BERT Goes to School: Improving BERT Embeddings Through Curriculum-Based Contrastive Learning and Synonym-Based Data Augmentation

Arnav Gangal
Department of Statistics
Stanford University
`agangal@stanford.edu`

Martin Pollack
Department of Statistics
Stanford University
`pollackm@stanford.edu`

Russell Tran
Department of Computer Science
Stanford University
`tranrl@stanford.edu`

Abstract

In this project we present an implementation of BERT to simultaneously perform sentiment analysis, paraphrase detection, and semantic textual similarity tasks. In particular, we focus on implementing a method to further pre-train BERT using curriculum-based contrastive learning using labelled triplets, and integrate these embeddings into the model's downstream multitask training loop. This model training method was investigated in combination with data augmentation to improve the model's robustness to data variations. However, our findings indicate that pre-training embeddings with curriculum-based contrastive learning as well as data augmentation can actually hurt performance on downstream tasks. Our best overall performing models was still highly successful and achieved accuracies of 0.540 and 0.900 on the test sets of the sentiment analysis and paraphrase detection classifications respectively, and a Pearson's Correlation of 0.844 on the semantic textual similarity test set. Further work in this area includes more complex data augmentation, perhaps by introducing new task-specific datasets.

1 Key Information

- Team contributions: Martin implemented all of minBERT, wrote the code for the triplet ranking system and the contrastive learning loss, and performed ablation studies. Arnav implemented the baseline multitask training framework, the contrastive learning/curriculum learning training loop, the multitask training loop using fine-tuned embeddings, and precision/recall/F1 score evaluation. Russell implemented data augmentation (SGD synonym replacement), conducted the Select Example Analysis, and helped team members run ablation experiments.
- Default Final Project
- Mentor: Johnny Chang
- Late days: 2 late days were used (2x3=6) and shared by Martin (6) to Arnav, Russell, and Martin.

2 Introduction

In the field of Natural Language Processing, significant progress has been driven by advancements in transformer-based models, such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019). Common use-cases involve taking the embeddings produced by a model's pre-trained weights and fine-tuning them for particular downstream tasks. The pre-training is usually done with extraordinary amounts of data and with lots of computation, but it allows the model to gain deep understanding of the target language semantics. Then fine-tuning can be done relatively cheaply and quickly, making the model flexible to do lots of different interesting tasks. Research has shown that the details of the fine-tuning methods and the datasets on which they are performed can significantly affect the performance of the model. In this project, we present an approach to improve the performance of a pre-trained BERT model in performing the tasks of sentiment classification (SC), paraphrase detection (PD), and semantic textual similarity (STS) simultaneously.

Our main extension is curriculum-based contrastive learning (CBCL) as a method to further pre-train and enhance BERT embeddings. Contrastive learning uses a loss function that makes similar embeddings closer and dissimilar ones farther apart. CBCL, suggested by Dehghan and Amasyali (2023), aims to imrpove upon this by systematically introducing training samples of increasing difficulty, allowing the model to learn from easier to harder examples, refining its understanding of semantic relationships. Post pre-training, we use a multitask classification framework, rotating through SC, PD, and STS tasks, each with its own classification head. This strategy aims to create more robust and nuanced embeddings, improving performance across all tasks. CBCL is a new extension introduced in 2023, with limited research and hyperparameter tuning. Thus, we experimented with pacing functions, temperature hyperparameters, and techniques for scoring sample difficulties. Additionally, we investigated using synonym replacement for data augmentation, a suggested improvement by CBCL's creators over their use of only using dropout as noise.

Our final model pipeline consists of two major sections: further pre-training using CBCL and synonym replacement as well as multitask fine-tuning. Our results indicate that the multitask fine-tuning architecture is much more important for maximizing performance on the downstream tasks. This includes the task heads, the losses used, and whether training is done with a sequential or round-robin approach. Contrastive learning and CBCL actually can detract from performance, perhaps since further pre-training on an objective not related to the tasks can take away important information. In addition to our main results, we classify common errors our model makes, do a nuanced error analysis, and perform ablation studies.

3 Related Works

3.1 Contrastive Learning

Our project builds on the contrastive learning approach for language models proposed by Gao et al. (2021), which optimizes a contrastive loss function to compare sentence embeddings in unsupervised or supervised settings. Unsupervised training processes the same sentence twice with different dropout rates to maximize embedding similarity, using dropout as data augmentation. Supervised training uses annotated pairs from datasets like Stanford NLI (Bowman et al., 2015), with "entailment" pairs as positives and "contradiction" pairs as hard negatives. This method aims to align positive pair embeddings more closely, potentially improving performance on downstream NLP tasks.

3.2 Curriculum Learning

Existing contrastive learning methods for fine-tuning language models do not distinguish between easy and hard negatives during training. Curriculum learning, formalized by Bengio et al. (2009), suggests that training with ordered examples improves the convergence speed and quality of minima found by deep learning models, theoretically improving their generalization to unseen data.

