Supercharging MinBERT with Contrastive Learning & Self-Distillation

Stanford CS224N Default Project

Cécile Logé Baccari
Department of Computer Science
Stanford University
`ceciloge@stanford.edu`

Abstract

This paper explores the integration of Contrastive Learning and Self-Distillation techniques to enhance MinBERT's ability to generate high-quality sentence embeddings for diverse NLP tasks. Our primary aim is to validate whether these methods, proven effective in domains like Computer Vision and in some cases for Semantic Textual Similarity (STS), can generalize to tasks such as Sentiment Classification and Paraphrase Detection. We implement various contrastive learning paradigms—unsupervised, semi-supervised, and supervised—alongside a self-distillation approach where a student model learns from the predictions of a finetuned teacher model. Our extensive experiments demonstrate that supervised pretraining consistently outperforms unsupervised methods, especially when leveraging external datasets. We show that self-distillation significantly enhances model performance, confirming its regularization benefits. However, improvements are task-dependent, with contrastive learning favoring tasks related to semantic similarity over sentiment analysis. Our best ensemble model achieves a notable performance gain, highlighting the potential of combining these techniques.

TA Mentor: Johnny Chang.
No external collaborator / mentor. No sharing. For CS224N only.

1 Introduction

In this project, we set to tackle the fundamental problem of generating high-quality sentence embeddings in order to improve performance on downstream tasks. While traditional approaches often rely on extensive labeled datasets and complex supervised methodologies, our work aims to explore the potential of unsupervised and minimally supervised techniques to achieve competitive results.

We propose an approach that integrates contrastive learning and self-distillation techniques to improve the performance of MinBERT, our lightweight variant of the BERT model. Contrastive learning, which learns representations by contrasting positive and negative pairs, has shown remarkable success in domains like computer vision and, more recently, semantic textual similarity (STS). By extending this technique to other NLP tasks, we seek to demonstrate its generalizability and effectiveness. Similarly, self-distillation, where a model learns from its own predictions, has shown potential in improving model efficiency and effectiveness. We conduct a comprehensive set of experiments involving unsupervised, semi-supervised and supervised methods, and achieve notable performance gains with our best ensemble model. Our findings contribute to the understanding of how these techniques can be applied across various NLP tasks.

2 Related Work

The idea behind contrastive learning is to minimize the distance between embeddings for similar sentences (positive pairs) while maximizing it for dissimilar sentences (negative pairs). A significant challenge in this approach lies in the identification and selection of appropriate positive and negative pairs within the dataset, especially in the unsupervised setting. While this is a relatively easy question in computer vision - as positive pairs can simply be cropped / flipped / rotated versions of the same image - it is much trickier in natural language processing. SimCSE (Gao et al., 2022) and SimCLR (Chen et al., 2020a), two contrastive learning frameworks relying partly on unsupervised techniques to build better representations, directly influenced our project's experiments. In particular, SimCSE introduces an ingenious way to generate positive pairs without needing any external data, complex manipulations or a dual encoder framework: they use dropout as a data augmentation technique to create positive pairs of embeddings from the same sentence. Prior work had also tried to make unsupervised contrastive learning work via synonym substitution, random word deletion (Meng et al., 2021), same document sampling (Giorgi et al., 2021) or dual encoding (Zhang et al., 2020; Carlsson et al., 2021), but these methods were either too complex or leading to relatively unconvincing results.

Despite the successes of these approaches, most existing work has focused primarily on STS tasks, with limited exploration of their applicability to other NLP tasks such as sentiment classification and paraphrase detection. This gap in research presents an opportunity to investigate whether the advantages of contrastive learning and self-distillation can extend beyond STS to improve performance across a broader range of NLP applications. On top of this, studies like SimCLRv2 (Chen et al., 2020b) in Computer Vision have demonstrated the effectiveness of combining contrastive learning with self-distillation, suggesting potential benefits in NLP as well.

