MOPS: Memory Optimization and Performance Surveying when Parameter Sharing in a Multi-Task minBERT Architecture.

Stanford CS224N Custom Final Project

Anthony Weng
Department of Computer Science
Stanford University
ad2weng@stanford.edu

Callum Burgess
Department of Computer Science
Stanford University
callumb@stanford.edu

Mark Bechthold
Department of Computer Science
Stanford University
markpb2@stanford.edu

1 Key Information

- External collaborators: None.
- Mentor (custom project only): N/A.
- Sharing project: No.

2 Research paper summary

Title	Multi-Task Learning Using BERT With Soft Parameter Sharing Between Layers [1]
Authors	Niraj Pahari, Kazutaka Shimada
Venue	International Conference on Soft Computing and Intelligent Systems (SCIS); International Symposium on Advanced Intelligent Systems (ISIS)
Year	2022
URL	`https://ieeexplore.ieee.org/document/10001943`

Background. With the advent of novel deep learning frameworks like BERT, methods for natural language processing (NLP) have been able to achieve high-water marks on a variety of tasks. However, models using such methods are often developed for single task applications [2] and require a huge amount of data to train the underlying architectures in order to achieve state-of-the-art performance.

Multi-task learning (MTL) is a model architecture paradigm which seeks to subvert the demands of task-specific model development and potential task-specific data scarcity. Through model construction and training choices, models which employ MTL seek to generalize well across related tasks while being trained using a singular data source. MTL can be employed in two ways: *hard parameter sharing* (where some early-stage model layers are shared for multiple tasks, and task-specific layers are kept separate) and *soft parameter sharing* (where task-specific model layers remain separate but parameters of the layers are regularized to encourage similarity).

MTL has shown promise in that learning multiple tasks simultaneously can indeed improve generalization performance [3]; however, well-defined strategies for employing MTL (e.g., answering questions like "what layers do should my task-specific models share?") and quantification of its effectiveness remain open to exploration. This work seeks to contribute to both domains. It does so by evaluating the performance of various MTL BERT architectures on three tasks: sentiment classification, category classification, and aspect-opinion sequence classification. The Restaurant-ACOS dataset [4] is used for training and evaluation of all tasks.

Summary of contributions.

1. A framework for conceptualizing and implementing MTL: a model layer taxonomy (frozen, shared, or individual layers) is provided and employed, enabling MTL regimes (specified by their layer groups–e.g., "Frozen-Shared-Individual") to be described and analyzed in a unified manner.
2. Efficacy analysis of different MTL regimes: MTL regimes like "Frozen-Shared-Individual" and "Shared-Shared-Shared" are implemented in a BERT architecture and evaluated on the tasks described in **Background**, providing quantified reference figures for performance gains when using varying MTL regimes.
3. Insight into the knowledge representation of the various layers in a BERT model: variations in the number of layers in each layer group for identical MTL regimes (e.g., a 4-4-4 versus a 1-7-4 layer grouping when employing the same "Frozen-Shared-Individual" MTL regime for a 12-layer BERT architecture) and comparisons of their performances on evaluation tasks provide clues as to what knowledge each BERT layer/layer group captures.

Limitations and discussion. Pahari and Shimada (2022) apply various MTL regimes (with some intra-regime layer grouping variation as well) to a 12-layer BERT architecture to appreciable performance gains across all three of their evaluation tasks (as measured by F1-score). While these results demonstrate promise for applying the MTL paradigm to other model architectures, notable limitations in the research methodology signal toward areas for further MTL exploration and development. Key limitations include:

1. Exploring MTL regimes via soft parameter sharing only: the MTL BERT architectures constructed by the authors conduct MTL via soft parameter sharing. As noted in '**Background**', MTL can also be employed via hard parameter sharing. Evaluation of the performance preservation/gains under MTL regimes using hard parameter sharing could be of particular interest given the capacity of hard parameter sharing to reduce both the model training and compute times and memory occupancy.
2. Restricting model assessment to absolute performance gains: the authors compare various MTL regimes to the baseline BERT models (i.e., individual BERT models trained for each evaluation task separately) on the basis of absolute F1-score for the three evaluation tasks. However, this means of evaluating the effectiveness of MTL regimes misses the opportunity to connect MTL to memory- and/or compute-bound contexts, where some performance sacrifice may be acceptable if memory and compute savings are significant.
3. Manual, unstructured exploration of the MTL regime space: the authors do explore intra-regime performance variations for a given MTL regime when altering the number of layers in each layer grouping, but they do so in a manual, unstructured manner. Though this type of exploration may be sufficient for MTL regimes with limited numbers of layer groupings and/or base model layers, it will scale poorly as these aspects increase.

