Exploring minBERT Performance Optimizations

Stanford CS224N Default Project

Marie Chu
Department of Computer Science
Stanford University
mariechu@stanford.edu

Emmy Thamakaison
Department of Computer Science
Stanford University
emmyst@stanford.edu

Abstract

With the rise in interest of transformers and natural language processing with ChatGPT and GPT4, we decided to revisit traditional problems in NLP such as sentiment analysis, paraphrase detection, and semantic textual similarity to see how we could incorporate minBERT with other optimizations to improve the performance of these tasks. To fine-tune our model's performance on these tasks, we explored substituting optimization algorithms and different regularization parameters. We also incorporated gradient surgery as well as incorporating additional pre-training tasks in hopes of improving the multitask learning. In improving our model's similarity score, which was the lowest of the three, we incorporated cosine similarity into our loss to improve the performance of the specific task. We found out that this significantly improved our model's performance on similarity prediction and was also useful for the paraphrase detection task. Training on all data sets with a round-robin approach, as well as combining gradient surgery (GS) and pre-trained weights on masked language modeling (MLM) of our domain data set and pre-training on an inference prediction task also led to big gains in overall performance over our baseline. Our final model achieved a significant improvement from our baseline (overall: 0.234), with a Sentiment Analysis accuracy of 0.452, Paraphrase accuracy of 0.732, Similarity correlation of 0.584, and overall test score of 0.590.

1 Key Information

Our mentor is Candice Penelton and Gabriel Poesia Reis e Silva. We have no external collaborators nor did we share projects.

2 Introduction

Bidirectional Encoder Representations from Transformers (BERT) was recently introduced as a simple yet empirically powerful language representation model [1]. Its transformer-based approach performs competitively on a variety of tasks, including question-answering and inference prediction. In this project, we aim to implement and improve upon a simpler, functional version of the model called minBERT. The tasks we aimed to optimize include sentiment analysis, semantic textual similarity, and paraphrase detection, which will be expanded upon in the following paragraphs.

Paraphrase detection is the task of comparing sentences in determining whether they share the same meaning. However, according to a recent review, current paraphrase detection methods fall short– a study examining a multitude of plagiarism detection tools concluded that most were unable to identify paraphrases [2]. An even more difficult task than paraphrasing is performing sentiment analysis. In this task the model needs to classify textual bodies (ie. "Positive", "Neutral", "Negative") based on their respective authors' opinions. Due to factors such as sentence complexity, there is still room for improvement on current State-of-the-Art models. [3] In fact, even humans struggle to semantically analyze a text due to double meanings, satire, etc. Semantic Textual Similarity (STS) is

a task that measures the degree of similarity (ie. 0 - being unrelated, 5 - being same meaning) between bodies of text. Though much progress has been made towards STS through semantically-aware word embeddings and transformer-based approaches, gaps remain in areas such as domain-specific word embeddings [4]. From plagiarism detection to news analysis, all three of these tasks present significant real-world applications. Thus, iterating upon them is an important task our team decided to undertake.

Our current project explores ways to improve upon our baseline minBERT's performance on the described tasks above. We performed experiments around optimization algorithms, regularization, round-robin training, and loss functions. Furthermore, we implemented minibatch stochastic gradient descent and extended the multitask classifier's pre-training to include inference classification and masked language modeling on the domain specific datasets. Finally, we assembled combinations of the above changes to produce a model that achieves results that are significantly better than a random model on all three tasks. This model is further described in section 4.

3 Related Work

As mentioned in the introduction, BERT was introduced by Devlin et al. in 2018 as a transformer-based language representation model [1]. It utilizes a masked language model (MLM) pre-training objective, in which the model attempts to predict randomly masked input tokens from its surrounding context. This translates into a versatile design that does not need heavy task-specific architectural modifications. BERT achieves competitive performance on a large suite of sentence-level and token-level tasks, including having a GLUE score of 80.5%, MultiNLI accuracy of 86.5% and SQuAD v1.1 question answering Test F1 score of 93.2 [1]. Since its publishing, BERT has become a ubiquitous baseline for NLP experiments [5].