3 Approach

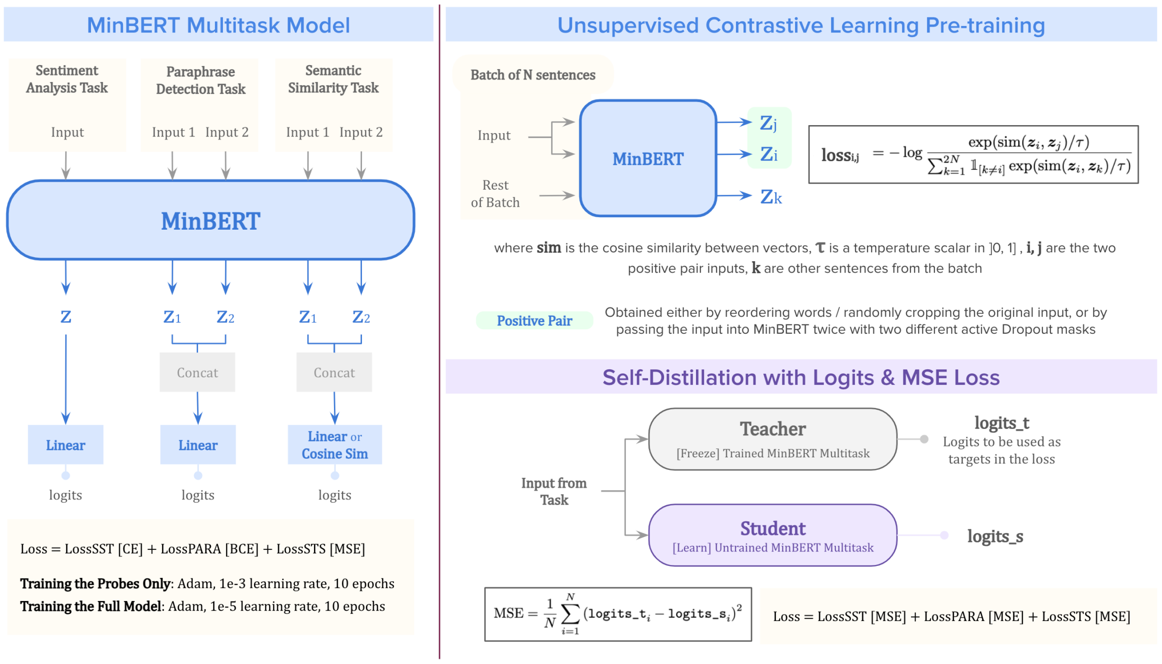

MinBERT Multitask Model
Sentiment Analysis Task
Paraphrase Detection Task
Semantic Similarity Task
Input
Input 1 Input 2
Input 1 Input 2
MinBERT
Z
Z1 Z2
Z1 Z2
Concat
Concat
Linear
Linear
Linear or Cosine Sim
logits
logits
logits
Loss = LossSST [CE] + LossPARA [BCE] + LossSTS [MSE]
Training the Probes Only: Adam, 1e-3 learning rate, 10 epochs
Training the Full Model: Adam, 1e-5 learning rate, 10 epochs
Unsupervised Contrastive Learning Pre-training
Batch of N sentences
Input
Rest of Batch
MinBERT
Zj
Zi
Zk
$\text{loss}_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(z_i, z_k)/\tau)}$
where sim is the cosine similarity between vectors, τ is a temperature scalar in]0, 1] , i, j are the two positive pair inputs, k are other sentences from the batch
Positive Pair
Obtained either by reordering words / randomly cropping the original input, or by passing the input into MinBERT twice with two different active Dropout masks
Self-Distillation with Logits & MSE Loss
Input from Task
Teacher
[Freeze] Trained MinBERT Multitask
logits_t
Logits to be used as targets in the loss
Student
[Learn] Untrained MinBERT Multitask
logits_s
$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\texttt{logits\_t}_i - \texttt{logits\_s}_i)^2$
Loss = LossSST [MSE] + LossPARA [MSE] + LossSTS [MSE]

Figure 1: Illustration of the Approach.
Left: Multitask Finetuning setup, **Top Right:** Contrastive Learning setup for unsupervised pre-training, **Bottom Right:** Self-Distillation procedure with logits + MSE loss.

