Grid Search for Improvements to BERT

Stanford CS224N Default Project

Harsh Goyal
Department of Computer Science
Stanford University
`hgoyal@stanford.edu`

Abstract

Grid Search can be a useful method for identifying the most and least impactful components of a system. Our goal is to first implement and improve a system based on miniBERT that effectively solves three downstream tasks (Sentiment Analysis, Paraphrase Detection, and Semantic Text Similarity). We then develop extensions to improve the system's performance, and finally perform a grid search to measure the contribution of individual components. In this report, we compare extensions we have implemented, such as **task-adaptive pretraining**, modifications to the **loss function** and **the non-shared layers**, and **prefixing**, yielding a system that improves upon our baselines. We also present a table with results from a grid search across the space of the improvements we made, which may serve as a useful resource for researchers considering including the same mechanisms into their own work. We find that a task-adapted BERT model with non-shared MLP layers performs the best on the tasks, with a score of 0.77 on the test leaderboard.

1 Introduction

The introduction of BERT (Devlin et al., 2018) was considered a revolutionary advancement because of the range of tasks at which the model was able to perform at a state-of-the-art level. This spurred a research stream applying the BERT model to an increasing number of tasks, and attempting to improve its performance on these tasks by adding extensions to the model. This means that today, Machine Learning engineers have a long menu of extensions to choose from when implementing their own systems. In this report, we aim to simplify these choices by first implementing a range of extensions to BERT, and then measuring the performance of our system on every possible combination of the improvements that we implemented.

We identified three components of the system to experiment with. We try two different parameter sets for the $\texttt{BERT}_{BASE}$ model, three different architectures for the post-BERT task-specific layers, and three different loss functions for our system, for a total of 18 different combinations. We provide quantitative results from these experiments, and perform qualitative analysis in service of hypothesizing the reasons behind the results we observe.

We will be performing our analysis on three downstream tasks. These includes: (i) a sentiment analysis task, modeled as a multi-class classification problem, (ii) a similarity measurement task, modeled as a regression problem, and (iii) a paraphrase detection task, modeled as a binary classification problem.

2 Related Work

In some ways, the Natural Language AI revolution was kicked of by Seq2Seq model (Sutskever et al., 2014). This Neural Machine Translation model outperformed specialized systems built by hundreds of engineers and linguists over many years. It could handle variable length input and output sequences. Crucially, it was a simpler system compared to the machine translation systems it replaced.

Another big step came in the form of ELMo (Peters et al., 2018), an RNN based encoder. This work introduced biderectionality in the context of computing word embeddings. This meant that the model embeddings captured more nuanced meanings of words and their contexts. This manifested in the form of the model performing better on downstream tasks. It also served as an inspiration for BERT (Devlin et al., 2018). BERT was groundbreaking because it showed the possibilities of scaling up the transformer architecture, introduced by Vaswani et al. (2017).

BERT performed well out of the box on many downstream tasks, and served as a foundational model for many research papers that came after it. One such significant paper was RoBERTa (Liu et al., 2019), which found that simply training BERT for longer and on more data could lead to significant gains in performance. While BERT and RoBERTa served as reliable foundational models at the time, today much research on foundational models has moved on to decoder-heavy or decoder only models, like the GPT series (Brown et al., 2020) and the Gemini series (Google AI Team, 2024).

However, we believe that ML research often goes in cycles, and encoder-heavy models will likely be back in fashion. Therefore, it's valuable to study extensions relevant to BERT. This includes regularization methods like SMART (Jiang et al., 2020) and uncertainity weighing (Kendall et al., 2017). Empirically, these methods have been shown to improve generalization, but might be considered heuristical in nature. In this report, we also attempted to replicate Task-Adaptive Pretraining, as shown to be effective by Gururangan et al. (2020). The main contribution of this paper is the measurement of combining these methods in different ways.

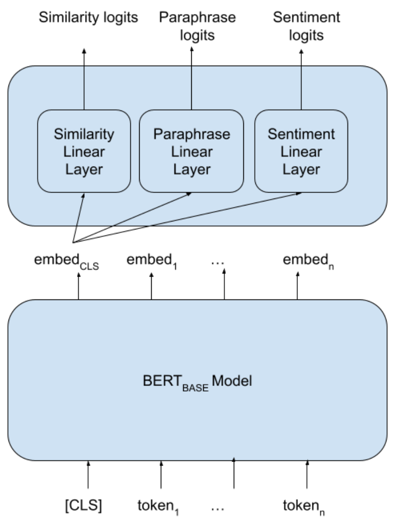
3 Approach

Our approach involves implementing multiple options for the loss function, BERT model, and the task-specific layers, and then running a grid search to find the best combination of options.

