System Report for CCL24-Eval Task 2: Chinese Parataxis Graph(CPG) Parsing Based on Large Language Models

YueYi Sun
Beijing Institute of Technology
No.5 Yard, Zhongguancun South Street,
Haidian District, Beijing
1120223534@bit.edu.cn

Yuxuan Wang
Beijing Institute of Technology
No.5 Yard, Zhongguancun South Street,
Haidian District, Beijing
wangyuxuanbilly@bit.edu.cn

Abstract

This paper presents the work submitted for the 23rd China National Conference on Computational Linguistics(Evaluation Workshop)(CCL24-Eval), focusing on the Chinese Parataxis Graph (CPG) Parsing task. CPG represents Chinese natural language hierarchically through relational triplets, providing a consistent representation for linguistic units of varying levels. Our approach has used large-scale language models through full fine-tuning, achieving the result with F1 value at 71.6% in the contest and 74.76% after the contest. Furtehrmore, our team has proposed a combined model that integrates multiple LoRA fine-tuned medium-scale models after the contest. This approach is able to minimize the time and space consumption while keeping the performance of CPG construction task relatively high.

1 Introduction

In recent years, the development of LLM fine-tuning technology has progressed rapidly. LLMs can be fine-tuned to adapt to new tasks and be customized for specific needs, making them easily applicable to downstream vertical domains. Fine-tuning techniques can fully utilize the knowledge in pre-trained models, achieve excellent performance with less training data, significantly improve training efficiency, and avoid retraining the model for each task. This also focuses the model more on the target task, reduces model development costs, and improves model application flexibility and scalability. Fine-tuning for specific tasks can effectively enhance the model's accuracy and generalization ability on those tasks.

Previous research (**?**; **?**) has introduced the definition and rules of CPG. Meanwhile, earlier studies (kommineni2024human, 1972; **?**; **?**; **?**) have experimentally and theoretically demonstrated the completion of corresponding tasks through fine-tuning single large model. Our goal is to find a method to construct CPG through fine-tuning LLMs， which is a combination of the works that are listed. However, there are quite a few significant limitations of previous research. Firstly, the adjustment of large model parameters is insufficient. Secondly, previous tasks are completed only through fine-tuning large models, leading to high space occupancy and time requirements. In our work, we take the task of CPG construction as an example to introduce the results of achieving high performance through fine-tuning large models, reaching an F1 score of 74.76%. Additionally, it proposes a method that achieves similar results through the interaction and integration learning of middle-scale models, significantly reducing

©2024 China National Conference on Computational Linguistics
Published under Creative Commons Attribution 4.0 International License

Proceedings of the 23rd China National Conference on Computational Linguistics, pages 51-61, Taiyuan, China, July 25 - 28, 2024.
Volume3: Evaluations
(c) Technical Committee on Computational Linguistics, Chinese Information Processing Society of China

time and space consumption during the task process.

According to the Scaling Law (**?**), as the number of model parameters and the size of data increases, the model's performance improves significantly. However, large-scale models require substantial computational resources and storage space. In contrast, middle-scale models require significantly fewer computational resources and storage space and usually have a much faster training speed than large models. They can iterate and adjust quickly, having lower deployment costs, and can more easily adapt to different tasks and environments, achieving good performance in multiple application scenarios with minimal adjustments and fine-tuning.

2 Preliminaries

In this task, we use the Wenxin Qianfan's large-scale model platform of Baidu AI Cloud and the ERNIE Speed closed-source model for fine-tuning. The model version is ERNIE-Speed-8K, released on February 5, 2024. It is a high-performance industry-level knowledge-enhanced large language model developed by Baidu.

This model has the following advantages for this task:

Chinese Context: ERNIE Speed is specially optimized and trained for the Chinese context, allowing it to better understand and process Chinese text.

Context Handling: ERNIE Speed has a high context length processing capability in inference scenarios, enabling it to better handle contextual dependencies in Chinese text.