Multitask Finetuning Setup: Our multitask finetuning setup involves three single linear heads, one for each dataset and task. We obtain logits from each heads: the Sentiment Analysis logits are passed into a Cross Entropy loss (and turned into probabilities with softmax during evaluation); the concatenated Paraphrase Detection logits are passed into a Binary Cross Entropy loss (and turned into

probabilities with sigmoid during evaluation); the concatenated Semantic Textual Similarity logits are passed into an MSE loss. See Figure 1 (Left) for an illustration.
For this last task, we also experiment with Cosine Similarity: instead of a feeding a linear head with the concatenated embeddings, we compute the cosine similarity of the two and rescale the results into logits from 0 to 5. We then pass these logits into an MSE loss.
The final overall loss to optimize is the sum of the three, such that we can finetune the model on all three tasks at once. Our **baseline model** consists in finetuning MinBERT with these linear heads without additional pretraining or self-distillation.

Unsupervised Contrastive Learning: We implement an unsupervised framework to experiment with two positive-pairing options: (1) random sentence reordering, and (2) passing the sentence input into MinBERT twice with two different active dropout masks.
For each option, we create N positive pairs for each batch of N sentences and pre-train MinBERT with the Normalized Temperature-scaled Cross Entropy loss:

$$\mathrm{NTXE}_{\mathrm{i,j}} = -\log \frac{\exp\left(\mathtt{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp\left(\mathtt{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_k\right)/\tau\right)}$$

where $\boldsymbol{z}$ are the MinBERT embeddings, $\mathtt{sim}$ is cosine similarity, τ is a temperature scalar, i, j are the two positive pair inputs, k are other sentences from the batch that can be considered "soft negative" matches.

Supervised Contrastive Learning: We can use the contrastive learning approach in a supervised way by using labels from the Paraphrase training set (1 means it's a positive pair, 0 means it's not) and the STS training set (> 3.5 means it's a positive pair, below means it's not). Non-positive pairs are still used during training as soft negative matches. Note that this approach implies ignoring the SST data entirely during pre-training, which as we'll see in the results is not ideal when it comes to performance on the Sentiment Analysis task.
We also experiment using an external dataset, the Jina AI NLI Negation dataset (Günther et al., 2023) where each data point consists of a triplet ('anchor', 'entailment', 'negative'). We use ('anchor', 'entailment') as a positive pair, and ('anchor', 'negative') as a hard negative pair, with the same NTXE loss as before, simply adding the hard negative examples to the batch of soft negatives. However, in order to make the most out of the hard negative pairs, we also experiment with a more standard Triplet Loss (Chechik et al., 2010):

$$\mathrm{TL}_{\mathrm{i}} = \max(\mathtt{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_i^{+}\right) - \mathtt{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_i^{-}\right) + 1, 0)$$

where $\boldsymbol{z}$ are the MinBERT embeddings, $^{+}$ and $^{-}$ are the positive ('entailment') and negative ('negative') matches resp., and $\mathtt{sim}$ is cosine similarity.

Semi-Supervised Contrastive Learning: Supervised contrastive learning, even when conducted on the task datasets, leads to ignoring the SST train data. To remedy that, we also experiment with a semi-supervised approach where positive pairs are formed from labels for the STS and Paraphrase train sets (similar to supervised learning) and from two different dropout masks for the SST train sets (similar to unsupervised learning). We use the NTXE loss, as described in the previous section.

Self-Distillation: Unsupervised self-distillation is a technique where a model is trained using its own predictions as part of the learning process. We set up self-distillation starting with *teacher* model t that's already been finetuned for the tasks at hand, and a *student* model s of similar architecture that hasn't been trained yet.
During training, the student model learns to mimic the teacher model's output instead of referring to the truth labels, meaning we replace the truth labels in the loss with the teacher model's predicted probabilities p^t. For example, Cross Entropy loss (for Sentiment Analysis and Paraphrase Detection) becomes:

$$\mathrm{CE}^{\mathtt{distill}} = -\frac{1}{N} \sum_{i=1}^{N} \left[\sum_{y} p^t\left(y \mid \boldsymbol{x}_i\right) \log p^s\left(y \mid \boldsymbol{x}_i\right) \right]$$

However, we also experiment with MSE loss using the teacher's output logits directly instead of probabilities, which leads to the following loss for all three tasks:

$$\text{MSE}^{\texttt{distill}} = \frac{1}{N}\sum_{i=1}^{N}(\texttt{logits\_t}_i - \texttt{logits\_s}_i)^2$$

and note that it is more efficient at improving the original performance (see Experiments).

