

Parameter Optimization in DeBERTa for Text Classification Using PCA on Limited GPU Resources

Yvheng Lin

School of physics and astronomy, Sun-Yatsen University, Guangzhou, China

liny63@mail2.sysu.edu.cn

Abstract. This study focuses on optimizing Decoding-enhanced Bidirectional Encoder Representations from Transformers with disentangled Attention (DeBERTa) for text classification tasks, particularly on smaller datasets with limited GPU resources, by using Principal Component Analysis (PCA) to adjust and explain parameter choices. The goal is to reduce the model's parameters without significantly compromising accuracy. Specifically, the research employs DeBERTav3 on the Internet Movie Database (IMDb) dataset, without pre-training, and uses PCA to analyse the embedding and attention components of the model. PCA is applied to identify and reduce parameters by observing the Number of smallest factors that has a given Neglected cumulative contributing Rate of principal components (NNR), iteratively refining the model until optimal efficiency is achieved. This study demonstrates that PCA can effectively guide parameter adjustments, allowing even a smaller model to maintain high accuracy. The IMDb dataset is used to validate the approach, showing that a reduced DeBERTa model can still achieve competitive performance, making it particularly useful for natural language processing tasks involving minority languages or constrained computational environments. The findings have broader implications for optimizing large language models, suggesting that future work could explore combining PCA with techniques like factorized embedding parameterization to further enhance efficiency.

Keywords: DeBERTa, Principal Component Analysis (PCA), Text Classification.

1. Introduction

In the development of Natural language processing (NLP), language models based on transformer [1] grow larger on parameters to deal with more complex tasks and give better performance, making pre-training more time-exhausting and expensive. Therefore, fetching an appropriate number of parameters representing a significant portion of the information is valuable work. The embedding layer is the initial part of the neural network and was visualized to help people understand how the model works, and the attention part tells how words relate to each other. A way to explain why a model or a choice of parameters is better than another helps us to construct a better model or choose a more appropriate set of parameters.

In text classification [2], Decoding-enhanced Bidirectional Encoder Representations from Transformers with disentangled Attention (DeBERTa) and eXtreme Language Modeling (XLnet) are two of the most successful algorithms based on BERT [3-6]. An application of variants of BERT on various datasets. DeBERTa used disentangled attention, it concatenated word embedding and absolute position embedding to improve the position embedding used in BERT which is inherited from the transformer. XLNet improved BERT by Two-Stream Self-Attention, but the demand for devices increased greatly due to the permutation process. In contrast, ALBERT [7] reduces the amount of calculation by factorized embedding parameterization and cross-layer parameter sharing, but the necessity of decomposing the large vocabulary embedding matrix into smaller ones is not discussed. In addition, both of the models are pre-trained with huge datasets in English (Wikitext) and advanced GPUs and can deal with various tasks. A smaller model based on BERTs with GPU limited to the personal computer level is useful in certain cases, such as a single text classification task in minority languages without appropriate pre-train datasets like English. Such a model is easier to analyse and optimize under a certain number of parameters. Many powerful statistical tools such as principal component analysis (PCA) [8] and t-distributed stochastic neighbor embedding(t-SNE)

[9] have been used to evaluate the importance of parameters before the explosion of transformer-based models. More embedding techniques are shown in [10, 11].

The primary objective of this study is to explore the application of DeBERTaV3 on a relatively small dataset within the computational limits of a personal computer's GPU. This research aims to demonstrate the model's performance while strategically reducing the number of parameters using Principal Component Analysis (PCA) without significantly compromising accuracy. Initially, DeBERTaV3 without pre-training is employed for text classification tasks on the Internet Movie Database (IMDb) dataset, with parameters optimized to fully utilize the available GPU resources. Subsequently, PCA is applied to both the embedding and attention components of the model to identify the smallest set of factors with negligible cumulative variance contribution. The corresponding parameters for these negligible factors are then reduced, and the modified model is trained. This process is repeated iteratively to optimize efficiency. Experimental results indicate that PCA can effectively guide the adjustment of parameters in the embedding and attention components, enhancing the efficiency of models like DeBERTaV3 in text classification tasks. The study also provides an example of an efficient parameter setting for binary text classification tasks, tailored for environments with limited GPU resources.