Dehghan and Amasyali (2023) integrate curriculum training with contrastive learning in "SelfCCL," a method for fine-tuning BERT. This approach sequentially orders training examples from simple to complex during training, mimicking human learning. The model's embeddings determine the difficulty of training examples, using triplets from Bowman et al. (2015) and Williams et al. (2018). The authors show improved performance on STS and sentiment evaluation tasks when using BERT and SentenceBERT, but do not explore multitask learning paradigms or tune their curriculum-related hyperparameters. Our project represents a meaningful extension of their work — not only do we aim to replicate their results on our new downstream tasks, but also we strive to investigate the impact of tuning difficulty labeling and data ordering in a multitask objective.

3.3 Synonym-based data augmentation

In previous work on contrastive as well as curriculum learning, the idea of data augmentation has been mentioned frequently. The seminal paper on contrastive learning, Gao et al. (2021), uses many forms of data augmentation including dropout, crop, and word deletion or replacement. In their future works section on CBCL, Dehghan and Amasyali (2023) report that a priority is adding a form of curriculum data augmentation where one "gradually [increases] the noise in the data to generate new data", and they cite synonym replacement as a prime example of this. They speculate that this will make the model more flexible and able to distinguish semantic connections between similar phrases and sentences more clearly. Since we are implementing CBCL in our model, we found it appropriate to test the hypothesis of these authors.

Synonym-based data augmentation has been applied to many different problems in NLP. For example, Tashu and Horváth (2022) found that the technique can dramatically improve a model's ability to perform automatic essay scoring as well as protect from and anticipate "adversarial attacks" in the form of plagiarism. Abdollahi et al. (2021) shows that it can aid in the classification of and extraction of meaning from medical discharge notes. These notes are usually highly unstructured and contain various jargon and phrases, but data augmentation allowed the model to avoid these pitfalls better. Lastly, the technique was also used to improve the performance of fake news detection in Riza Rizky and Suyanto (2021).

4 Approach

Our approach can be divided into four key stages: establishing a baseline BERT model, implementing and iteratively improving a multitask training framework, implementing curriculum-based contrastive learning to fine-tune embeddings, and augmenting the training data using synonym replacement.

4.1 Baseline BERT

Our work builds directly on top of the original BERT model, detailed in Devlin et al. (2019) and the default handout at CS 224N Staff (2024). A full description of the model's architecture can be found in Devlin et al. (2019). As a high-level summary, the baseline BERT architecture consists of an embedding layer, followed by 12 multi-headed transformer blocks. We have implemented the multi-head attention mechanism, the classifier pipeline, and the Adam optimizer as per the default project handout (CS 224N Staff (2024)). In addition, we have implemented a multitask classification pipeline that features classification heads for all three tasks and trains based on their combined loss. Using this as a baseline, we added first only contrastive learning and then curriculum-based contrastive learning to evaluate how these methods improved our three task scores. In addition, data augmentation methods were added to the baseline as well as the CBCL model to test its effectiveness. All of these extensions were coded by us.

4.2 Multitask Framework

To establish baseline performance levels on the simultaneous tasks of SC, PD and STS, we chose to split our BERT model into three separate task heads. The architecture of each of the heads is summarized below.

4.2.1 Sentiment Classification Head

The input to this head is the pooled BERT embedding corresponding to the `<CLS>` token of the training example, a sentence taken from the Stanford Sentiment Treebank dataset. This embedding is passed through a dropout layer, followed by a fully connected linear layer with 512 units and ReLU activation. This is followed by a second fully connected layer with 256 units and ReLU activation. Finally, the output is passed through a final fully connected layer that maps the 256 units to the number of sentiment classes, producing the final classification logits. We then computed the cross-entropy loss between the predicted logits and the ground truth sentiment labels, using the cross-entropy loss function (Appendix Equation 3).

4.2.2 Paraphrase detection head

For the PD head, the input embeddings and attention masks of each paired sentence were concatenated and passed through the BERT model to obtain the pooled embedding output. Similarly to the sentiment classification head, this output was then passed through a drop-out layer, a fully connected linear layer with 512 units and ReLU activation, another dropout layer, a second fully connected linear layer with 256 units and ReLU activation, and a final fully connected layer with a single output unit. The loss for this task was evaluated using binary cross-entropy with logits (Appendix Equation 4).

4.2.3 Semantic textual similarity head

Our STS task implementation closely mirrors the PD head. It starts by concatenating the input embeddings and attention masks from the two sentences, which are then passed through the BERT model to obtain the pooled output. This output goes through a dropout layer, a linear layer with 512 units and ReLU activation, another dropout layer, and a second linear layer with 256 units and ReLU activation. A final fully connected layer maps the 256 units to a single output unit, passed through a sigmoid to normalize the score between 0 and 1. This output is scaled by 5.0 to match the similarity score labels, ranging from 0 to 5. Unlike the other tasks, STS is trained as a regression task using Mean Squared Error as the loss function (Appendix Equation 5).