A key component of the Devlin et al.'s BERT model is the Encoder Transformer layer, which was originally introduced in Vaswani's paper *Attention is All You Need*. The BERT transformer layer consists of multi-head self attention, an additive and normalization layer with a residual connection, a feed forward layer, and another additive and normalization layer with a residual connection [6]. Multi-head self attention is a scaled dot-product across multiple heads which allows the model to jointly attend to multiple representation subspaces [6]. The simpler version of BERT, minBERT was the basis of our model that we sought to improve upon.

Furthermore, our project draws inspiration from Yu et al's work on gradient surgery for multi-task learning. Learning multiple tasks at once presents a challenging optimization problem, as training efforts may lead to worse overall accuracy or lack computational efficiency. This may be due to conflicting gradients of different tasks in certain optimization landscapes. Under such circumstances, the multi-task optimization landscape may be dominated by one task's gradient, which leads to degraded performances for other tasks [7]. Thus, Yu et al. proposes the Projecting Conflicting Gradients (PCGrad) technique, a form of gradient surgery in which conflicting gradients are altered by projecting onto the normal plane of one another. They demonstrate PCGrad's effectiveness in improving the efficiency and performance of multi-task reinforcement learning models.

We also drew inspiration from Sun et al.'s work on fine-tuning BERT for text classification [8]. In this paper, Sun et al. proposed to pretrain BERT on within-task training data, and fine-tune BERT with multitask learning and on the target task itself. Furthermore, we drew from Williams et al.'s work on sentence understanding through inference as an additional pretraining task. Like the authors of these papers, we hoped that our model would have a better understanding of the language it's working with by incorporating these training tasks.

Lastly we drew from Reimers and Gurevych's work on Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks and their measure of cosine similarity in our approach [9] to better capture the similarity between sentences in the similarity and paraphrase detection tasks.

4 Approach

Our baseline model is the multi-task minBERT trained on only the STS dataset. This corresponds to the minBERT starter code created by Stanford University's CS 224N staff and Carnegie Mellon

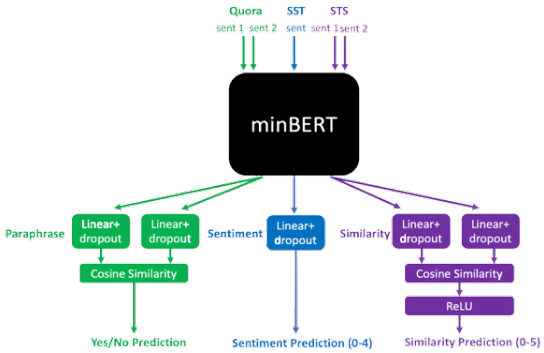

Figure 1: minBERT with three separate heads for downstream tasks of sentiment analysis, paraphrase detection, and semantic textual similarity.

University's CS11-711 staff, which is a more compact, functional version of Devlin et al.'s original BERT model.

In attempting to improve the model, we first explored smaller changes. Our baseline utilized AdamW optimization, a stochastic gradient descent method extended from the Adam (Adaptive Movement Estimation) algorithm that additionally decays weights per techniques described in Loshchilov and Huttner's 2019 paper [10]. We explored substituting AdamW with NAdam (Nesterov-accelerated Adaptive Moment Estimation), another extension of Adam that employs Nesterov momentum [11]. The momentum smooths out noisy objective functions and theoretically improves convergence by adding an exponentially decaying moving average of the gradient to the gradient descent algorithm [11]. Furthermore, we also explored the impact of dropout rate on accuracy. Dropout is a technique of randomly dropping units during training to prevent model over-fitting, and varying the dropout rate has been shown to affect error [12]. We also tested out adding more layers by using two linear layers and a non-linear ReLU activation function for our three tasks to increase the expressiveness of the model.

