in the training set. This diverse range of data limitations enables us to systematically assess the adaptability and performance of our proposed methodology across varying degrees of data scarcity. In terms of training specifics, we fix the maximum length of the sequence at four during the experiments. When training on an intermediate task within the sequence, we limit the training steps rather than allowing the model to fully converge. This strategy is employed to strike a balance between training efficiency and obtaining meaningful insights from the intermediate tasks. In the context of Monte Carlo Tree Search (MCTS), simulations can be computationally intensive as they involve iterative fine-tuning of the model. To mitigate this, we reduce the number of steps during simulation, aiming for a more efficient approximation of the true performance. Table 2 presents the results for each task treated as the target task. Notably, these results reflect the performance of a fully converged model on the target task.

The limitations imposed by the scarcity of data make direct fine-tuning ineffective, resulting in suboptimal outcomes. Random sequences sometimes exhibit slightly improved results, aligning with the understanding that incorporating intermediate training tasks in data-limited scenarios can offer some benefits. In contrast, our proposed methods demonstrate significant success in enhancing the performance of the target task across all tasks in the graph. Notably, Monte Carlo Tree Search (MCTS) outperforms Greedy Best-First Search (GBFS) in most tasks, indicating that the iterative nature of MCTS likely contributes to its superior performance in navigating the task graph and identifying more effective adaptation sequences. This observation underscores the effectiveness of our task-adaptation curriculum learning framework in comparison to baseline methods.

TACL on a graph of nine tasks In the extension of the previous experiment, we expanded the graph to include three additional GLUE tasks (STS-B, CoLA, WNLI), resulting in a total of nine tasks. Unlike the previous experiment where all tasks were treated as target tasks, in this case, we focused on observing the performance of our method on three specific tasks: MRPC, QNLI, and RTE. The experimental settings remained consistent with the smaller graph experiment, ensuring a fair comparison.

The results of the experiments are presented in figure 4. As indicated by the results, Task-Adaptation Curriculum Learning (TACL) appears to derive some benefits from a more diverse range of available auxiliary tasks, with slightly improved performance. Upon examining the new paths, it is noteworthy that STS-B is often included in the sequence of adaptation tasks. The Semantic Textual Similarity Benchmark involves sentence pairs sourced from news headlines and other texts, annotated with a score indicating the semantic similarity between the two sentences on a scale from 1 to 5. Given the nature of the STS-B task, which assesses the general semantic knowledge of a model, we can hypothesize that training on this task could be beneficial for other downstream tasks. The universal knowledge acquired during the learning process of STS-B may contribute to the model's improved adaptability and performance.

Analysis of paths and structures within the task graph In addition to evaluating performance,

Methods	SST-2	MRPC	MNLI	QNLI	QQP	RTE
Fine-tune	81.8	81.2	60.2	78.6	70.6	68.6
Random	74.0	80.1	62.1	77.8	71.7	70.1
GBFS	84.2	**83.2**	**64.9**	79.0	73.0	71.5
MCTS	**85.0**	83.1	64.2	**79.9**	**73.6**	**72.8**

Table 2: Target task performance (%) achieved by different transfer learning strategies on a small six-task's graph.

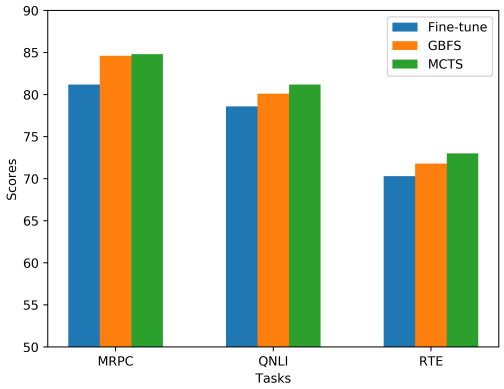

Figure 4: Performance scores (%) of three target tasks achieved by GBFS/MCTS-searched curriculum learning on a nine-task graph. Scores refer to accuracy or F1 score. MCTS-curriculum achieves the best performance, while both MCTS and GBFS outperform direct finetuning.

