Smoothing Multi-Stage Fine-Tuning in Multi-Task NLP Applications

Oleg Rudenko (orudenko@stanford.edu) and Amir Ziai (amirziai@stanford.edu)

Introduction

A recent trend in many NLP applications is to fine-tune a network pre-trained on a language modeling task in multiple stages. This approach has been very effective partly due to the success of models such as BERT [1]. However, to the best of our knowledge, no systematic study has been conducted to understand the effects of the training schedule in a multi-task setting with a shared representation [3].

In this work we show that the choice of the training schedule in a multi-task setting with a shared representation among tasks is an important factor for performance and convergence speed. The input to our scheduling algorithm is a set of NLP tasks and the output is which task to train on next. Once this selection is made, we feed a batch of inputs from the selected task, which is a text document encoded using either binary bag-of-words or word embeddings into a model for training and output a binary classification. We have used a variety of tasks such as sentiment classification, answer entailment, and paraphrasing.

Related work

Curriculum learning [16] is the idea of improving generalization and convergence speed of a model by sequencing training examples from easy to difficult. This framework has been extensively used in reinforcement learning and there's empirical evidence suggesting that it can improve performance in some cases. However, defining difficulty is not straightforward and requires a deep understanding of the domain. Also, for some training problems it may be more beneficial to start with more difficult problems [12].

An alternative approach is to weigh the tasks statically [13] and treat this weighting as a hyper-parameter that can be tuned. This approach is very expensive since we need to train the entire model multiple times in order to find the best weights. Also the search space grows exponentially as we add more tasks. Guo et al. [12] define the difficulty of a task as the inverse of performance and develop a framework for changing each task's loss objective during training. This is a very elegant approach but introduces some complexity to the training loop. A more promising line of research, which our work draws inspiration from, is to use an agent for learning from progress signals. This approach has been successfully used for learning the optimal architecture for neural networks [15].

Datasets

The data for this study comes from the publicly available GLUE benchmark [2], which we used without modification.

The Stanford Sentiment Treebank (SST-2)
Each record in this dataset is a movie review with a corresponding human-annotated positive or negative label. There are 67k training and 872 validation instances.

The Microsoft Research Paraphrase Corpus (MRPC)
This dataset consists of pairs of sentences extracted from the news and the label is whether one sentence is a paraphrasing of the other. There are 3.6k training and 408 validation instances.

Quora Question Pairs (QQP)

The Quora Question Pairs dataset is curated from the question answering website Quora. The task is to determine whether two questions are semantically equivalent with 363k training and 40k validation examples.

Stanford Question Answering (QNLI)
This dataset consists of pairs of questions and sentences. There are 105k training and 5k validation instances. The label is whether the answer to the question is entailed in the sentence.

Features

We used three sets of features for the models we explored in this work. For all of these feature sets we first converted all characters to lowercase and then split each document on white spaces. The first feature set we use is binary bag-of-words representation where the feature value is set to 1 in the document's feature vector if the token is present and 0 otherwise for the top 10k most occurring tokens. For multi-layer perceptrons we used a 300 dimensional embedding layer which was trained from scratch with a 10k vocabulary size. For BERT (details in the next section), we used the WordPiece segmentation used in the original paper which segments tokens at the sub-word grain and helps with handling of rare words [11].

Methods

Multi-Layer Perceptron (MLP)
We use multi-layer perceptrons with a single hidden layer, hyperbolic tangent activation, and dropout layer with probability 0.2 between the hidden and the output layer, which is trained with binary cross entropy loss.

Bidirectional Encoder Representations from Transformers (BERT)
BERT is a language representation model that is trained on vast amounts of unlabeled text data and subsequently fine-tuned to specific supervised tasks. The model is based on a network architecture called the transformer [10] and does not involve recurrent or convolutional layers. The model takes a sequence of tokens and each token is passed serially through the transformer blocks. Each token is eventually turned into a 768 dimensional vector, we capped the number of tokens to 20. For classification tasks a special token [CLS] is passed in the beginning of the sequence and we pass the first 768-dimensional output vector corresponding to this position to a linear classification layer.