Lightweight Design: For this task, we introduced the ERNIE-Speed-8K version, which is a lightweight large language model with strong performance and high processing speed.

Efficient Training: Due to its lightweight design, ERNIE Speed has a relatively short training time, improving model iteration efficiency.

During the task, we also tried fine-tuning other large language models provided by the Wenxin Qianfan's large-scale model platform, such as Baidu's lightweight large language model ERNIE Lite and the Qianfan-Chinese-Llama-2-13B-v1 large model, which is based on Llama-2, with an expanded Chinese vocabulary and enhanced pre-training and instruction fine-tuning using large-scale Chinese-English data. Although these models also achieved good performance, the final results were not as good as the ERNIE Speed model. We believe this is mainly due to the parameter scale, so we ultimately chose the ERNIE Speed model.

3 Single-LLM Task

In this section, we mainly present the methods we implemented in the CCL24-Task2 competition.

The core process of fine-tuning large models includes prompt engineering and parameter adjustment. This section will focus on demonstrating the methods of prompt engineering and parameter adjustment.

3.1 Prompt Engineering

During prompt engineering, we tested various prompts, such as removing word segmentation and using the entire paragraph as a prompt. However, we found that this caused significant redundancy in

CCL 2024

the model output and mismatched results with the original word segmentation, making post-processing difficult. Therefore, we explicitly specified word segmentation information. On the other hand, we found that overly detailed and complex prompts may caused the model output to ”repeat” itself, such as including numbering in the prompt and adding prompts about the numbers. Hence, we ultimately adopted a concise prompt.

In our model, the data used to train the large language model consists of a prompt and a response. During the training of the large model, the prompt is the input, and the response is the expected output. In this task, we define the prompt as the sentence being studied and the response as the relationship set in the intention diagram, specifically defined as follows:

Prompt: Composed of the original sentence in segmented form, prefixed with ”segmentation as,” as shown in the example below:

"prompt": "分词为[中国,的,南方,水乡,, ,一旦,进入,“,梅雨,季节,”,, ,阴雨天,往往,会,持续,数,日,。]"

Response：Presented in the form of triplets, showing two words and their relationship, as shown in the example below:

"response": "(中 国,南 方,EntityRel),(南 方,水 乡,EntityRel),(水乡,进 入,A0),(一 旦,进 入,Conj),(进 入,ROOT,CoreWord),(进 入,条 件 关 系,条 件事 件),(梅 雨,季 节,EntityRel),(季 节,进 入,状 态 终 点),(阴 雨 天,持 续,A0),(往往,持 续,Time),(会,持 续,Mod),(持 续,ROOT,CoreWord),(持 续,条 件 关 系,推 论 事件),(数,日,EntityRel),(日,持续,Time)"

Overall：


```
[
    {
        "prompt": "分词为[中国,的,南方,水乡,, ,一旦,进入,“,梅雨,季节,”,, ,阴雨天,往往,会,持续,数,日,。]",
        "response": "(中 国,南 方,EntityRel),(南 方,水乡,EntityRel),(水 乡,进 入,A0),(一 旦,进 入,Conj),(进 入,ROOT,CoreWord),(进入,条件关系,条件事件),(梅雨,季节,EntityRel),(季节,进入,状态终点),(阴雨天,持续,A0),(往 往,持 续,Time),(会,持 续,Mod),(持 续,ROOT,CoreWord),(持 续,条 件 关系,推论事件),(数,日,EntityRel),(日,持续,Time)"
    }
]
```


3.2 Data Preprocessing

We clean the data by removing words corresponding to incomplete or missing examples. Then, we convert each dataset into the prompt format.

