Minhbert

Stanford CS224N Default Project

Minh Vu
Department of Computer Science
Stanford University
minhavu@stanford.edu

Abstract

This project ventures into the intricate realm of natural language processing (NLP), implementing minBERT—a refined adaptation of the BERT model. Conducted by Minh Vu under the mentorship of Yuan Gao, the initiative customizes minBERT to three NLP tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. The development of minBERT is marked by a series of deliberate enhancements and exhaustive testing, transforming it from a foundational multitask framework into a model fine-tuned for specific tasks.

The culmination of these efforts is reflected in the model's performance on various benchmarks, achieving a sentiment analysis test accuracy of 0.518, a paraphrase detection accuracy of 0.851, and a semantic textual similarity (STS) correlation of 0.818. Collectively, these results yield an aggregate score of 0.759, positioning our project at 44th on the leaderboard, a competitive score for a solo team.

1 Introduction

In recent years, the field of natural language processing (NLP) has witnessed significant advancements, largely fueled by the introduction of deep learning techniques and particularly the development of transformer-based models. Among these, BERT (Bidirectional Encoder Representations from Transformers) has emerged as a groundbreaking architecture, demonstrating unparalleled proficiency in capturing the intricacies of language through its deep bidirectional understanding. This project aims to explore the capabilities of BERT by implementing a scaled-down version, minBERT, to tackle a variety of NLP tasks. Specifically, we focus on developing a multitask classifier that addresses sentiment analysis, paraphrase detection, and semantic textual similarity.

The extensive computational resources required by BERT pose limitations for its widespread application, which motivates the development of minBERT. Through this project, we first aim to implement a working version of minBERT that meets baseline metrics for sentiment analysis. Subsequently, we extend its application to more complex tasks such as paraphrase detection and semantic textual similarity. Finally, we delve into experimenting with various methods to refine minBERT's performance across these tasks. Our exploratory efforts encompass a range of techniques including cosine similarity for assessing textual similarity, class weights for balancing task influence, and the implementation of task-specific architectural enhancement. This exploration not only underscores the model's potential in a multifaceted NLP landscape but also contributes to the ongoing discourse on making advanced AI models more accessible and efficient.

2 Related Work

2.1 Attention Is All You Need

"Attention Is All You Need" by Vaswani et al., introduces the Transformer, a novel architecture that fundamentally alters the landscape of sequence transduction models by exclusively relying on attention mechanisms, sidestepping the need for recurrent or convolutional neural networks. This paper marks a significant shift towards utilizing attention mechanisms over traditional recurrent or convolutional neural networks for sequence transduction tasks, most notably in machine translation. The Transformer model, characterized by its reliance on self-attention to process sequences in parallel, eliminates the need for recurrence, showcasing an efficient alternative for capturing long-range dependencies within text.

2.2 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Devlin et al. introduces BERT (Bidirectional Encoder Representations from Transformers), a new method for pre-training language representations. Unlike previous models that pre-train on a unidirectional language model, BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right contexts in all layers. This allows the pre-trained BERT model to be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial modifications to the model architecture.

BERT's key innovation lies in its ability to train a language model that can understand the context of a word based on all of its surroundings (left and right of the word). The paper demonstrates BERT's superiority through its performance on eleven natural language processing tasks.

2.3 Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics

In "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics", Kendall, Gal, and Cipolla propose a novel approach to multi-task deep learning that dynamically weights multiple loss functions by considering the homoscedastic uncertainty of each task. This method enables the simultaneous learning of various quantities with different units or scales across both classification and tasks. By interpreting homoscedastic uncertainty as task-dependent weighting, their model optimally balances the contribution of each task's loss to the overall objective, leading to superior performance compared to training separate models for each task or manually tuning loss weights.

3 Approach

3.1 Implementation of minBERT

We first implemented minBERT, a streamlined variant of the original BERT model as outlined in the project handout. At the core of the implementation was the multi-head attention mechanism, formulated as $MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$, with each head processing inputs through scaled dot-product attention. This mechanism was crucial in enabling the model to capture various contextual nuances by attending to different positions within the input sequences. By integrating this with the transformer layer's architecture—which includes residual connections, layer normalization, and position-wise feed-forward networks—we were able to construct a minBERT implementation.