4 Experiments

No Pre-training | Self-Distillation Setup *Results on the Dev Set*	**No Pre-training | Self-Distillation**			
	Sentiment Acc	Paraphrase Acc	STSimilarity Corr	Score
Last Linear Layer | Using Linear Head for STS Task				
Baseline | *finetuned only*	0.383	0.659	0.268	**0.559**
Self-Distilled | *with original losses + teacher probs*	0.376	0.640	0.253	**0.548**
Self-Distilled | *with MSE losses + teacher logits*	0.381	0.638	0.235	**0.546**
Full Model | Using Linear Head for STS Task				
Baseline | *finetuned only*	0.501	0.738	0.355	**0.639**
Self-Distilled | *with original losses + teacher probs*	0.485	0.721	0.364	**0.629**
Self-Distilled | *with MSE losses + teacher logits*	0.509	0.732	0.375	**0.643**

Contrastive Pre-training | Parameter Setup with NTXEnt Loss *Results on the Dev Set*	**Contrastive Learning | Unsupervised with Dropout Pairs**			
	Sentiment Acc	Paraphrase Acc	STSimilarity Corr	Score
Full Model | Using Linear Head for STS Task				
Embed Dropout p= 0.3, Temperature tau = 1	0.500	0.734	0.355	**0.637**
Embed Dropout p= 0.1, Temperature tau = 1	0.495	0.734	0.382	**0.640**
Embed Dropout p= 0.05, Temperature tau = 1	0.506	0.737	0.383	**0.645**
Embed Dropout p= 0.025, Temperature tau = 1	0.501	0.736	0.372	**0.641**
Embed Dropout p= 0.05, Temperature tau = 0.5	0.501	0.747	0.372	**0.645**
Embed Dropout p= 0.05, Temperature tau = 0.1	0.500	0.737	0.360	**0.639**

Figure 2: Parameter Optimization, on the Dev Dataset.
Top: Shows the impact of Self-Distillation on baseline performance, using different loss setups.
Bottom: Shows the results of Unsupervised Contrastive Pretraining, with hyperparameter search for the Dropout Pairing procedure (p) and for the NTXEnt Loss temperature (tau τ).

4.1 Data & Evaluation

Pretraining data: During our experiments with supervised contrastive learning, we use the Jina AI NLI Negation dataset (Günther et al., 2023) as an external task-agnostic dataset. Each data point consists of a triplet ('anchor', 'entailment', 'negative') where ('anchor', 'entailment') are positive pairs taken from SNLI, and 'negative' contradicts both 'anchor' and 'entailment'.

Finetuning data & Evaluation: For finetuning as well as some of the contrastive learning experiments, we use the SST dataset for Sentiment Analysis (with accuracy as our evaluation metric), the Quora dataset for Paraphrase Detection (with accuracy as our evaluation metric) and the SemEval STS Benchmark dataset for Semantic Textual Similarity (with pearson correlation as our evaluation metric). During cross-validation on the Dev Set, we only keep the models that achieve the best performance on the `score` metric defined as $\texttt{score} = \frac{1}{3}(\texttt{acc}_{\text{Para}} + \texttt{acc}_{\text{Sent}} + 0.5 + 0.5 * \texttt{corr}_{\text{STS}})$. This `score` metric is also used to rank, interpret and analyse our results (see 4.3 Results section).

4.2 Experimental details

Code: We wrote the code for these several approaches entirely ourselves (Pairing prodedures, Contrastive Learning functions, Self-Distillation model, Ensembling), leveraging the loss functions from `Pytorch` and `PytorchMetricLearning` to make everything work.

Loss & Parameter Optimization: We conduct a hyperparameter search to determine the optimal setup for unsupervised contrastive learning. In particular, we look for the best p to use with the Embed Dropout layer of MinBERT to obtain positive dropout pairs, and for the best temperature τ to use with the NTXE loss. Despite computational limitations, we manage to go through different combinations methodically (see Figure 2, Bottom table) and settle on $p = 0.05$ and $\tau = 0.5$ as the optimal parameters.
As described in the approach section, we also experiment with two different methods for self-distillation, and find that using the MSE loss on logits directly brings the best results (see Figure 2, Top table).