3.3 Data post-processing

The output of the large language model is constructed as follows, we first display an example:


```
[
    {
        "index":1,
        "inference_dataset_id":"ds-gf1cr64223rfekth",
        "prompt":"分词为[培养,一条,优秀,的,警犬,是,极,不,容易,的,事,,
,我们,的,警犬,当然,首先,要,为,中国,军队,服务,。,”]",
        "response":"",
        "completion":"(培养,是,A0),(一条,警犬,EntityRel),(优秀,警
犬,EntityRel),(警 犬,培 养,A1),(是,ROOT,CoreWord),(极,容 易,Mod),(不,容
易,Mod),(容 易,事,EntityRel),(事,是,A1),(我 们,警 犬,EntityRel),(警 犬,服
务,A0),(当然,服务,Mod),(首先,服务,Time),(要,服务,Mod),(为,军队,PN),(中国,军
队,EntityRel),(军队,服务,A2),(服务,ROOT,CoreWord)"
    }
]
```


We extract the useful parts, including the prompt section and the completion section, and then we convert them into the final required output format.

3.4 Model Tunning

Considering that with sufficient computing power, full fine-tuning yields the best results among various LLM fine-tuning strategies, but it also requires the most computational resources and storage space. With the cloud computing services provided by the Wenxin Qianfan's large-scale model platform, we have ample computing resources for convenient fine-tuning. Therefore, we ultimately choose the full parameter fine-tuning strategy.

4 Multi-LLM Ensembling and Tuning

In this section, we introduced some smaller-scale LLMs into the task. When using these smaller-scale LLMs separately, they perform not as well as large-scale models. Our approach can be mainly described as two parts: Multi-Model Reinforcement Circum-block and IU-Block. Through these two tools, we came up with a similar result compared with using large-scale LLM.

4.1 Multi-Model Reinforcement Circum-block

We integrate large models with smaller parameter scales through a serial approach, using this method to train the model. We designed a training model named reinforcement circum-block(RC-Block) and the model is illustrated in Figure 1. The main structure of the model consists of two different large language models with small parameter scales(no more than 7B). Our training process is carried out in the following steps:

CCL 2024

We first introduce the notations in this section. Each notation s or t denotes a subset of the test set. The superscripts of each notation represent the step number from which the set is generated, and the subscripts of each notation represents the LLM from which the set is generated.

Step 1: First, we input the dataset into LLM1(i.e. Qwen1.5-7B). The data that generates correct outputs after training is passed to LLM2(i.e. ChatGLM3-6B))for further training, with the results (including prompt and request) denoted as s_2^1. The data that does not generate correct outputs after training is fed back into LLM1 for retraining, with the results denoted as t_1^1. LLM2 is then trained with s_2^1. The results that generate correct outputs after training are recorded as z_2^1, and those that generate incorrect outputs are recorded as t_2^1.

Step 2: z_2^1 is used to train LLM1, generating the correct result set s_2^2 and the incorrect result set t_1^2. Then s_1^2 is used to train LLM2, generating the correct result set z_2^2 and the incorrect result set t_2^2.

Subsequent steps follow this pattern, and the entire model structure is shown in the figure below.

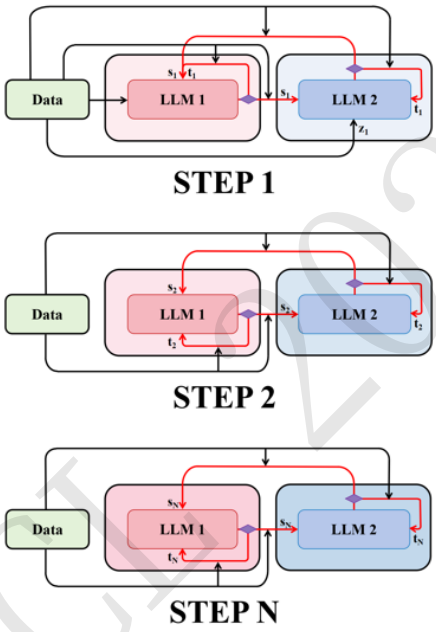

Figure 1: Training Process

4.2 Running and IU-Block

In this section, we mainly show how we design a Result Intersection and Union Block(IU-Block) model. We trained multiple different models and fused the results of these models by taking the intersection ($\cap$) and union ($\cup$).

