Improving minBERT Performance on Multiple Tasks through In-domain Pretraining, Negatives Ranking Loss Learning, and Hyperparameter Optimization

Stanford CS224N {Default} Project

Addison Jadwin
Department of Computer Science
Stanford University
ajadwin@stanford.edu

Catherine Huang
Department of Computer Science
Stanford University
chuang75@stanford.edu

Abstract

BERT models have seen a recent explosion in use cases, but an understanding of how to optimize BERT for various tasks is developing. The present study aims to improve the performance of minBERT, a smaller version of the original BERT model, on a variety of sentence-level tasks (sentiment classification, paraphrase detection, and semantic contextual similarity) simultaneously. To do so, we employ an in-domain pretraining strategy in which minBERT is pretrained on a Masked Language Model (MLM) objective on the datasets which it performs tasks on. We also employ Negatives Ranking Loss Learning to improve baseline BERT embeddings. Both strategies, along with optimal learning rate selection, significantly improved minBERT's performance.

1 Key Information to include

- Mentor: Xiaoyuan Ni
- External Collaborators (if you have any): N/A
- Sharing project: No

2 Introduction

BERT (Bidirectional Encoder Representations from Transformers), released in 2018, is a state-of-the-art pre-trained language model that has achieved impressive performance on a wide range of natural language processing tasks. BERT uses a transformer-based architecture that allows it to capture bidirectional context in language data, making it particularly effective at handling tasks that require understanding of the overall meaning of a sentence or document (Devlin et al., 2019).

Our present work aims to improve the performance of minBERT, a minimalist version of BERT that we implemented with many of important components of the BERT model and its architecture, across a variety of sentence-level tasks including sentiment analysis, paraphrase detection, and semantic contextual similarity. By focusing on the important aspects of the BERT model architecture and training process with minBERT, we can better analyze and understand the underlying mechanisms that may make BERT more effective across sentence-level tasks with our extensions.

We implemented several extensions beyond the baseline minBERT model; applying in-domain pretraining through a masked language model (MLM) objective, fine-tuning through multiple negatives ranking loss learning, hyperparameter changes, and further finetuning on the sentiment analysis task. Our implementations drew conceptual inspiration from previously published works (Devlin et al., 2019) and (Henderson et al., 2017), but were adapted developed internally from scratch without relying on third-party frameworks.

3 Related Work

Since their inception, BERT models have proved powerful and useful for a variety of tasks, including text classification (Devlin et al., 2019), question answering (Schwager and Solitario), named entity recognition (Iovine et al., 2022), and machine translation (Garg et al., 2019). How to optimize BERT for a variety of tasks, however, is a different question. Sun et al. (2020) present several key findings on the further pre-training of BERT. In particular, the authors find that pre-training BERT with data of a target task ("within-task" pretraining) was helpful in their experiment with text classification. Additionally, the authors found in their experiments that additional training with in-domain data ("in-domain" pretraining) was also helpful in improving BERT performance, but additional pre-training with cross-domain data did not significantly improve BERT performance.

One potential method of in-domain pretraining on task-specific datasets comes from the original introduction of BERT (Devlin et al., 2019), in which the authors pretrain the model on a Masked Language Model objective which randomly masked tokens of the input sequences in the dataset and trains the model to predict the correct tokens in the masked locations. Pretraining using this method on in-domain data is a method of improving BERT performance for a specific task.

Another relevant method for improving a model's text classification abilities is Multiple Negatives Ranking Loss Learning, detailed by Henderson et al. (2017). The authors introduce a learning strategy which maximizes the similarity between the model's representation (e.g., embeddings) of a training example and a paired similar example while minimizing its distance from multiple other dissimilar examples. Using this version of loss reduced their error by 20% in a natural language response suggestion model focused on message-response pairs. The same training objective is relevant for minBERT embeddings in the context of judging the similarity or paraphrase status of a pair of sentences, since maximizing the similarity of the model's representation of positive pairs of sentences and minimizing the similarity for negative pairs of sentences can help the model discriminate between the two during the task.