Additionally, we experimented with a round-robin training method versus a sequential training method. To employ the round-robin method, for each epoch, we selected a batch corresponding to each task and trained them, repeating this process until we ran out of batches. As the datasets included a different number of batches, we accounted for this by indefinitely cycling through the batches of the smaller datasets until we finished all the batches in the largest dataset (Quora). In the sequential method, we trained the model on all of the batches of a data set before moving onto the next data set in the epoch.

On top of round-robin training on the datasets, we also incorporated the PyTorch implementation of gradient surgery by Wei Cheng Tseng [13]. As described above, this technique alters conflicting gradients of multiple tasks by projecting them onto the normal plane of each other [7]. In incorporating this, we hoped to improve the efficiency and accuracy of multitask minBERT.

We also implemented an additional inference prediction task into the pre-training of the multitask model. We utilize the Multi-Genre Natural Language Inference (MultiNLI) corpus created by Williams et al., which contains over 433k broad-coverage sentence pairs annotated with inference labels. We loaded in this dataset and implemented the collate function to pretrain on the MultiNLI dataset in batches. We used a cross entropy loss to update the parameters of our model on the training dataset by how well our model was able to label two sentences as entailment, contradiction, or neutral. By training the model on this dataset, we hoped to improve its natural language understanding and have it translate into better performance on the three downstream tasks [14].

Further, we implemented the code for MLM on the domain specific datasets of Stanford Sentiment Treebank, Semantic Textual Similarity, and the Quora dataset. We loaded in these three datasets to create a new dataset that consisted of only the sentences from each of the three datasets. We then followed the specifications from the original BERT paper, masking out 15% of the tokens in each sentence that weren't [PAD] tokens, [CLS] token, or [SEP] tokens. However, following the note

mentioned in the default project specifications, these tokens weren't masked all the time but rather from that 15%, 80% of the cases are replaced with the [MASK] token, 10% of the tokens are replaced with a random token and 10% remain unchanged.

We pretrained our MLM and inference prediction task with a round robin approach and gradient surgery. We used these pretrained weights for our model and froze the minBERT layers on the subsequent finetuning for the tasks of paraphrase, sentiment analysis, and similarity. Furthermore our loss functions for the three tasks were binary cross entropy, cross entropy, and mean squared error respectively as the first two tasks are classifications and the last task is a regression.

We also looked at incorporating cosine similarity for our similarity and paraphrase tasks. This calculates the cosine similarity between two embeddings, across the specified dimension encapsulated by Equation 1. To do this, we created two linear layers that projected the embeddings of minBERT down to a dimension of 500 in which we subsequently calculated the cosine similarlity and scaled respectively to fit our paraphrase task (0/1 outputs) and semantic textual similarity task (0-5 outputs). We believed this would help the performance of paraphrase detection and semantic textual similarity since two featurized vectors in embedding space that are similar most likely translate to two sentence that are similar in sentence space.

$$\frac{x_1 \cdot x_2}{max(\|x_1\|_2 \|x_2\|_2, \epsilon)} \tag{1}$$

After performing the experiments detailed below, we chose our final model to include frozen pretrained minBERT weights from training on our two pretrain tasks of inference classification and MLM of domain specific data. Furthermore, we utilized gradient surgery and a round robin approach of training on the STS dataset, Quora dataset, and SST dataset with details provided in the next section. We also incorporated cosine similarity in both our semantic textual similarity task as well as our paraphrasing task. A diagram of our final model is provided in Figure 1.

5 Experiments

5.1 Data

For the Sentiment Classification task, we used the Stanford Sentiment Tree (SST) bank [15] and CFIMDB data sets for training and testing. For the multitask portion we only used SST. SST consists of 11,855 single sentences from movie reviews, and each sample was annotated to have a negative, somewhat negative, neutral, somewhat positive or positive label. We used 8,544 examples for training, 1,101 examples for dev, and 2,210 examples for testing, respectively. Additionally, we used 2,434 examples from CFIMDB, which consists of highly polar movie reviews that are either negative or positive. For each phase, we used 1,701 to train, 245 examples for dev, and 488 examples for testing purposes.