The purpose of taking the intersection of the output results of two models is that we believe the results predicted to be the same by both models have a higher probability of being correct. At the same time, it excludes redundant incorrect predictions, reducing the total number of predictions and improving the recall rate of the model. The purpose of then taking the union of these intersections is to include as many correct outputs as possible, thereby improving the precision rate of the model.

Our IU-Block model is as follows: First we set the answer generated by the first model(i.e. Qwen1.5-7B) and the second model are A and B, then we may set $A \cap B$ or $A \cup B$ the ultimate answer.

However, the experimental results indicate that none of them are good enough.

This is not difficult to explain, as $A \cap B$ can lead to the model being too "conservative", meaning it tends to make fewer decisions to ensure the accuracy of the predictions made, resulting in higher recall rates. However, it can also lead to lower accuracy due to fewer correct predictions. On the other hand, $A \cup B$ can lead to the model being too "reckless", meaning it tends to make more decisions to ensure that all predictions made cover the majority of correct results, resulting in higher accuracy. However, it can also lead to lower recall rates due to the low proportion of correct predictions made among all predictions made.

So in order to balance the above two situations and fully utilize $A \cap B$ and $A \cup B$, we consider introducing a third large language model (here we introduce Yi1.5-6B, which has been fine tuned on the dataset) for supervision. Firstly, the correct predictions in $A \cap B$ account for a large proportion, so we consider directly incorporating them into the answer. For $A \cup B$, we consider using C (i.e. the predicted result of Yi1.5-6B on the test set) for supervision. If an element in $A \cup B$ appears in C, we believe it is likely to be a correct prediction and incorporate it into the answer. Otherwise, we consider it to be an incorrect prediction. In the end, we obtain the answer:

$$Answer = (A \cap B) \cup ((A \cup B) \cap C)$$

Simplifying the above equation, we have

$$Answer = (A \cap B) \cup (A \cap C) \cup (B \cap C)$$

This equation is not difficult to explain, as the above steps are based on the assumption that the results predicted to be the same by both models have a higher probability of being correct, which means that we take the union of the predicted results of any two models

Similarly, we can introduce a fourth model D (DeepSeek-7B introduced in this article) for supervision and the sets of their output results are denoted as A, B, C, and D, respectively. Then our IU-Block model is as follows: We trained four different large models, and the sets of their output results are denoted as A, B, C, and D, respectively. We perform the following operations on the output results of the four large models.

$$Answer = (A \cap B) \cup (A \cap C) \cup (A \cap D) \cup (B \cap C) \cup (B \cap D) \cup (C \cap D)$$

We take $Answer$ as the final output.

The establishment of the IU-Block model can improve both the recall rate and precision rate of the model, thereby enhancing the overall performance of the model.

5 Experiments

5.1 Experiments on Single-LLM Fine-tuning

5.1.1 Hyperparameter Tuning Experiment

Based on the parameter interface of the ERNIE-Speed-8K model, we made the following adjustments to the model parameters. The experimental data is shown in table 1, table 2 and table 3. below.

CCL 2024

It should be noted that the result of this experiment is tested on the blind test set. The last parameter combination achieved the highest F1 score, indicating the best model performance.

Training Set Size	Validation Set Size	F1/%
3000	1000	40.2
3000	1000	56.28
3000	1000	66.1
3600	400	71.6
4000	0	74.76

Table 1: Hyperparameter Tuning Process Part 1

Learning Rate	Epoch	Learning Rate Adjustment Plan
0.00002	3	linear
0.00002	3	linear
0.00002	6	linear
0.000025	6	cosine
0.000025	6	cosine

Table 2: Hyperparameter Tuning Process Part 2

Number of Cosine Cycle	Regularization Coefficient	Temperature	Diversity
–	0.01	0.95	0.8
–	0.01	0.95	0.8
–	0.01	0.95	0.8
0.5	0.009	0.95	0.8
0.5	0.009	0.6	0.6

Table 3: Continuation of Hyperparameter Tuning Process

5.1.2 Hyperparameter Tuning Process

Utilizing the Wenxin Qianfan's large-scale model platform, we visualized the loss and perplexity (ppl) during the fine-tuning process as shown in Figure 2.

