minBERT and Downstream Tasks Optimization with Disentangled Attention

Stanford CS224N Default Project

Sean Bai
Department of Computer Science
Stanford University
seanbai@stanford.edu

Jeremy Linfield
Department of Computer Science
Stanford University
jeremyml@stanford.edu

Abstract

In this project, we investigate the effect of a disentangled attention on three downstream NLP tasks: sentiment analysis, paraphrase detection, and semantic text similarity. We implement the disentangled attention mechanism as well as enhanced mask decoder, proposed by He and colleagues in their 2021 paper "DeBERTA: Decoding-enhanced BERT with Disentangled Attention" (He et al., 2020). Disentangled attention introduces a novel approach in which each word is depicted through a position vector and content vector. Attention weights between words are then computed using disentangled matrices on their content and relative positions. The enhanced mask decoder integrates a word's absolute position within the decoding layer to predict masked tokens during model pre-training. After implementation, we saw significantly improved performance on sentiment analysis and semantic textual similarity, and marginally worse performance on paraphrase detection.

1 Key Information to include

- Mentor: Hamza El Boudali
- Team contributions: Work was split up evenly. Jeremy implemented classifier.py, multitask_classifier.py, optimizer.py, helped debug disentangled attention mechanism, showed how enhanced mask decoder should be implemented, and contributed heavily to the write-up. Sean implemented bert.py, implemented both the disentangled attention mechanism and enhanced mask decoder for DeBERTa, and contributed heavily to the write up.

2 Introduction

In recent years, natural language processing (NLP) has witnessed remarkable advancements, particularly with the advent of transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). BERT, introduced by Devlin and colleagues, has revolutionized the field by utilizing both the left and right context of unlabeled text in all layers of pretraining.

Motivated by the success of BERT, our study focuses on further enhancing the capabilities of multitask learning using the DeBERTa (Decoding-enhanced BERT with disentangled attention) algorithm. DeBERTa, an extension of BERT proposed by (He et al., 2020), refines the pretraining strategy and introduces a novel self-attention mechanism to more effectively capture the relative position of words in their surrounding contexts, as well as utilizing their absolute positions, in order to better represent the meaning and effect of words.

In this paper, we investigate the effectiveness of multitask learning with DeBERTa across

three fundamental downstream tasks in NLP: sentiment analysis (the task of determining the feeling expressed in sentence, which can range from negative to neutral to positive), paraphrase detection (identifying whether two sentences convey similar meanings), and semantic textual similarity (quantify the degree of similarity between two texts). These tasks are pivotal in understanding and processing natural language, with applications spanning sentiment understanding, information retrieval, and text summarization.

By evaluating DeBERTa's performance on these tasks, we aim to provide insights into the model's ability to capture nuanced linguistic features and generalize across diverse datasets.

In the subsequent sections of this paper, we reference related work, describe our methodologies, present our experimental setup, analyze the results obtained, and discuss implications for future research and applications in the field of NLP.

3 Related Work

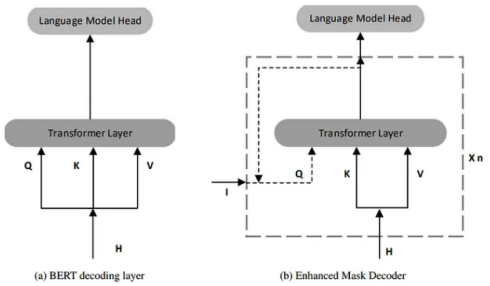

Figure 1: BERT vs. DeBERTa decoding layer (He et al., 2020)

The DeBERTa algorithm elaborates on years of Natural Language Processing (NLP) research. The DeBERTa model builds off of the original BERT paper (Devlin et al., 2019) and later RoBERTa (Liu et al., 2019) by using a similar algorithm, with some clear differences. The DeBERTa model incorporates a disentangled attention mechanism to compute attention weights, whereas BERT and RoBERTa use bidirectional self-attention. Additionally, BERT and RoBERTa incorporate absolute word positions in the input layer, while DeBERTa uses absolute word positions after all of the Transformer layers, but before the softmax layer (enhanced mask decoder), as shown in Figure 1. The DeBERTa model builds upon papers that have already shown the effectiveness of relative position encoding (Shaw et al. (2018); Huang et al. (2018)) with its utilization of the "position-to-content" term, which helps fully capture the attention weight of a word pair. Using SuperGLUE (Wang et al., 2020) as a baseline, the DeBERTa model outperformed human performance.