Training Details: During finetuning and self-distillation, we run the training loop for 10 epochs, with a learning rate of $\mathtt{1e}^{-3}$ for only the linear probes, and $\mathtt{1e}^{-5}$ for the full model. During pretraining, we run the contrastive learning procedure for 10 epochs with a learning rate of $\mathtt{1e}^{-5}$.
For all of the above, we set up the training loop such that each iteration goes through one batch of each task's train dataset - with the idea of preventing the gradient update to lean to much towards one specific task at once. Batch sizes are 8 for STS, SST, and 16 for Paraphrase. Note that this means the Paraphrase dataset cannot be used to its full extent, being much larger than the other two; however, using the random shuffle option in the dataloader ensures we still go through roughly 40% of the dataset.

Ensembles: Once we have trained all our models, we select the best performing ones and experiment with ensembling. The idea is to combine models together so as to pool their strengths and offset each other's soft spots. We try two approaches: task-based ensembling, where we use one selected model per task, thus obtaining the best recorded performance for each task; and average ensembling, where we average the output probabilities (for SST, Paraphrase) or logits (for STS) of each selected model across all tasks.

4.3 Results

Results from our experiments come with some great insights about contrastive learning, self-distillation and ensembling. (See Figure 3 for Results from Experiments on the Dev Set; see Figure 4 for Final Ensembling & Results on the Test Set)

	Experiment Results | *On the Dev Set*	Full Model | Finetuning			
		Sentiment Acc	Paraphrase Acc	STSimilarity Corr	Score
	Full Model | Using Linear Head for STS Task				
	No Pre-training	0.501	0.738	0.355	**0.639**
M1	Unsupervised Contrastive PT | NTXEnt Loss | Random Reorder Pairs	0.500	0.728	0.347	**0.634**
M2	Unsupervised Contrastive PT | NTXEnt Loss | Dropout Pairs	0.501	0.747	0.372	**0.645**
M3	Semi-Supervised Contrastive PT | NTXEnt Loss | Dropout Pairs + Labels	0.518	0.750	0.367	**0.651**
M4.1	Supervised Contrastive PT | NTXEnt Loss | Labels Only	0.497	0.773	0.390	**0.655**
M5	Supervised Contrastive PT | NTXEnt Loss | Hard Negatives [External NLI Data]	0.517	0.778	0.358	**0.658**
M6.1	Supervised Contrastive PT | Triplet Loss | Hard Negatives [External NLI Data]	0.516	0.779	0.389	**0.663**
	Full Model | Using Cosine Similarity Head for STS Task				
	No Pre-training	0.499	0.708	0.550	**0.661**
M4.2	Supervised Contrastive PT | NTXEnt Loss | Labels Only	0.468	0.733	0.599	**0.667**
M6.2	Supervised Contrastive PT | Triplet Loss | Hard Negatives [External NLI Data]	0.488	0.715	0.535	**0.657**
	Full Model | with Self-Distillation + Cosine Similarity Head for STS Task				
M4.3	Supervised Contrastive PT | NTXEnt Loss | Labels Only	0.459	0.731	0.644	**0.671**

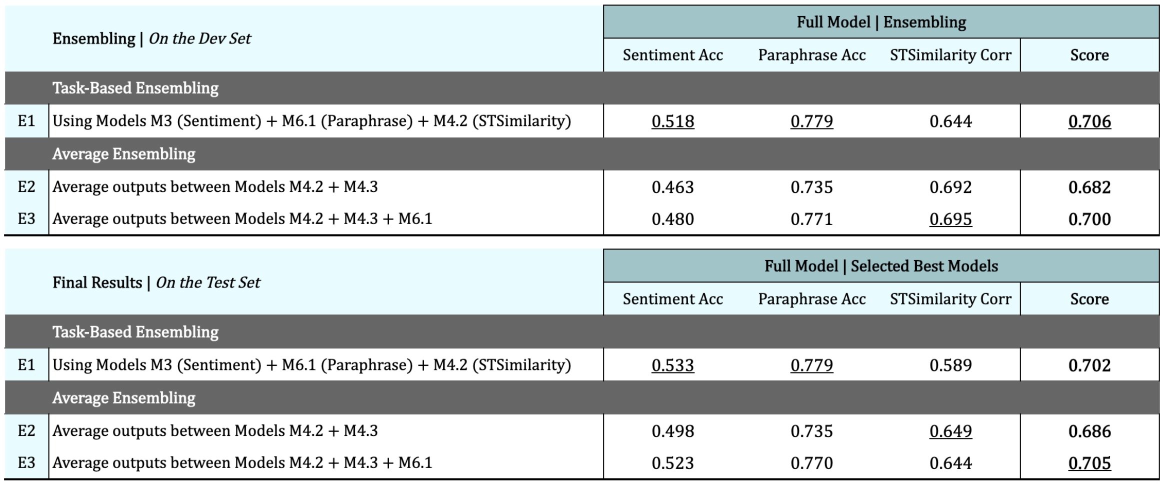
Figure 3: Result Overview from Experiments on the Dev Dataset. Unsupervised, Semi-Supervised or Supervised; with Linear Head or Cosine Similarity Head for STS; with Self-Distillation for the final model.