4.2.4 Sequential vs Round Robin Training

In our approach, the three classification heads — SC, PD, and STS — can be trained either sequentially or using round-robin training. Sequential training involves focusing on one task at a time until completion, until the maximum number of epochs has been reached, before moving on to the next. While this approach allows the model to fully focus on one task at a time, it has some drawbacks. For example, it can miss synergies between tasks and lead to information loss as earlier training is overwritten by learning from later tasks. Additionally, training sequentially can lead to longer overall training times, as each task is handled separately. Round-robin training, on the other hand, involves alternating between tasks in a cyclic manner, updating the model after each batch. Training in this way ensures continuous exposure to all tasks, potentially promoting better generalization and more robust embeddings.

We considered two approaches to round-robin training, due to differing dataset sizes between tasks. Our initial approach was a hybrid between sequential learning and true round-robin training, where in each epoch we would loop through one batch of each dataset until the batches for the smaller datasets ran out. This approach, while effective to some extent, resulted in an imbalanced training process where the larger datasets would dominate the later stages of each epoch. To address this, we adopted a method where, upon reaching the end of a smaller dataset, we restarted and reshuffled it, continuing until iterating over every batch in the largest dataset. This balanced training across tasks, providing diverse training examples and resulting in improved generalization and performance. We implemented all this code ourselves and found that this refined round-robin approach demonstrated the best results, validating its effectiveness in our multitask learning setup.

4.3 Curriculum-based contrastive learning (CBCL)

As an extension to BERT, we implemented curriculum-based contrastive learning (CBCL) to fine-tune the model's embeddings and improve its performance on downstream tasks. This method follows the self-supervised CBCL procedure laid out in Dehghan and Amasyali (2023), in which example triplets (x, x^+, x^-) (anchor, entailment, and contradiction) are labeled as:

$$\textbf{Easy: } d(x, x^+) + m < d(x, x^-)$$
$$\textbf{Semi-hard: } d(x, x^+) < d(x, x^-) < d(x, x^+) + m$$
$$\textbf{Hard: } d(x, x^-) < d(x, x^+)$$

where d is the cosine distance between embeddings, and m is a tunable hyperparameter called the distance margin. The BERT embeddings are then trained using the Normalized Temperature-Scaled Cross-Entropy (NT-Xent) loss as proposed in Chen et al. (2020) with the addition of a hard negative:

$$-\log \frac{e^{sim(x_i, x_i^+)/\tau}}{\sum_{j=1}^{n} (e^{sim(x_i, x_j^+)/\tau} + e^{sim(x_i, x_j^-)/\tau})}, \tag{1}$$

where sim is the cosine similarity, and τ is a temperature hyperparameter that scales the cosine similarity.

Curriculum learning expands on this approach by using a pacing function that sets a threshold for the proportion of difficulty-sorted examples eligible for training at epoch t. Specifically, at epoch t, the proportion of trainable examples $g(t)$ is given by Equation 2:

$$g(t) = \left(\frac{t}{T}\right)^{\lambda} \cdot k \tag{2}$$

where T is the total number of epochs, t is the current epoch, λ is a tunable hyperparameter (set to 1 in Dehghan and Amasyali (2023), but other common values are $\frac{1}{2}$ and 2 for root and quadratic functions, respectively), and k is the total number of examples in the dataset.

In theory, fine-tuning embeddings in this way allows the model to learn from progressively more challenging examples, thereby refining its understanding of semantic relationships. As training progresses, the difficulty of the examples that the model is trained on increases according to the pacing function, enabling the model to gradually tackle more complex examples. This staged approach helps the model build on its previous knowledge incrementally, potentially leading to more robust and nuanced embeddings.

To establish a baseline for how much improved performance we could expect to see when fine-tuning the BERT embeddings directly, we initially trained without using any type of curriculum pacing, i.e. all examples were made available to be trained on in each epoch. We then experimented with ordering the training examples using the `easy, semi-hard, hard` labelling framework of Dehghan and Amasyali (2023); as well as ordering them in descending order of cosine similarity between premise and entailment ($d(x, x^+)$), or increasing order of cosine similarity between premise and contradiction ($d(x, x^-)$). Note that all of the code to further pre-train BERT embeddings using CBCL, including the loss function, curriculum pacing methods, and training loops, were implemented by our team in PyTorch.

4.4 Synonym Replacement

To enhance the robustness and generalization of our model, we employed synonym replacement as a data augmentation strategy. We implemented three methods via our own code. The first method involved a naive approach where each word in the training data was replaced with a synonym from WordNet with a probability p. This precomputation method was simple and allowed for a diversified dataset.

The second method involved using a part-of-speech (POS) analyzer based on the Penn Treebank tags by Marcus et al. (1993) to replace all nouns and adjectives in a sentence with their corresponding WordNet synonyms, again with a probability p. This method ensured syntactic accuracy while introducing variability.

