BERT-based Multi-task Learning

Stanford CS224N Default Project

Sahar Kazemzadeh
Department of Computer Science
Stanford University
saharkaz@stanford.edu

Abstract

While large language models like BERT[1] have greatly advanced language modeling, effective multi-task learning with generalizable sentence embeddings continues to be an important area of research. We implemented a BERT Transformers [2] model that predicts sentiment, detects paraphrases and predicts semantic textual similarity (STS). On the test sets our extended-BERT model achieves 86.5% accuracy on paraphrase detection, 50% accuracy on sentiment analysis, and an STS correlation coefficient of 76.1%, for an average metric of 70.9%. We found that for the tasks with small datasets, heavy regularization via SMART [3] and leveraging additional in-domain datasets via SimCSE contrastive loss [4] were paramount to improving perfomance. We further found that head architecture is very important, and that by prompting the model we were able to expedite learning. Our implementations of our extensions resulted in an approximately 12% improvement over our base-BERT model.

1 Introduction

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based model that generates contextual word representations that capture context from both left and right directions[1] . With its release in 2018, BERT took a large leap forward for large language models. The release of the model enabled its use for fine-tuning on downstream tasks with fewer computational resources and smaller datasets. However, fine-tuning effectively on multiple tasks continues to be an open research problem.

We created a minimalist implementation of the BERT model (minBERT), including multi-head self-attention and Transformer layers. Using pre-trained weights from *bert-base-uncased* [1], we created a multi-task baseline model that predicts sentiment, classifies paraphrases, and gauges semantic textual similarity (STS). We also implemented the Adam [5] optimizer with weight decay [6]. We applied extensions to this model in order to obtain robust and generalizable sentence embeddings that allow for strong multi-task performance.

2 Related Work

The General Language Understanding Evaluation (GLUE) [7] benchmark is a collection of natural language understanding (NLU) tasks designed to evaluate and compare the performance of various machine learning models on a wide range of NLU tasks. Introduced by Wang et al [7], the GLUE benchmark aims to promote research in developing general-purpose language understanding models that can perform well across multiple tasks. We selected 3 of these tasks to focus on. Sentiment classification involves determining the polarity of a given text. Paraphrase detection aims to identify whether two given texts convey the same meaning using different wording. The goal of semantic textual similarity (STS) prediction is to measure the degree of semantic similarity between two pieces of text.

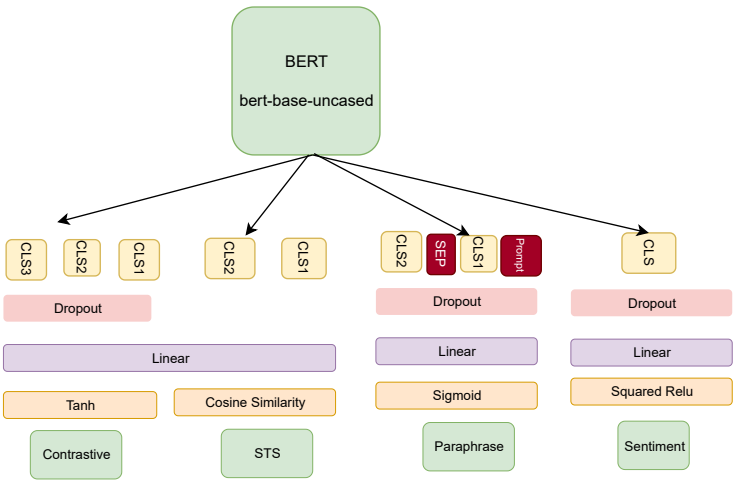


Figure 1: Architecture diagram of our network. BERT (bert-base-uncased) was used as the basis for fine-tuning. The CLS token produced by BERT for each input sentence was provided to each of the sentiment classification, paraphrase detection, regular STS, and STS contrastive learning tasks.

One of the challenges with multi-task training is that aggressive weight updates can cause over-fitting; an aggressive weight update to accommodate one task can come at the expense of the other tasks. To reduce the likelihood of training in a zero-sum game, Jiang [3] et. al propose **SM**oothness inducing **A**dversarial **R**egularization and B**R**egman PRoximal poin**T** OpTimization (SMART) regularization.