In short, the limitations of this work prompt three key tasks for our exploration: (1) defining and constructing MTL regimes employing hard parameter sharing; (2) evaluating said regimes on both absolute and memory- and compute-relative bases; and (3) defining a structured methodology or guiding heuristics for efficient MTL regime exploration. Responding to these ideas may improve the accessibility of cutting-edge NLP architectures like BERT in resource-bound contexts.

Why this paper? Many of the strides in modern NLP architectures are born from computing environments sponsored by commercial and institutional entities where memory and/or runtime demands are a second order consideration. However, not all computing environments have the luxury of institutional sponsorship. As this project's authors believe democratizing and improving accessibility to NLP architectures is critical to ensuring equity in the distribution of benefits from and future development of such models, project extensions with the potential for compute-friendly effects were of particular interest. Parameter sharing rose as a standout extension due to its simplicity in principle, allowing for analytic thoroughness to be more readily attained. This project's authors sought an example of parameter sharing and MTL applied to the BERT architecture as well as a conceptual taxonomy of parameter sharing regimes to guide their extension design, which is precisely what the reviewed paper by Pahari and Shimada (2022) provides.

Wider research context. Language is fundamentally hard for computers to model due to the inter-context variation of word meanings. Though we common prescribe "context" to mean the tokens surrounding a given word, a human reader understands that a word's context is far more than that, including textual elements like preceding/following paragraphs, social elements like differences in word meaning when used in certain settings and by certain actors, etc. In an effort to learn contextually-robust word representations, this paper—and more fundamentally, strategies for multi-task learning like parameter sharing—implicitly expand the definition of context space in NLP to include not only different sets of surrounding tokens but also different tasks being accomplished. Ultimately, by doing so, the NLP architectures which learn word representations will (ideally) become one step closer to the gold standard NLP attempts to approximate: human language processing.

3 Project description (1-2 pages)

Goal. The goals of this project are to address the following questions:

1. How can a hard parameter sharing regime be deployed in a multitask minBERT model architecture?
2. On both an absolute and time- and memory-adjusted basis, what is the efficacy of a multitask minBERT model architecture utilizing a hard parameter sharing regime as compared to the same minBERT model with no parameter sharing?
3. *Stretch:* How can the space of hard parameter sharing regimes be searched efficiently? Relatedly, what characteristics do effective parameter sharing regimes share and how can these characteristics be used to design candidate parameter sharing regimes requiring little iterative improvements?

The motivation for pursuing these goals is described in detail in the '**Why this paper?**' and '**Wider research context**' sections and, for convenience, will be summarized here as follows:

- Modern NLP architectures are able to achieve state-of-the-art results on a plethora of language tasks, but developing and training these architectures is memory and compute time-intensive task, often requiring an institutional or commercial backer/sponsorship.
- Parameter sharing is a multi-task learning (MTL) technique which (ideally) both encourages model generalization/performance preservation across related tasks and reduces the memory and compute resources required to development and train even state-of-the-art NLP architectures like BERT.
- Pahari and Shimada (2022) find *soft* parameter sharing within a BERT architecture can lead to performance gains (as measured by F1-score) in a multi-task context, but these authors do not explore the potential of *hard* parameter sharing.
- Accomplishing the previously-stated goals of this project will build off Pahari and Shimada's 2022 work to provide effective strategies for implementing MTL via both soft and hard parameter sharing.
- If effective strategies for both soft and hard parameter sharing (as well as the potential performance variation–compute/memory reduction tradeoff of the latter) are well-studied and documented, said strategies can be deployed to improve accessibility to cutting-edge NLP architectures. Improved accessibility is desirable both for the broader engagement with and advancement of the field as well as for the improvement in distribution equity of the benefits derived from modern NLP technologies.