3.1 Baseline

We began by completing the implementation of the miniBERT system, including AdamW (Loshchilov and Hutter, 2017), and passing provided sanity tests. We then implemented a system containing a single shared BERT model with per-task Linear layers on top. The input to this linear layer is the final embedding of the `[CLS]` token, and the outputs are logits. The loss function for the baseline was a sum of Cross-Entropy loss for the Paraphrase Detection and Sentiment Analysis tasks, and Mean Squared Error Loss for Similarity Detection task. The baseline is visualized in figure 1.

Figure 1: The Baseline setup, containing a shared BERT model and three task-specific Linear layers.



3.2 Extension: MLP and BERT Non-Shared layers

All our experiments consisted of a shared BERT model with some non-shared layers stacked on top. In addition to the baseline Linear layers, we also experimented with a Multi-Layer Perceptrons (Rosenblatt, 1958) with GELU (Hendrycks and Gimpel, 2016) activation function. We chose GELU because it's seems to work well in practice, and because it is less susceptible to gradient vanishing problem than some other activation functions. The input to the MLP is the embedding for the `[CLS]` token, same as that of the Linear Layer. We also experimented with adding task-specific BERT layers, which took as input all the embeddings from the shared BERT model.

3.3 Extension: Confidence weighing and SMART Loss Functions

In addition to the baseline simple sum of losses, we also tried normalizing the loss functions by hand based on observed values during training. This option was discarded for the final grid search due to underperformance. We experimented with uncertainity modeling as performed by Kendall et al. (2017). This involves modeling the loss $\mathcal{L}$ as $\mathcal{L}(\theta) = \sum_{i=1}^{n} \frac{1}{2\sigma_i^2}\mathcal{L}_i(\theta) + \ln(\sigma_i)$, where $\mathcal{L}_i$ is the loss for the ith task, and each σ_i is a learned parameter. The original authors posit that this allows the model to express uncertainty about certain tasks, hypothetically leading to better learning.

We also implemented SMART regularization (Jiang et al., 2019). In this method, we optimize $min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta)$, where $\mathcal{L}$ is the original loss, λ_s is a tuning parameter and $\mathcal{R}_s(\theta)$ is a measure of difference between the model's output for some input x and another adversarially chosen input $\hat{x}$ ($|x - \hat{x}| < \epsilon$, where ϵ is a tuning paramter). We use the mean squared error as the difference measure for the similarity and paraphrase tasks, and symmetrized KL divergence for the sentiment classification task. We also arbitrarily choose $\hat{x}$ instead of adverserially choosing it for performance reasons. One important detail with our implementation is that we turned off Dropout when using SMART loss, because the stochasticity introduced by Dropout layers interfere with divergence calculations.

3.4 Extension: Task Adaptive Pretraining

Gururangan et al. (2020) showed that performing additional pretraining of BERT on unlabeled task data can lead to an improvement in performance. Therefore, we attempted to perform additional pretraining on the unlabeled training data from the provided Sentiment Treebank (SST), Quora Question Pairs (QQP) and SemEval datasets.

Similar to the original paper, we used a shared BERT model for performing additional pretraining. The additional pretraining was performed using the Masked Language Modeling ("MLM") task used by Devlin et al. (2018). For this task, we mask out 15% of tokens, pass it through BERT, and pass the final embedding of the masked tokens through a linear layer. The final output has the same number of logits as the vocabulary size. These are passed through a softmax layer. The original token is used as the ground truth, and cross entropy loss is used to train the model. Masking is done by replacing the original token with the `[MASK]` token 80% of the time, a random token 10% of the time and making no change 10% of the time.

In order to prevent catastrophic forgetting, we freeze all parameters in BERT except the first embedding layer, and choose the output at the epoch that performs best on the MLM task on the dev set. The training task is visualized in figure 3, and the training and dev loss over 40 epochs is shown in figure 2.

3.5 Extension: Prefixing

We attempted to add a prefix to the input to the BERT model. We added special per-task tokens (`[STS]`, `[SEM]`, and `[PAR]`). We also tried using existing vocabulary terms ("similarity," "paraphrase," and "sentiment"). The motivation for adding a prefix is that it may enable the model to attend to the right tokens based on the target task right in the early layers. However, in practice this failed to produce satisfactory results. Adding fresh special tokens severly hindered performance and using existing vocabulary terms did not improve performance. This approach might be considered reminiscent to prompt tuning as explored by Brown et al. (2020). However, it is **original** in the context of encoder-only models to my knowledge.