4 Approaches

4.1 Base BERT

As described in the default project handout, each of the three tasks had an independent basic implementation that used the embeddings from an already implemented BERT model:

Sentiment classification: We took sentence embeddings for a batch of sentences, passed them through a linear layer, and returned five logits for each sentence. We measured loss through cross-entropy loss.

Paraphrase detection: Given a batch of pairs of sentences, we produced sentence embeddings for each sentence. We then separately passed each sentence's embeddings through a linear layer. Next, we performed cosine similarity on the outputs of the separate linear layers. On this result, we then applied the ReLU activation function, returning a logit in the range of [0, 1]. We measured loss through binary cross-entropy loss.

Semantic Textual Similarity This algorithm was similar to the paraphrase detection algorithm. The only differences were that the logit (after the cosine similarity function) was scaled by a factor of 5, returning a logit in the range of [0, 5], and that loss was measured using the mean square error (MSE) or L2 loss.

While we considered applying dropout before the linear layer in each task, we ultimately decided not to after noticing decreased performance in sentiment classification with dropout applied.

4.2 Disentangled Attention Mechanism with Enhanced Mask Decoder

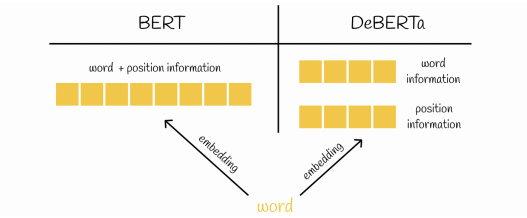

Figure 2: Word Representations in BERT vs. DeBERTa (Efimov, 2023)

We sought to improve upon the BERT baselines by introducing a disentangled attention mechanism. The key difference of disentangled attention is the integration of relative positional information as independent embeddings, as shown in Figure 2. The default BERT attention mechanism combines content and positional information through the addition of word and absolute positional vectors.

$$Q_c = HW_{q,c}, \quad K_c = HW_{k,c}, \quad V_c = HW_{v,c}, \quad Q_r = PW_{q,r}, \quad K_r = PW_{k,r}$$

$$\tilde{A}_{i,j} = \underbrace{Q_i^c K_j^{cT}}_{content-to-content} + \underbrace{Q_i^c K_{\delta^r_{(i,j)}}^{T}}_{content-to-position} + \underbrace{Q_j^c K_{\delta^r_{(j,i)}}^{T}}_{position-to-content}$$

$$\mathrm{H}_o = \mathrm{softmax}\left(\frac{\tilde{A}}{\sqrt{3d}}\right) V_c$$

The pseudocode for the disentangled attention mechanism can be found in the Appendix is Figure 11.

Shifting to disentangled attention required key modifications of our base BERT implementation. First, we removed the absolute positional embedding from the embedding layer. We calculated the relative positional encodings by utilizing log bucketing:

$$\delta_{(i,j)} = \begin{cases} 0 & \text{for } i \neq j \leq k \\ 2k-1 & \text{for } i \neq j \geq k \\ i-j+k & \text{others} \end{cases}$$

By grouping relative positions into buckets, the model is able to focus on local context with more granularity. Naturally, immediate context in language tends to be more important than distant context. For our log bucket implementation, we took source code from the DeBERTa repo, creating a log bucket dictionary, mapping relative positions to log bucket indices. After clamping each relative positions tensor to the specified maximum position range, we mapped each relative position to its corresponding log-bucket index using our created log bucket dictionary.