4 Approach

We implemented minBERT, a simplified version of BERT that maintains its key aspects, as per the default project handout. We then aimed to improve minBERT's performance across sentiment analysis, paraphrase detection, and measuring semantic similarity, through implementing in-domain pre-training, multiple negatives ranking loss learning, and hyperparameter optimization. Our baselines were derived from the dev accuracies from our multitask classifier model without training: an accuracy of 0.318 for Sentiment Analysis, an accuracy of 0.380 for Paraphrase Detection, and an accuracy of 0.019 for Semantic Textual Similarity. Refer to §5.2 for additional details about our baseline.

4.1 Baseline minBERT

We have implemented a baseline version of the minBERT model described in the default project handout. The model tokenizes sentence inputs into individual word pieces, feeds them through a trainable embedding layer which assigns them position and token embeddings, and then passes them through bert encoder layers which consist of a multi-head self-attention layer, an add and norm layer, then a feed forward linear layer, and another add and norm layer.

During multitask classification, when minBERT is performing 3 different tasks (sentiment analysis, semantic contextual similarity, and paraphrase detection), there are 3 different 'heads', or final layers, which determine the output of the minBERT model. For sentiment analysis, the head is one final linear layer which transforms the minBERT's final pooled encoding to a 5-unit output vector. For semantic contextual similarity, the head takes minBERT's final pooled encoding from the two inputs and computes their cosine similarity, then feeds them through a ReLU and multiplies them by 5 to scale them to the labels in the STS dataset. For paraphrase detection, the head concatenates the minBERT final pooled encodings from each input and passes them through a linear layer which outputs 1 unit indicating their probability of being paraphrased. The model's performance before extensions, compared with the baselines, is detailed in §5.4.

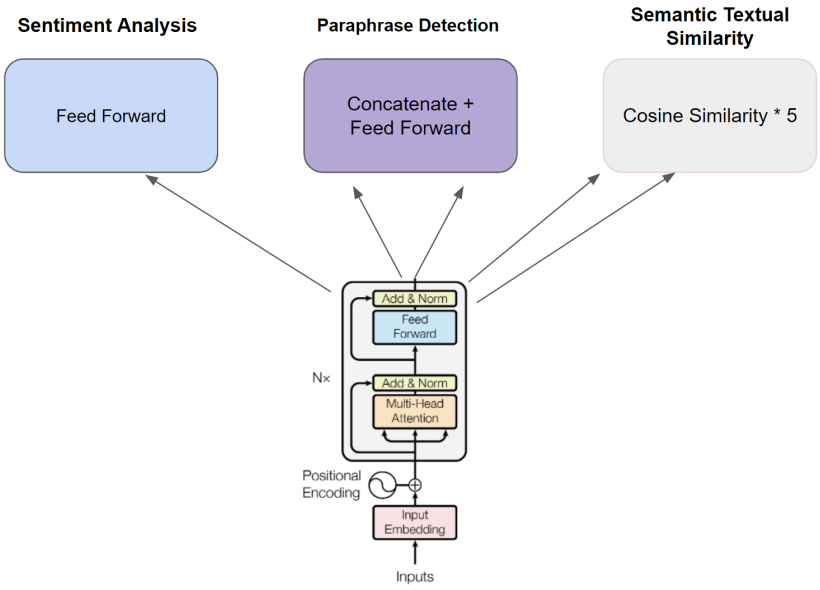

Figure 1: Multitask BERT heads. After being passed through 12 BERT encoder layers, the sentence encodings are passed to 3 different heads perform sentiment analysis, paraphrase detection, and semantic textual similarity predictions separately. Note that the sentiment analysis layer uses one sentence's encoding while the other layers use two. BERT layer image credit: Vaswani et al. (2017)

4.2 In-Domain Pretraining

We also implemented several extensions to this model; namely, we implemented in-domain pretraining through a Masked Language Model (MLM) objective with our own code, where we set the chance of each token in the input text being masked to 15% (this differs from how MLM is implemented in the original version of BERT, where 15% of tokens are masked), and train the model to predict the masked tokens based on the surrounding context, in order for the model to develop a deeper understanding of the structure and meaning of the language it is trained on. An additional MLM head is added on top of the bert model which takes in the sequence encoding (an encoding for each word in the input sequence) and transforms it to a sequence of predictions for the word at each index of the input sequence. We train the model with cross-entropy loss between the predicted words in the masked positions and their respective target one-hot labels representing the correct word.