2. Methodology

2.1. Dataset Description and Preprocessing

The IMDb dataset is a popular dataset for NLP tasks with 25,000 highly polar movie reviews for training and 25,000 for testing from IMDb, as well as each movie review's binary sentiment: positive or negative [12]. There is additional unlabeled data for use that this paper will not go into. Some features of the training dataset are shown in Fig.1 and Fig.2. 63.3% of the reviews have a length under 256 and 89.9% have a length under 512. 190 reviews appear more than once in the training set. Repeated reviews should be masked during the construction of vocabulary. The size of the total vocabulary of the IMDb training set is 101518. As a comparison, BERT has 30522 words, and DeBERTaV3 has 128100 words. To avoid overfitting, only the words that appear more than 4 times are counted in the vocabulary. The appearing times are counted after removing the repeated reviews, and the vocabulary size becomes 31271, so that only 1.33% words are marked as unknown.

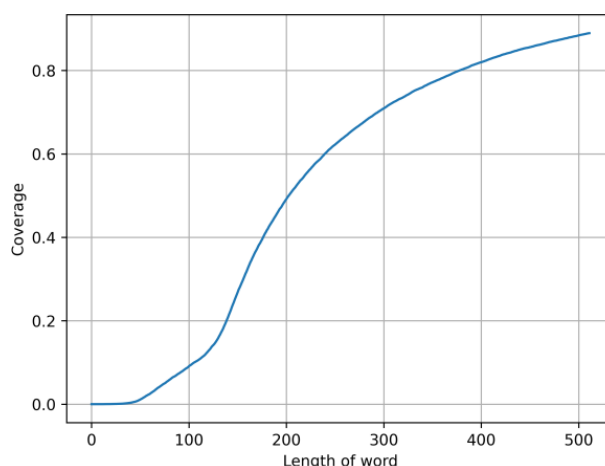


Figure 1. Length-wise coverage of reviews

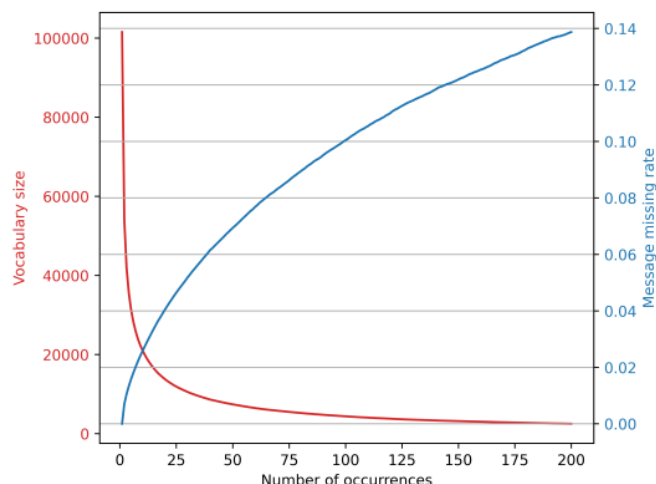


Figure 2. Vocabulary size and missing message

2.2. Proposed Approach

The prediction accuracy of BERT and DeBERTa reach 90+% with pre-train parameters. The original BERT has $30522 \times 768 = 23\text{M}$ parameters for embedding, DeBERTaV3 has 98M, and they have around 85M parameters in transformer layers. Such a large number of parameters prevents a device with 4070 from pre-training. A natural question is: Do so many parameters all contribute to the task positively? How many parameters are already enough to deal with a downstream task? How should they have reduced appropriately instead of blindly adjusting the parameters one by one? The answer is to adjust parameters such as vocabulary size, input length, embedding dimension, and MLP in the BERT encoder. Use PCA to test if a dimension is redundant or not. After getting the optimized set of parameters, change the dropout weight to improve the converged accuracy. Following this idea, the flow chart is as follows Fig. 3.