Moreover, contrastive learning is a technique that aims to learn effective representations by pulling semantically close neighbors together and pushing apart non-neighbors [8]. This mode of learning, which mimics the way humans learn, has shown promising results in deep learning for both leveraging large, unlabeled datasets and for providing an alternative and sometimes better mode of learning for labeled datasets. Meng et al [9] demonstrated that a contrastive objective can be extremely effective when coupled with pre-trained language models such as BERT [1]. They presented SimCSE, a simple contrastive sentence embedding framework that can produce superior sentence embeddings from either unlabeled or labeled data.

Furthermore, the authors showed that SimCSE can alleviate the anisotropy problem that is common in language modeling. Anisotropy is when learned embeddings occupy a narrow cone in the vector space, which severely limits their expressiveness. SimCSE regularizes pre-trained embeddings' anisotropic space to be more uniform and better aligns positive pairs when supervised signals are available. In theory this should help improve multi-task learning in which shared embeddings are used. We hypothesized that using SimCSE to leverage additional in-domain datasets can help create generalizable sentence embeddings.

3 Approach

Sentences were used as input for three different downstream tasks of sentiment analysis, paraphrase detection and semantic textual similarity. The output was a categorical label from 1 (negative) to 5 (positive) for the sentiment task, a binary label for the paraphrase task, and a continuous score from 0 to 5 indicating degree of semantic textual similarity for the STS task.

3.1 Baseline Model

Sentences were passed through BERT's tokenizer and resultant sentence embeddings were fed through our network. Our baseline model's architecture was similar to 1. It had one head for each task and an MLP comprised of a dropout layer, linear layer and activation function for each task. The CLS token $X \in \mathbb{R}^{(B \times 784)}$ for each sentence produced by BERT was passed into each of the downstream tasks. This model served as the basis for extensions, some of which we ultimately adopted to result in the architecture shown in Figure 1, which is discussed in Section 3.2.

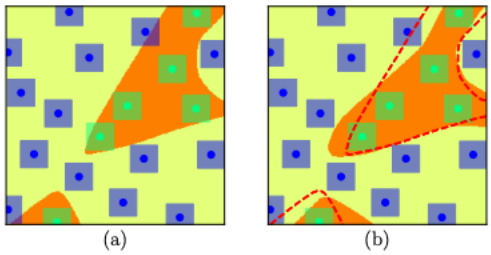


Figure 2: Decision boundaries learned without (a) and with (b) SMART regularization. The red dotted line in (b) represents the decision boundary in (a). As can be seen, the output f in (b) does not change much within the neighborhood of training data points. Image borrowed from [3].

3.2 Head Architecture and Model Prompting

Different head architectures were explored for each of these tasks. The sentiment classification head had a dropout layer followed by a linear layer and squared relu activation function. Its output was of shape $Nx5$, where N is the batch size.

The model was prompted for the paraphrase task. The tokenized form of the prompt "Are these two sentences paraphrases ?" was concatenated with the tokenized form of the first sentence, the SEP token and the tokenized form of the second sentence along the sequence length dimension. The combined input was fed through BERT and the resultant CLS token was passed through dropout, linear and sigmoid layers.

The architecture for the STS task is described in section 3.6.

3.3 SMART Regularization

One extension that proved valuable was our implementation of smoothness-inducing regularization component of SMART regularization, which was employed for sentiment classification; this task in particular initially suffered from a high degree of overfitting due to the small size of its initial datasets. Its loss is characterized in 1.

$$\min_\theta F(\theta) = L(\theta) + \lambda_s R_s(\theta) \tag{1}$$

where $L(\theta)$ is the standard loss function and

$$R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\substack{k \\ \|e_i^x - x_i\|_p \leq \epsilon}} s(f(e_i^x;\theta), f(x_i;\theta)) \tag{2}$$

$$s(P,Q) = D_{\text{KL}}(P\|Q) + D_{\text{KL}}(Q\|P) \tag{3}$$

At train time, the input sentence embeddings were perturbed α times such that the p-norm of the difference between the input embeddings and perturbed embeddings was $<= \epsilon$. Each of these perturbed sentence embeddings was run through the model to produce $f(e_i^x;\theta)$, and the maximum symmetric KL-divergence between the original output's CLS token and the α $f(e_i^x;\theta)$ CLS tokens was added to the loss.