Our most effective method, the third one, was integrated directly into the training process. Inspired by Algorithm 1 from Jungiewicz and Smywiński-Pohl (2019), we applied synonym replacement at the batch level during stochastic gradient descent training. With probability p=0.25, each batch had one adjective in every sentence replaced with a synonym from WordNet. To prevent overfitting and ensure diversity, we alternated between replacing the first and last adjectives found in sentences. This dynamic augmentation, performed live during training, mitigated overfitting risks and maintained fresh sentence variations, leading to better generalization and performance.

5 Experiments

5.1 Data

For our three different downstream tasks we used three different data sources. First, for sentiment analysis we used the Stanford Sentiment Treebank (SST) dataset, which compiles movie reviews from Rotten Tomatoes. It contains 11,855 single sentences, which we split into 8,544 training sentences, 1,101 dev sentences, and 2,210 training sentences. The Stanford parser was then run over the dataset, leading to 215,154 unique phrases that were labeled as 'negative', 'somewhat negative', 'neutral', 'somewhat positive', or 'positive'. With our models we aimed to predict which of these four labels a sentence would have, giving us a multiclass classification task to train our model on. Second, for paraphrase detection we used a dataset from the online question answering site Quora. It contains pairs of sentences that are labeled as either paraphrases of one another or not. Of the 404,298 pairs, we allocated 283,010 for train, 40,429 for dev, and 80,859 for test splits. Thus, this dataset produced a binary classification task where we would predict if sentence pairs are paraphrases. Third, we used the SemEval STS Benchmark Dataset for semantic textual similarity prediction which contains sentence pairings of varying similarities from image captions, news headlines and user forums. Sentence pairings are ranked on a scale of 0 (unrelated) to 5 (equivalent meaning). Of the 8,628 pairings, we assigned 6,040 to the training dataset, 863 to the dev dataset, and 1,725 to the testing dataset. Thus, this dataset presented us with a regression task, allowing our model to output values in the range of 0 to 5 which was compared to a true value within the same range.

Lastly, for our extension of CBCL we introduce two new datasets: the Stanford Natural Language Inference (SNLI) corpus and Multi-Genre Natural Language Inference (MultiNLI) corpus. The former was written by humans doing a "novel grounded task based on image captioning" and contains 570,000 pairs of sentences that are labeled as either entailment, contradiction, or neutral as specified in Bowman et al. (2015). The latter according to Williams et al. (2018) has a similar format but contains 433,000 additional sentence pairs that are significantly more difficult and taken from ten different genres including telephone conversations, travel guides, and government documents. We wrote the code ourselves to combine the two datasets and create sentence triplets (x_i, x_i^+, x_i^-) that merged entailment and contradiction pairs on the first sentence of the pair. Specifically, we have that x_i is the premise or anchor of the triplet, x_i^+ is the positive taken from entailment sentences, and x_i^- is the hard negative taken from contradiction sentences. In the end, this yielded a total of 275,600 triplets that we used to further pre-train our BERT model using CBCL before fine-tuning on the three downstream tasks. Thus, for this part of the analysis the inputs were sentence triplets (x_i, x_i^+, x_i^-) and the outputs were contrastive loss values, with the equation for these values explained found in Equation (1) above.

5.2 Evaluation method

The tasks of sentiment analysis and paraphrase detection were evaluated using prediction accuracy, or the percentage of output values that correctly matched the true class label. For the STS task, we used Pearson correlation between the predicted and true similarity scores as our evaluation metrics, identical to what was used in the original SemEval paper Agirre et al. (2013).

To evaluate how the various hyperparameters pertaining to CBCL affected the model's performance, we needed a different evaluation metric. The seminal paper on CBCL only used a single combination of hyperparameters, and we wanted to experiment with tuning these hyperparameters to see if we could get any improvements in model performance. To measure this, we compared the mean NT-Xent loss, Equation (1) above, over the entire training dataset at the last epoch of pre-training for each combinations of hyperparameters. This is the loss we used to in order to perform the pre-training as described in the Approach section, and thus a lower mean loss should indicate better tuning of the hyperparameters for the BERT embedding model.

5.3 Experimental details

Both the pre-training using CBCL and the fine-tuning on the three downstream tasks were done using similar experimental setups. All experiments were run using Google Cloud Compute Engine virtual machines with NVIDIA T4 GPUs. Some hyperparameter tuning was done for each of the two parts of our model pipeline, and we found that in both sections using 10 epochs achieved sufficiently low loss and a batch size of 32 lead to acceptable runtime speed without overloading computer memory. Then for pre-training using CBCL we found that a learning rate of 2×10^{-5} led to the lowest loss, whereas for task fine-tuning a lower learning rate of 1×10^{-5} was better. Also, for the latter step a hidden layer dropout probability of 0.5 worked well.

5.4 Results

Our baseline model, which only takes the BERT embeddings and fine-tunes them on our three tasks and does not implement contrastive learning or CBCL, performs best as seen in Table 1. In fact, on the test leaderboard our baseline performs extremely well, with our overall test score being less than 0.015 below the top performing model. Our SA task seems especially well-tuned, as our accuracy of 0.540 puts us level with the third-placed model at the time of writing. The results of the same models on the dev set can be found in Appendix Table A1.