For the Paraphrase Detection task, we used the Quora dataset, consisting of 400,000 question pairs wtih labels indicating whether they are paraphrases of one another. This includes a train (141,506 examples), dev (20,215 examples), and test (40,431 examples) split.

For similarity detection, we trained on the SemEval STS Benchmark [16] dataset which consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). For the STS dataset, we have the following splits: train (6,041 examples), dev (864 examples), test (1,726 examples).

Additionally, we also used the SST, STS, and Quora datasets for our pretraining MLM task.

Regarding the inference prediction task, we used MultiNLI, which consists of labeled sentence pairs from ten distinct genres of written and spoken English. As our source of truth, we used the "Gold label", an assigned label for a single pair with the majority vote of the annotators, that could be "contradiction", "entailment", or "neutral". The train, dev, and test datasets consisted of 392,702, 20,000, and 20,000 pairs, respectively.[14]

5.2 Evaluation method

For evaluation of paraphrase detection and sentiment classification, we used accuracy. For semantic textual similarity, we compared the Pearson correlation of the true similarity values against the predicted similarity values across the test data set. The Pearson correlation measures the strength of the linear relationship between two variables.

5.3 Experimental details

For the fine-tuning, tasks we ran our experiments for one epoch with a learning rate of 1e-3 and a dropout rate of 0.1 (except when we were experimenting with different dropout rates, learning rates, and training time). For all the experiments that weren't pretraining on the domain specific dataset or inference prediction, we froze the layers of minBERT. By training on one epoch, this allowed us to quickly test out different additions without being hindered by slow feedback loops.

For pretraining on the domain-specific dataset and inference task, we used a learning rate of 1e-5 and first trained for 5 epochs to obtain the pretrained weights for minBERT. We subsequently loaded these weights and froze the layers of minBERT to use in the subsequent target tasks. These pretrained weights were used in all the experiments shown in Table 2. For the final model we attempted to train for more epochs but ran into the problem of overfitting as described below. Our final model includes pretrained weights with a learning rate of 1e-5, dropout rate of 0.3, and 13 epochs.

5.4 Results

Method	Sentiment	Paraphrase	Similarity
Baseline (Sentiment training only)	0.332	0.380	-0.009
Sequential training on all datasets	0.312	0.627	0.249
Round Robin (RR) on all datasets	**0.378**	**0.661**	**0.282**

Table 1: Training on Multi-task Baseline for 1 epoch

Our training on domain-specific datasets instead of only the SST dataset significantly improved our minBERT performance from the baseline. Employing a round-robin approach instead of a sequential approach to training improved our model's accuracies further. We think this helped the model update its parameters more evenly across all tasks, as the last dataset doesn't "dominate" the predictions of the targeted tasks due to the model "forgetting" what it learned from previous datasets.

Additions	Sentiment	Paraphrase	Similarity
RR baseline	0.378	0.661	0.282
Dropout 0.2	0.387	0.626	0.270
Dropout 0.3	0.361	0.636	0.258
Added Layers	0.375	0.671	0.277
Gradient Surgery	0.361	0.656	0.291
GS + pretraining on domain dataset	0.354	0.654	0.323
pretraining on both	0.431	0.689	0.278
GS + pretraining on both	0.414	0.694	0.268
NAdam	0.381	0.649	0.280
GS + NAdam + pretraining on both	**0.432**	0.693	0.278
pretrain + cosineSim on similarity	0.427	**0.708**	**0.429**

Table 2: Experimental Results for Round-Robin Baseline training on Multi-task for 1 epoch

As seen in Table 2, our implementation of MLM and inference prediction pre-training also incrementally improved the model's performance (Sentiment: 0.431, Paraphrase: 0.689, Simliarity: 0.278), albeit incrementally. This aligns with our initial predictions that this pre-training will help improve the model's lexical understanding and translate to better performance in downstream tasks.