We trained using the provided training datasets from each of the three tasks (ids-sst-train, quora-train, and sts-train), and utilized a round-robin strategy to switch between datasets, in which the model alternates between each dataset across each batch. To address different dataset length, we took a batch from each dataset; if a dataset had no more batches, we would alternate between the remaining datasets until we trained through every batch for each dataset.

4.3 Multiple Negatives Ranking Loss Learning

We also fine-tuned our model to the semantic similarity measurement task by implementing multiple negative ranking loss with our own code. As detailed by Henderson et al. (2017), the multiple negative ranking loss learning is implemented by a loss function where we train to minimize the approximated mean negative log probability of the data. With this loss function, the training data consists of K sentence pairs $[(a_1, b_1), ..., (a_n, b_n)]$ where a_i, b_i are labeled as similar sentences and all a_i, b_j where $i \neq j$ are not similar sentences. The loss function then minimizes the distance between a_i, b_i while simultaneously maximizing the distance between a_i, b_i where $i \neq j$.

For a single batch, this is calculated as

$$J(x, y, \theta) = \frac{-1}{K} \sum_{i=1}^{K} \left(S(x_i, y_i) - \log \sum_{j=1}^{K} e^{S(x_i, y_j)} \right)$$

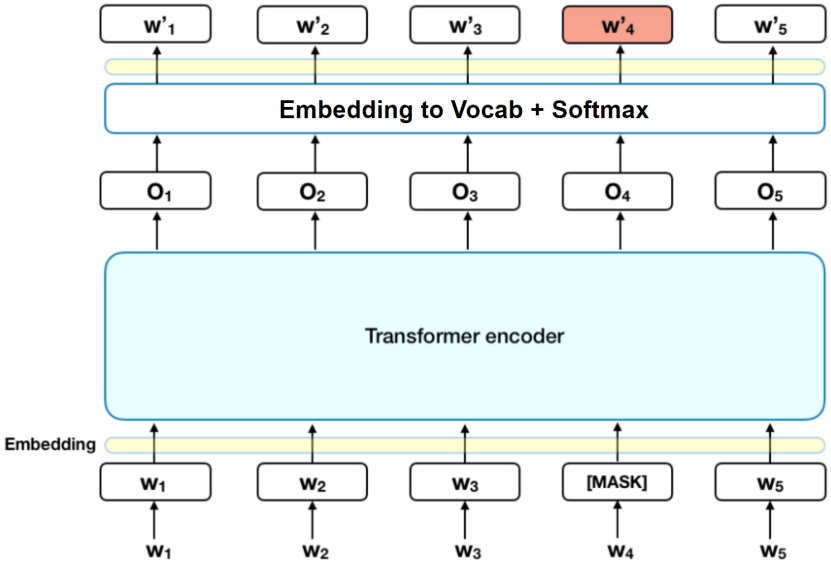

Figure 2: Structure of the MLM objective. Sentences with masked tokens (w) are embedded and encoded through BERT (o), and a final layer converts the embedding size to vocab size and computes softmax for each word to form a prediction (w'). Only the masked token predictions (highlighted in red) are used to compute cross-entropy loss. (Image credit: Rani Horev)

where S is a scoring function that outputs similarity between inputs x and y. We made slight modifications to the loss function calculation to make the loss function compatible with our batch organization:

$$J(x, y, \theta) = \frac{-1}{K_s} \sum_{i=1}^{K_s} \left(S(x_i, y_i) - \log \sum_{j=1}^{K} e^{S(x_i, y_j)} \right)$$

where K is our batch size (and effectively, our total number of sentence pairs), and K_s is the number of similar pairs, where the paraphrase score between the sentences in the pair is 1.0 or the similarity score between the sentences in the pair is greater than or equal to 4.0. Our scoring function utilized cosine similarity between the two sentences.