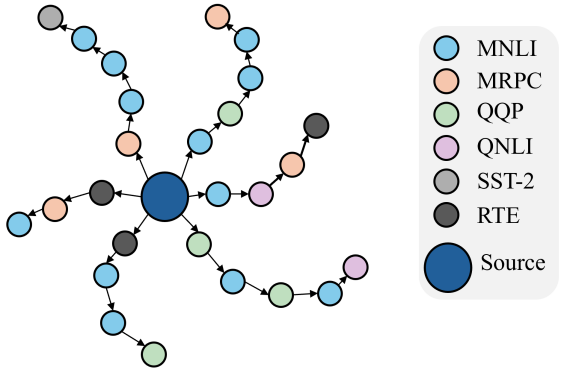

Figure 5: Curricula for six different target tasks by greedy best first search.

our investigation aims to determine whether our method can uncover specific structures within the graph that are relevant to the target task. Figure 5 depicts the paths discovered by Greedy Best-First Search (GBFS) to all target tasks. Figure 6 demonstrates some paths to QQP by Monte Carlo tree search. While the paths are not entirely deterministic due to the choice of random seed, we are still able to discover some important patterns and structures within the graph of tasks.

As shown in Figure 5, MNLI is the most frequently chosen task in task sequences, and its placement at the end of the sequence may be crucial for the performance on the target task. Furthermore, MNLI tends to be associated with high-value paths produced by MCTS, as illustrated in Figure 6. Multi-Genre Natural Language Inference (MNLI) is a large-scale entailment classification task. In MNLI, given a pair of sentences, the objective is to predict whether the second sentence entails, contradicts, or is neutral with respect to the first one. Based on these observations, we can formulate a hypothesis that the model becomes more proficient in processing and analyzing semantic information after training on MNLI. The frequent inclusion of MNLI suggests its importance in enhancing the model's ability to understand and reason about semantic relationships between sentences. This enhanced capability is expected to translate into improved performance on target tasks.

5.4 Secondary Findings and Additional Analyses

Influences of training steps in TACL In addition to the final results of Task-Adaptation Curriculum Learning (TACL), our curiosity extends to understanding the factors that may affect the performance of TACL. Throughout the course of experiments, we observe that the number of training steps on each task within the task sequence is sometimes important in determining the final results. For a fixed sequence of tasks, varying the number of training steps can lead to different out-

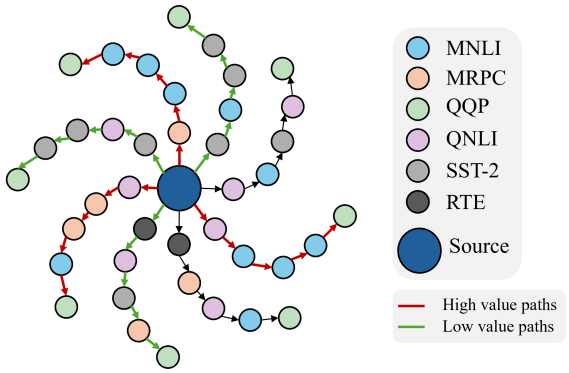

Figure 6: Illustration of some possible task-curricula for QQP (target task) by Monte Carlo tree search. Better curricula with higher values are highlighted in red.

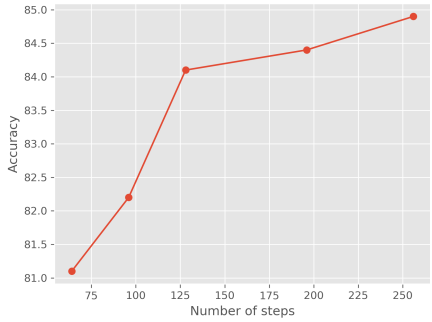

Figure 7: Different number of training steps and its impact on the performance of SST-2.

comes. As depicted in Figure 7, more training steps may help the model in acquiring and preserving more knowledge from the task, resulting in greater improvements on the target task. This observation emphasizes the importance of this hyperparameter to the effectiveness of TACL.