Figure 2: Training and dev loss during task adaption. The loss is measured as a sum of cross entropy losses for the three datasets in the batch.

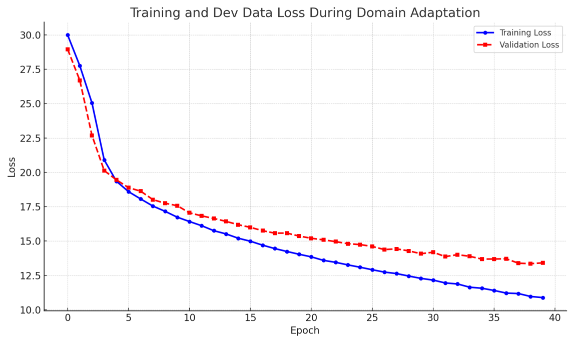
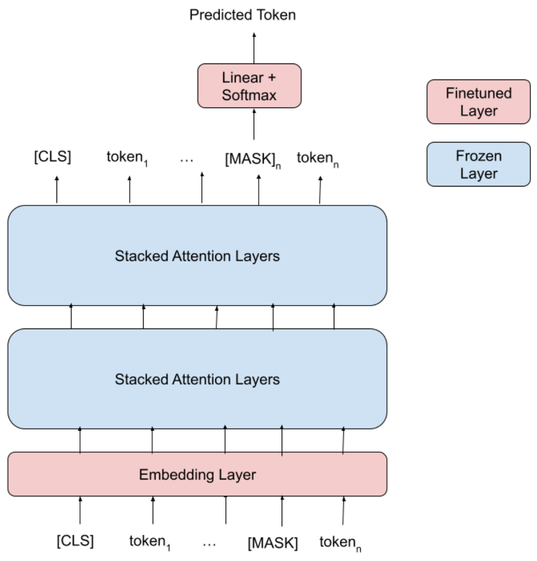


Figure 3: Masked Language Modeling Task. 12% of tokens are replaced with the [MASK] token, 1.5% are replaced with a random token from the vocabulary, and 1.5% are kept as-is.



Since these approaches did not succeed in our experiments, we discarded them for the grid search, in favor of more promising alternatives.

4 Experiments

We compared the various approaches mentioned in the previous section by conducting a grid search. The search space is visualized in figure 4.

4.1 Datasets

The datasets that we are using are enumerated in the table 1, along with the label types and references.

4.2 Evaluation Method

For our evaluation metric, we used Pearson Correlation for the Similarity task and accuracy for the other two tasks. We first scale Pearson Correlation to $[0, 1]$, similar to other two metrics. We then

Figure 4: The search space for grid search. There are 18 possible configurations for our system.

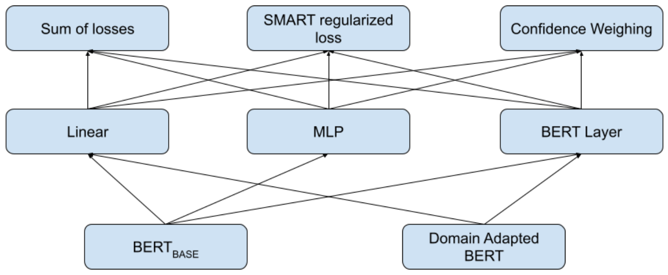


Table 1: Datasets used in this report. Here, "Cl" means Classification and "Reg" means Regression.

Dataset	**Task**	**Type**	**Range**	**Examples**
Stanford Sentiment Treebank (SST-5)	Sentiment Analysis	Cl	1-5	11,853
Quora Question Pairs Dataset (QQP)	Paraphrase Detection	Cl	0-1	202,157
SemEval STS Dataset (STS)	Semantic Text Similarity	Reg	0-5	8,630

take the average of the accuracy scores and the scaled pearson correlation score to arrive on the final "overall" score.

4.3 Experimental details

For each training run, we:

- Used a learning rate of 2×10^{-5}, and a minibatch size of 8, both chosen by hand.
- Concatenated the input tokens for the Paraphrase Detection and Semantic Text Similarity tasks, with a `[SEP]` special token in between and with random ordering.
- Trained with 1068 batches per epoch, and for 15 epochs. This means 16020 training examples for each task, including double counting in some cases.
- Trained on a VM (2 vCPUs, NVidea V100 GPU), taking about 15 minutes per experiment.

We finetuned BERT on the three unlabeled task datasets for 20 epochs (1068 examples per task in each epoch), and chose the model weights from the epoch with the lowest loss on the dev dataset.