CCL 2024

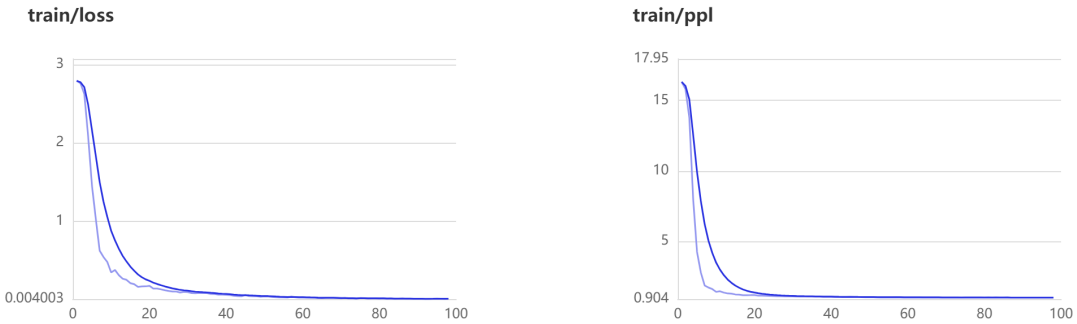

Figure 2: Changes in Loss and Perplexity During Training

5.1.3 Results

Our hyperparameter tuning results are shown in table 4.

Hyperparameter	Value
Number of Iterations	6
Gradient Accumulation Steps	4
Learning Rate	0.000025
End LR for Polynomial Strategy	1e-7
Learning Rate Adjustment Plan	cosine
Sequence Length	4096
Number of Cosine Cycles	0.5
Power for Polynomial Strategy	1
Fake Multi-Turn Probability	0
Checkpoint Saving Interval	256
Random Seed	42
Warmup Proportion	0.1
Regularization Coefficient	0.009
temperature	0.6
top-p	0.6

Table 4: Parameter Description

It should be noted that $F_1 = 74.76\%$ is the result we obtained after the competition. Compared to $F_1 = 71.6\%$, we adjusted the temperature and top-p from both 0.8 to 0.6, and adjusted the gradient accumulation steps from 0 to 4, while incorporating the validation set into the training set.

5.2 Experiments on Multi-LLM Ensembling and Tuning

This section mainly focuses on the performance of large models with smaller parameter scales. In our experiments, we introduced four large models: Qwen1.5-7B, ChatGLM3-6B, Yi1.5-6B, and DeepSeek-7B. First, we present the performance of these four large models after hyperparameter tuning

CCL 2024

and training 6 epochs when independently completing the task, as shown in table 5. The result of the experiment is tested on the test set.

LLM	Precision	Recall	F1
Qwen1.5-7B	0.6101	0.7053	0.6542
ChatGLM3-6B	0.5924	0.6196	0.6057
Yi1.5-6B	0.6337	0.6755	0.6539
DeepSeek-7B	0.6118	0.6757	0.6421

Table 5: Precision, Recall, and F1 Scores of Different LLMs

5.2.1 RC-Block and Cross-Fusion Experiment

In this experiment, we seperately trained the Qwen1.5-7B model, the ChatGLM3-6B model, the Yi1.5-6B model and the DeepSeek-7B model as candidates of the RC-Block, which is shown in figure 1. We recorded their results on the test set as A, B, C and D, respectively.

We compared introducing the models above with or without the method of RC-Block and whether the results are intersected after running the model. The results are shown in the table 6. The result of the experiment is tested on the test set.