We also found in our initial experiments that GS and NAdam, when used individually, both decreased the model performance, with the overall performance being 0.436 for both GS and NAdam, and the RR baseline being 0.440. However, when combined with some other extensions, such as NAdam + MLM and inference pre-training, they led to superior performance than when they were used individually (Sentiment: 0.432, Paraphrase: 0.693, Simliarity: 0.278).

Interestingly, we found that purely adding more layers had decreased the overall accuract accuracy compared to the Round-Robin (RR) baseline (Sentiment: 0.375, Paraphrase: 0.671, Simlarity: 0.277). This may be attributed to unnecessary layers contributing to overfitting due to our smaller training datasets, leading to higher error. We also found that dropout probabilities of p = 0.2, 0.3 led to lower accuracy compared to RR baseline as well, which can be alternatively explained by the model not being able to fit properly.

The result of Table 2's pretrained weights were obtained with training over 5 epochs. However when we were pretraining the network on the inference prediction task and MLM for 10 epochs, we realized that the inference prediction was overfitting to the training dataset while MLM was still showing gains in the train and dev sets. To combat this, we retrained it with slowed down learning for the inference task so there would be smaller updates to the parameters due to inference. Our final pretrained weights were obtained after 13 epochs and we used them for the subsequent experiments. Before assembling our final model, we trained the combinations in Table 3 for more epochs as we believed they would be promising based on the results of Table 2. This was to confirm or dispel our initial one-epoch results, as well as explore other additions such as cosine similarity, before committing to our final model.

Additions	Sentiment	Paraphrase	Similarity
GS + pretrain on both (9)	0.444	0.712	0.272
GS + pretrain on both + NAdam (4)	**0.449**	**0.715**	0.288
GS + pretrain + cosineSim on similarity (8)	**0.449**	0.710	**0.512**

Table 3: Training on multiple epochs with round robin. *Note*: (n) where n is number of epoch that produced best results

As mentioned, we explored cosine similarity after we had already experimented with training for more epochs on our other models. However, from the results of Table 3 it was clear that cosine similarity was the most promising. As a result, we did a few more shorter experiments with it. The cosineSim in the following experiments refers to a modified cosineSim compared to the one in Table 3.The cosineSim in Table 4 includes an extra ReLU layer after the cosine similarity layer for the similarity task as pictured in Figure 1. We observed that this achieved better results in the first epoch compared to the old cosineSim as values in our dataset were not really "dissimilar". Therefore, instead of linearly shifting and scaling the -1 to 1 outputs of cosine similarity, we instead passed it through a ReLU layer so all "dissimilar" items get mapped to 0 (unrelated sentences) instead.

Additions	Sentiment	Paraphrase	Similarity
pretrain + cosineSim on similarity	0.427	0.708	0.472
pretrain + cosineSim on both	**0.443**	0.717	0.503
NAdam + pretrain + cosineSim on both	0.434	0.718	0.504
GS + pretrain + cosineSim on both	0.418	0.712	0.501
GS + pretrain on both + scaled similarity loss + cosineSim on both	0.441	**0.725**	**0.546**

Table 4: Training different variants of cosineSim. *Note*: these values were obtained with ranges between 1-3 epochs of training

As we were training for multiple epochs, we noticed it was overfitting for the similarity task and there were significant improvements in the dev and train set of similarity while the performance of the other tasks began to degrade, as a result we decided to do a similar process as we did with pretraining and scaled down the loss of the similarity task so its updates to the parameters weren't as aggressive. The improvements of this is showed in Table 4. We also chose to use gradient surgery in our final model because although the results on the short experiment appear to be worse, with more training time, the performance of gradient surgery seems to balance out the performance between the tasks as the parameters are only updated in the direction of their common losses.