3.2 Baseline Model

As a baseline model for multitask learning, we implemented the `MultitaskBERT` class, which encapsulates a BERT-based architecture adapted for sentiment classification, paraphrase detection, and semantic textual similarity.

- **Sentiment Classification:** The baseline model employs a linear layer that processes BERT's [CLS] token embedding, mapping it to the number of sentiment classes. Cross-entropy loss is used to assess the prediction accuracy against multiple sentiment labels.
- **Paraphrase Detection:** A linear layer that merges embeddings from two input sentences is utilized to produce logits. These are evaluated against binary labels using binary cross-entropy loss with logits.

- **Semantic Textual Similarity:** This task is approached with a linear layer for regression, taking embeddings from two input sentences to compute a continuous value indicative of the degree of similarity between the sentence pairs. Mean squared error loss is applied to evaluate the model's performance.

For sentiment classification, dropout is applied subsequent to retrieving the embeddings. For paraphrase detection and similarity textual similarity, dropout is applied after concatenating the embeddings.

Within the train_multitask function, the model adopts a sequential training strategy, processing each task one after the other within an epoch.

3.3 Final Model Architecture

Task-Specific Linear Layers:

- For sentiment classification, an intermediate linear layer maps BERT's output to a 768-dimensional space, followed by normalization, dropout and another linear layer that classifies the sentiment into one of five classes.
- Paraphrase detection utilizes a concatenation of embeddings from two sentences, passed through an intermediate layer of size 768, normalization and dropout, culminating in a binary classification via a linear layer.
- For semantic textual similarity, we explored various methods for comparing sentence pair embeddings, including absolute difference, cosine similarity, and a combination of concatenating cosine similarity with some form of difference. Ultimately, taking the squared difference between embeddings was found to yield the best results. This is followed by dropout and regression via a linear layer

An activation function, specifically ReLU, is applied after the intermediate layers in paraphrase detection and similarity prediction.

Pooling Strategies Across Tasks:

The final model employs distinct pooling strategies tailored to the requirements of each task it addresses:

- **Sentiment Classification:** Utilizes the [CLS] token embedding from BERT, leveraging it as a comprehensive representation for analyzing overall sentence sentiment.
- **Paraphrase Detection and Semantic Textual Similarity:** These tasks adopt mean pooling across all token embeddings to incorporate broader contextual information necessary for assessing sentence relationships and similarity. Mean pooling allows the model to consider the contribution of each token in the sentence.

3.4 Sequential and Joint Multitask Learning Strategies

We explored two multitask training strategies: Sequential and Joint Multitask Learning.

3.4.1 Sequential Multitask Learning

Sequential Multitask Learning sequentially processes each task, dedicating a portion of each training epoch to individually updating the model based on the specific task's loss. This method's primary advantage is its ability to focus deeply on one task at a time, potentially leading to better task-specific performance due to the targeted optimization. Sequential Multitask Learning is more computationally efficient within each epoch, as it considers each example across the three datasets only once. However, the isolation of tasks might prevent the model from effectively capturing and leveraging cross-task insights during training, possibly limiting the development of robust generalized representations. Algorithm 1 details Sequential Multitask Learning.

Algorithm 1 Sequential Multitask Learning

```
1:  for each $epoch$ do
2:      $epochLoss \leftarrow 0$
3:      for $sstBatch$ in $sstDataLoader$ do
4:          optimizer.zero_grad()
5:          $loss \leftarrow ComputeSSTLoss(sstBatch, model)$
6:          $weightedLoss \leftarrow ...$          ▷ Weight loss by uncertainty or inverse to task performance
7:          weightedLoss.backward()
8:          optimizer.step()
9:          $epochLoss \leftarrow epochLoss + weightedLoss.item()$
10:     end for
11:     for $paraBatch$ in $paraDataLoader$ do
12:         ...                              ▷ Similar logic to training for semantic prediction
13:     end for
14:     for $stsBatch$ in $stsBatch$ do
15:         ...                              ▷ Similar logic to training for semantic prediction
16:     end for
17:     ...                                  ▷ Additional training logic and evaluation
```