RC-Block	Intersect	Precision	Recall	F1
False	False	0.6720	0.7390	0.7039
True	False	0.6694	0.7418	0.7037
False	True	0.6638	0.7395	0.6996
True	True	0.6780	0.7456	0.7102

Table 6: RC-Block Results with Intersection

The experimental results indicate that both the RC-Block and Cross-Fusion methods can effectively improve the performance of large models.

5.2.2 IU-Block Experiment

We completed the training of four models using the RC-Block method and generated outputs A and B for the Qwen1.5-7B and ChatGLM3-6B models using the Cross-Fusion method. The results C and D were obtained by running the test set on the other two models. The results were processed using the IU-Block in different ways, as shown in Table 7. The result of the experiment is tested on the test set.

CCL 2024

Method	Precision	Recall	F1
$A \cap B$	0.4527	0.8513	0.5911
$A \cup B$	0.7257	0.5649	0.6353
ABC+IU-Block	0.6152	0.7941	0.6933
ABD+IU-Block	0.6109	0.7956	0.6911
$ABCD$+IU-Block	0.6780	0.7456	0.7102

Table 7: Results of Different Methods

The experimental results indicate that the IU-Block can effectively improve the performance of large models.

6 Conclusion

This paper proposes a novel method for constructing CPG, significantly improving the accuracy of their construction. By fine-tuning large language models, a model capable of constructing CPG with high accuracy was developed. Considering the substantial time and space requirements of large language models with extensive parameters, this paper explores the use of smaller-scale models for this task. A system integrating multiple small-scale models and a fusion runtime system was constructed. This approach successfully used smaller-scale models to construct CPG, achieving similar results to larger models but with significant savings in both runtime and space utilization.

CCL 2024

Acknowledgements

We would like to express our sincere gratitude to Professor Xin Xin from Beijing Institute of Technology, for his guidance and instruction in the Knowledge Engineering course, which provided the foundation for this project. Additionally, we would like to thank the CPG team at Beijing Language and Culture University, particularly Professor Endong Xun, Gaoqi Rao, Gongbo Tang and graduate studednt Mengxi Guo, for their support and assistance in this project.

References

Kommineni, Vamsi Krishna and König-Ries, Birgitta and Samuel, Sheeba 2024. *From human experts to machines: An LLM supported approach to ontology and knowledge graph construction* arXiv preprint arXiv:2403.08345.

Vizcarra, Julio and Haruta, Shuichiro and Kurokawa, Mori. 2024. 2024 IEEE 18th International Conference on Semantic Computing (ICSC), 231–232. IEEE.

Carta, Salvatore and Giuliani, Alessandro and Piano, Leonardo and Podda, Alessandro Sebastian and Pompianu, Livio and Tiddia, Sandro Gabriele. 2023. terative zero-shot llm prompting for knowledge graph construction. arXiv preprint arXiv:2307.01128.

Yu, Shuang and Huang, Tao and Liu, Mingyi and Wang, Zhongjie. 2023. BEAR: Revolutionizing Service Domain Knowledge Graph Construction with LLM, 339–346. Springer.

郭梦溪 and 荀恩东 and 李梦 and 饶高琦. 2024. 意合图：中文多层次语义表示方法. 第二十三届中国计算语言学大会.

郭梦溪 and 荀恩东 and 李梦 and 饶高琦. 2024. 基于意合图语义理论的结构标注体系与资源建设. 第二十三届中国计算语言学大会.

Aghajanyan, Armen and Yu, Lili and Conneau, Alexis and Hsu, Wei-Ning and Hambardzumyan, Karen and Zhang, Susan and Roller, Stephen and Goyal, Naman and Levy, Omer and Zettlemoyer, Luke. 2023. Scaling Laws for Generative Mixed-Modal Language Models. Proceedings of Machine Learning Research.

CCL 2024