Training schedules
The training schedule is a sequential decision-making problem. At every training epoch we have to choose which task to feed to the model next. We considered four training schedules.

1- Round robin
This training schedule simply cycles between batches of the available tasks.

2- Proportional
For this schedule we sample batches weighted by the size of the training set. For instance, let's say that we only have tasks A and B with 90k and 10k instances respectively. In this example task A will be selected 90% of the time and task B will be selected 10% of the time.

3- ε-greedy
Our objective is to train the model using a sequence of batches so that we maximize a metric of interest. Let's say that we are mainly concerned with improving the validation F1 score for the SST-2 task. Therefore, we can observe the improvement for this metric at step T relative to the previous step ($\Delta {F_1}^T = {F_1}^T - {F_1}^{T-1}$) after each batch and decide which task to pick next. The ε-greedy algorithm [4] is a simple approach for tackling this problem. ε refers to the only parameter of this algorithm where $\varepsilon \in [0, 1]$. For each decision we generate a random value $p \in [0, 1]$. If $p <= \varepsilon$, then we randomly pick one of the tasks. If $p > \varepsilon$ we pick the task with the highest average ΔF_1 where we maintain averages for each task separately.

4- Upper-Confidence Bound (UCB)

The ε-greedy algorithm can be wasteful in scenarios where we have collected a lot of data to have confidence in that some of the tasks are better than others. The UCB algorithm first tries all of the available choices. Once this exploration is over we select the task that maximizes this quantity:

$$\overline{\Delta F_1}^{a} + \sqrt{\frac{2 \log t}{N_a}}$$

where a is a task, N_a is the number of times that the task a has been selected, and t is the epoch number. The intuition for this formulation is that the algorithm should explore tasks that have not been explored a lot.

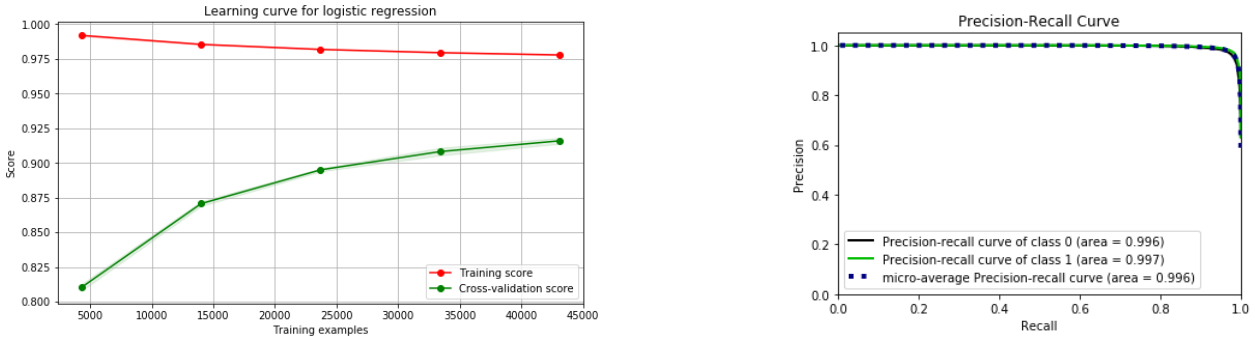
Experiments

Evaluation Metric

Our primary metric is the validation F1 score, which is calculated as the harmonic mean of precision and recall: $F_1 = 2 \times P \times R/(P + R)$ where P is the precision and R is the recall.

Baseline

To establish a baseline for our main task (SST-2), we trained a logistic regression model with L2 regularization and regularization parameter equal to 1. This model achieves a mean 5-fold cross-validation F1 score of 0.91, which is very close to the state-of-the-art for this task at 0.96. Figure 1(a) depicts the learning curves for training and cross-validation as more data is made available to the model, which points to a continual improvement with more data. Figure 2(b) depicts the precision-recall curve.



(a) (b)

Figure 1- Learning curves and the precision-recall curve for the baseline model

Figure 2 depicts examples of correct and incorrect predictions (visualized using LIME [5]). As shown in this example, the model has trouble with cases where the semantics of the review is different from the linear combination of the sentiment of the present words. In the portrayed examples blue words have a negative weight and orange ones have a positive weight.

