Even Language Models Have to Multitask in This Economy

Stanford CS224N Default Project

Jacqueline Pang
Department of Computer Science
Stanford University
pangjac@stanford.edu

Paul Woringer
ICME
Stanford University
paulwrg@stanford.edu

Abstract

This project investigates the application of Multi-Task Learning (MTL) using the BERT pre-trained model to address three distinct tasks: Sentiment Analysis (SST), Paraphrase Detection (Para), and Semantic Textual Similarity (STS). While single-task models excel at specific tasks, they necessitate separate training for each, leading to inefficiencies.

We conduct a comparative analysis between MTL and single-task approaches, emphasizing MTL's potential for enhanced efficiency and performance owing to shared representations across tasks.

Our experimentation is structured around two distinct design choices: first, we implement a baseline model, augmenting the existing architecture with an additional layer to perform the three task-specific transformations on the output provided by the BERT model. In the initial set of experiments, we explore traditional NLP solutions, such as incorporating supplementary similar datasets and employing Part-of-Speech (POS) tagging and Named Entity Recognition (NER) during tokenization. The second set of experiments focuses on employing different layers tailored to each specific task. For instance, in Paraphrase Detection (Para) and Semantic Textual Similarity (STS), both tasks relying on sentence token comparison, we modify the neural structure to directly extract tokens from the input, bypassing the pooled output tokens.

Finally, we address a key design challenge: how to balance dataset sizes across the three tasks. To tackle this, we employ methods such as random sampling, weighted loss, transfer learning (initializing the model with pre-trained BERT weights and fine-tuning it on a specific problem), and task-specific architectures.

1 Key Information to include

- Mentor: Yuhui Zhang
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Large Language Models (LLMs) tend to get larger and larger with time, as many have found that this greatly increases their performance. However, larger models require more data, time, and energy to train. This costs a lot of money and significantly increases the environmental impact of such models. However, the opposite trend has begun to appear lately, with companies trying to develop relatively smaller models that are able to reach similar performance levels as their larger counterparts (Workshop, 2023) (Tay et al., 2023). Our main objective in this project is to explore how to maximize the performance of a very small language model, the BERT pre-trained model, on a small set of tasks.

The tasks that we will be focusing on are sentiment analysis on the Stanford Sentiment Treebank (SST) data set from Socher et al. (2013), paraphrase detection on a set of questions from the Quora website, and Semantic Textual Similarity (SST) as described in Agirre et al. (2013).

This is a challenging task, as the model is rather small in size and should be able to perform all three tasks. Furthermore, our training data sets are rather small as well, limiting our ability to train our model extensively. This is why we chose to explore and compare the performances of multi task training and of single task training, to make the most of our very limited resources.

3 Related Work

As mentioned in the introduction, Workshop (2023) and Tay et al. (2023) are some examples of the initiatives to create smaller but well performing language models. These models still have a much larger number of parameters (respectively 20 billion and 176 billion) than even the largest BERT model, BERT-Large, which has 340 million parameters.

3.1 NER/POS Tagging

Named Entity Recognition (NER) is employed to extract entities like people, locations, and dates from text. Part of Speech Tagging (POS), is utilized to label words in a sentence with their respective parts of speech. NER and POS are sometimes combined with co-reference resolution,and they can help categorize or filter documents based on mentioned entities, like people and organizations. Additionally, POS tagging aids in identifying verb and noun chunks, which can be valuable for metadata enrichment or enhancing document retrieval.

Both NER and POS tasks are essential upstream tasks that contribute to co-reference resolution and entity linking. Some research Peinelt et al. (2020)has even proposed combining topic models with BERT.

3.2 Scheduling

One classification approach for multi-task learning is to employ round-robin sampling Moseley and Vardi (2022). This method involves cycling through different tasks one after the other until all data are exhausted. However, round-robin scheduling poses certain challenges. Firstly, it aims for fairness by allocating a fixed time slice (or resources) to each task in a sequential manner. Yet, this may not be optimal for multi-task training if one dataset contains more observations for a particular task, leading to overfitting. Secondly, it fails to consider variations in batch sizes, which are essential for addressing potential imbalances in dataset sizes.

