YNU-HPCC at SemEval-2024 Task 9: Using Pre-trained Language Models with LoRA for Multiple-choice Answering Tasks

Jie Wang, Jin Wang, and Xuejie Zhang
School of Information Science and Engineering
Yunnan University
Kunming, China
wangjie_qpqj@stu.ynu.edu.cn, {wangjin, xjzhang}@ynu.edu.cn

Abstract

This study describes the model built in Task 9: brainteaser in the SemEval-2024 competition, which is a multiple-choice task. As active participants in Task 9, our system strategically employs the decoding-enhanced BERT (DeBERTa) architecture enriched with disentangled attention mechanisms. Additionally, we fine-tuned our model using low-rank adaptation (LoRA) to optimize its performance further. Moreover, we integrate focal loss into our framework to address label imbalance issues. The systematic integration of these techniques has resulted in outstanding performance metrics. Upon evaluation using the provided test dataset, our system showcases commendable results, with a remarkable accuracy score of 0.9 for subtask 1, positioning us fifth among all participants. Similarly, for subtask 2, our system exhibits a substantial accuracy rate of 0.781, securing a commendable seventh-place ranking. The code for this paper is published at: https://github.com/123yunnandaxue/Semveal-2024_task9.

1 Introduction

The human reasoning process includes two types of thinking: vertical and horizontal. Vertical thinking is a sequential analysis process based on rationality, logic, and rules. Horizontal thinking is a divergent and creative process. The success of language models has inspired the natural language model (NLP) community to focus on tasks that require implicit and complex reasoning. Although this type of vertical thinking task is widespread, horizontal thinking puzzles have received little attention (Jiang et al., 2024). Task 9 in the SemEval-2024 competition: brainteaser is a multiple-choice task that tests the model's ability to demonstrate horizontal thinking and challenge default common sense associations. The task consists of two subtasks, sentence and word puzzles (Jiang et al., 2023).

- Subtask 1: Sentence-type brain teaser where the puzzle defying commonsense is centered on sentence snippets.
- Subtask 2: Word-type brain teaser where the answer violates the default meaning of the word and focuses on the letter composition of the target question.

In recent years, machine learning models have garnered significant attention. Traditionally, these models have employed a two-step process involving the extraction of hand-crafted features from documents followed by classification using algorithms like Naïve Bayes (Zhang, 2004), SVM (Cortes and Vapnik, 1995), HMM (Trabelsi et al., 2012), or random forests (Ren et al., 2015). However, this approach presents limitations, such as the need for meticulous feature engineering and reliance on domain knowledge for feature design. To address these shortcomings, neural approaches have emerged. Early attempts, such as latent semantic analysis (LSA) (Dumais et al., 1988) and neural language models, initially underperformed compared to classical models but paved the way for developing more powerful embedding models. Significant advancements were made with the introduction of word2vec (Mikolov et al., 2013), ELMo (Peters et al., 1802), RoBERTa (Liu et al., 2019), GPT (Radford et al., 2019), BERT (Devlin et al., 2018), and subsequent models like GPT-3 (Brown et al., 2020) and GShard (Lepikhin et al., 2020), which boast increasingly more significant parameters and training datasets (Minaee et al., 2021).

This paper proposes a deep learning system for Task 9 in SemEval-2024, titled brainteaser. We use the decoding-enhanced BERT (DeBERTa) (He et al., 2020) model with disentangled attention as the base model and use LoRA (Hu et al., 2021) to fine-tune the model. Focal loss was used to address the issue of label imbalance. The back-translation

Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), pages 471–476

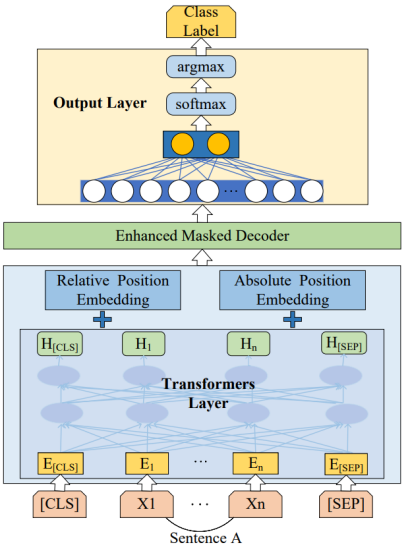

Figure 1: The overall architecture of the proposed method.

method is used to enhance the original dataset, and the processed dataset is used to train the model. The experimental results of this paper were ultimately presented in Task 9 of the SemEval-2024 competition. On the original dataset, the accuracy of Task 1 was 0.9, ranking fifth; The accuracy of Subtask 2 was 0.78, ranking seventh. The rest of this paper is organized as follows. In Section 2, we provided a detailed description of the proposed system and model. The experiment and results are discussed in Section 3. Finally, Section 4 presents the conclusion.