Using these relative position embeddings, we then constructed a relative position embedding layer (P) and new projection matrices $W_{q,r}$, $W_{k,r}$ in order to generate separate key and query vectors corresponding to relative position. From there, we computed the dot product of content-to-content, content-to-position, and position-to-content, summing the results. The resulting attention scores are then passed through the softmax function as in the original implementation, with the exception of our scale reduction being three times larger to account for the increased dimensionality.

For our implementation of the enhanced mask decoder, we incorporated the absolute positional embeddings right before the softmax layer, which is where the model utilizes the combined contextual embeddings of word contents and positions to decode the masked words. Specifcally, we added the aboslute positions to the contextual embeddings in the decoding layer before the softmax layer. However, it did not perform as well as we expected, and we ultimately decided to reintroduce absolute positional embeddings in a later iteration.

5 Experiments

5.1 Data

The datasets used for the minBERT investigation (sentiment analysis) are the SST dataset and the CFIMBD dataset. The SST dataset consists of 11,855 single sentences from movie reviews, extracted from movie reviews. These sentences were parsed to produce 215,154 unique phrases. Each phrase has a label of **negative**, **somewhat negative**, **neutral**, **somewhat positive**, or **positive** The CFIMBD dataset consists of 2,434 highly polar movie reviews, each with a binary label of **negative** or **positive**.

The datasets used for our multitask BERT and disentangled attention implementations were the SST dataset, the Quora dataset, and the SemEval STS Benchmark Dataset. The Quora dataset consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another. This dataset is split into **train** (141,506 examples), **dev** (20,215 examples), and **test** (40,431 examples). The STS dataset consists of 8,628 different sentence pairs of varying similarity ranging from 0 (unrelated) to 5 (equivalent meaning). This dataset is split into **train** (6,041 examples), **dev** (864 examples), and **test** (1,726 examples).

5.2 Evaluation method

For the sentiment classification and paraphrase detection tasks, accuracy was used as our evaluation metric.

For the semantic textual similarity task, the Pearson correlation of the true similarity values against the predicted similarity values (Agirre et al., 2013) was used as our evaluation metric.

5.3 Experimental details

Parameter	Value
Batch Size	8
Learning Rate	1 x 10^{-3} for pretrain and 1 x 10^{-5} for finetune
Seed	11,711
Epochs	10
Dropout Probability	0.3

Table 1: Bert and DeBERTa Model Parameters

The training times for BERT and disentangled attention varied by a large amount. For the BERT implementation, pre-training took around 2.5 hours and fine-tuning around 5 hours. After implementing disentangled attention, our training times increased tremendously. We were not expecting the added complexity to have such a large effect. Introducing the parameters $W_{q,r}, W_{k,r} \in \mathbb{R}^{d \times d}$ and $P \in \mathbb{R}^{2k \times d}$ for disentangled attention resulted in an increase of model parameters by $2L \cdot d^2 + 2k \cdot d$. Additionally, disentanglement requires two additional sets of calculation for position-to-content and content-to-position attention probabilities. Computational complexity is therefore increased on the order of $O(Nkd)$. These two factors combined led to a pre-train training time of 6.5 hours, and over 13 hours for fine-tuning. These long training times were intractable as we continued to iterate on our model. We initially attempted to use larger batch sizes, but doing so overloaded our GPUs. The use of fewer epochs and lower learn rates degraded our results too much. Our only remaining option was to trim the size of the QQP dataset by 70%, as it was the largest by far and the clear bottle neck in our training.

5.4 Results

Our best model, incorporating Disentangled Attention, log bucketing, as well as absolute position was able to outperform our baseline minBERT implementation in SST dev accuracy as well as Pearson Coefficient score for STS. As mentioned, we had to randomly remove 70% of the QQP dataset due to resource constraints. If we had the full training dataset available, we are confident that our disentangled implementation would have performed just as well if not better than our baseline.