4.4 Hyperparameter Optimization

Within our pre-training extension with the MLM objective, we pre-trained our model across several learning rates: $1e-3$, $1e-4$, and $1e-5$. We found that pre-training the model with a learning rate of $1e-5$ qualitatively lead to optimal loss decrease across epochs; thus, we saved the weights of the pre-trained model with a learning rate of $1e-5$.

Similarly, within our fine-tuning extension using multiple negatives ranking loss learning, we fine-tuned our model across several learning rates: $1e-3$, $1e-4$, and $1e-5$. We found that fine-tuning the model with a learning rate of $1e-5$ qualitatively lead to optimal performance (a learning rate of $1e-5$ lead to decreasing, converging embedding values); thus, we saved the weights of the fine-tuned model with a learning rate of $1e-5$.

4.5 Training the Multi-Task BERT Heads

After pretraining on MLM and/or fine-tuning with Multiple Negatives Ranking Loss, we finally train the two multi-task heads which contain learnable weights: the sentiment analysis and paraphrase detection heads. We train both of these heads on the training data, while freezing all BERT parameters

as to only update the weights and biases in the heads. We first train the sentiment prediction head and then the paraphrase detection head.

4.6 Further Finetuning on the Sentiment Analysis Task

After training the sentiment analysis and paraphrase heads while freezing the BERT parameters, we further finetune the model, including the BERT parameters, on the sentiment analysis task. We do this because without finetuning on the sentiment analysis task, our results on sentiment analysis are unsatisfactory. See the results section below.

5 Experiments

5.1 Data

We use the three datasets included in the default project handout, for each of the three tasks in our multi-task model.

Our first dataset is the Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013), which consists of 11,855 single sentences extracted from movie reviews, parsed with the Stanford parser and including a total of 215,154 unique phrases from those parse trees. Each of these phrases were labeled by human judges with an integer score between 0 and 4, indicating the sentiment of the sentence, with 0 indicating a "negative" sentence and a 4 indicating a "positive sentence".

The second dataset we used is the Quora dataset (Iyer et al.), which consists of 400,000 question pairs with binary labels of 1 or 0 indicating whether particular instances are paraphrases of one another. Our given training dataset was a subset of the Quora dataset that contained 141,506 examples; the given dev dataset contained 20,215 examples, and the given test dataset contained 40,431 examples.

The third dataset we used is the SemEval STS Benchmark dataset (Agirre et al., 2013), which consists of 8,628 sentence pairs of varying similarities with integer labels on a scale from 0 and 5, indicating the degree of similarity between the two sentences, with 0 indicating unrelated sentences and 5 indicating sentences with equivalent meanings. Our given training, dev, and test datasets were divided accordingly: training contained 6,041 examples, dev contained 864 examples, and test contained 1,726 examples.

We also use the CFIMDB dataset, which consists of 2,434 highly polar movie reviews with binary labels of "negative" or "positive" review sentiment. This dataset was used to evaluate our sentiment classifier in our base minBERT model, but was not used in our multitask classifier and across our extensions.

5.2 Evaluation method

To evaluate our models of minBERT (single-task sentiment classification and multi-task), we compare our dev and test accuracies for the models to the baselines.

When training minBERT for the sentiment classification task, the baselines provided in the default project handout are as follows, with the accuracy bolded and the standard deviation in parentheses:

Pretraining for SST: Dev Accuracy: **0.390** (0.007)
Pretraining for CFIMDB: Dev Accuracy: **0.780** (0.002)
Finetuning for SST: Dev Accuracy: **0.515** (0.004)
Finetuning for CFIMDB: Dev Accuracy: **0.966** (0.007) .