3.3 Fine-tuning regularization

Given that we are tasked with handling three distinct subtasks, a natural inquiry arises regarding how to manage conflicting gradients, which are typically encountered in multi-task learning.

Gradient surgery for multi-task learning refers to a technique utilized to tackle the challenge of conflicting gradients when training a neural network on multiple tasks concurrently. In this context, the network endeavors to optimize for various objectives, but the gradients stemming from these diverse tasks may sometimes interfere with one another, resulting in inadequate convergence or subpar performance.

The concept of "gradient surgery for multi-task learning" encompasses several fundamental principles and methodologies aimed at enhancing the fine-tuning process of large-scale pre-trained natural language models. Jiang et al. (2019) introduce a framework known as SMART, standing for Smoothness-inducing Adversarial Regularization, and Bregman proximal point optimization. SMART seeks to manage model complexity by applying explicit regularization during the fine-tuning phase. This approach encourages minimal alteration in the model's output when subjected to minor perturbations in the input, thereby enhancing the model's smoothness and generalization capabilities. By preventing overly aggressive updates during fine-tuning, the SMART framework enhances training stability and generalization performance across a spectrum of NLP tasks.

3.4 Multiple negatives sampling

This is an describe in Henderson et al. (2017) to train their model in the objective of assigning a high dot product to message-response pairs, and a lower dot product to pairs consisting of a message and another random input (a negative). For each input, they sample K possible responses: one which is the correct response, and $K - 1$ negatives. To do so, they use batches of size K and consider all other responses from the batch as negatives. They chose the mean negative log probability of the data as the loss function:

$$\mathcal{L}(x, y, \theta) = -\frac{1}{K}\sum_{i=1}^{K} \log P(x_i, y_i) = -\frac{1}{K}\sum_{i=1}^{K}\left[S(x_i, y_i) - \log\left(\sum_{j\neq i} e^{S(x_i, y_j)}\right)\right]$$

where $S(x, y)$ is the similarity between x and y, which is computed by a given function f. This function can be the output of a neural network for example.

4 Approach

The BERT language model provided in the project is the standard BERT model from the open-source project, Transformers. The tokenization process is comparable, including the distinctive BERT keywords like 'CLS' and 'SEP'. We utilized this pre-trained model in our work.

4.1 Baseline

As a foundation for our experiments, we constructed a straightforward classifier built upon a BERT model. Atop the BERT structure, we added a dropout and a linear layer, each specifically trained for distinct tasks. It's worth noting that the BERT model is composed of a sequence of 12 BERT layers. Each of these layers, termed as a 'BERT-Layer', can be mathematically represented as BL(h) = LN(h + SA(h)), where LN is a layer normalization step, and SA is a self-attention layer. The self-attention mechanism is further defined as SA(h) = FF(LN(h + MH(h))), where FF is a feed-forward network, and MH signifies a multi-head attention layer.

We tackled sentiment classification and paraphrase detection as classification tasks, while semantic similarity was approached as a regression problem. For all three tasks, our strategy involved leveraging the pooled output layer of the BERT model (denoted by 'CLS'). We then applied a single linear projection, incorporating dropout, tailored to each task.

The choice of loss function varied depending on the type of task. For the sentiment classification task, we chose the CrossEntropyLoss function, as it is well suited for multiclass classification problems. For paraphrase detection, BinaryCrossEntropyLoss was used to fit its binary classification nature. Conversely, for the STS task, we opted for the Mean Squared Error (MSE) loss, as it aptly interprets the output as a continuous similarity score.

Out of curiosity, for the STS task, we also have explored an alternative solution: despite being aware that STS is a regression task, we considered discretizing our predictions and formulating it as a multi-class classification problem. Instead of predicting a semantic similarity score ranging from 0 to 5, we considered predicting six probabilities, each corresponding to a separate class. We tested the CrossEntropyLoss for this task, but the MSE loss proved more effective, as we expected.

In our initial approach for the baseline model of multitask training, we would sequentially train the model on each dataset (one epoch on SST, one on paraphrase detection, one on STS, and repeat). Although our experiments indicated that doing so did not significantly affect the final performance, we must recognize this approach as sub-optimal. Essentially, the model tends to overfit on one dataset before moving on to the next, consequently diminishing its performance on the previously trained data. We observed the loss function fluctuating extensively, failing to converge to a stable level. This challenge led us to our first methodology: employing a 'round-robin' strategy to manage the dataset training process.