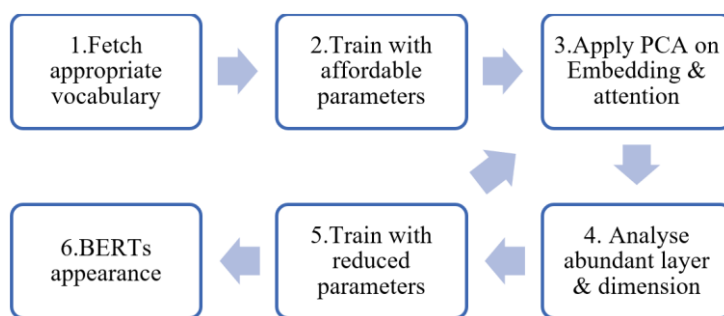


Figure 3. Flow chart of the programme

2.2.1 DeBERTaV3

Basing on DeBERTa, DeBERTaV3 use Gradient-Disentangled Embedding Sharing (GDES) rather than disentangled attention is a state-of-the-art natural language processing model developed by Microsoft Research. The training optimizer is AdamW, and the activation function of the last pool is GeLU instead of Tanh. DeBERTaV3 improves upon previous versions by incorporating advancements in model architecture and training techniques, resulting in better performance on various NLP tasks. DeBERTaV3 uses a disentangled attention mechanism, where the model separately considers the content and position of each word in a sentence. This allows the model to capture the relationship between words better, leading to improved contextual understanding. It also used a new pre-train task and is beyond the discussion of this paper. The DeBERTaV3 and its configuration setting function are imported from the transformer package of Huggin face.

2.2.2 PCA

PCA is a widely used statistical technique in data science and machine learning for dimensionality reduction. It transforms a dataset into a new coordinate system where the greatest variances by any projection of the data come to lie on the first few coordinates (called principal components).

PCA is a method used to simplify high-dimensional data. Assuming it has a standardized p -dimensional random vector x , the goal of PCA is to find an orthogonal matrix that maximizes the variance of this vector in a certain direction. When the variance in this direction is maximum, it corresponds to the maximum eigenvalue of the covariance matrix. In PCA, first calculate the covariance matrix of x and find the eigenvalues and eigenvectors of this matrix. Then arrange these feature values in descending order, and the corresponding feature vectors are also arranged accordingly. This gives us the direction of each principal component, which is these eigenvectors. Next, 'Loading matrix' is consisted of the first r eigenvectors multiplied by the square root of their corresponding eigenvalues. This allows us to represent the raw data x as a linear combination of these principal components. By selecting the first few principal components (i.e. choosing the eigenvectors with the highest eigenvalues and corresponding values), it can approximate the original data and ignore those components that contribute less to the variance. It uses an indicator called 'cumulative contribution rate of principal components' to measure the effectiveness of the principal components it has selected. If the sum of the remaining eigenvalues is relatively small compared to the sum of the overall eigenvalues, then these ignored components have weak explanatory power for the overall data.

In a string of matrix multiplications, an orthogonal matrix and its inverse can always be inserted between every two matrices. With an appropriate choice of orthogonal matrices, each matrix in the string can make the sum of the square of elements in the last few columns as small as possible just like PCA does in varimax rotation. However, the neural network is quite different from matrix multiplication: activation functions are inserted between linear transform layers. Hence PCA should only applied to the embedding and attention parts since there's no activation function inside them. For any matrix, if Number of smallest factors that have a given Neglected cumulative contributing Rate of principal component (NNR) is large, it is assumed that the matrix has the potential to contain abundant information, and the corresponding dimension should be reduced. As an example, given Neglected cumulative contributing Rate of 0.001, the Number of smallest factors that has a given Neglected cumulative contributing Rate of principal components is denoted as NNR (0.01).

2.2.3 Loss function

Cross-entropy is widely used as a loss function in text classification tasks due to its effectiveness in measuring the difference between the predicted probability distribution and the true distribution of class labels. Positive/negative label are converted to 1/0. The true distribution is denoted as $p = [y, (1 - y)]$. Denote the output of classifier as $out = [o_1, o_2]$, and the distribution of output is $q = \text{sigmoid}(out)$. The result is denoted as $q = [\hat{y}, (1 - \hat{y})]$, and the cross entropy is $H(p, q) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$.