For the Multi-Layer Perceptrons (MLP), we used a 2 hidden layers (chosen arbitrarily), with a hidden size of 768 (same as the hidden size of BERT). For non-shared BERT layers, we used a single additional BERT layer per task, and fed the output embedding of the `[CLS]` token for this additional layer to an MLP.

For SMART regularization, we used many of the same parameters used by Jiang et al. (2019), with $\lambda_s = 5$, $S = 1$ and $T_{\bar{x}} = 1$.

4.4 Results

The results of the grid search are shown in table 2. The task-adapted model using sum of losses as the loss function and an Multi Layer Perceptron on top performs the best based on the overall score. In table 3, we show the average score by design choice. MLPs consistently outperform the other two non-shared layer designs. The difference between loss types seems to be quite small, and the task adapted BERT seems to consistently outperform the base BERT.

Table 2: Grid Search Results

Name	Overall	Sentiment	Para	Similarity
Sum of Losses / Task Adapted / MLP	**0.769**	**0.527**	0.855	0.851
Sum of Losses / Task Adapted / Linear	0.758	0.495	0.856	0.848
Sum of Losses / Task Adapted / BERT	0.763	0.513	0.854	0.843
Sum of Losses / Base BERT / MLP	0.763	0.494	0.865	**0.861**
Sum of Losses / Base BERT / Linear	0.761	0.503	0.857	0.848
Sum of Losses / Base BERT / BERT	0.757	0.492	0.852	0.852
SMART / Task Adapted / MLP	0.767	0.520	0.859	0.840
SMART / Task Adapted / Linear	0.759	0.487	**0.867**	0.845
SMART / Task Adapted / BERT	0.762	0.501	0.862	0.844
SMART / Base BERT / MLP	0.761	0.500	0.863	0.839
SMART / Base BERT / Linear	0.760	0.498	0.863	0.839
SMART / Base BERT / BERT	0.761	0.501	0.863	0.838
Confidence Weighted / Task Adapted / MLP	0.762	0.509	0.851	0.851
Confidence Weighted / Task Adapted / Linear	0.759	0.497	0.853	0.853
Confidence Weighted / Task Adapted / BERT	0.764	0.513	0.851	0.854
Confidence Weighted / Base BERT / MLP	0.761	0.508	0.854	0.842
Confidence Weighted / Base BERT / Linear	0.759	0.493	0.860	0.845
Confidence Weighted / Base BERT / BERT	0.756	0.487	0.853	0.853

Table 3: Average Overall Scores by system design choice

Category	Average Overall Score
Loss Type	
Confidence Weighted	0.760
SMART	0.762
Sum of Losses	0.762
Adaptation	
Base BERT	0.760
Task Adapted	0.763
Non Shared Layers	
BERT	0.761
Linear	0.759
MLP	0.764

5 Analysis

5.1 Paraphrase Detection

Here are three **false positives** from our best performing model:

1. Why do I fall asleep when sitting? **v.s.** Why do I keep falling asleep?
2. What are some really cool working science models that I can make? **v.s.** I am in 10th grade. I need to make a working model for a science fair. Any suggestions?
3. What are the best and profitable ways of making money? **v.s.** What are your best ways to save money?

All these questions are quite similar, differing only in small nuances. Indeed, one may argue that the second and the third examples are mislabeled in the given dataset. Therefore, we are not surprised at these misclassifications. Next, we show some **false negatives**.

1. What makes one angry? **v.s.** What is the one thing that makes you most angry?
2. What is the physical meaning of operating voltage? **v.s.** What is the physical meaning of operating voltage in diodes?
3. How do you paint your rims black? **v.s.** How do you paint chrome rims black?

We argue that all these pairs are mislabeled in the dataset. In the first pair, there is an important difference between what make any random person angry v.s. what makes the reader angry. In the second pair, operating voltages might have different connotations in diodes v.s. other appliances. Finally there could be differences between how chrome rims are painted v.s. how other rims are painted.