Overall, we saw significant improvements in our multi-task model's (depicted in Figure 1) overall performance (dev: 0.598) compared to the baseline (dev: 0.234). We achieved that result by running experiments with different learning rates as well as longer training times as depicted in Figure 2 and Figure 3. Our final model achieved the following results on the test set: **SST test accuracy: 0.452, paraphrase test accuracy: 0.732, STS test correlation: 0.584, and overall test score: 0.590**.

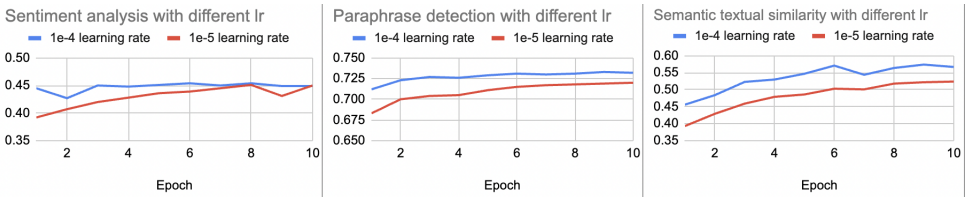

Figure 2: different learning rates effect on performance of each task

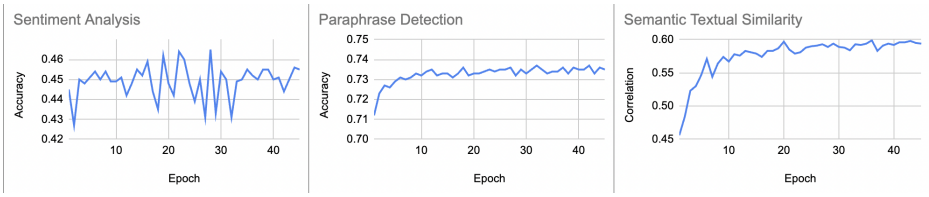

Figure 3: effects of training time on performance of each task.

6 Analysis

Examining the model's performance across the tasks, we see that the model can solve basic instances of a presented problem but struggles with complex prompts. Regarding the sentiment classification task, we illustrate this using the following examples:

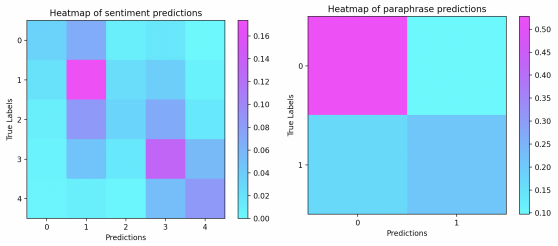

Figure 4: heatmaps of sentiment and paraphrase predictions with the results scaled to the percentage of predictions in each category.

Correct:
Sentence: "Exquisitely nuanced in mood tics and dialogue , this chamber drama is superbly acted by the deeply appealing veteran Bouquet and the chilling but quite human Berling."
Predicted: 4
True: 4
Incorrect:
Sentence: "A rewarding work of art for only the most patient and challenge-hungry moviegoers."
Predicted: 0
True: 3

As seen above, the model appears to understand the task to some extent when presented with an overtly positive/negative prompt. However, the model struggles with nuanced sentences. In the above incorrect example, it may be only focusing on the word "only" and "challenge-hungry", which may dominate the sentence's underlying positive sentiment. As seen in Figure 4, the darker colored diagonal of the heatmap demonstrates that the model for the most part can correctly predict positive and negative sentiment. The model however has difficulty determining the degree of positivity or negativity in the statement.