When training multi-task minBERT across sentiment classification, paraphrase detection, and sentence similarity, we compare the dev and test accuracies to the baseline of our multitask classifier model without training. Those baselines are as follows, with the accuracy bolded:

Sentiment Analysis (SST): Dev Accuracy: **0.318**
Paraphrase Detection (Quora): Dev Accuracy: **0.380**
Semantic Textual Similarity (STS): Dev Accuracy: **0.019** .

To evaluate our model, we will compare our accuracies against these baselines; significant increases in accuracy indicate a successful improvement to our baseline minBERT model.

5.3 Experimental details

To evaluate the efficacy of our strategies, we trained 3 different models in addition to the baseline minBERT: one which used just MLM to pretrain the embeddings, one which just used Multiple Negatives Loss Learning to fine-tune the embeddings, one which first pretrained using MLM and then further fine-tuned using Multiple Negatives Loss Learning.

To pre-train using the MLM objective, we used the following parameters: epochs = 10, batch size = 8, hidden dropout probability = 0.3, learning rate = 1e-5. To fine-tune using multiple negatives loss learning, we used the following parameters: epochs = 10, batch size = 8, hidden dropout probability = 0.3, learning rate = 1e-5. In our run with both MLM pre-training and multiple negatives loss learning, we used the following parameters: epochs = 10, batch size = 8, hidden dropout probability = 0.3, learning rate = 1e-5.

When training across our sentiment analysis and semantic contextual similarity heads, we kept the BERT embeddings and parameters frozen, and used the following parameters: epochs = 10, batch size = 8, hidden dropout probability = 0.3, learning rate = 1e-3.

We then decided to fine-tune for sentiment analysis to improve dev performance in the task, and used the following parameters: epochs = 10, batch size = 8, hidden dropout probability = 0.3, learning rate = 1e-5.

Our total training time for MLM took approximately 19 hours. Total training time for Multiple Negatives Loss Learning took approximately 6.6 hours. Total training time for training the multitask heads took approximately 1.2 hours. Training time for the sentiment analysis fine-tuning took approximately 10 minutes.

5.4 Results

Tables 1 and 2 detail our accuracies on the dev set across each task (Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity) with multi-task classification. Table 3 details our accuracies on the test set across each task (Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity) with multi-task classification. Baselines accuracies are obtained from original BERT embeddings without further pretraining or finetuning on the tasks.

Task (multi-task classification)	Baseline	MLM-Only	MNRL-Only	MLM+MNRL
Sentiment Analysis	0.318	0.332	0.279	0.246
Paraphrase Detection	0.380	0.738	0.769	0.768
Semantic Textual Similarity	0.019	0.172	0.569	0.503

Table 1. Model dev set accuracies for each task with multi-task classification.

We see significant improvement in the paraphrase detection task across each model, and significant improvement in semantic textual similarity task in models using multiple negatives ranking loss learning only and using both MLM and MNRL. The improvement on the paraphrase and STS tasks from MNRL finetuning was expected. This demonstrates MNRL's efficacy in improving tasks which rely on comparing the similarity of two sentence embeddings. An unexpected result, however, was that neither MLM nor MNRL improved sentiment analysis performance, which demonstrates that some tasks may require finetuning of the whole bert model. Due to the relatively low accuracies in our models for the sentiment analysis task, we decided to further fine-tune our model on the sentiment analysis task. Our resulting dev set accuracies across each of our three tasks after this fine-tuning can be seen in Table 2, compared to our existing baseline.

Task (multi-task classification)	Baseline	MLM-Only	MNRL-Only	MLM+MNRL
Sentiment Analysis	0.318	0.503	0.500	0.484
Paraphrase Detection	0.380	0.722	0.727	0.748
Semantic Textual Similarity	0.019	0.309	0.605	0.544

Table 2. Model dev set accuracies for each task with multi-task classification, after additional sentiment analysis fine-tuning.