(a) Correctly predicted negative (b) Incorrectly predicted positive

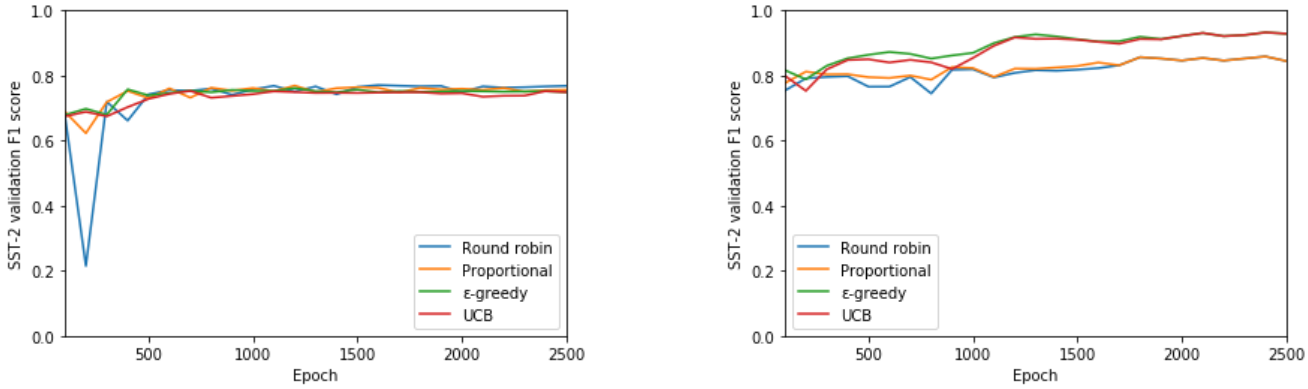
Figure 2- Logistic regression predictions explained for two SST-2 reviews.

Experiment 1

For our first experiment we recorded the validation F1 score for the SST-2 after each epoch of training. In all experiments we trained the algorithms for 2,500 epochs. We found that most configurations plateaued around this number. We trained both MLP and BERT, and limited training to batches from SST-2 and MRPC tasks. For MLP we selected a single hidden-layer with 256 units, Adam optimizer with learning rate of 0.0003, and batch size of 64. We found these values

empirically by running randomized search over a wide range for each hyper-parameter and selected the ones based on the highest average performance over 5 runs. We used the uncased base BERT with dropout equal to 0.1 and learning rate set to 0.0003, following the paper. We also experimented with tuning these hyper-parameters but the gains were minor.

Figure 3 depicts the evolution of the validation F1 score for four training strategies separately for (a) MLP and (b) BERT. Each point on the graph is the average of 5 different runs for the same configuration.



(a) MLP (b) BERT

Figure 3- Evolution of validation F1 score for round robin, proportional, ε-greedy, and UCB

From this initial experiment we noticed that ε-greedy and UCB performed very similarly, which continued in the subsequent experiments as well. We made the same observation between proportional and round robin strategies and decided to only report round robin and ε-greedy going forward. Similarly, we can see in Figure 3 that MLP is not making progress after some initial training even though training F1 gets very close to 1.0. In other words MLP is overfitting to the data. Our interpretation of this finding is that BERT is pre-trained on related language tasks and has a more appropriate architecture for exploiting structure in sequential data relative to MLP. Based on this finding we'll only report BERT results in the next experiment.

Experiment 2

For the second experiment we fine-tuned BERT with SST-2 and one of MRPC, QQP, QNLI, or a combination of all the three tasks. Table 1 shows the train and validation F1 score for the SST-2 task at 500 and 2,500 epochs. At around 500 epochs, as evident in the prior experiment, most algorithm and training schedule combination have gone through a period of initial training and we get a read on how quickly the configuration is making progress. As mentioned in the previous experiment we have capped training at 2,500 epochs.