After our first implementation of the Disentangled Attention mechanism, we were expecting to see immediate improvements upon baseline. After all, the literature had shown its superiority to

Downstream Task	Sentiment (SST)	Paraphrase (QQP)	Similarity (STS)	Av.
BASELINE				
minBert Dev (Finetune)	0.435	0.781	0.515	0.658
DeBERTa (Finetune)				
Disentangled Attention Dev	0.302	0.399	0.012	0.402
Log Bucketing	0.336	0.422	0.241	0.460
Enhanced Mask Decoder Dev	0.371	0.439	0.510	0.522
Absolute Position Dev	0.515	0.682	0.729	0.687
Absolute Position Test	0.499	0.681	0.738	0.681

Table 2: Dev and Test Accuracy Scores of Notable Models. Note: DeBERTa models are in order of features added. IE Enhanced Mask Decoder is the model with Disentangled Attention, Log Bucketing, and Enhanced Mask Decoder. Exception Being Absolute Position, which was implemented to replace EMD. Additionally, see Appendix for distribution of minBERT performance

predecessors in benchmarks. In retrospect, we had placed too much emphasis on the attention mechanism itself, rather than the DeBERTA model being well-performing due to the sum of its parts. After we implemented log bucketing, and noticed some improvements in performance, we were sure the Enhanced Mask Decoder was the final step. However, much to our surprise, it did not bring about the performance gains we were anticipating. While investigating the reasons behind this, we found a peculiar detail in the DeBERTa spec sheet that shows that, by default, content embeddings actually have absolute positions added in the same way as our baseline implementation. Putting back absolute positions immediately bolstered the performance of our model. While we are not completely sure why this is the case, we will explore some hypothesises.

6 Analysis

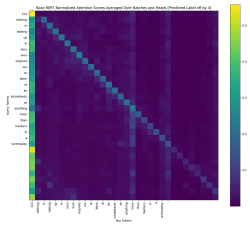

Figure 3: Base BERT Normalized Attention Scores Averaged Over Batches and Heads [Predicted Label off by 4]

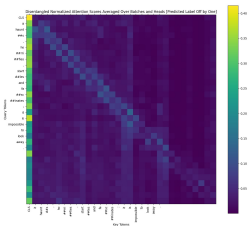

Figure 4: Disentangled w/ Enhanced Mask Decoder Normalized Attention Scores Averaged Over Batches and Heads [Predicted Label off by 1]

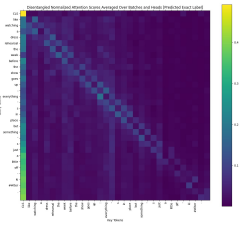

Figure 5: Disentangled w/ Absolute Position Embeddings Normalized Attention Scores Averaged Over Batches and Heads [Predicted Exact Label]

Our analysis begins with selected examples of attention distributions for our best model, the Enhanced Mask Decoder Model, as well as minBert. The attention score samples were taken during the evaluation phase, representing the application of the model's full learning. There are some qualities shared between all three models, despite them being from three distinct sample classes (great prediction, reasonable prediction, poor prediction). First, we notice high density of attention being directed to the first column, which represents the CLS token. Given the use of CLS represents the entire sequence for downstream tasks, this is unsurprising. The second similarity is there is a visible line across the diagonal of varying degrees, representing a token attending to itself. The most clear diagonal line can be seen in the minBERT implementation, where tokens are almost exclusively attending to themselves. We conjecture that this is due to BERT's self attention mechanism. On the opposite side of the spectrum, we have the Enhanced Decoder DeBERTA implementation, where the diagonal line is still visible, but noticeably more spread than BERT. This can be attributed to the mechanism behind the mask decoder, whereby absolute positional information is only added after all the transformer layers have completed. Finally, there is Disentangled BERT with absolute position

embeddings, which is somewhere between the two. We feel that this is the case because absolute position embeddings lay the foundation for the attention mechanism in our model, but is offset by weighting to learned patterns in relative position as well. We see that the varying degrees of token self-attention have a noticeable impact on the ability for tokens to attend to others in the sequence. minBERT is very sparse on the off diagonals, Absolute Disentangled BERT less so, and Enhanced Mask Decoder even less than that.