4.2 Round-Robin Finetuning

The initial assumption of the baseline model (see Figure 1) was that fine-tuning on sentiment classification would adequately generalize to the paraphrasing and similarity prediction tasks. However, an analysis of the loss trends disproves this assumption. To overcome the issue of generalization across varied tasks, we introduced a batch-by-batch round-robin Multi-Task Learning (MTL) strategy (see Figure 2). Starting with the pre-trained BERT weights, the process involves cycling through batches from datasets for SST, paraphrase, and STS tasks in each epoch.

We have also tested two different approaches: maintaining a consistent batch size for all three datasets, and strategically increasing the batch size for the paraphrase data while decreasing the batch sizes for the SST and STS datasets. The latter approach aims to balance the number of batches in each dataset, despite their size disparities, so that the round-robin scheduler does not run out of batches from one dataset halfway through the epoch.

In each iteration, the model updates are made for each batch in a single pass. The prediction structure remains the same as the baseline. This round-robin technique ensures that the fine-tuned model is trained on data from all tasks equally.

While we anticipate that a balanced round-robin fine-tuning approach will lead to efficient learning across multiple tasks, it does inherently limit us to using the same amount of data from each task. The ideal outcome would enhance the performance of the paraphrasing task without detrimentally affecting the sentiment analysis accuracy.

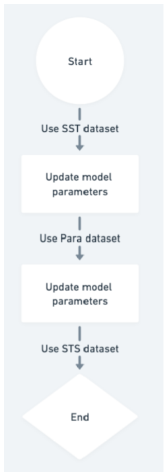


Figure 1: Baseline Architecture

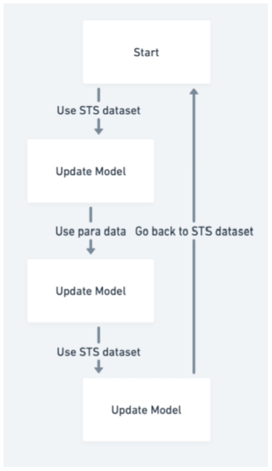


Figure 2: Round-Robin Architecture

4.3 POS/NER tagging combined with BERT

We aim to investigate the extent to which POS/NER tagging can enhance the performance of the BERT model. The tokenization process for BERT models is distinct, especially since these models are designed to predict tokens, which would lead to an impractically large model size if Universal Dependencies (UD) tokenization were used. Instead, these models tokenize text into subword units. Common words are tokenized as whole units, whereas less frequent words are divided into smaller segments, often corresponding to morphemes like affixes. Punctuation is also tokenized at the character level. This approach helps maintain a relatively low number of unique tokens (around 40,000 for BERT) while retaining significant linguistic information. BERT models also utilize specific representations for particular tasks, such as CLS, SEP, and UNK tokens.

When incorporating spaCy to apply POS/NER tagging, a key question arises: how can we use multiple representation vectors generated by BERT and its POS/NER tag ? Our approach is to associate the tag with the last subtoken of the word and assign a special to any <pad>/<cls>/<sep>/<unk> tag to the remaining subtokens (they plays a different role during sentence batching and is disregarded when calculating the accuracy of the predicted tags). For example, if BERT tokenizes the word "workked" into "work" and "##ked", the "##ked" part inherits the POS/NER tag from "work" or recursively looking for its nearest preceding valid word with an appropriate POS/NER tag.

4.4 Fine-tune with Regularized Optimization

We utilize the SMART regularization to avoid the possible over fitting. The SMART implementation is adopted from SMART-Pytorch library. Generally speaking the SMART step optimization can be expressed as:

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_{\text{s}} \mathcal{R}_{\text{s}}(\theta)$$

where $\mathcal{L}(\theta)$ is the loss function, λ_{s} is a weighting parameter for regularization. Due to the time limitations, we have not attempted to make λ_{s} a learnable parameter, and we used a fixed default value instead.