Additional task	Training schedule	Train N=67k,4k, 105k, and 363k for SST-2, MRPC, QNLI, and QQP		Validation Baseline: 0.90 N=872	
		@500 epochs	@2,500 epochs	@500 epochs	@2,500 epochs
MRPC	Round robin	0.84	0.92	0.79	0.81
	ε-greedy	0.87	0.96	0.83	0.93
QQP	Round robin	0.85	0.93	0.81	0.82
	ε-greedy	0.88	**0.97**	**0.82**	0.94
QNLI	Round robin	0.81	0.88	0.74	0.84
	ε-greedy	0.84	0.94	0.80	0.91

All	Round robin	0.85	0.92	0.72	0.83
	ε-greedy	**0.89**	0.96	0.81	**0.95**

Table 1- Train and validation F1 score for BERT trained jointly on SST-2 and additional tasks

Discussion

These experiments suggest that the choice of training schedule is an important factor for fine-tuning. The results in Table 2 show that by using ε-greedy relative to round robin, which feeds more batches of the task with the highest gain for the task of interest, the model converges faster and to higher validation scores. This is especially important when training large models since the training time per epoch may be long and we can save time and computational resources by achieving similar results more quickly. Moreover, ε-greedy really shines when multiple tasks are involved so valuable resources are not wasted on training epochs for tasks that do not add a lot of value. Finally, we noticed that the QNLI task was the least helpful among all tasks for the SST-2 task since it consistently had the lowest average gain in F1 score. This is possibly due to the fact that QNLI consists of more formal language and SST-2 and QQP are more conversational. Effectively, the ε-greedy method was able to empirically find the most relevant task and exploit the training of that task for maximizing the validation score.

Conclusion and future work

In this work we explored the effects of the choice of the training schedule in a multi-task NLP setting. We have shown that using an ε-greedy approach can result in a faster convergence of BERT, a large model pre-trained on massive unlabeled text data, for fine-tuning on new tasks. We will expand the list of tasks to include sentence similarity (STS-B) and coreference resolution (WNLI). We want to both train on an expanded set as well as evaluate on other tasks. Also, we want to explore different choices of network architecture, hyper-parameters, and pre-trained embeddings.

Contributions

Oleg and Amir contributed equally to the experiments and the write-up.

Acknowledgements

We would like to thank Nimit Sharad Sohoni and Sen Wu from the DAWN Lab for providing very valuable feedback. We would also like to thank the jiant[6] team at NYU for providing a very powerful open-source framework which facilitated our experiments with GLUE.

Code

The code for this work is available at https://github.com/amirziai/cs229-project.

References

[1] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[2] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O. and Bowman, S.R., 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.

[3] Ratner, A., Hancock, B., Ré, C., 2019. Emerging Topics in Multi-Task Learning Systems. https://hazyresearch.github.io/snorkel/blog/mtl_systems.html

[4] Kuleshov, Volodymyr, and Doina Precup. "Algorithms for multi-armed bandit problems." arXiv preprint arXiv:1402.6028 (2014).

[5] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, 2016.

[6] Wang, Alex, et al. "jiant 1.1: A software toolkit for research on general-purpose text understanding models." Note: http://jiant. info/Cited by: footnote 4 (2019).

[7] Joulin, Armand, et al. "Fasttext. zip: Compressing text classification models." arXiv preprint arXiv:1612.03651 (2016).

[8] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
[9] Garivier, Aurélien, and Eric Moulines. "On upper-confidence bound policies for switching bandit problems." International Conference on Algorithmic Learning Theory. Springer, Berlin, Heidelberg, 2011.
[10] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.
[11] Wu, Yonghui, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." arXiv preprint arXiv:1609.08144 (2016).
[12] Guo, Michelle, et al. "Dynamic task prioritization for multitask learning." Proceedings of the European Conference on Computer Vision (ECCV). 2018.
[13] Kokkinos, Iasonas. "Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.
[14] Xu, Wei, et al. "Multi-task classification with sequential instances and tasks." Signal Processing: Image Communication 64 (2018): 59-67.
[15] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).
[16] Bengio, Yoshua, et al. "Curriculum learning." Proceedings of the 26th annual international conference on machine learning. ACM, 2009.
[17] Yang, Yi, et al. "Multi-class active learning by uncertainty sampling with diversity maximization." International Journal of Computer Vision 113.2 (2015): 113-127.