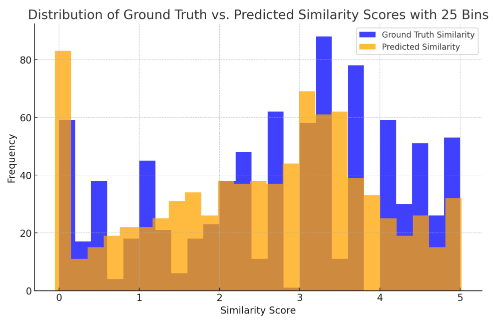
5.2 Similarity Detection

tabularx

Table 4: Analysis of Sentence Similarity Predictions vs. Ground Truth

Sentence 1	**Sentence 2**	**Ground Truth Similarity**	**Predicted Similarity**
Work into it slowly.	It seems to work.	0.0	3.507442
In these days of googling, it's sloppy to not check first.	I agree with Kate Sherwood, you should be able to check first with Google; it's sloppy not to do so.	3.2	0.047869
The villains I absolutely hate are selfish and egotistical.	A villain you want to take down is, at his/her core, selfish and egotistical.	3.0	0.126704

Table 4 has the top 3 sentences where the prediction is furthest off from the ground truth. The model might have made the wrong prediction for the first one because of confusion about the meaning of "work." For the 2nd two pairs, the model predicts similarity close to zero. Graphing the distribution of precdictions v.s. ground truth in figure 5.2, we see that the model is generally predicting zero similarity two often. Perhaps when it is not confident about the actual score it just predicts zero similarity.



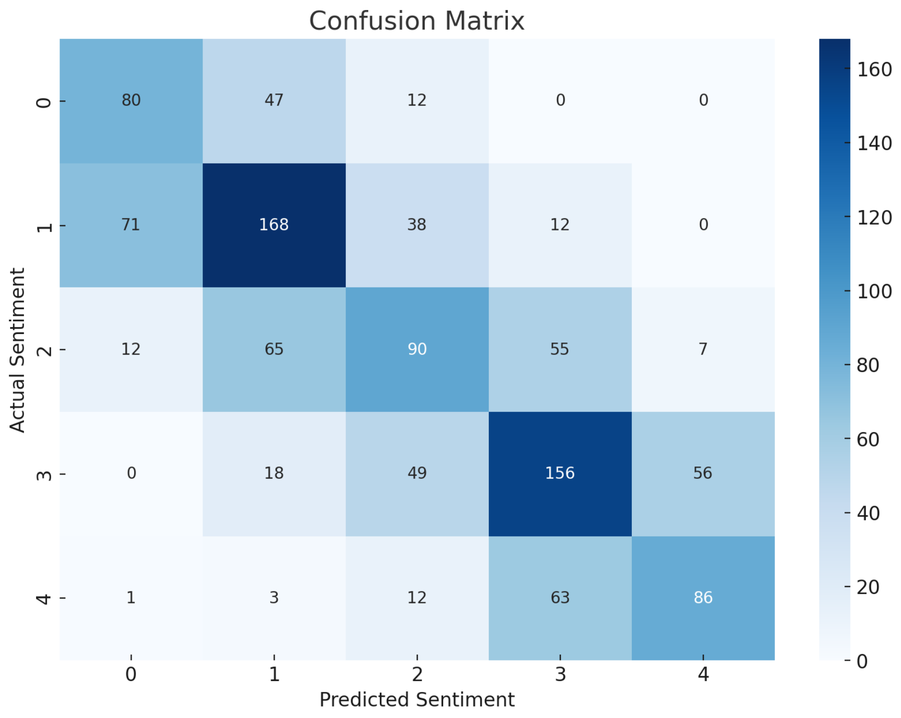
5.3 Sentiment Analysis

Top 3 sentences for which our predictions were furthest from the ground truth are:

1. If Steven Soderbergh 's ' Solaris ' is a failure, it 's a noble, honorable failure. (Actual "Positive" Predicted: "Negative")
2. No screen fantasy-adventure in recent memory has the showmanship of Peter Jackson 's King Kong. (Actual "Positive" Predicted: "Somewhat Negative")
3. This flick is about as cool and crowd-pleasing as they come. (Actual "Positive" v.s. Predicted "Somewhat Negative")

These are all examples of positive sentiment reviews but the model could be confused because the reviews is comparing the movie to other movies of perhaps lower quality. In the case of the third sentence, the model might be confused because of the term "crowd-pleasing" which can have a negative connotation.

In Figure 5.3 we can see a confusion matrix from our predictions. An encouraging thing to note is that even in cases where the model misjudges the sentiment, the predicted class is usually near the actual class.



6 Conclusion

In this work, we experimented with various ways of building a system to perform three downstream tasks and showed evidence for certain design decisions might be better than others. We presented the raw data from our experiments, which may help others make similar decisions. One drawback of our approach might be that all experiments were done once due to limited compute power, which means that we don't have error bars available for researchers.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz

Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Google AI Team. 2024. Gemini: Next generation model. Technical report, Google.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

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. *CoRR*, abs/1911.03437.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CoRR*, abs/1705.07115.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, volume 30.