We see that after addition fine-tuning on sentiment analysis, we see more significant improvement on the sentiment analysis task, and still see significant improvement on the paraphrase detection and semantic textual similarity tasks. The boost on sentiment analtis performance is expected. We did not expect the boosts on paraphrase detection and STS results to remain intact so well, which demonstrates the robustness of the embeddings learned from MLM and MNRL. Based on the outputted accuracies on the dev set in Tables 1 and 2, we decided to submit the predictions from our best-performing model. The test-set accuracies we achieved from these predictions can be found in Table 3, compared to our existing baseline.

Task (multi-task classification)	Baseline	MNRL Only
Sentiment Analysis	0.318	0.514
Paraphrase Detection	0.380	0.727
Semantic Textual Similarity	0.019	0.565

Table 3. Model test set accuracies for each task with multi-task classification, using our best performing (on the dev-set) model. Our overall test score is 0.602.

6 Analysis

The first set of key findings, from the data before further sentiment analysis finetuning (see Table 1) is that MLM pretraining alone seems to significantly boost paraphrase detection performance and slightly boost semantic textual similarity (STS) while using MNLR finetuning alone or in tandem with MLM significantly boosts both paraphrase detection and STS performance. In addition, it seems that using MNLR finetuning is slightly detrimental to sentiment analysis performance. MLM probably boosts paraphrase performance since it allows the model to learn more meaningful word embeddings which are rooted in which words appear in which contexts. A big part of detecting a paraphrase is likely deciding whether the two sentences are in the same 'context' by examining the word embeddings. STS performance also receives the slight boost from MLM since the context of each sentence reflected in the word embeddings can be a sign of their semantic similarity as well. MNRL finetuning probably gives the additional boost to STS performance because it allows the model to adjust the cosine similarity of various sentence embeddings in a way which reflects their similarity or paraphrase status before it performs the discrimination task on them. As to why MNLR is slightly detrimental to the sentiment analysis task, we suggest that adjusting the cosine similarity of sentence embeddings based on paraphrase/STS may cause the embeddings to shift in directions which are often irrelevant or tangential to sentiment, such as whether two words appear in the same context.

Additional finetuning on the sentiment analysis task seemed to greatly improve the model's sentiment analysis performance while largely keeping performance on paraphrase and STS intact across all 3 combinations of MLM/MNRL, even slightly boosting STS performance in the case of the MLM-only model. This may be because pretraining on the MLM task and/or finetuning using MNRL is able to produce robust enough word embeddings that maintain their relative cosine similarity and contextual information even when they are adjusted for the specific sentiment analysis task.

7 Conclusion

Masked Language Model (MLM) pretraining, MNRL (multiple negatives ranking loss) finetuning, learning rate selection, and further finetuning on the sentiment analysis all for the most part boosted our model's performance significantly. Overall, it seems that MNRL finetuning gave the model the biggest boost as compared to MLM alone or a combination of the two. Furthermore, we were able to further finetune on the sentiment analysis task alone to boost performance on that task while maintaining MLM/MNRL's benefits on the other tasks. These findings suggest that in-domain pretraining and finetuning with an MNRL function may be good standard practices for improving BERT's performance in a multitask domain. One limitation of the present study is that the model was only further finetuned on the sentiment analysis task. Future work may explore further finetuning on all relevant tasks, rather than just one.

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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT 2019*, pages 4171–4186, Online. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4453–4462, Online. 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. Google Research.

Andrea Iovine, Anjie Fang, Besnik Fetahu, Oleg Rokhlenko, and Shervin Malmasi. 2022. Cyclener: An unsupervised training approach for named entity recognition. In *Proceedings of the ACM Web Conference 2022 (WWW '22)*, Online. Association for Computational Linguistics.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question-pairs.

Sam Schwager and John Solitario. Question and answering on squad 2.0: Bert is all you need.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification? In *Proceedings of the Chinese Computational Linguistics: 18th China National Conference*, pages 194–206. Springer International Publishing.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.