Tasks. For the base component of the final project, we will address the tasks outlined in the "Default Project Handout." For our group-specific extension tasks, please see the concluding paragraph of '**Limitations and discussion.**' (i.e., "In short, the limitations of this work prompt three key tasks for our exploration...").

Data. For the base component of the final project, the following datasets will be used: Stanford Setniment Treebank (SST) and CFIMDB datasets. For the extension component of the final project, the following datasets will be used: SST, Quora, and SemEval STS Benchmark datasets.

All datasets will likely be passed through a preprocessing pipeline consisting of (but not limited to): cleaning non-alphabetic characters, case and space standardization, tokenization (handled by `tokenizer.py`), stop words removal, and stemming and/or lemmatization. Part-of-speech tagging and named entity recognition may also be conducted.

Methods. The primary method we will employ is *hard parameter sharing* (defined in the '**Background**' of the '**Research paper summary**').

As hard parameter sharing is a general strategy for model construction in MTL contexts rather than a specific method, all aspects of any hard parameter sharing regime will be implemented by our project group.

We may borrow from Pahari and Shimada (2022)'s layer taxonomy (i.e., conceptualizing layers and/or groups of layers as "frozen," "shared," or "individual"), but this will be purely conceptual borrowing and their implementation of soft parameter sharing will not be utilized at all.

Baselines. In completing the defined project tasks and accomplishing the outlined goals, multiple iterations of a minBERT model with some type of hard parameter sharing sub-structure will be generated as a key output of the project. These model iterations (and the embeddings generated from them) will be compared using the metrics described in the "Default Project Handout" to the following baselines:

1. The performance of the base minBERT implementation (and associated embeddings) resulting from the base component of the final project.
2. The performance of the same model as in item 1, but with fine-tuning performed on one of the task-specific datasets (e.g., SST, Quora, SemEval STS Benchmark). Three of these fine-tuned models (and associated embeddings) will be generated and used as baselines.

These baselines (i.e., those using only evaluation metrics defined in the "Default Project Handout") will be implemented by our group.

Also, as noted in '**Evaluation**,' we will compute percentage performance gains/losses (as measured by task-specific precision, recall, and F1-score) of the parameter-sharing models relative to the base minBERT implementation with no or some fine-tuning. Computing these metrics will enable comparisons for the performance gains/losses induced by hard parameter sharing versus those induced by soft parameter sharing as computed and previously published by Pahari and Shimada (2022).

Finally, for the weighted performance score described in '**Evaluation**,' we are currently unaware of baselines for this exact type of metric but are looking for qualitatively similar research domains and their potential baselines.

Evaluation. For both the base and extension components of the final project, we will evaluate our minBERT model using the metrics and methods described in the "Default Project Handout."

For the extension component of the final project, we may also create an index combining the relative fractions of train/evaluation times and memory footprint of parameter-sharing models versus some baseline (i.e., a minBERT model with no parameter sharing and separate layers for each evaluation task) to generate weighted performance scores: e.g.,

$$\text{weighted performance score} = \text{accuracy on some task} \times \text{memory savings multiplier} \times \text{train/evaluation time savings multiplier}$$

We may also compute percentage performance gains/losses of models using hard parameter sharing vis-a-vis the performance of our base minBERT implementation with no or some fine-tuning performed. For these percentage-based measurements, we will compute the task-specific precision, recall, and F1-score of each model and use these as our performance metrics through which percentage improvements/losses are computed.

References

[1] Niraj Pahari and Kazutaka Shimada. Multi-task learning using bert with soft parameter sharing between layers. In *2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS)*, pages 1–6. IEEE, 2022.

[2] Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Revisiting multi-task learning in the deep learning era. *arXiv preprint arXiv:2004.13379*, 2(3), 2020.

[3] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008.

[4] Hongjie Cai, Rui Xia, and Jianfei Yu. Aspect-category-opinion-sentiment quadruple extraction with implicit aspects and opinions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 340–350, 2021.