Our observations motivate a discussion on how attention mechanisms should optimally allocate weighting to tokens in the context of sentiment analysis. minBERT did a poor job identifying the sentiment of the input sequence. The heatmap indicates that perhaps tokens overly attending to themselves is undesirable in this downstream application. In many cases, sentiment is determined by the interplay of words in context. The use of negations is a great example. The BERT model may see a token for "Great" and hastily assign it a positive sentiment, neglecting the fact that the token that preceded it was "Not". Furthermore, tokens considered in isolation may not reflect sentiment either. Homophones are an example of this, and they need to be disambiguated with the proper context. This is not to say that self attention is useless, all three models display stronger self-attention in relation to other tokens. Enhanced Mask Decoder DeBERTa displayed the least token self-attention, and was also unable to predict the exact label. EMD may have gone slightly too far in terms of disregarding absolute positions. It has a much greater spread of attention to off-diagonal tokens, diluting the focus on key words that convey a lot of sentimental meaning. We conjecture that Disentangled BERT with absolute position embeddings was able to predict the exact label because it strikes a balance. The model shows focused attention on tokens it believes indicate strong sentiment, while also having connections to adjacent token. This allows it to capture individual meaning while also attending to contextual cues such as negations.

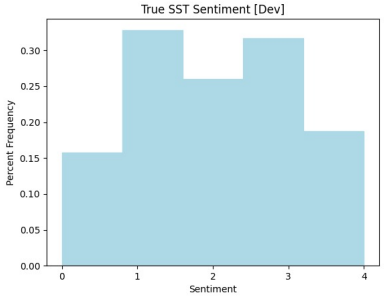

Figure 6: True Distribution of Sentiment Values (DeBERTa)

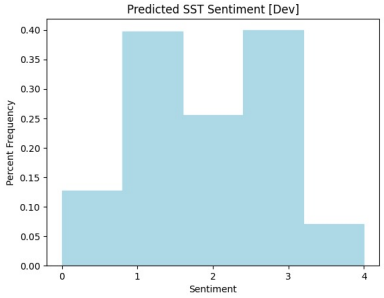

Figure 7: Predicted Distribution of Sentiment Values (DeBERTa)

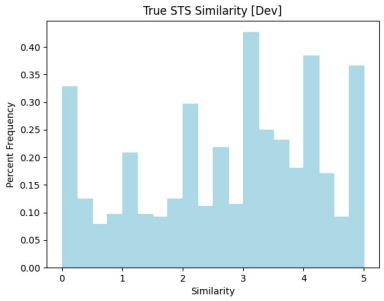

Figure 8: True Distribution of Similarity Values (DeBERTa)

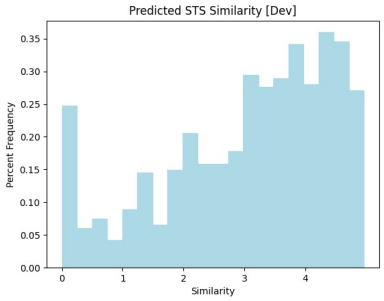

Figure 9: Predicted Distribution of Similarity Values(DeBERTa)

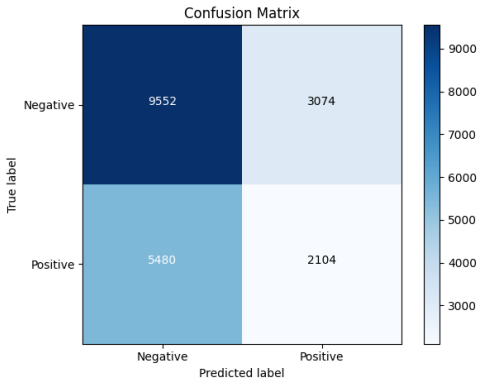

Figure 10: Paraphrase Detection Confusion Matrix (DeBERTa)

Beyond the attention mechanism, we explored where our model excelled and failed through generating histograms of true and predicted label distributions for SST and STS, as well as a confusion matrix for Paraphrase detection. Upon seeing the results, we were satisfied that the model was able to recognize the general patterns of the dev sets for SST and STS.