4.5 A small architecture change

Introducing SMART results in modifications to the architecture. The previous sections discussed the layer architecture, noting that all tasks were based on the BERT <CLS> output. However, we are exploring an alternative approach for the PARA and STS tasks, which typically require two different sets of input IDs. Consequently, rather than relying on the BERT CLS output, we extract the embeddings from the second to last layer. We then concatenate these two embeddings in the format <input_id_1> <SEP> <input_id_2> <SEP>. Our objective is to determine whether utilizing vector embeddings directly can enhance performance. This method also aligns with the requirements of SMART-Pytorch, which necessitates the input from the perturbation layer and the final output layer for its operation.

4.6 Multiple negatives sampling

We implemented this in a first step of training the model for the paraphrase detection task, as it seemed to be the most adequate to test it out. For each sentence pair, we used the dot product of their embeddings produced by the BERT model as similarity measure. We estimated that sentences with similar meanings should already have similar embeddings, and that we could strengthen this pattern. Doing so would allow us to have more representative embeddings, that would make it easier for the paraphrase detection layer to discern them. A second phase of training was then the normal end-to-end training for the paraphrase detection.

We note that the pairs in the Quora dataset are not all positive paraphrases. Therefore, to generate the training data for this training we filtered the pairs of sentences that actually correspond to paraphrases. We then used the same strategy discussed in Henderson et al. (2017) of using the other sentences in a batch as negative samples.

5 Experiments

5.1 Data

Provided datasets The BERT model we are utilizing has been pre-trained on Wikipedia articles using two unsupervised tasks: masked language modeling and next sentence prediction. For fine-tuning on downstream prediction tasks, we employ three datasets: the Stanford Sentiment Treebank (SST), Quora, and the SemEval STS Benchmark (STS). It is apparent that the Quora dataset is more substantial in size compared to the other two, as shown below.

SST This dataset comes from Socher et al. (2013). It consists of a set of 11,855 single-sentence movie reviews. It takes the following form:

- Input ID (integer),
- Sentence (string),
- Sentiment label (integer).

The sentiment label corresponds to one of five possible values: negative, somewhat negative, neutral, somewhat positive, or positive. These labels have been produced by human judges.

Quora dataset This dataset consists of a list of 400,000 pairs of questions from the website, that may or may not be paraphrases of each other. The goal of the website is to redirect users to pages that may already contain the answer to their question instead of having duplicates.

- Input ID (integer),
- Sentence 1 (string),
- Sentence 1 (string),
- Paraphrase label (boolean).

STS We used the SemEval STS Benchmark dataset from Agirre et al. (2013). The dataset has 8,628 sentence pairs, with the following features:

- Input ID (integer),
- Sentence 1 (string),
- Sentence 1 (string),
- Similarity measure (float).

Contrarily to paraphrase detection, STS is not a binary decision. The label for this task is a measure of similarity, a floating-point entry that ranges from 0 (completely dissimilar) to 5 (extremely similar).

Additional datasets Additionally, we have integrated two more datasets for the paraphrase and STS tasks. Both datasets had been

MRPC For paraphrasing, we included the Microsoft Research Paraphrase Corpus, which contains 5,800 sentence pairs extracted from online news sources, along with human annotations to denote whether the sentences are paraphrases or semantically equivalent. We formatted this dataset in order to give it the same shape as the Quora dataset.

SICK The Sentences Involving Compositional Knowledge (SICK) dataset from Zhang et al. (2019) is very similar to the STS dataset, except that the similarity measure ranges from 1 to 5 (instead of 0 to 5). It is used for compositional distributional semantics, features a vast number of sentence pairs annotated for relatedness and entailment. To use it in our training, we filtered the features so that it corresponds to the STS dataset, and scaled the label value to have the exact same structure as STS.

5.2 Evaluation method

For sentiment classification and paraphrase detection, we use simple accuracy metric, which is defined by the correctly classified observations percentage. For the STS task, we use Pearson correlation measure.

One significant experiment we were unable to complete due to time constraints is the investigation of the intrinsic relationship between the PAR and STS tasks. These tasks both necessitate the comparison of sentence token similarities (albeit on different scales). We hypothesized that it might be beneficial to explore the potential of a shared layer preceding the divergence into the specific tasks of PAR and STS. Theoretically, knowledge gained from the PAR task should enhance the performance on the STS task.