2.3. Implementation Details

The study used Python 3.11 and torch 2.1.0+cu121 with NVIDIA GeForce RTX 4070 GPU. 20% of the training set is taken as the validation set. The prediction accuracy on the validation set is used to make early stopping. During the tokenization process, the length of reviews is truncated or padded into 512. The attention mask is added inside the model. With a batch size of 32, a dropout rate of 0.3, patience of 5, learning rate of 1e-4, the 1st training almost made full use of GPU. Loading matrix and Networked Value Chains (NNVC) in PCA are calculated by NumPy. The embedding part applied NNR (0.01) and attention part applied NNR (0.001). The history of all the parameter choices and training evaluations is recorded to be reviewed. The training optimizer is AdamW, and the activation function of the last pool is GeLU instead of tanh [13]. The time cost to run an epoch differs from 187s to 154s.

3. Result and Discussion

3.1. Apply PCA on Embedding & Attention

In the PCA process of attention part, the embedding dimension is regarded as the number of variables, and each word is taken as a sample. In the 1st attempt, NNR (0.01) is $1 \ll 256$, suggesting that most of the dimensions in the embedding layer are candidate to contribute to the prediction. In the PCA process of attention part, the in-feature dimension is regarded as the number of variables, and each column is taken as a sample, as shown in the Fig. 4.

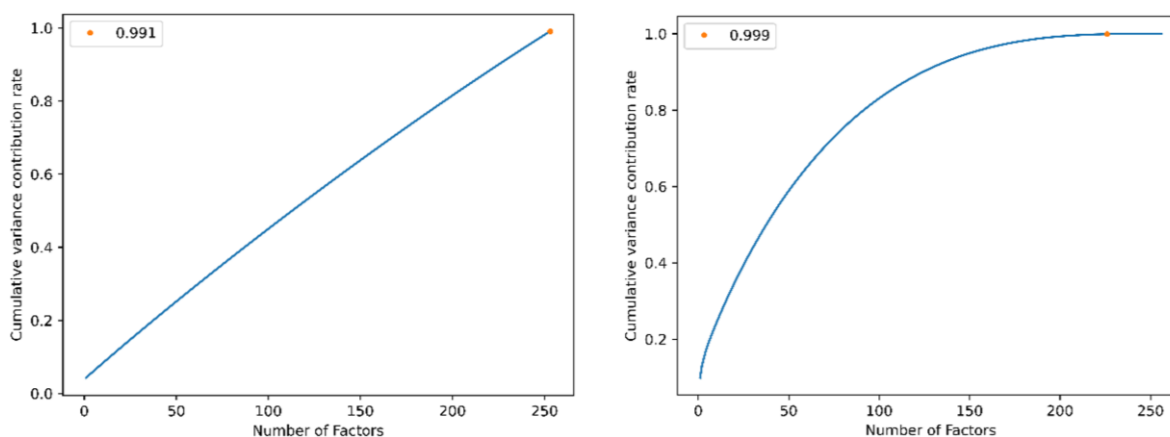


Figure 4. Cumulative variance contribution of embedding; cumulative variance contribution of query matrix in the first layer

3.2. Analyse Abundant Layer & Dimension

In the attention encoders, the projection matrices of position are not explicitly separated in the model's code, hence the paper focuses on the projection matrices of content. Applying the linear transform in code on suitable eigenmatrices gives the projection matrices. Average NNR (0.001) and NNR (0.01) of attention layer is 30.4 and 61.8. It's reasonable to check if the test accuracy remains unchanged after reducing the embedding dimension by 32, as shown in the Table 1.

Table 1. NNR (0.001) of attention part

Layer	Query	Key	Value	Dense
1	32	32	32	31
2	30	30	31	31
3	31	30	31	30
4	30	30	31	30
5	30	30	31	29
6	30	30	29	30
7	30	30	29	31
8	30	30	30	30

3.3. Train with Reduced Parameters

The 2nd attempt with N_EMBED = 224 gets higher accuracy (88.3) is not a surprise, since the training accuracy can easily reach 0.98 before the early stopping “patience”. Average NNR (0.001) is 30.0. The 3rd attempts with N_EMBED = 196 gets lower accuracy (87.3). Also, average NNR (0.001) becomes 24.0. This significant decline indicates that 196 might be a suitable parameter that should not be reduced further. The 4th, and 5th attempts with N_EMBED = 128 with 4, 2 heads get lower accuracy (87.8, 87.3). Average NNR (0.001) becomes 16.7, 16.9 respectively. This result firmly supports the conclusion in the 3rd attempt. Referring to the second attempt, where 13.3% of the dimensions were found to be negligible, hence the plan for the next attempt is to reduce the number of layers by 13.3%, hence the 6th attempt with N_EMBED = 224 and 7 layers gets a higher

accuracy of 77.5. With N_EMBED same as the 2nd attempt, average NNR (0.001) reduced from 30.0 to 27.1, indicating that the usage of parameters indeed raised, as shown in the Table 2. Conversely, applying the parameter adjusting process by adding the parameters is also capable of reaching high efficiency.