However, surprisingly both contrastive learning and CBCL hurt the model's ability to perform the three downstream tasks. These were extensions that were supposed to improve BERT embeddings by making them more distinguishable in the embedding space. This can possibly be explained by the fact that further pre-training of BERT on a different loss, NT-Xent loss, which is not related to our downstream tasks of interest can harm the performance on these tasks. Instead, we found that adjusting the multitask fine-tuning phase with different learning rates, number of epochs, and task heads as well as using round-robin training led to much better improvements than performing additional pre-training using CBCL. This shows us the importance of fine-tuning: no matter how much pre-training is done to a model, it still needs to learn the specific tasks well during fine-tuning to be able to perform adequately. This also tells us that our approach of using CBCL is not effective for SC, PD, and STS tasks, and instead we were correct in spending significant time developing our multitask fine-tuning pipeline.

Finally, when doing Synonym Replacement in addition to CBCL there is improvement on PD and STS tasks, but this translates to an improvement of less than 0.010. This conforms with our tempered expectations as Jungiewicz and Smywiński-Pohl (2019) had reported at best a 1.2% improvement in their CNN model via their synonym replacement methodology, from which we took inspiration.

Model	SA Accuracy	PD Accuracy	STS Correlation	Overall Test score
Baseline	**0.540**	**0.900**	**0.844**	**0.788**
Curriculum-Based Contrastive Learning (CBCL)	0.518	0.896	0.833	0.777
CBCL with Synonym Replacement	0.518	0.900	0.838	0.779

Table 1: Results on Test Dataset

6 Analysis

6.1 Studying Select Examples

Inspired by Table I of Zhou et al. (2022) and Table 1 of Wahle et al. (2023), we developed a set of custom categories to analyze why our language model failed to detect paraphrases. The criteria for an example to be assigned to each category is detailed in section A.2 of our Appendix. To assess our model's performance and identify improvement areas, we manually categorized 100 random examples from the Quora Dev Set in which our best performing model failed to make the correct prediction. To check if the frequency of these failure categories was consistent with the overall distribution, we compared these with 100 random examples from the Quora Training Set and another 100 random examples from the Quora Dev Set, all categorized under the same criteria (Figure 1). By including these additional samples, we aimed to see if the support for each category was roughly consistent with the failed examples. For binary classification tasks like paraphrase detection, the reference sample set was categorized as if the model's predictions had failed.

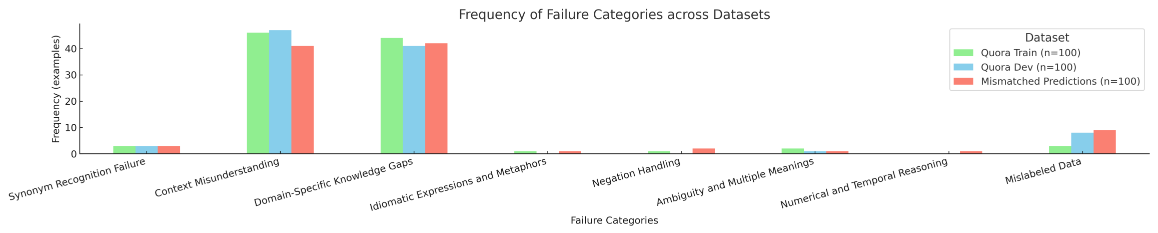

Figure 1: Frequencies of failure categories in paraphrase detection examples.

As shown in Figure 1, the most prominent category of failures was Context Misunderstanding. This was expected, as it indicates that the language model has not yet developed a comprehensive understanding of sentence directionality and contextual relationships. The second most prominent category, however, was Domain-Specific Knowledge Gaps, which implies that the model has not been adequately exposed to specific facts, proper nouns, or specialized knowledge areas. It should also be noted that 13% of the model's failed predictions in the sample were actually correct, as can be seen in the right-most category of Figure 1. This highlights some concern with the Quora datasets but also demonstrates the power of our pre-training since the model prevailed and made correct predictions even given mislabeled data during fine-tuning.

The prominence of Domain-Specific Knowledge Gaps as a failure mode led us to the hypothesis that this could be attributed to limitations in the model's training data. In particular, we theorize that the "bert-base-uncased" model (Devlin et al., 2019), pre-trained on BooksCorpus (800M words) and English Wikipedia (2,500M words), might not perform well on examples involving proper nouns lacking prevalence in the fine-tuning data. To investigate this, we examined 1,300 Quora Dev Set predictions, in which we compared three correctly classified PD examples against three incorrectly classified ones of similar topics. We traced the frequency of proper nouns in all fine-tuning sentences, hypothesizing that failed examples would show fewer proper nouns. However, our manual analysis, summarized in Table 2, did not support this hypothesis. Some failed examples actually had higher proper noun frequencies compared to successful ones of similar topics, suggesting that the multiplicity of pronouns and unaccounted frequencies of proper nouns in the large pre-training corpus might be a confounding factor.