2 DeBERTa

Transformer (Vaswani et al., 2017) has become the most effective neural network architecture for neural language models. Unlike recurrent neural networks (RNNs) (Zaremba et al., 2014) that process text sequentially, transformers apply self-attention functionality to parallelly calculate the attention weight of each word in the input text. Therefore, compared to RNNs, they can perform large-scale model training in parallel. In this paper, the DeBERTa model we use is a new transformer neural language model that improves the Bert model using two novel techniques: a disentangled attention mechanism and an enhanced mask decoder. Figure 1 shows the structure of the system.

2.1 Tokenizer

Given a training data $\mathcal{D} = \{X^{(m)}, y^{(m)}\}_{m=1}^{M}$, $X^{(m)}$ is the input text, $y^{(m)}$ is the corresponding ground-true label, tokenizer is applied to transform $X^{(m)}$ as,

$$X = \{[\text{CLS}], x_1, x_2, ..., x_n, [\text{SEP}]\} \quad (1)$$

where x_i is the token in the text, [CLS] represents the classified characters, and [SEP] represents the terminating characters.

2.2 Encoder

DeBERTa's encoder is mainly composed of multilayer transformer encoders, and each transformer encoder is composed of multiple sub-layers. The following are the main components of the encoder of the DeBERTa model.

Token embeddings. Each token in the input text is first converted into the corresponding word embedding vector. First, we use an embedding layer to map each token x_i to its corresponding word embedding vector. The embedding matrix E is with dimension $V \times d$, where V is the size of the vocabulary and d is the dimension of the word embedding. Then, the embedding vector corresponding to the i-th token x_i can be expressed as $e_i = E[x_i]$.

Positional encoding. Positional encoding represents the absolute position of each word in the input sequence. Suppose we have a position encoding matrix P with dimensions $N \times d$, where d is the dimension of the word embedding. Then, the position encoding vector corresponding to the i-th position p_i can be expressed as $p_i = P[i]$. After adding positional encoding, the new word embedding sequence we get is $E(X) + [p_1, p_2, ..., p_N]$.

Relative positional encoding. The relative position encoding matrix is a learnable parameter matrix with dimensions$L \times 2D$, where L is the maximum sequence length and D is the word embedding dimension. In DeBERTa, the calculation process of relative position encoding is as follows:

For each pair of words (i, j), we calculate its relative position relationship vector r_{ij}.

$$r_{ij} = PE_{(i-j)} \quad (2)$$

where $PE_{(i-j)}$ represents the encoding vector at position $(i - j)$ in the relative position encoding matrix. Finally, the input text sequence X that needs to be sent to transformer encoder layers can be obtained by adding the word embedding vector, position encoding, and relative position encoding.

$$x_i = E[x_i] + p_i + r_{ij} \quad (3)$$

$$X = \{x_1, x_2, ..., x_n\} \tag{4}$$

Transformer encoder layers. Each transformer encoder layer contains the following sub-layers:

1. Multi-head self-attention. This sub-layer allows the model to focus on different parts of the input sequence simultaneously to capture global information.

2. Feed-forward neural network. This sub-layer contains a feed-forward neural network for non-linear transformation and feature extraction of the context vector at each position.

When the input text sequence passes through the encoder of the DeBERTa model, it can be expressed as:

$$H = \text{Encoder}(X) \tag{5}$$

where H is the encoded context representation, X is the input text sequence, and Encoder() is the Encoder part of the DeBERTa model.

2.3 Output Layer

In the DeBERTa model, the output layer is usually used to predict downstream tasks, such as text classification, named entity recognition, etc.

Linear transformation. First, map the output of the transformer encoder to the output space, usually through a linear transformation (fully connected layer). Assuming we have a weight matrix W and a bias vector b, the calculation of the linear transformation can be expressed as:

$$Z = H \cdot W + b \tag{6}$$

where Z is the output after linear transformation.