The constraint enforces that the perturbation should not exceed a certain distance (measured by the p-norm) from the original input. Its effect is summarized in Figure 2

3.4 Task Weighting

Joint optimization of multiple tasks is challenging due to unbalanced dataset sizes and variations in task difficulties. We monitored training loss and adapted task weights throughout training according to loss magnitude in order to prioritize harder tasks over easier tasks.

3.5 Further In-domain Training

As described in Table 1, there was severe dataset size imbalance with the original datasets of SST, CFIMDB, Quora and SemEval. The Quora dataset for paraphrase detection is 406x the size of the SemEval STS dataset and 28x the size of the Stanford and CFIMDB sentiment datasets combined. As a result initial experiments showed much weaker results for the sentiment and STS tasks. To combat this, we made use of the Stanford Natural Language Inference (NLI) [10] dataset for the STS task and the Twitter Sentiment Analysis Training Corpus [11].

3.6 Contrastive Learning to Enable Further In-Domain Training

Another extension for which we wrote an efficient implementation was the use of both unsupervised and supervised SimCSE. In unsupervised SimCSE, the authors take a collection of sentences $\{x_i\}_{i=1}^m$ and use identical positive pairs $x_i^+ = x_i$. The key ingredient is the use of independently sampled dropout masks for x_i^+ and x_i. They denote $h_i^z = f_\theta(x_i, z)$, where z is a random mask for dropout. They feed the same input to the encoder twice and get two embeddings with different dropout masks z, z'. The training objective for a mini-batch of N sentences is

$$l_i = -log \frac{e^{sim(h_i^{z_i}, h_i^{z_i'})/\tau}}{\sum_{j=1}^N e^{sim(h_i^{z_i}, h_i^{z_i'})/\tau}} \tag{4}$$

where sim is the cosine similarity and z is the standard dropout mask in Transformers. Dropout serves as a minmal form of data augmentation. Interestingly, the authors note that when they use no dropout ($p = 0$) and "fixed 0.1" dropout (i.e. the same dropout masks for the pair), the resulting embeddings for the pair are exactly the same and there is a dramatic performance degradation.

In supervised SimCSE, the authors propose the loss function

$$l_i = -log \frac{e^{\text{sim}(h_i, h_i^+)/\tau}}{\sum_{j=1}^N e^{sim(h_i, h_j^+)/\tau} + e^{\text{sim}(h_i, h_i^-)/\tau}} \tag{5}$$

where sim is cosine similarity, h_i and h_j^+ are the embeddings for a positive pair and h_i and h_j are embeddings for a negative pair. Per Table 1, the SemEval train set had only 6k examples. To combat overfitting we downloaded the Natural Language Inference (NLI) dataset, which had 275,600 triplets: two sentences per triplet had similar meaning, and a third sentence had an opposing meaning and was deemed a hard negative. Because this dataset did not have labels with continuous scores between 0 and 5, we leveraged it using SimCSE supervised contrastive loss and had this task share a linear layer with the STS task trained on SemEval.

The SemEval STS head separately ran each of the two input sentences' CLS token through the linear layer it shared with the NLI task, performed cosine similarity on the tokens, and clamped the output between 0 and 1 before scaling it by a multiple of 5 as the output to feed to MSE loss. This architecture is reflected in Figure 1.

4 Experiments

4.1 Data

For sentiment analysis, the Stanford Sentiment Treebank (SST) dataset [12], CFIMDB [13] and Twitter Sentiment Analysis Training Corpus (TSATC) [11] datasets were used. SST was parsed with the Stanford parser and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive. The CFIMDB dataset consists of 2,434 highly polar movie reviews. Each movie review has a binary label of negative or positive, and many of these reviews are multi-sentence. TSATSC is an English language corpus of Tweets with labels of negative, neutral and positive. For consistency with the output of categorical labels 0, 1, 2, 3, and 4 for our sentiment task, we mapped negative to 0, neutral to 2 and positive to 4.