Predicted correctly		Predicted incorrectly	
Question Pair	**Noun Frequency**	**Question Pair**	**Noun Frequency**
Are there any genetic basis for an IndoEuropean ('Aryan') migration theory? / Who were the Indo-Europeans?	IndoEuropean: 0, Indo-Europeans: 2, Aryan: 79	Is Ashwathama still seen in Himalayas? / Is Ashwatthama of Mahabharat still alive?	Ashwathama: 3, Ashwatthama: 3, Mahabharat: 194, Himalayas: 15
Is diphenhydramine soluble in water? Why or why not? / Is zinc sulfate soluble in water? Why are most sulfates soluble in water?	Diphenhydramine: 6, Sulfate: 50, Zinc: 59	What is the different between the Prozac and Zoloft? / What is the difference between Zoloft and Xanax?	Prozac: 38, Zoloft: 14, Xanax: 57
What are some good books on Joseph Goebbels? / What are the best books on Joseph Goebbels?	Goebbels: 27, Joseph: 55	Where is Anna Hazare these days? / Where is Anna Hazare now?	Hazare: 5, Anna: 520

Table 2: Comparison of Correctly and Incorrectly Predicted Paraphrase Pairs

6.2 Performance on Data Subsets

Previous papers on CBCL have only reported single overall numbers like accuracy to measure performance. This can be misleading and does not provide insights into the strengths and weaknesses of a model. Thus, to further examine how different elements of the training dataset contributed to the model's overall performance metrics across the three downstream tasks, we looked at the model's precision, recall, and F1 scores across different classes of the classification tasks SST and PD using the Dev dataset. The performance of our key models on the SST dataset for sentiment analysis can be seen in Table 3.

	Precision					**Recall**					**F1 Score**				
Model	**0**	**1**	**2**	**3**	**4**	**0**	**1**	**2**	**3**	**4**	**0**	**1**	**2**	**3**	**4**
Baseline	0.53	0.54	0.43	0.48	0.54	0.32	0.63	0.32	0.61	0.48	0.40	0.58	0.37	0.54	0.51
Contrastive Learning	0.55	0.51	0.41	0.47	0.57	0.31	0.59	0.34	0.62	0.47	0.40	0.55	0.37	0.54	0.51
Curriculum-Based Contrastive Learning	0.46	0.52	0.45	0.50	0.56	0.27	0.61	0.33	0.65	0.51	0.34	0.56	0.38	0.56	0.54
Synonym Replacement	0.44	0.53	0.42	0.49	0.55	0.34	0.60	0.33	0.63	0.43	0.38	0.56	0.37	0.55	0.48
CBCL with Synonym Replacement	0.47	0.49	0.42	0.49	0.59	0.26	0.73	0.19	0.64	0.45	0.33	0.59	0.26	0.56	0.51

Table 3: Precision, Recall, and F1 Scores per Class for Sentiment Analysis on Dev Set

From this table, we can see that our models generally performed best (in terms of F1 score, a balanced measure of both precision and recall) at identifying slightly negative or slightly positive sentiments (classes 1 and 3). All of our models uniformly struggled at identifying neutral sentiment (class 2), which we found unsurprising: sentences that express neutral sentiment may by nature simply be more subjective or difficult to identify than sentences that express a clear sentiment. The most surprising result is that all of our models struggled to identify extreme sentiment, particularly extremely negative sentiment (class 0). By considering the context of the dataset as consisting of movie reviews, we suspect that this is due to the fact that negative movie reviews can often be sardonic or sarcastic, making it challenging for the models to accurately capture the intensity of the sentiment. Additionally, extreme sentiments may involve more nuanced language or cultural references that the models might not interpret correctly, leading to lower performance in these categories.

A similar analysis was conducted for the two classes in the PD task. The differences in performance were minor and likely due to imbalances in class support within the training dataset. For instance, on the Quora dev dataset, the F1 score for the "True" class was 0.86 when training with CBCL, while the "False" class achieved a higher F1 score of 0.92. This discrepancy can be attributed to the support levels, with the "True" class having 104,579 instances compared to the "False" class' 178,424 instances.

6.3 Ablation Studies

Since CBCL is a relatively new technique for improving BERT embeddings introduced in 2023, we wanted to understand better how tuning hyperparameters pertaining to this technique would affect performance. This is especially important since the paper introducing this method only uses a single combination of hyperparameters without justification.

The first ablation study focused on λ which relates to the pacing function that decides when samples of varying difficulty are introduced into the training process. Values of λ of 0.5, 1, and 2 relate to root, linear, and quadratic pacing functions, respectively. The results of this study are below in Table 4, and we see that mean contrastive loss decreases as λ decreases. Thus, choosing $\lambda = 0.5$ led to the best results as it made entailment embedding pairings more distinct from contradiction embedding pairings.