Activation function. The softmax function is a commonly used activation function in the output layer in multi-classification problems. The formulation of the softmax function is as follows:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{7}$$

where the x_i is the i-th element in the input vector Z, and N is the length of the input vector. Finally, the category label is obtained by the following formula.

$$\hat{y} = argmax(softmax(H \cdot W + b)) \tag{8}$$

Focal loss. Focal loss is a dynamically scaled cross-entropy loss. A dynamic scaling factor can dynamically reduce the weight of easily distinguishable samples during training, thereby quickly focusing the center of gravity on those difficult-to-distinguish samples. The formula for focal loss is as follows.

$$FL(p_t) = -\alpha_t(1 - p_t)^{\gamma} \log(p_t) \tag{9}$$

where the α_t is a trainable parameter, the γ is a hyperparameter, and the p_t represents the probability of the category of t obtained by softmax function.

2.4 LoRA

The low-rank adapter (LoRA) significantly reduces the number of trainable parameters for downstream tasks by freezing the weights of pre-trained models and injecting trainable rank decomposition matrices into each layer of the transformer architecture. Research has shown that the model quality and fine-tuning of LoRA on RoBERTa, DeBERTa, GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020) are equivalent or better.

LoRA injects trainable low-rank matrices into transformer layers to approximate the parameter update. For a pre-trained weight matrix $W \in \mathcal{R}^{n \times d}$, LoRA decomposes the update with a low-rank factorization,

$$W + \Delta W = W + W^{down} W^{up} \tag{10}$$

where W^{down} and W^{up} are both trainable parameters. Specifically, LoRA applied such an update to the query and value projection matrix in the multi-head attention. For a specific input H_{l-1} to the linear projection in multi-head attention, LoRA can be defined as,

$$H_l = H_l + \gamma \cdot H_{l-1} W^{down} W^{up} \tag{11}$$

where γ was used to scale the contribution of LoRA.

3 Experimental Results

Datasets. The training set for subtask 1 (507 data) and subtask 2 (396 data) are processed using back-translation to enhance the model's efficiency. The dataset is translated into Chinese, Russian, Arabic, French, German, Spanish, Portuguese, Italian,

Method	Loss	Subtask 1	Subtask 2
LoRA	CE	0.93	0.51
	Focal	**0.83**	**0.67**
AdaLoRA	CE	0.37	0.37
	Focal	0.51	0.32
Prompt-Tuning	CE	0.17	0.22
	Focal	0.17	0.27
R-Drop	CE	**0.96**	**0.80**
	Focal	0.34	0.40

Table 1: Accuracy of each strategy in dev data. Results in bold are the best performance.

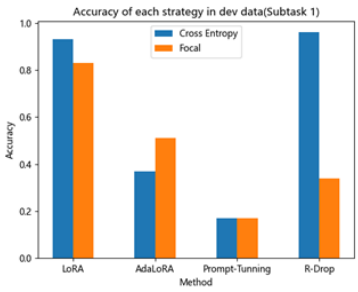

Figure 2: Accuracy of each strategy in dev data (Subtask 1).

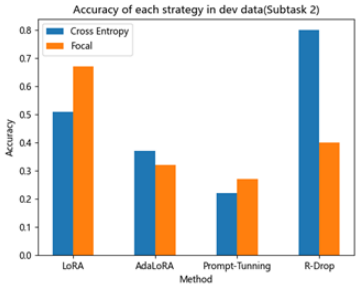

Figure 3: Accuracy of each strategy in dev data (Subtask 2).

Dataset	Subtask 1	Subtask 2
Back-translation method dataset	1.0	1.0
Original dataset	0.96	0.98

Table 2: Accuracy of LoRA in dev data.

and Japanese and re-translated into English. The translated results are added to the dataset to obtain the enhanced dataset: the training set for subtask 1 (5070 data) and the training set for subtask 2 (3960 data).

Evaluation Metrics. In this paper, the task will be evaluated based on the following two accuracy indicators.

- Example-based accuracy: treat each problem (primitive/adversarial) as a separate instance.
- Based on group accuracy: each problem and its related adversarial instances form a group.

Implementation Details. In this paper, we use the back-translation method for data augmentation to improve the model's efficiency. After obtaining more data, use focal loss to address the issue of category imbalance in the data. LoRA is used to reduce the trainable parameters of downstream tasks to save computational costs. This is the DeBERTa-V2-xxlarge model with 48 layers and a 1536 hidden size. The total parameters are 1.5B, and it is trained with 160GB of raw data.