The Quora dataset was used for paraphrase detection. The subset that was used consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another.

The SemEval STS Benchmark dataset [14] was used for semantic textual similarity classification. It consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning).

The NLI dataset came in triplets: two sentences with similar semantic similarity (considered an entailment pair) and a sentence with an opposing meaning (considered a hard negative). This made it a great fit for contrastive learning, so we trained it with a supervised SimCSE objective. This augmented performance on our much smaller SemEval STS dataset.

The dataset sizes and train, dev and test split sizes are listed in Table 1. All splits for CS224N datasets are consistent with the splits designated for the course. There wasn't any discernible distribution shift between the train and dev sets. For the datasets we introduced we created random splits for train and dev splits.

Dataset	Task	Total	Train	Dev	Test
Stanford	Sentiment	11,855	8,544	1,101	2,210
CFIMDB	Sentiment	2,434	1,701	245	488
Quora	Paraphrase	400,000	141,506	20,215	40,431
SemEval	STS	8,628	6,041	864	1,726
NLI	STS	275,600	261,820	13,780	N/A
Twitter	Sentiment	27,481	24,731	2,750	N/A

Table 1: Our datasets and their split sizes. We added the NLI dataset for STS and the Twitter TSATC dataset for sentiment classification.

4.2 Evaluation method

For sentiment classification and paraphrase detection we monitored accuracy because these tasks have categorical labels. For the STS task we calculated the Pearson correlation between the true similarity and predicted similarity values. We compared results with our own baseline model. Our main metric that we maximized during training was the average performance of these three tasks on the SST, Quora, and SemEval dev datasets.

4.3 Experimental details

Experiments used a Tesla p100 GPU with batch size of 64. We experimented with different learning rates and decay schedules and ultimately used a constant learning rate of 3e-5, dropout rate of 20% and weight decay of 1e-2. Training was done in a round robin fashion between the different tasks. Cross-entropy loss was used for all tasks other than the regression task of STS, for which MSE loss [15] was used.

4.4 Results

Table 3 shows the dev set results of ablation experiments that were completed with the same seed to evaluate the efficacy of our extensions before we adopted them. Each of these experiments was run to 1500 steps due to compute constraints. We found that for the paraphrase task, changing the architecture of the head to procure a single CLS token for a combined input of the prompt and the two tokenized sentences separated by the SEP token expedited learning for that task. We also found that SMART regularization on the sentiment task helped alleviate overfitting considerably, as did leveraging the NLI dataset via SimCSE learning for the STS task. Task weighting helped overall performance. Sometimes one extension improved results on one task at the expense of the others; this underscores an inherent challenge in multi-task learning.

In "pretrain" mode BERT's weights were frozen and only the head weights were updated. In "finetune" mode all layers were updated. We found that it was nearly impossible to significantly improve performance on the STS task in "pretrain" mode, which is consistent with what we expected. Keeping BERT's weights frozen gives relatively little ability to learn and does not allow for updating the sentence embeddings.

Dev Set Performance Ablation Experiments				
	Average	Sentiment	Paraphrase	STS
Baseline	0.616	0.477	0.747	0.623
Prompting (para)	0.623	0.463	**0.786**	0.619
SMART reg	0.616	0.489	0.749	0.610
Task weighting	0.620	**0.482**	0.752	0.625
SimCSE	0.630	0.453	0.753	**0.683**
Add TSATC sentiment dataset	0.617	0.480	0.741	0.631

Figure 3: Ablation experiments used to initially gauge whether an experiment is useful. Run with batch size 64, 1500 steps and the same seeds.

Regularization from SMART and weight decay proved critical to improving dev set performance for the Sentiment and STS tasks. This is consistent with our expectations because prior to the addition of the NLI and TSATCs datasets these two tasks had much smaller datasets; without regularization the model was prone to overfitting to the training data.