In sentiment analysis, the model had difficulty predicting sentiments at both extremes. This may have been in part due to the use of cross-entropy loss, which aims to maximize the log probability of the correct class. Given classes 1,2,3 were far more common in the underlying distribution, the model may have been biased towards the more common classes. This may also have been exacerbated by the use of the sigmoid function in cross entropy loss.

In regards to the similarity downstream task, we saw that the predicted labels somewhat matched the underlying distribution, with a bias towards higher similarity scores. It was interesting to see that the number of predicted labels near zero was very similar to the true distribution, as our sentiment classification struggled at the extremas. We attribute this to the use of the RELU function in our prediction function setting all negative values to zero. We also see that the dataset is skewed towards whole numbers, something we did not account for when we implemented our prediction algorithms. This presents a very interesting problem, and we think an interesting continuation would be biasing our loss function to output similarity scores closer to whole numbers. Once again, we see loss aversion at play, where the model minimizes overall loss by predicting more common outputs.

Finally, in regards to paraphrase detection, we see that when our model was incorrect, it was mostly due to falsely predicting two sentences as not being paraphrases when they in fact were. One contributing factor to this was the fact that our training set was reduced by 70%. Had the model had the opportunity to see more training examples, perhaps it would have been able to more accurately predict paraphrases. Furthermore, we also could have done a more thoughtful job deciding the split of our training set. Our random reduction of the data resulted in a training set that was much more balance (45% paraphrase vs 55% not paraphrase) than the underlying distribution, which skewed towards not paraphrases. This could have led to our model to be overly tuned to positive labels.

7 Conclusion

Implementing disentangled attention was a very challenging endeavor, and we are very satisfied that the implementation works and actually outperforms minBERT in two of the three chosen metrics. Our biggest finding was that in the context of smaller transformer models, absolute position is an irreplaceable element to build robust models. This was emphasized through our initial implementation of DeBERTa, with it underperforming in the absence of absolute positional information. Delving deeply into attention distributions, we discovered key insights about what makes a good attention mechanism. In the domain of sentiment analysis, self-attention is important, but cannot come at the expense of understanding broader context. We believe that our disentangled approach to attention allowed the model to be better tune into these contexts, which surely applies to tasks outside of sentiment analysis. A limitation of our project is the inability to train on the full Quora paraphrase set. Additionally, our model was optimized specifically for sentiment analysis, paraphrase detection, and semantic textual similarity. It is possible that our model would not perform well on other major

downstream tasks. If given more time, we would focus on improving our DeBERTa algorithm for paraphrase detection, as it performed slightly worse than our base BERT implementation.

References

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.

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

Vyacheslav Efimov. 2023. Large language models: Deberta — decoding-enhanced bert with disentangled attention.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music transformer.

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

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems.

A Appendix

Algorithm 1 Disentangled Attention

Input: Hidden state $\boldsymbol{H}$, relative distance embedding $\boldsymbol{P}$, relative distance matrix $\boldsymbol{\delta}$. Content projection matrix $\boldsymbol{W}_{k,c}, \boldsymbol{W}_{q,c}, \boldsymbol{W}_{v,c}$, position projection matrix $\boldsymbol{W}_{k,r}, \boldsymbol{W}_{q,r}$.

$\boldsymbol{K}_c = \boldsymbol{H}\boldsymbol{W}_{k,c}, \boldsymbol{Q}_c = \boldsymbol{H}\boldsymbol{W}_{q,c}, \boldsymbol{V}_c = \boldsymbol{H}\boldsymbol{W}_{v,c}, \boldsymbol{K}_r = \boldsymbol{P}\boldsymbol{W}_{k,r}, \boldsymbol{Q}_r = \boldsymbol{P}\boldsymbol{W}_{q,r}$