Comparative Results. In addition to using LoRA, this paper also attempted methods such as AdaLoRA (Zhang et al., 2023), Prompt-Tuning (Lester et al., 2021), R-Drop (Wu et al., 2021), using cross-entropy and focal loss as losses, with accuracy as the evaluation metric. The results are shown in Table 1.

Figure 2 depicts validation set accuracy for subtask 1 across various methods. Models employing LoRA and R-Drop exhibit higher accuracy with cross-entropy loss. Transitioning to focal loss saw a 0.1 drop for LoRA, whereas R-Drop experienced a significant decrease. AdaLoRA's accuracy increased by 0.14 with focal loss adoption, though performance remains subpar. Prompt-Tuning's accuracy remains stagnant regardless of the loss function, indicating poor performance.

Figure 3 shows the accuracy of various methods on the validation set for subtask 2. From the figure, we can see that the accuracy of the model using LoRA increased by 0.16 after using focal loss. After using focal loss, AdaLoRA's accuracy dropped by 0.05. Moreover, no matter which method the model uses, its performance on subtask 2 is worse than on subtask 1.

Finally, we found that when using cross-entropy, R-Drop achieved the best results, with LoRA ranking second. However, after using focal loss, the accuracy of R-Drop decreased significantly. Based on the results of cross-entropy and focal loss, using LoRA yields the best result. Therefore, LoRA was chosen for model fine-tuning, and then the data augmentation dataset was used to train the model. The obtained model was retrained using the orig-

Dataset	Subtask 1	Subtask 2
Original dataset	0.900 (5)	0.781 (7)
Semantic reconstruction	0.825 (8)	0.719 (9)
Recontextualization	0.800 (7)	0.812 (6)
Original dataset + Semantic reconstruction	0.825 (8)	0.719 (9)
Original dataset + Recontextualization	0.725 (8)	0.625 (10)
Original dataset + Semantic reconstruction + Recontextualization	0.842 (12)	0.771 (13)

Table 3: Result in the Test

inal dataset, and the results on dev are shown in Table 2.

Table 3 shows the competition results on the Test, with rankings displayed in parentheses. To ensure that the task evaluates reasoning ability rather than memory ability, adversarial versions of the original data are constructed in two ways.

- Semantic reconstruction: rephrasing the original question without changing the correct answer and interfering factors.
- Context reconstruction: maintains the original reasoning path but changes the question and answer to describe the new contextual context.

As shown in Table 3, our model achieved good results on both subtask 1 and subtask 2 on the original dataset. In subtask 1, the accuracy reached 0.9, ranking fifth, and in subtask 2, the accuracy reached 0.781, ranking seventh. Except for the final dataset, the accuracy of our model ranks in the top ten. Moreover, our model performs better on subtask 1 except for the context reconstruction dataset. It is well proven that our system has demonstrated competitive performance.

4 Conclusions

This paper describes a deep learning model for a multiple-choice task (Task 9: brainteaser in the SemEval-2024 competition), using DeBERTa as the base model and achieving good results, ranking fifth in accuracy in subtask 1 and ranking seventh in accuracy in subtask 2. However, there is still considerable room for improvement in the model. Therefore, we will try more methods to improve the model's efficiency in the future.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments. This work was supported by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61966038 and 62266051.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20:273–297.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Susan T Dumais, George W Furnas, Thomas K Landauer, Scott Deerwester, and Richard Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–285.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yifan Jiang, Filip Ilievski, and Kaixin Ma. 2024. Semeval-2024 task 9: Brainteaser: A novel task defying common sense. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1996–2010, Mexico City, Mexico. Association for Computational Linguistics.

Yifan Jiang, Filip Ilievski, Kaixin Ma, and Zhivar Sourati. 2023. BRAINTEASER: Lateral thinking puzzles for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14317–14332, Singapore. Association for Computational Linguistics.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

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. *arXiv preprint arXiv:1907.11692*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learning–based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3):1–40.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 1802. Deep contextualized word representations. corr abs/1802.05365 (2018). *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Shaoqing Ren, Xudong Cao, Yichen Wei, and Jian Sun. 2015. Global refinement of random forest. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 723–730.

Chiraz Trabelsi, Bilel Moulahi, and Sadok Ben Yahia. 2012. Hmm-care: Hidden markov models for context-aware tag recommendation in folksonomies. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 957–961.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021. R-drop: Regularized dropout for neural networks. *Advances in Neural Information Processing Systems*, 34:10890–10905.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Harry Zhang. 2004. The optimality of naive bayes. *Aa*, 1(2):3.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.