3.4.2 Joint Multitask Learning

Joint Multitask Learning, conversely, integrates training across all tasks within the same iteration by aggregating their losses. This is done by iterating through batches in the paraphrase training data while simultaneously cycling through the SST and STS data. This approach's most significant advantage is its ability to promote shared representation learning, potentially enhancing model generalizability across tasks. By training on multiple tasks simultaneously, the model can identify and exploit common features, which might lead to improved performance on individual tasks and a more cohesive understanding of the language. However, due to cycling through the SST and STS data rather than iterating through them once, Joint Multitask Learning is less efficient than Sequential Multitask Learning. Our model uses Joint Multitask Learning after we found that Joint Multitask Learning performed better than Sequential Multitask Learning. Algorithm 2 details Joint Multitask Learning.

Algorithm 2 Joint Multitask Learning

```
    for each $epoch$ do
2:      $epochLoss \leftarrow 0$
        Initialize data loaders $sstLoader, stsLoader$
4:      for paraBatch in paraDataLoader do
            optimizer.zero_grad()
6:          $sstBatch \leftarrow$ next batch from $sstLoader$
            $stsBatch \leftarrow$ next batch from $stsLoader$
8:          $sstLoss \leftarrow ComputeSSTLoss(sstBatch, model)$
            $paraLoss \leftarrow ComputeParaLoss(paraBatch, model)$
10:         $stsLoss \leftarrow ComputeSTSLoss(stsBatch, model)$
            $weightedSSTLoss \leftarrow ...$   ▷ Weight loss by uncertainty or inverse to task performance
12:         $weightedParaLoss \leftarrow ...$  ▷ Weight loss by uncertainty or inverse to task performance
            $weightedSTSLoss \leftarrow ...$   ▷ Weight loss by uncertainty or inverse to task performance
14:         $totalLoss \leftarrow weightedSSTLoss + weightedParaLoss + weightedSTSLoss$
            $totalLoss.backward()$
16:         optimizer.step()
            $epochLoss \leftarrow epochLoss + totalLoss.item()$
18:     end for
        Adjust task weights based on performance
20:     ...                                  ▷ Additional training logic and evaluation
```

3.5 Task Weighting

3.5.1 Uncertainty Weighting

One method of task weighting we explored was uncertainty weighting in accordance with the approach proposed by Kendall, Gal, and Cipolla. Task-specific uncertainty is quantified by log variance terms ($\log \sigma^2$), each initialized to zero, which are learned during the training process. These terms not only scale the contribution of each task's loss to the total loss function but also act as a regularization mechanism to promote an optimal balance of prediction confidence across tasks. The model's adjusted task-specific loss is calculated as follows:

$$AdjustedLoss = \frac{L}{2\sigma^2} + \frac{1}{2}\log \sigma^2 \tag{1}$$

In this equation, L denotes the original task-specific loss, a measure of the model's prediction error for the task. The term $2\sigma^2$ represents the task's variance and is related to the model's confidence in its task-specific predictions; as variance increases, indicating greater uncertainty, the loss for the task is down-weighted. The term $\frac{1}{2}\log \sigma^2$ is the regularization term that penalizes high uncertainty, thereby preventing the model from assigning too much weight to tasks where it is less certain. This balance enables the model to focus learning where its certainty is higher, thus optimizing overall performance.

3.5.2 Dynamic Performance-Based Task Weighting

We devised an alternative weighting scheme that dynamically adjusts the contribution of each task to the overall learning process based on task performance. This approach seeks to allocate more resources and attention to tasks where the model underperforms, thereby promoting a more balanced improvement across tasks.