5.3 Experimental details

The majority of our experiments adhere to a set of common parameters: they are conducted over 10 epochs with a fine-tuning learning rate set at 1e-5 and with a pretraining learning rate set at 1e-3. The dropout probability used for the hidden layers is 0.3. The hidden layer dimension is 768, we have tried to change to other values N*768, but only change the hidden layer dimension does not help the performance, so we choose to stick on 768.

Memory Management Following the baseline experiments, including those where the baseline model was supplemented with extra data, we encountered "CUDA Out of Memory" errors. We used a batch size of 64 for the paraphrase detection task, and 8 for the other two tasks. We implemented mixed precision training (AMP) and utilized gradient accumulation. We observed that while using AMP didn't affect model performance, gradient accumulation did influence it. Due to time constraints, we have not thoroughly examined the mechanisms behind using gradient accumulation yet, but it certainly merits further exploration.

Robin-Round Scheduling In our Round-Robin Multitask Fine-Tuning approach, we implemented two distinct strategies for equally-weighted round-robin. The first strategy involves setting a consistent batch size of 32 across all datasets. For each epoch, we cycle through the datasets sequentially, refraining from revisiting any dataset until all others have been processed for that epoch. The second strategy adjusts the batch sizes to 64 for the PAR dataset and 8 for both the STS and SST datasets, aiming to ensure that each dataset is evenly represented in all batches of a single epoch.

When incorporating SMART regularization into our model, the optimal configuration we found was to use a batch size of 64 for PAR, but only 3 for STS and 2 for SST. Our findings indicate that batch size significantly affects the performance of the model under SMART regularization.

SMART Regularization We found that the values suggested by the SMART-Pytorch library were the most optimal for all parameters except for the regularization weight λ. Specifically we set num_steps = 1, step_size = 1e-3, ϵ = 1e-6 and noise_var = 1e-5. We experiment the regularization weight λ in 0.01, 0.05, 0.1, and use 0.1 for the best model. As mentioned above, to accommodate for memory constraints, we adjusted the default batch size from 64 to 32, and also reduced the batch size of SST/STS to 8.

5.4 Results

Method	Dev. SST	Dev. Paraphrase	Dev. STS	Dev. Mean	Runtime(s)
Baseline	0.487	0.767	0.369	0.541	31
Baseline + extradata	0.459	**0.784**	**0.393**	0.545	49
Robin + extradata	**0.542**	0.780	0.372	**0.565**	92
Robin + extradata + SMART-Regulaization	0.490	0.313	0.062	0.290	2593
Robin + extradata + POS-tokenization	0.499	0.776	0.386	0.554	5989

Table 1: Dev set accuracy based on Fine-tune. Runtimes per training epoch.

We observed that simple round-robin scheduling did not significantly improve upon the baseline approach. We also noted that adding single-task training after multi-task training did not improve our results.

Our results on the SST and paraphrase detection tasks are overall satisfying (test accuracy scores of **0.542** and **0.780** respectively on the leaderboard), as we focused the majority of our efforts on these two tasks. For the SST task in particular, at the time of this submission we rank in the top 10 of the leaderboard. However our STS score stayed pretty low, which brings our overall score down.

6 Analysis

It seems that our biggest flaw in this project is how we handled the STS task. This might be due in great part to the data we used and how we used it. The paraphrase detection dataset was very large, and in our round-robin multitask training approach we gave the same weight to each incoming batch in the computation of the gradient. This might have led this large dataset to dominate the computation of the gradients, despite our efforts to alternate between different datasets. We should maybe have weighted these batches to give the same overall importance to each dataset, this is something we will investigate in future tests.

Data still matters Furthermore, the SICK dataset we used for additional training on this task might not transfer extremely well to the main STS dataset we use for evaluation. We noted that training on the SICK dataset yielded a smaller rate of improvement than training on the STS dataset. By manually examining the two datasets, it feels like the sentence pairs in SICK are not generally more similar than in STS (see Figure 3) - our hypothesis is that the multimodal nature of data distribution might be attributable to labeling bias: people tend to round labels to whole numbers, and when assessing similarity, they apply different subjective criteria. This aligns with what we have learned from lectures, namely that data labeling, when subject to human judgment, can be biased by various factors, such as time or economic environments.