λ	Mean Contrastive Loss
0.5	**3.577**
1	3.606
2	3.646

Table 4: Pacing Function λ Ablation Study Results

Then the second ablation study was on the effect of the scoring technique used to sort training samples from easy to hard. In the original CBCL paper Dehghan and Amasyali (2023) only a single technique is used: labeling samples as easy, semi-hard, and hard triplets using a distance margin of 0.2 and then grouping by these three groups. We experimented with changing this value of m. However, if the goal of curriculum-based learning is to train the model on the easiest triplets first and the hardest last, we thought it made more sense to sort triplets based on increasing entailment similarity or decreasing contradiction similarity. Thus, the triplets where the anchor and entailment embeddings are closest or the anchor and contradiction embeddings are furthest should be easiest and thus fed to the model first. But using the grouping technique used in the paper, within easy, semi-hard, and hard groups there can still be a large variation in similarities between triplets which is not ideal.

The results of the varying the triplet scoring technique are seen in Table 5. We can see that the choice of m does not greatly affect the mean contrastive loss at the end of further BERT pre-training since all values are within 0.001 of each other. Sorting by contradiction cosime similarity again led to almost identical results. However, changing the technique to sorting triplets using entailment cosine similarities did improve results as mean contrastive loss decreased by around 0.003. Although this number looks small, when using the model resulting from this new technique on our three downstream tasks, we saw improvements of a few percentage points on the two tasks evaluated on accuracy and a few tenths on STS.

Scoring Technique	Mean Contrastive Loss
Grouping with $m = 0.3$	3.577
Grouping with $m = 0.2$	3.577
Grouping with $m = 0.1$	3.576
Grouping with $m = 0.05$	3.577
Entailment cosine similarity	**3.573**
Contradiction cosine similarity	3.577

Table 5: Triplet Scoring Technique Ablation Study Results

The third ablation study we performed was on the effect of the temperature hyperparameter τ in the NT-Xent loss objective (Equation 1). Initially, we started with no temperature scaling (value of 1) to observe the baseline performance. We then experimented with different values around the range of 0.05, the value used in Gao et al. (2021), to understand the impact of varying the temperature hyperparameter. Through these experiments, we found that a value of 0.05 yielded the best results. The results of our experiments varying τ on the NT-Xent loss objective, when trained without curriculum pacing, can be seen in Table 6.

τ	Mean Contrastive Loss
1	3.833
0.025	3.713
0.05	**3.663**
0.075	3.801

Table 6: Temperature Hyperparameter τ Ablation Study Results

7 Conclusion

Overall we built a very high-performing model that gets an overall score close to the top models on the test leaderboard. This was done by building robust heads for our three downstream tasks SC, PD, and STS and implementing the round-robin multitask training technique. We found that further pre-training BERT using both contrastive learning and curriculum-based curriculum learning did not lead to significant improvements. In contrast, these extensions actually detracted from performance for all three tasks. We hypothesize that this is due to the fact that these extensions use a different loss function, and so pre-training BERT parameters using this loss takes information away or distracts from the tasks we are interested in. Using data augmentation has also been suggested in the literature to improve performance, but again we see no signficant changes in model performance. Thus, our findings directly contradict those of Dehghan and Amasyali (2023) which introduced CBCL in 2023. They used different evaluation metrics and different downstream tasks, and thus we have shown that CBCL does not improve a model attempting to do all three tasks SC, PD, and STS at once.

For future work, we suggest exploring curriculum-based training with datasets and difficulty objectives more closely related to the downstream tasks. This could involve tailoring the difficulty metrics to better match the characteristics of SC, PD, and STS. Additionally, incorporating more complex data augmentation techniques, such as paraphrasing, back-translation, and contextual word replacements, might help improve model robustness and performance. These approaches could provide a better alignment between the pre-training phase and the specific requirements of the downstream tasks, potentially leading to more significant improvements.

8 Ethics Statement

One ethical challenge is avoiding the perpetuation of biases in our data. This issue is extremely pertinent to our project since we perform contrastive learning. This technique spreads out word embeddings in the embedding space. Thus, if our model learns bias in the pre-training phase in the form of embeddings differing depending on race, ethnicity, gender, religion, etc., then these embeddings will become more dissimilar. For example, if "she is a receptionist" and "he is a receptionist" have very different representations because the model is trained that one is more frequent than the other, then our further pre-training with contrastive learning will only exacerbate this gender bias. A strategy to mitigate this risk is the incorporation of bias detection tools after our training on the contrastive learning objective. Some examples include applying the Word Embedding Association Test (WEAT) or debiasing vectors using subspace projection according to Bolukbasi et al. (2016). One potential limitation of the latter strategy, however, is that it has been primarily tested on gender-based bias and not been empirically shown to be generalizable to all types of implicit bias.