Global property of TACL While our primary objective is to enhance the model's performance on the target task, we also anticipate potential benefits for the model on other tasks, even if they are not the primary targets. After the completion of the intermediate training sequence, we save the model checkpoints and conduct fine-tuning on all tasks that are not included in the searched sequence. As illustrated in the figure 8, the model demonstrates enhanced performance not only on the target task but also on other tasks. This finding emphasizes the efficacy of TACL not only in optimizing for specific tasks but also in enhancing the overall generalization capabilities of the model, indicating that TACL can uncover important global structures within the task graph.

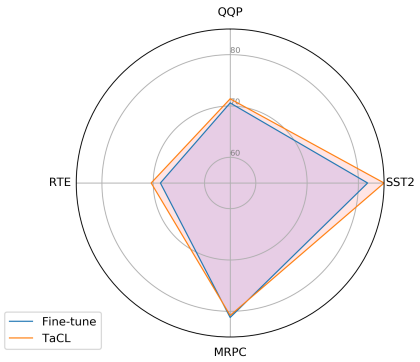

Figure 8: Comparison between direct finetuning and TACL trained models evaluated on **tasks not included in the curriculum**. SST-2 is the target task.

6 Conclusion

In summary, we have introduced the framework of task-adaptation curriculum learning as a solution to challenges associated with directly fine-tuning pretrained models. Our approach offers several advantages: it is both simple and flexible, allowing for the incorporation of various search algorithms on graphs. Furthermore, it serves as an extension of intermediate training, leveraging a broader set of tasks to enhance the model's generalizability, particularly in scenarios with limited data.

The adaptability provided by a sequence of tasks may play a crucial role in addressing the disparity between a pretrained model and a highly specific downstream task. We believe that our methodology contributes some insights to the realm of task adaptation in Natural Language Processing (NLP).

References

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. 2019. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6430–6439.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Rémi Coulom. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Lucio M Dery, Paul Michel, Ameet Talwalkar, and Graham Neubig. 2021. Should we be pre-training? an argument for end-task aware training as an alternative. *arXiv preprint arXiv:2109.07437*.

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.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. 2015. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. 2020. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7882–7926, Online. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

A Appendix

Algorithm 2 Monte Carlo Tree Search

Require: $\{\mathcal{T}_n\}$: A set of n auxiliary tasks
Require: f_θ: Current model

function MCTS(f_θ)
  while within computation budget **do**
    $\mathcal{T}_l \leftarrow$ TREEPOLICY(f_θ, null)
    $r \leftarrow$ SIMULATE($\mathcal{T}$)
    BACKUP($\mathcal{T}_l, r$)
  end while
  return $\arg\max_{\mathcal{T}}$ UCT(null, 0)
end function
function TREEPOLICY($f_\theta, \mathcal{T}$)
  while $\mathcal{T}$ is nonterminal **do**
    if $\mathcal{T}$ not fully expanded **then**
      choose an untried tasks $\mathcal{T}'$
      add a new child $\mathcal{T}'$ to $\mathcal{T}$
      Train f_θ on T': $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(T')$
      return $\mathcal{T}'$
    else
      $\mathcal{T} \leftarrow \arg\max_{\mathcal{T}}$ UCT($\mathcal{T}, c$)
      Train f_θ on T: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(T)$
    end if
  end while
  return $\mathcal{T}$
end function
function SIMULATE($\mathcal{T}$)
  while $\mathcal{T}$ is nonterminal **do**
    choose $\mathcal{T}'$ randomly
    Train f_θ on T: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(T)$
    $\mathcal{T} \leftarrow \mathcal{T}'$
  end while
  Train f_θ on $\mathcal{T}^*$: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(T)$
  $r \leftarrow$ evaluate f_θ on $\mathcal{T}^*$
  return r
end function
function BACKUP($\mathcal{T}, r$)
  while $\mathcal{T} \neq$ null **do**
    $N(\mathcal{T}) \leftarrow N(\mathcal{T}) + 1$
    $Q(\mathcal{T}) \leftarrow Q(\mathcal{T}) + r$
    $\mathcal{T} \leftarrow$ parent of $\mathcal{T}$
  end while
end function
function UCT($\mathcal{T}, r$)
  return $\frac{Q(v')}{N(v')} + c\sqrt{\frac{2\log N(v)}{N(v')}}$
end function