Using recommendations from [3] and our own fine-tuning we found that $\alpha = 7$, $\epsilon = 5e - 3$, and $\lambda = 50$, and $p = \inf$ works best. With our implementation of SMART regularization, the performance drop between the train and dev sets for STS went from 45% to 26%, which is a 42% improvement.

Surprisingly, the large TSATC sentiment dataset did not help improve performance on SST. We suspect three reasons for this: 1) it's possible that Tweets are too dissimilar to movie reviews in SST, 2) the TSATC dataset may need further cleaning, and 3) TSATC only had labels for negative, neutral and polar rather than 5 more granular labels of polarity. We mapped negative, neutral and polar to classes 0, 2 and 4. On the other hand, leveraging the large NLI STS dataset via supervised contrastive learning with $\tau = 0.05$ did prove important for improving performance on SemEval.

Ultimately our baseline model achieved a mean metric of 64.0% on the dev set, and our extended BERT model achieved a mean metric of 71.7% on the dev set and 70.9% on the test set. Our extensions thus resulted in a 12% improvement on the dev set.

	Average	Sentiment	Paraphrase	STS
Base BERT (baseline)	0.640	0.481	0.781	0.656
Extended BERT (dev)	0.717	0.500	0.865	0.785
Extended BERT (test)	**0.709**	0.500	0.865	0.761

Table 2: Performance of our baseline BERT and extended BERT. Our extensions thus resulted in a 12% improvement on the dev set.

5 Analysis

Of the three tasks, sentiment accuracy was the lowest. This is probably due to a number of factors, including subjectivity in the labels, as bolstered by the fact that a small degree of label smoothing helped that task. The misclassifications were not egregiously wrong. Per figure 5(b), 80% of the misclassifications were off by 1 category, 18% were off by 2 categories and 2% were off by 3 categories. As shown in Figure 4, the model underpredicted the extreme polarity classes 0 and 4.

The paraphrase task had the highest accuracy. This is likely due to the fact that it had the largest training dataset and that it required a 2-class categorization rather than 5-class or continuous score

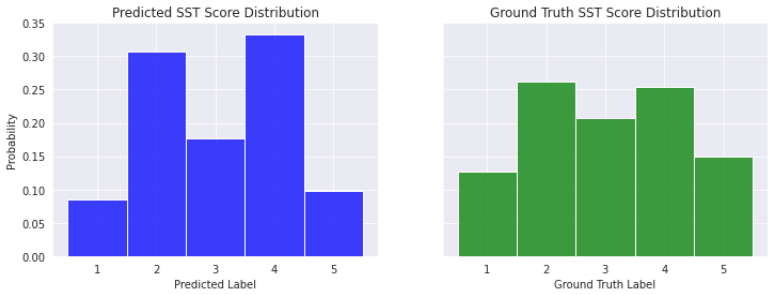
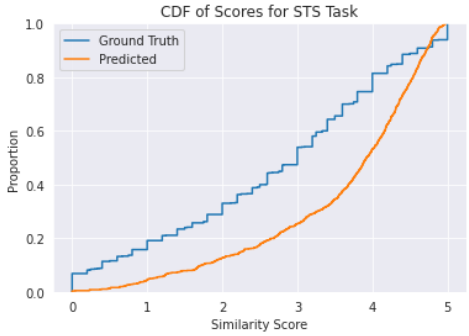
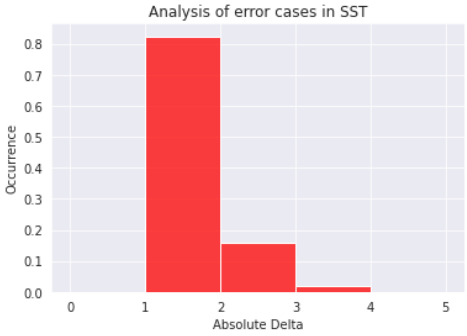


Figure 4: Predicted and ground truth score distribution on the dev set for the sentiment task. The model underpredicted the extreme polarity classes 0 and 4.



(a) Cumulative distribution function of predicted and ground truth scores on the SemEval dev set for SST.