Correct:
Sentence 1: "What are the latest exciting hollywood movies?"
Sentence 2: "What's the latest Hollywood movie?"
Predicted: 1
True: 1
Incorrect:
Sentence 1: "What are some gift ideas for a female friend?"
Sentence 2: "What are some gift ideas for female friend?"
Predicted: 0
True: 1

For paraphrase detection, we see that the model appears to do pretty well with cases that have parallel syntax and are easily identifiable as paraphrases, however it tends to struggle with examples containing typos and bad grammar. Although the two sentences in the incorrect case are almost identical (except the missing article in front of "female friend"), the model still predicts this incorrectly as not paraphrases. These noisy data points could cause our model trouble as not only does it need to detect paraphrases, it would also have to detect typos and bad grammar. We also noticed the model had difficulty with determining paraphrases when one sentence is significantly longer than the other/one sentence is the summary of another as this is another task in NLP in and of itself. Furthermore, by analyzing Figure 4 we see that our model tends to be conservative in its predictions of paraphrases, often opting for "not a paraphrase" when it is unsure and generating more false negatives than false positives. Lastly, we illustrate some aspects of the model's behavior on the similarity task:

Correct:
Sentence 1: "Two black dogs are playing on the grass."
Sentence 2: " Two black dogs are playing in a grassy plain."
Predicted: 4.1
True: 4.6
Incorrect:
Sentence 1: "UN chief welcomes peaceful presidential elections in Guinea"
Sentence 2: "UN chief condemns attack against peacekeepers in Mali"
Predicted: 3.48
True: 1.0

As demonstrated above, the model seems to fixate on parallel syntax between two sentences rather than the semantics.

7 Conclusion

Our multi-task minBERT based model demonstrates sufficient understanding of our downstream tasks, with scores of 0.452 for sentiment analysis, 0.732 for paraphrase detection and 0.584 for similarity correlation. This was achieved with frozen weights from pre-training tasks, gradient surgery to combine the losses of multiple tasks, task specific improvements like cosine similarity in calculating the loss, hyper-parameter tuning, and regularization. We discovered that individually, optimizations may seem to worsen or improve the performance of the model but when used in conjunction with other methods, the improvements and degradation could differ. Although our model already performs significantly better than the baseline, the examples highlighted in the "Analysis" section signals that there is still room for improvement.

Regarding future directions, we foresee infusing word sense– word categories corresponding to multiple definitions– into the input embeddings as a potential avenue towards model improvement. Levine et al. created a Sense-BERT model that attempted to predict a word's supersenses (ie. "noun.food") in conjunction with the pre-existing MLM tasks in the pre-training. Demonstrating their resultant model's superior performance in tests such as Word in Context (WiC) task from the SuperGLUE benchmark, they created a sense-informed model that appears to achieve "significantly improved lexical understanding" when evaluated compared to vanilla BERT [17]. Although non-task-specific, incorporating word sense embeddings may translate into better performance in our downstream tasks.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Toutanova Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. October 2018.

[2] Chao Zhou, Cheng Qiu, and Daniel E. Acuna. Paraphrase identification with deep learning: A review of datasets and methods, 2022.

[3] Meenu Bhagat and Brijesh Bakariya. Sentiment analysis through machine learning: A review. In Garima Mathur, Mahesh Bundele, Mahendra Lalwani, and Marcin Paprzycki, editors, *Proceedings of 2nd International Conference on Artificial Intelligence: Advances and Applications*, pages 633–647, Singapore, 2022. Springer Nature Singapore.

[4] Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity - A survey. *CoRR*, abs/2004.13820, 2020.

[5] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[7] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning, 2020.

[8] Yige Xu Xuanjing Huang Chi Sun, Xipeng Qiu. How to fine-tune bert for text classification? May 2019.

[9] Iryna Gurevych Nils Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. EMNLP, 2019.

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

[11] Timothy Dozat. Incorporating nesterov momentum into adam, Feb 2016.

[12] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[13] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[14] Adina Williams, Nikita Nangia, and Samuel Bowman. 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, June 2018. Association for Computational Linguistics.

[15] Jean Wu Jason Chuang Christopher D Manning Andrew Y Ng Richard Socher, Alex Perelygin and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 conference on empirical methods in natural language processing*, page 1631–1642, 2013.

[16] Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California, June 2016. Association for Computational Linguistics.

[17] Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. Sensebert: Driving some sense into BERT. *CoRR*, abs/1908.05646, 2019.