A second concern is the potential for the model to produce and propagate misinformation, particularly in the context of the downstream paraphrase detection and STS tasks. The model may incorrectly identify paraphrases or label two bodies of text as more similar than they actually are, leading to the spread of inaccurate or outright counterfactual information. This is especially dangerous if the model is used for tasks involving political figures, historical topics, or other sensitive subjects where misinformation can have significant consequences. For example, models trained for STS are commonly applied in news aggregation and content moderation, where precise language is critical to preserving meaning. Our model training process is particularly vulnerable to this issue because we do synonym replacement. Thus, we may make imperfect replacements that lead to us generating paraphrases that are not truly equivalent in meaning. Mitigating this issue requires a systematic analysis of standard synonym replacement databases with human-in-the-loop evaluations at each stage of the training process, an approach suggested by Amershi et al. (2014). Additionally, implementing robust monitoring systems, such as anomaly detection algorithms to identify unusual model outputs and automatic alerts to flag potentially misleading information for human review, can help minimize the impact of any misinformation generated by the model. Applying multi-layered strategies such as these, developed in consultation with misinformation experts, is essential to maintaining the accuracy and trustworthiness of the outputs of any public-facing language model.

References

Mahdi Abdollahi, Xiaoying Gao, Yi Mei, Shameek Ghosh, Jinyan Li, and Michael Narag. 2021. Substituting clinical features using synthetic medical phrases: Medical text data augmentation techniques. *Artificial Intelligence in Medicine*, 120:102167.

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.

Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *AI magazine*, 35(4):105–120.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Somaiyeh Dehghan and Mehmet Fatih Amasyali. 2023. Selfccl: Curriculum contrastive learning by transferring self-taught knowledge for fine-tuning bert. *Applied Sciences*, 13(3):1913.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Michał Jungiewicz and Aleksander Smywiński-Pohl. 2019. Towards textual data augmentation for neural networks: synonyms and maximum loss. *Computer Science*, 20.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Lalu M. Riza Rizky and Suyanto Suyanto. 2021. Improving stance-based fake news detection using bert model with synonym replacement and random swap data augmentation technique. In *2021 IEEE 7th Information Technology International Seminar (ITIS)*, pages 1–6.

Tsegaye Misikir Tashu and Tomáš Horváth. 2022. Synonym-based essay generation and augmentation for robust automatic essay scoring. In *Intelligent Data Engineering and Automated Learning – IDEAL 2022*, pages 12–21, Cham. Springer International Publishing.

Jan Wahle, Bela Gipp, and Terry Ruas. 2023. Paraphrase types for generation and detection. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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. Association for Computational Linguistics.

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

A Appendix

Model	SA Accuracy	PD Accuracy	STS Correlation
Baseline	0.501	**0.900**	**0.853**
Curriculum-Based Contrastive Learning (CBCL)	**0.504**	0.898	0.842
Synonym Replacement	0.493	0.899	0.841
CBCL with Synonym Replacement	0.496	0.897	0.827

Table A1: Results on Dev Dataset

A.1 Key equations

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{3}$$

Equation 3: Cross-Entropy Loss Function, where y_i is the true label and $\hat{y}_i$ is the predicted probability for the i-th instance.

$$\mathcal{L} = -\left[y \log(\sigma(\hat{y})) + (1-y)\log(1-\sigma(\hat{y}))\right] \tag{4}$$

Equation 4: Binary Cross-Entropy Loss Function, where y is the true label (1 for paraphrase, 0 for non-paraphrase), $\hat{y}$ is the raw output (logit) from the model, $\sigma(\hat{y})$ is the sigmoid function applied to $\hat{y}$ defined as $\sigma(\hat{y}) = \frac{1}{1+e^{-\hat{y}}}$.

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5}$$

Equation 5: Mean Squared Error (MSE) Loss Function, where N is the total number of samples, y_i is the true similarity score for the i-th sample, $\hat{y}_i$ is the predicted similarity score for the i-th sample, scaled to the range 0 to 5.

A.2 Descriptions for Paraphrase Detection Failure Categories

Synonym Recognition Failure: The model fails to recognize synonyms that could be directly replaced by each other. Example: “automobile” vs. “car”.

Context Misunderstanding: The model does not understand the directionality or context of the sentences, leading to incorrect interpretations. The model may think that two completely unrelated sentences are the same. Example: Questions beginning with “how” vs. “why”, or confusing two people in a sentence.

Domain-Specific Knowledge Gaps: The model lacks the specific knowledge required to understand domain-specific topics, academic concepts, jargon, or proper nouns. Generally speaking, a failure that can qualify for either (D) or (C) is categorized as (D) because (C) is a catch-all for complete misunderstandings of sentences.

Idiomatic Expressions and Metaphors: The model fails to understand idiomatic expressions or metaphors. Example: Misinterpreting phrases like "layman’s terms."

Negation Handling: The model struggles with sentences that involve negation or opposites. Example: "The dog is happy” vs. “The dog is not happy”.

Ambiguity and Multiple Meanings: The model cannot correctly disambiguate words with multiple meanings based on context. Example: “bank” vs “bank”.

Numerical and Temporal Reasoning: The model struggles with sentences involving numerical or temporal elements. Example: "How to get rich in 3 months” vs. “how to get rich in 4 months.” If the failure pertains to a math equation with mostly algebra, categorize it as (d) instead.

Mislabeled Data: Failures due to incorrect labeling in the dataset, whereby the model’s prediction is actually determined correct upon our manual review of the dataset.