(b) Absolute delta between ground truth and predicted sentiment labels for error cases in SST. 80% of error cases were off by 1 category.

output. Error analysis showed that the model had an approximately equal rate of false positives as false negatives. Some of the "error" cases actually turned out to be cases of mislabeling. The pair "how do compasses work? and "how does a compass work?" was erroneously labeled as not a paraphrase, while "It healthy to eat a whole avocado every day?" and "What is a cheap healthy diet i can keep the same and eat every day?" were erroneously labeled as paraphrases.

Some false positives were understandably difficult. One such example is "What are the best books to read by Indian authors?" and "Wat are good book of poetry by an Indian author?"; they would have been paraphrases if not for the specification of poetry in the latter sentence. Some false negatives indicate that the model has trouble with very long statements. One such example is the pair "how far back in time could we go in the uk before we would start to be unable to understand the english of the day?" and "how far back would we have to go for a current english native speaker to have trouble understanding an english speaker from the past?".

Sentiment + Paraphrase	0.77
Sentiment + STS	0.27
Paraphrase + STS	0.08

Table 3: Correlation coefficients between dev performance on the tasks.

Per Figure 5(a), the errors in STS tended to come from systematically underestimating similarity except at the very high end of the similarity range. This suggests that in future work this task might benefit from a different head structure. A lot of the errors were understandable. For the pair "a woman opens a window." and "woman is looking out a window" the label was 2.0 but the prediction was 4.6 and for the pair "the note 's must-reads for friday, december 6, 2013" and "the note 's must-reads for friday, july 12, 2013" the label was 1.8 but the score was 4.9. It appears that the model has difficulty when a pair of statements has a lot of words in common but has subtle nuance that changes their meaning.

We computed the correlation coefficients between the dev set performances of the different tasks. Per Table 3, the the sentiment and paraphrase tasks had a high correlation coefficient of 0.77. The sentiment and STS tasks were weakly correlated with a coefficient of 0.27, and the paraphrase and STS tasks had a very low correlation of 0.08. This reveals the inherent challenge in multi-task learning. Unfortunately, in this context a rising tide does not float all boats by the same amount.

6 Conclusion

In conclusion, we implemented minBERT and extensions. We found that for the tasks with small datasets, aggressive regularization via SMART and leveraging additional in-domain datasets via SimCSE contrastive learning was critical. We further found that head architecture is very important. For the paraphrase task, adding a prompt and SEP token to the input expedited learning. On the test leaderboard our model achieves 86.5% accuracy on paraphrase detection, 50% accuracy on sentiment analysis, and an STS correlation coefficient of 76.1%, for an average metric of 70.9%. We found that improving performance on one task correlated positively but sometimes weakly with performance on the other tasks. Good areas for future work would be to test if model prompting can help with the sentiment and STS tasks, reduce noise in the labels, and to delve deeper into the literature on effective multi-task learning.

References

[1] Bert: Pre-training of deep bidirectional transformers for language understanding. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5900–5909, 2017.

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

[4] Xingcheng Yao Tianyu Gao and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *Empirical Methods in Natural Language Processing*, 2021.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015.

[6] Diederik P. Kingma and Jimmy Ba. Adamw: A method for weight decay regularization. *arXiv preprint arXiv:1907.06913*, 2019.

[7] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[8] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[9] Payal Bajaj Saurabh Tiwary Paul Bennett Jiawei Han Yu Meng, Chenyan Xiong and Xia Song. Coco-lm: Correcting and contrasting text sequences for language model pretraining. *arxiv preprint arXiv:2102.08473.*, 2021.

[10] Samuel Bowman, Li Li, and Kristina Toutanova. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05807*, 2015.

[11] Ibrahim Naji. TSATC: Twitter Sentiment Analysis Training Corpus. In *thinknook*, 2012.

[12] 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*.

[13] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[14] Mona Diab Aitor Gonzalez-Agirre Eneko Agirre, Daniel Cer and Weiwei Guo. sem 2013 shared task: Semantic textual similarity. *Second joint conference on lexical and computational semantics (* SEM) volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

A Appendix (optional)

If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the 6-8 page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc., that you couldn't fit into the main paper. However, your grader *does not* have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.