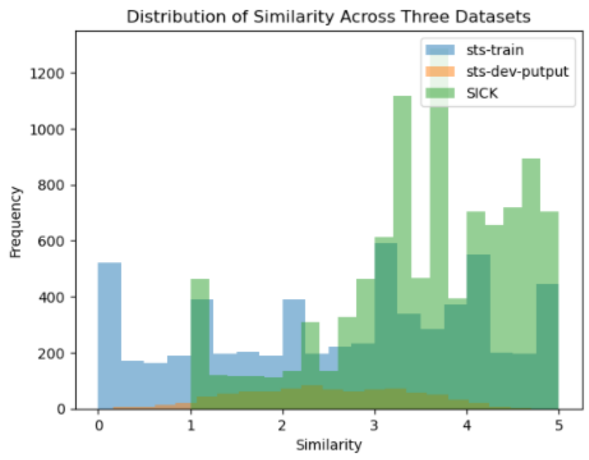
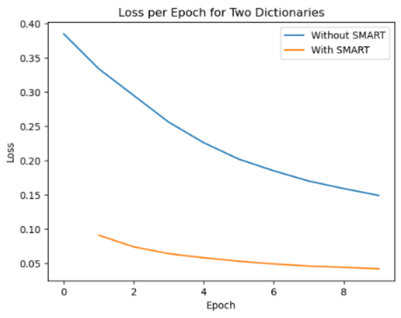


Figure 3: Analysis on STS label(Similarity Score) distribution



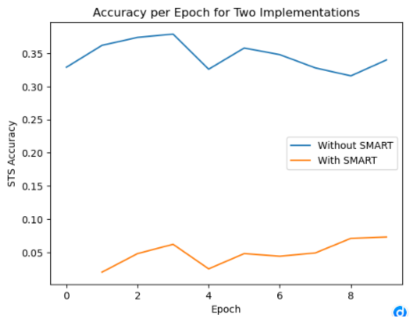


Figure 4: STS Loss Trend W/WO SMART Implementation

Figure 5: STS ACC Trend W/WO SMART Implementation

Why SMART regularization does not work well We aimed to enhance the STS task performance by balancing the training data size across tasks. However, we encountered rapid overfitting, which manifested after approximately 5 epochs. To address this, we implemented SMART regularization, anticipating it would improve STS performance. This necessitated a change in the embedding layer strategy: instead of utilizing the CLS output, we shifted to employing the last embedding layer, concatenating two sentence embeddings with a [SEP] token. SMART's role was to optimize the embedding layer, aiding the model in learning semantic relationships. While the performance on the SST task remained relatively stable, we observed a significant decline in the performance of both the PARA and STS tasks. We suspect that the issue may lie in the choice of the last embedding layer as the strategy, but time constraints prevented a thorough investigation into the optimal embedding strategy. While the performance of SST did not improve, it did not worsen too much either. This outcome can be attributed to the fact that, with a fixed data size, regularization might limit the learning capacity of the model. Additionally, it is plausible that the requirement for a smaller batch size in the SMART implementation (where we reduced the default batch size from 64 to 32 or 16) might introduce some degree of learning instability or "vibrations." This change in batch size could potentially affect the model's ability to generalize by impacting the gradient estimates during training.

7 Conclusion

The architecture we chose to add to the BERT model to enable it to perform these three tasks is rather simple, as we focused mainly on leveraging the structure of minBERT and trying different training paradigms than complex models.

Overall, we managed to reach very good accuracy on the SST and paraphrase detection tasks, which confirms that our approach was good. However, we did not reach such levels of performance on the

STS task, which could be indicative of insufficient data or simply that we did not weigh this task strongly enough in our training to improve our results.

Our future focus will naturally be paying closer attention to the STS task. We are very excited to see how some of the other teams overcame these difficulties and achieved better results on the STS task than us.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Benjamin Moseley and Shai Vardi. 2022. The efficiency-fairness balance of round robin scheduling. *Operations Research Letters*, 50(1):20–27.

Nicole Peinelt, Dong Nguyen, and Maria Liakata. 2020. tbert: Topic models and bert joining forces for semantic similarity detection. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 7047–7055.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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*.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. Ul2: Unifying language learning paradigms.

BigScience Workshop. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Li Zhang, Steven Wilson, and Rada Mihalcea. 2019. Multi-label transfer learning for multi-relational semantic similarity. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*. Association for Computational Linguistics.