Central to this weighting is the computation of task weights inversely proportional to each task's performance metrics. For the sentiment analysis and paraphrase detection tasks, performance is directly measured using accuracy. For the semantic textual similarity (STS) task, we utilize the Pearson correlation coefficient as a measure of performance. We first adjust the STS correlation score using the formula:

$$\text{Adjusted Correlation}_{\text{Similarity}} = \frac{\text{Correlation}_{\text{Similarity}} + 1}{2}$$

We determine the weight for each task as follows:

$$\text{Weight Task} = \frac{1}{\text{Task Performance Metric} + \epsilon}$$

After computing the weights for each task, we normalize these weights to ensure they sum to 1. Empirically we saw minor improvements in scores across the three tasks when using performance-based weighting over uncertainty, so our final model employs this weighting scheme.

4 Experiments

4.1 Data

The SST (https://nlp.stanford.edu/sentiment/treebank.html) dataset, consisting of 11,855 sentences from movie reviews, provides a basis for sentiment analysis with its five-level sentiment labels (0, 1, 2, 3, 4). The data is split into 8,544 training, 1,101 development, and 2,210 test examples.

We used a subset of The Quora dataset (https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs) consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. We used a subset of this dataset with 202,152 sentence pairs segmented into 141,506 training, 20,215 development, and 40,431 test examples with labels indicating whether particular instances are paraphrases of one another.

Lastly, the SemEval STS dataset(https://aclanthology.org/S13-1004.pdf), consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). It is split into 6,041 training, 864 development, and 1,726 test examples.

4.2 Evaluation method

To evaluate our model's effectiveness in we use accuracy metrics and the Pearson correlation coefficient. Accuracy metrics gauge the model's ability to classify sentiments and detect paraphrases, while the Pearson coefficient measures its performance in assessing textual similarity.

We also utilized a unified metric combining accuracy to evaluate overall performance. This metric, averaging the two accuracies and the adjusted Pearson correlation (adjusted to a [0, 1] scale), ensures balanced improvement across tasks.

$$\text{Unified Metric} = \frac{1}{3}\left(\text{Accuracy}_{\text{Sentiment}} + \text{Accuracy}_{\text{Paraphrase}} + \frac{(\text{Correlation}_{\text{Similarity}} + 1)}{2}\right)$$

4.3 Experimental details

In our experiments we ran the model with –pretrain with a learning rate of 1e-3 for 5 epochs as a way to rapidly prototype different approaches and extensions. For finetuning, we would use a learning rate of 1e-5 over 10 epochs. We saw that that peak performance was generally achieved around the 6th or 7th epoch, with performance leveling off or slightly declining thereafter. This pattern suggested the onset of overtraining, characterized by continuous improvements in training scores unaccompanied by corresponding gains in development scores.

We also explored ptimizing regularization through dropout, where a rate of 0.3 was found to offer the best compromise between preventing overfitting and maintaining sufficient model complexity for effective learning. Another aspect of our experimental setup was determining the ideal dimensionality for the intermediate layers of our model. Empirical exploration led us to settle on a dimensionality of 768.

For our best performing model, we adjusted our final round of finetuning to 12 epochs. This adjustment was made with the anticipation that a prolonged training period could potentially unveil additional performance benefits, however, the best performance was achieved after 6 epochs.

4.4 Results

Results from running `python3 multitask_classifier.py -option finetune -lr 1e-5 -epochs 12 -use_gpu`

Method	**Test SST**	**Test Paraphrase**	**Test STS**	**Test Unified**
Final Model Architecture + Performance-Based Weighting + Joint Multitask Learning + Square Difference	0.518	0.851	0.818	0.759
Method	**Dev. SST**	**Dev. Paraphrase**	**Dev. STS**	**Dev. Unified**
Final Model Architecture + Performance-Based Weighting + Joint Multitask Learning + Square Difference	0.487	0.853	0.847	0.755
Method	**Train SST**	**Train Paraphrase**	**Train STS**	**Train Unified**
Final Model Architecture + Performance-Based Weighting + Joint Multitask Learning + Square Difference	0.999	0.971	0.930	0.978

Table 1: Best Finetune Performance on Test, Dev, and Train Sets

Results from running `python3 multitask_classifier.py -option pretrain -lr 1e-5 -epochs 5 -use_gpu` across different approaches