Supervised Pretraining is more efficient overall than Unsupervised Pretraining: As expected, using Dropout as a positive-pairing procedure worked better than a more typical approach like randomly reordering words in a sentence. However, it only led to a slightly improved score over the baseline (model M1 at 0.645 v. 0.639), while using labels, whether from our task datasets or from our external NLI dataset, showed a lot of promise, even managing to improve on all three tasks in some cases (e.g models M5 and M6.1).

Contrastive Learning does not benefit all tasks equally: Except when using the Random Reordering procedure to create positive pairs, all our Contrastive Pretraining results show an improvement in performance compared to the No Pre-training baselines. It is rarely improving all tasks at once though, with Sentiment Analysis even showing weaker performance in some cases compared to the baseline. While this is in parts due to the multitask setup, and the fact that this task is the most unique out of the three and thus does not benefit from a shared goal around similarity. However, it does challenge our hypothesis that Constrastive Pretraining would be able to improve performance beyond similarity tasks. This is particularly visible when using the Cosine Similarity head for the STS Task, thus allowing the model to improve even more on that task and lean further away from Sentiment Analysis.

Self-distillation is a performance-booster: Due to computational limitations, self-distillation was only applied to our best model (i.e. model M4.2, using supervised pretraining, task data labels for positive pairings and a cosine similarity head for the STS task) and managed to improve the overall score by 0.04 points. This confirms the idea that self-distillation is a simple step that can boost performance and act as a regularization technique, as described in the SimCLRv2 approach (Chen et al., 2020b).

Ensembling & Final Results on the Test Set: We find that ensembling - especially when selecting models with different areas of strength - systematically leads to an improved score. In particular, averaging the outputs can even lead to new best performances, acting as a consultation between different models collectively leaning towards the most certain predictions. Our best model achieves a score of 0.705 on the test set, and 0.700 on the dev set, thus a $+0.061$ over our first baseline at 0.639.



	Ensembling | *On the Dev Set*	Full Model | Ensembling			
		Sentiment Acc	Paraphrase Acc	STSimilarity Corr	Score
	Task-Based Ensembling				
E1	Using Models M3 (Sentiment) + M6.1 (Paraphrase) + M4.2 (STSimilarity)	0.518	0.779	0.644	0.706
	Average Ensembling				
E2	Average outputs between Models M4.2 + M4.3	0.463	0.735	0.692	**0.682**
E3	Average outputs between Models M4.2 + M4.3 + M6.1	0.480	0.771	0.695	**0.700**

	Final Results | *On the Test Set*	Full Model | Selected Best Models			
		Sentiment Acc	Paraphrase Acc	STSimilarity Corr	Score
	Task-Based Ensembling				
E1	Using Models M3 (Sentiment) + M6.1 (Paraphrase) + M4.2 (STSimilarity)	0.533	0.779	0.589	**0.702**
	Average Ensembling				
E2	Average outputs between Models M4.2 + M4.3	0.498	0.735	0.649	**0.686**
E3	Average outputs between Models M4.2 + M4.3 + M6.1	0.523	0.770	0.644	0.705

Figure 4: Final Result Overview on the Dev & Test Datasets.
Top Table (1) shows the impact of Self-Distillation on baseline performance, with different losses.
Bottom Table (2) shows the Contrastive Learning performance results for different pairing procedures.