$\boldsymbol{A}_{c\to c} = \boldsymbol{Q}_c\boldsymbol{K}_c^\intercal$

for $i = 0, ..., N-1$ **do**

  $\tilde{\boldsymbol{A}}_{c\to p}[i,:] = \boldsymbol{Q}_c[i,:]\boldsymbol{K}_r^\intercal$

end for

for $i = 0, ..., N-1$ **do**

  for $j = 0, ..., N-1$ **do**

    $\boldsymbol{A}_{c\to p}[i,j] = \tilde{\boldsymbol{A}}_{c\to p}[i, \delta[i,j]]$

  end for

end for

for $j = 0, ..., N-1$ **do**

  $\tilde{\boldsymbol{A}}_{p\to c}[:,j] = \boldsymbol{K}_c[j,:]\boldsymbol{Q}_r^\intercal$

end for

for $j = 0, ..., N-1$ **do**

  for $i = 0, ..., N-1$ **do**

    $\boldsymbol{A}_{p\to c}[i,j] = \tilde{\boldsymbol{A}}_{p\to c}[\delta[j,i], j]$

  end for

end for

$\tilde{\boldsymbol{A}} = \boldsymbol{A}_{c\to c} + \boldsymbol{A}_{c\to p} + \boldsymbol{A}_{p\to c}$

$\boldsymbol{H}_o = \text{softmax}(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}})\boldsymbol{V}_c$

Output: $\boldsymbol{H}_o$

Figure 11: Disentangled Attention Algorithm

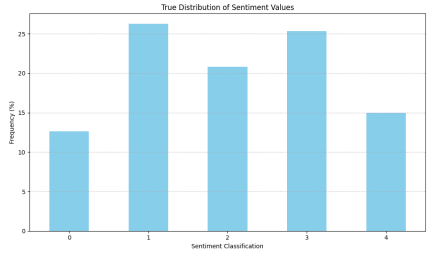

Figure 12: True Distribution of Sentiment Values (Base BERT)

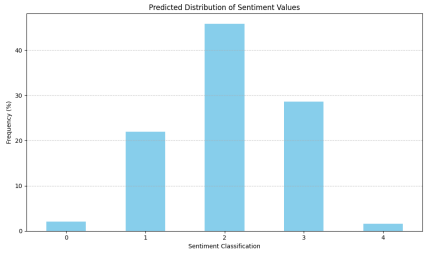

Figure 13: Predicted Distribution of Sentiment Values (Base BERT)

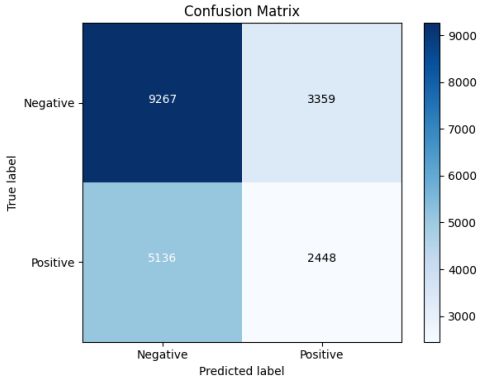

Figure 14: Paraphrase Detection Confusion Matrix (Base BERT)

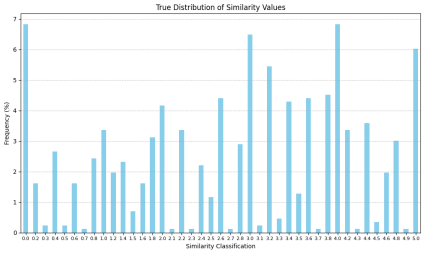

Figure 15: True Distribution of Similarity Values (Base BERT)

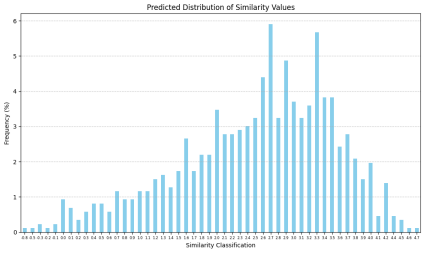

Figure 16: Predicted Distribution of Similarity Values (Base BERT)