The same parameter for the first three attempts is also applied to BERT, and each epoch runs for 129s. The results are inferior compared to DeBERTaV3. The accuracy with the 2nd attempt only reaches 86.7, with an average attention part NNR (0.001) of 27.2. The lower abundance indicates that DeBERTaV3 captured more figures since it reaches higher accuracy with fewer non-neglectable factors. Inversely, the result reveals that NNR represents the efficiency of parameters in models.

Table 2. Parameter Selection for Training. N_EMBED: Word embedding dimension; N_HEADS: Number of multi-head attention mechanisms; N_LAYER: Number of encoder layers; D_INTER: Hidden size of the MLP following the encoders; EPOCH: Epoch at which validation accuracy reaches its maximum; N_PARA: Total number of parameters, excluding word embeddings, which contribute most to the training time

No.	N_EMBED	N_HEADS	N_LAYER	D_INTER	EPOCH	N_PARA	ACC
1	256	4	8	1024	2	6.5M	87.2
2	224	4	8	896	2	5M	88.3
3	196	4	8	784	3	3.8M	87.3
4	128	4	8	512	4	1.7M	87.8
5	128	2	8	512	6	1.7M	87.3
6	224	4	7	512	4	4.4M	87.5

4. Conclusion

This study demonstrates the application of PCA in adjusting and explaining parameter choices for text classification tasks using DeBERTa, specifically focusing on optimizing the model for the IMDb dataset. By employing PCA, the research analyses how effectively information is captured within the model's matrices, allowing for an informed adjustment of parameters based on the NNR rates in different components of the model. The experimental results confirm that PCA provides valuable insights into parameter adjustments, enabling even a reduced model to achieve relatively high accuracy without pre-training. This approach proves particularly beneficial for NLP tasks involving minority languages or scenarios with limited computational resources, though it is also applicable to larger language models. Future research could explore combining PCA with techniques like factorized embedding parameterization from models such as ALBERT, potentially aligning the NNR rates in both the embedding and attention components of DeBERTa. This combination might further reduce the number of redundant parameters, enhancing model efficiency without compromising performance.

References

- [1] Vaswani, Ashish, et al. Attention is all you need. Advances in neural information processing systems, 2017, 30.
- [2] Li Q, Peng H, Li J, et al. A survey on text classification: From traditional to deep learning. ACM Transactions on Intelligent Systems and Technology (TIST), 2022, 13 (2): 1-41.
- [3] He P, Liu X, Gao J, et al. Deberta: Decoding-enhanced bert with disentangled attention. arXiv preprint arXiv: 2006.03654, 2020.
- [4] He P, Gao J, Chen W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. arXiv preprint arXiv: 2111.09543, 2021.
- [5] Yang Z, et al. Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 2019, 32.

- [6] Devlin J, Chang M, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805, 2018.
- [7] Lan Z, Chen M, Goodman S, et al. ALBERT: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv: 1909.11942, 2019.
- [8] Lebre R and Ronan C. Word emdeddings through hellinger pca. arXiv preprint arXiv: 1312.5542, 2013.
- [9] Liu S, et al. Visual exploration of semantic relationships in neural word embeddings. IEEE transactions on visualization and computer graphics, 2017, 24 (1): 553-562.
- [10] Karl F. Transformers are short text classifiers: a study of inductive short text classifiers on benchmarks and real-world datasets. CoRR abs/2211.16878, 2022.
- [11] Selva S, Kanniga R. A review on word embedding techniques for text classification. Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020, 2021: 267-281.
- [12] Lakshmipathi N, "Imdb-dataset-of-50k-movie-reviews", 2019, <https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>.
- [13] Semi O, "BERTs-PCA", 2023, <https://github.com/yv-h-Lin/DeBERTa-PCA>.