5 Analysis

Our initial hypothesis was that contrastive learning techniques would improve results overall but also across all tasks. However, results show that the dynamic between tasks is complex, such that when both Paraphrase Accuracy and STS Correlation improve, it is likely to see Sentiment Accuracy decrease, sometimes even below the baseline level. As it turns out, our best model before ensembling (M4.3) is also the model with the lowest Sentiment Accuracy at 0.459 compared to 0.501 for the baseline. To understand the dynamics behind this phenomenon, we conduct a qualitative study on the SST Dev set using three models:

- the baseline model that received no pretraining (Sentiment Accuracy of 0.501),

- model M3, finetuned after a semi-supervised contrastive pretraining using the Dropout Pairing approach on the SST dataset (highest recorded Sentiment Accuracy of 0.518),
- model M4.3, finetuned and self-distilled after a supervised contrastive pretraining ignoring the SST dataset completely (lowest recorded Sentiment Accuracy of 0.459)

5.1 Alignment & Uniformity

Two properties, *alignment* and *uniformity* (Wang and Isola, 2022), were recently introduced to measure the quality of embeddings, especially in the context of contrastive learning. Alignment relates to how close features from positive pairs should be, while uniformity measures how well the embeddings are uniformly distributed on the hypersphere.
Our qualitative study suggests that as M4.3 got better in the two other tasks, it also lost nuance in Sentiment Analysis, with accuracy in the neutral category (2) falling to 0.135 from the baseline at 0.305, and accuracy in the most positive category (4) increasing to 0.739 from 0.538. On the other end, model M3 managed to maintain a stronger balance across categories, thus allowing a better accuracy overall. See Figure 5 for an overview. These insights coincide with the SimCSE findings (Gao et al., 2022) that (1) Unsupervised Contrastive Learning, and the Dropout Pairing method in particular, can effectively improve the uniformity of pre-trained embeddings thus maintaining more balance across the various sentiment levels, and (2) incorporating supervised data in the Contrastive Learning process can improve alignment, thus favoring tasks that leverage the notion of closeness, like paraphrase detection and semantic similarity. Results from our experiments also hint that using Cosine Similarity can also skew embeddings towards alignment rather than uniformity.

5.2 Ambiguity in Sentiment Analysis

While language in general is an ambiguous field (Jusoh, 2018), one could easily argue that Sentiment Analysis is one of the trickiest tasks in NLP, as efficiently navigating challenges like sarcasm, irony, context-specific jokes or insights, is something even humans struggle with. In the SST dataset (Socher et al., 2013), inputs consist of single sentences extracted from movie reviews from real users, which makes them even more prone to sarcasm, colloquialisms, and intricate references. On top of this, they are presented with barely any context about the film, genre or era, which makes it even more difficult. As an example, the review "Cool?" had a ground truth sentiment of 3, meaning somewhat positive, but it doesn't take much imagination to possibly hear sarcasm and derision in this one-word review, thus making it negative. Another review - "Do not see this film." - was rated 0 by all three models, with a questionable ground truth of 1.
Given our main criteria of success was the overall score, it is thus not surprising that our best models leaned towards improving the less subjective paraphrase detection and STS tasks first.

6 Conclusion

In this work, we investigated the efficacy of contrastive learning and self-distillation techniques for generating high-quality sentence embeddings with minimal reliance on labeled data. Our experiments demonstrated that supervised pre-training is more efficient than unsupervised pre-training, with the latter showing somewhat limited improvements. We also found that contrastive learning, while beneficial overall, did not enhance all tasks equally, highlighting the complexity of multitask setups. However, we noted that the application of self-distillation significantly boosted performance, confirming its role as a powerful regularization and enhancement technique. Our results indicated that integrating these methods can lead to substantial improvements, especially when combined with ensembling strategies. They also unveiled the nuanced dynamics between different NLP tasks, emphasizing the need for balanced approaches when using a multitask framework. Future work could explore more sophisticated techniques for generating positive pairs, address the observed task-specific discrepancies, and further refine self-distillation processes. Moreover, extending this framework to other NLP tasks and larger datasets could provide deeper insights and broader applicability.