Method	Dev. SST	Dev. Paraphrase	Dev. STS	Dev. Unified
Baseline Model	0.470	0.699	0.314	0.608
Final Model Architecture + Uncertainty Weighting + Sequential Multitask Learning + Square Difference	0.469	0.727	0.585	0.662
Final Model Architecture + Uncertainty Weighting + Sequential Multitask Learning + Absolute Difference	0.469	0.727	0.598	0.665
Final Model Architecture + Uncertainty Weighting + Joint Multitask Learning + Square Difference	0.473	0.783	0.632	0.690
Final Model Architecture + Uncertainty Weighting + Sequential Multitask Learning + Absolute Difference	0.475	0.78	0.626	0.689
Final Model Architecture + Performance-Based Weighting + Sequential Multitask Learning + Square Difference	0.450	0.785	0.598	0.678
Final Model Architecture + Performance-Based Weighting + Sequential Multitask Learning + Absolute Difference	0.450	0.781	0.610	0.678
Final Model Architecture + Performance-Based Weighting + Joint Multitask Learning + Square Difference	**0.479**	**0.779**	**0.632**	**0.691**
Final Model Architecture + Performance-Based Weighting + Joint Multitask Learning + Absolute Difference	0.476	0.779	0.629	0.689

Table 2: Pretrain Performance of Different Approaches on Development Sets

Our model's position—55th out of 162 on the development leaderboard and 44th out of 138 on the test leaderboard—indicates strong performance. The test performance, in particular, highlights the effectiveness of the optimization strategies we've implemented, including a performance-based weighting strategy in conjunction with joint multitask learning.

This strategic choice in optimization appears to have been crucial in enhancing the model's generalization capabilities, as shown by our model's close proximity to the benchmark scores set by the leading submission, "BBB." The differences in individual scores are minor: 0.038 points for sentiment classification accuracy, 0.044 points for paraphrase detection accuracy, and 0.069 points for semantic textual similarity correlation. Such a tight margin suggests that the choice of task-specific modifications, notably the use of a square difference in semantic textual similarity (STS) tasks, has been significantly beneficial.

Our model's consistent pretrain performance across diverse weighting schemes was surprising. The uniformity across the different weighting schemes did not yield the varied learning outcomes anticipated, prompting us to consider a reevaluation of the weighting schemes' calibration and to explore other approaches.

An unexpected observation was the similarity in Sentiment Analysis Task (SST) accuracy between our model and the baseline model. Initially, this resemblance in performance was surprising, considering the extensive efforts and strategic implementations we made to enhance our model's capabilities beyond the baseline standards. However, upon closer examination of the leaderboard standings and the performances of other teams, we discovered that this phenomenon was not unique to our model. Many teams reported SST accuracy scores within a similar range, indicating a broader trend in the competition.

The stark contrast between our model's sentiment analysis task (SST) training accuracy of 0.999 and its development and test accuracies of 0.487 and 0.518, respectively, indicates a clear case of overfitting. While overfitting has been prominently identified in our model's performance on the sentiment analysis task (SST), it emerges as a less pronounced concern for the other tasks—paraphrase detection and semantic textual similarity (STS)—given their considerably better development and test scores. The disparity between the training accuracy for SST and its corresponding development and test accuracies starkly highlights overfitting; however, the more balanced accuracies observed in paraphrase detection and STS indicate that our model has a stronger ability to generalize its learned patterns to unseen data in these areas.

4.5 Analysis

Task	Example	Ground Truth	Prediction	Analysis
Sentence similarity	if Steven Soderbergh's 'Solaris' is a failure it is a glorious failure.	4	0	The model entirely missed the nuanced positive spin on the word 'failure', highlighting a gap in recognizing contextual sentiment.
Paraphrase Detection	what is the physical meaning of operating voltage detectors ? what is the physical meaning of operating voltage of detectore ?	1	0	The model incorrectly classified the sentences as non-paraphrases, due to a minor misspelling.
Sentence Similarity	you guys are making this all waaaaay too complicated. you are making this too complicated.	2.608	5.0	The model underestimated the similarity between the sentences, likely due to the informal language and length difference ("waaaaay too complicated" vs. "too complicated"). This suggests a sensitivity to sentence structure and limitations in handling informal language variations.

Table 3: Analysis of examples

In analyzing the performance of our model across various tasks, specific issues emerged that shed light on its strengths and limitations. Notably, in the task of sentence similarity, our model demonstrated a tendency to misinterpret the intensity of sentiment conveyed in sentences. In the example "every nanosecond of the new guy reminds you that you could be doing something else far more pleasurable," the model's prediction significantly deviated from the ground truth, indicating an overestimation of positivity. This discrepancy points to a specific challenge: the model's potentially limited capacity to understand contextual language cues such as tone and implication. Furthermore, our model exhibited sensitivity to informal language and lexical variations, which affected its predictions in certain instances. Examples with informal language or minor misspellings led to deviations in predictions, highlighting the model's reliance on surface-level features and its struggle to effectively handle linguistic variations.

5 Conclusion

The implementation of our model and the extensions we explored project successfully demonstrates the application of a scaled-down BERT model to address complex NLP tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity. Through experimentation, we have learned the value of a balanced dataset and the challenges that arise from overfitting. Our model shows promise in its ability to easily outperform baseline metrics and achieve competitive standings on leaderboards, it also uncovers critical areas for enhancement. Particularly, the challenges of overfitting in sentiment analysis, the nuanced understanding of context in sentence similarity, and handling lexical variations in informal language point to the intricacies of language that still pose considerable hurdles for even advanced models like minBERT.

For future improvements, we have a number of extensions to explore. First, incorporating a more diverse and expansive dataset that includes a variety of linguistic nuances could improve the model's generalizability and robustness, especially in understanding informal and contextually rich language. Second, experimenting with different architectures or attention mechanisms may provide new insights into capturing subtle linguistic cues. Finally, delving into and incorporating advanced training methodologies such as transfer learning and domain adaptation could further enhance the model's ability to generalize across various tasks while also mitigating the risk of overfitting.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 2017.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[3] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. *arXiv preprint arXiv:1705.07115*, 2018.

[4] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 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, 2013.

[5] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *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, 2013.

A Appendix

Method	**Dev. SST**	**Dev. Paraphrase**	**Dev. STS**	**Dev. Unified**
Final Model Architecture + No Task Weighting + Sequential Multitask Learning + Square Difference	0.457	0.778	0.603	0.678
Final Model Architecture + No Task Weighting + Sequential Multitask Learning + Absolute Difference	0.457	0.778	0.612	0.680
Final Model Architecture + No Task Weighting + Joint Multitask Learning + Square Difference	0.478	0.779	0.631	0.690
Final Model Architecture + No Task Weighting + Joint Multitask Learning + Absolute Difference	0.480	0.776	0.626	0.689

Table 4: Pretrain Performance With No Task Weighting on Development Sets

Task	Example	Ground Truth	Prediction	Analysis
Sentence similarity	every nanosecond of the new guy reminds you that you could be doing something else far more pleasurable.	1	4	The model's prediction deviates significantly from the ground truth, likely due to misinterpreting the intensity of sentiment conveyed in the sentence. A difference of 3 suggests the model may have recognized some sentiment but exaggerated its positivity.
Paraphrase Detection	what are the best ways to improve English ? how can i improve my english vocabulary ?	0	1	The model misclassified the sentences as paraphrases, and interpreted the similarity in semantic meaning as indicative of paraphrasing. This misinterpretation highlights the model's struggle to capture subtle contextual nuances, leading to an incorrect prediction.
Sentence Similarity	it's also a matter of taste. it's definitely just a matter of preference.	2.5175	5.0	The model underestimated similarity, possibly due to lexical variation ("taste" vs. "preference"). This highlights the model's reliance on surface-level features and its struggle to grasp deeper semantic meaning.

Table 5: Additional Analysis of examples