7 Ethics Statement

Ethical concerns and potential societal risks surrounding this project mostly relate to the generation of biased embeddings and, more specifically, finding out that our approach perpetuates or, even

On the Dev Set	Overall		Sentiment Accuracy across Sentiments (Ground Truth)				
	Score	Sentiment Acc	0	1	2	3	4
Baseline	0.639	0.501	0.353	0.667	0.305	0.542	0.538
Model M3	0.651	0.518	0.633	0.519	0.328	0.616	0.515
Model M4.3	0.671	0.459	0.309	0.616	0.135	0.470	0.739

Examples

Sentiment = 0 | "Try as I may , I can't think of a single good reason to see this movie , even though everyone in my group extemporaneously shouted , ` Thank you ! '" Rated 2 by M4.3, Rated 0 by Baseline, M3

Sentiment = 2 | "Ah yes , and then there's the music ..." Rated 3 by M4.3, Rated 2 by Baseline, M3

Sentiment = 3 | "Cool?" Rated 3 by M4.3, Rated 2 by Baseline, M3

Sentiment = 1 | "Do not see this film." Rated 0 by all three models

Sentiment = 4 | "The performances take the movie to a higher level." Rated 3 by all three models

Figure 5: Qualitative Results for the Sentiment Analysis task. Comparing our Baseline model with the model obtaining the highest Sentiment Accuracy on the Dev set (Model M3) and the one obtaining the lowest (Model M4.3).

worse, amplifies already existing biases encoded in the BERT model. This could potentially lead to unfair treatment or discrimination in applications such as sentiment analysis.
Unintended usage is also a key risk, with organizations/people leveraging our embeddings to make decisions in the context of sensitive tasks such as employment, justice, and medical diagnostics (e.g. leveraging embeddings to assess and rate resumes, cover letters, medical notes). As these models are made widely available, the risk becomes larger that they end up being used the wrong way in real-life settings, thus potentially having catastrophic consequences.

There are ways we can think of to mitigate these risks, including starting with *measuring* bias in language embeddings as we cannot fight what we don't know. Once we have properly defined a bias metric, we can answer questions such as: How biased are the initial BERT embeddings, and how biased are the embeddings we generate with our approach? Are we moving in the right direction with our model? What types of bias remain? Once we have a clearer picture of the situation, it can be useful to assess the impact of these on each of the intended tasks and think about (for example) post-processing correctional steps.
Additionally, educating the world about the limitations of AI would be a crucial step to prevent wrong usage. Based on society's reactions and excitement towards LLMs, it is clear AI is seen by some as this objective solution to all our problems. As researchers and analysts in the domain of AI, it is also on us to make sure we educate society's decision-makers and leaders, but also the broader public (starting with the next generation, potentially even talking about AI at school around the world).

References

Fredrik Carlsson, Amaru Cuba Gyllensten, Evangelia Gogoulou, Erik Ylipää Hellqvist, and Magnus Sahlgren. 2021. Semantic re-tuning with contrastive tension. In *International Conference on Learning Representations*.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. 2010. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(3).

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020a. A simple framework for contrastive learning of visual representations.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. 2020b. Big self-supervised models are strong semi-supervised learners.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. 2021. DeCLUTR: Deep contrastive learning for unsupervised textual representations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 879–895, Online. Association for Computational Linguistics.

Michael Günther, Louis Milliken, Jonathan Geuter, Georgios Mastrapas, Bo Wang, and Han Xiao. 2023. Jina embeddings: A novel set of high-performance sentence embedding models.

Shaidah Jusoh. 2018. A study on nlp applications and ambiguity problems. *Journal of Theoretical & Applied Information Technology*, 96(6).

Yu Meng, Chenyan Xiong, Payal Bajaj, Saurabh Tiwary, Paul Bennett, Jiawei Han, and Xia Song. 2021. Coco-lm: Correcting and contrasting text sequences for language model pretraining.

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

Tongzhou Wang and Phillip Isola. 2022. Understanding contrastive representation learning through alignment and uniformity on the hypersphere.

Yan Zhang, Ruidan He, Zuozhu Liu, Kwan Hui Lim, and Lidong Bing. 2020. An unsupervised sentence embedding method by mutual information maximization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1601–1610, Online. Association for Computational Linguistics.