Self-Instruct CAAFE

Yanheng He, Jiahe Jin, Yuxuan Zhang

Abstract

With the advancement of large language models (LLMs), integrating them effectively into data science workflows is becoming increasingly important. Based on the previous work of Context-Aware Automated Feature Engineering (CAAFE), we propose an improved approach, Self-Instruct CAAFE, enhancing the capability of LLM as feature engineering agent by supervised fine-tuning (SFT). Specifically, we leverages a base LLM to iteratively generate semantically meaningful features based on dataset descriptions like CAAFE, and automatically log and filter high-quality responses for SFT.

Experimental results demonstrate that the CAAFE performance of the fine-tuned model improves significantly — boosting mean ROC AUC performance from 0.812 (with no feature engineering) to 0.860 on 10 OpenML datasets - far surpassing the performance of the base model, which is 0.823. This proves the effectiveness of our Self-Instruct method.

We also replace the GPT-4 (or GPT-3.5) model used in CAAFE with a smaller open-source model Llama3-8B-Instruct as the base model. By simplifying the prompt templates and optimizing feedback mechanism, we achieve comparable CAAFE performance with lower hardware requirements, making advanced feature engineering much more accessible. Code and data are available at GitHub [1].

1 Introduction

Large Language Models (LLMs), such as ChatGPT, have become essential agents in the field of data science. They provide a highly universal interface for a general-purpose assistant, capable of understanding and generating natural language, making them useful for various data science tasks. Despite their vast potential, effectively integrating them into specific data science workflows, particularly for tabular data feature engineering, remains challenging. This integration is crucial as feature engineering significantly impacts machine learning model performance.

Context-Aware Automated Feature Engineering (CAAFE) has emerged as a promising approach to address this challenge. CAAFE utilizes an LLM to iteratively generate additional semantically meaningful features for tabular data based on the description of the dataset and retains those good. This simple method improves machine learning performance across multiple datasets and enhances interpretability by creating both Python code for new features and explanations for their utility.

In this work, we introduce *Self-Instruct CAAFE*, an enhanced approach of CAAFE which combines its foundational principles with the innovative Self-Instruct method, as visualized in Figure 1. The Self-Instruct method engages in supervised learning by automatically logging and filtering the high quality responses generated by the LLM during the CAAFE process. These data are then used to perform a supervised fine-tuning of the original model, and successfully enhancing the effectiveness of CAAFE.

In particular, our work makes the following contributions:

[1] https://github.com/LLM-class-group/Self_instruct_CAAFE

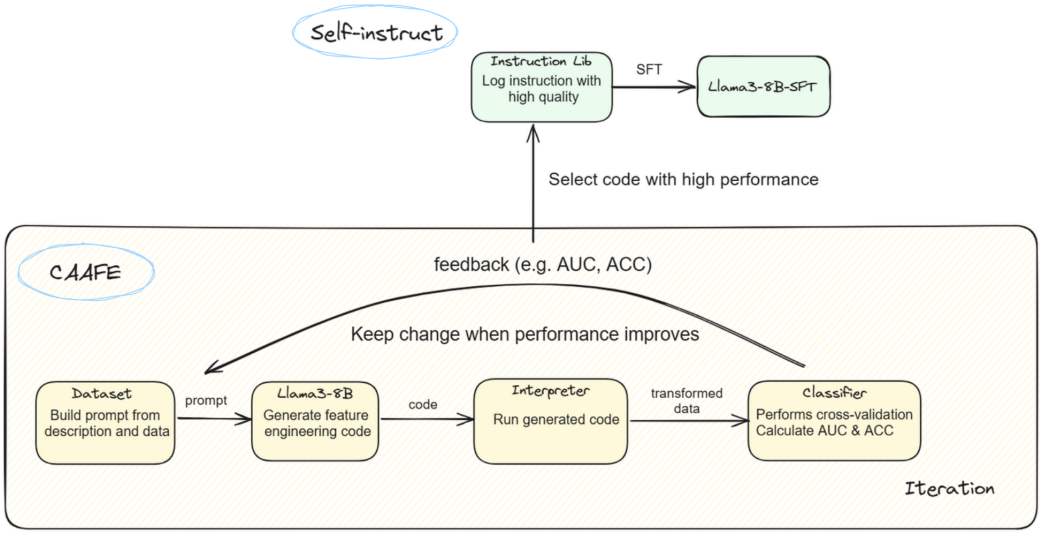

Figure 1: Self-Instruct CAAFE

- **Self-Instruct Method**. We present an experience accumulating method and pipeline for building fine-tuning data automatically. Through supervised fine-tuning of the base model, we elevate the performance of CAAFE to new heights.
- **Smaller Open-Source Base Model**. We replace the GPT-4 (or GPT-3.5) model used in CAAFE with a much smaller open-source model, Llama3-8B (Instruct), while maintaining comparable results. Llama3-8B can be deployed on a single consumer-grade GPU (e.g. RTX 3090), significantly lowering computational barriers and making advanced feature engineering accessible on more affordable hardware.
- **Optimized CAAFE Pipeline**. We simplify the prompt templates used in CAAFE for faster inference and lower memory requirements of SFT, and also optimize the code execution error feedback mechanism for better performance, especially on smaller model like Llama3-8B.

2 Related Work

Context-Aware Automated Feature Engineering (CAAFE) [1]. CAAFE is an automatic feature engineering method for tabular datasets by using an LLM to generate semantically meaningful features and explanations of their utility iteratively, while retaining those features that improve classification performance. CAAFE combines the scalability and robustness of classical machine learning classifiers (e.g. Random Forest) with the extensive domain knowledge embedded in LLMs.

Self-Instruct [3]. Self-Instruct is a classic machine learning strategy aimed at enhancing a model's performance on specific tasks with minimal reliance on manually annotated data. This approach leverages the model's own capabilities to generate training instructions, making the learning process more efficient and scalable. Our work builds on the principles of Self-Instruct by employing a similar self-guided enhancement methodology to improve the capability of LLMs.

3 Method

We present Self-Instruct CAAFE, an enhanced approach of CAAFE with shorter prompts, better feedback mechanism and innovative self-instruct method, achieving unprecedented automatic feature engineering performance. We use Llama3-8B-Instruct as the base model, which is an open-source LLM developed by Meta with much smaller parameter size compared to GPT-4 or GPT-3.5.

3.1 Shorter Prompts

In CAAFE, the prompt used to instruct LLM to create new features includes the dataset description, existing feature names, data types and samples, as well as explicit instructions for generating code and explanation. The dataset and its description are directly downloaded from websites like OpenML or Kaggle. Hence, sometimes we encounter descriptions that are excessively long and filled with redundant information. This would lead to exceptionally lengthy prompt (reaching over 2000 tokens), reducing inference speed and demanding more GPU memory for fine-tuning.

To address this issue, we let the LLM to summarize the dataset description before generating features, retaining only the parts that are meaningful for feature engineering. We also reduced the number of samples for each feature and shortened the prompt template used in CAAFE. Together, without compromising the quality of the model's responses, we reduced the average length of the prompts by nearly 50%.

Dataset description:

The dataset contains tablebases for the game "Jungle Chess" (Dou Shou Qi) with different piece combinations, such as Rat vs Rat, Rat vs Panther, and so on. The dataset is suitable for classification tasks.

Feature:

```
white_piece0_strength (float64): Sample [6.0]
white_piece0_file (float64): Sample [1.0]
white_piece0_rank (float64): Sample [6.0]
black_piece0_strength (float64): Sample [5.0]
black_piece0_file (float64): Sample [2.0]
black_piece0_rank (float64): Sample [6.0]
class (category): Sample [0.0]
```

Code and explanation:

```
# Average piece strength ratio
# Usefulness: The average strength ratio can provide a better indication of the game's outcome.
# Input samples: ((white_piece0_strength + black_piece0_strength) / 2) / ((white_piece0_strength /
# black_piece0_strength) + (white_piece0_strength - black_piece0_strength)), ((6.0 + 5.0) / 2) / ((6.0 / 5.0) +
# (6.0 - 5.0)), ((4.0 + 6.0) / 2) / ((4.0 / 6.0) + (4.0 - 6.0))

df['average_piece_strength_ratio'] = ((df['white_piece0_strength'] + df['black_piece0_strength']) / 2) /
((df['white_piece0_strength'] / df['black_piece0_strength']) + (df['white_piece0_strength'] - df['
black_piece0_strength']))

# Drop redundant columns
df.drop(columns=['white_piece0_strength', 'black_piece0_strength'], inplace=True)
```

Figure 2: Prompt example (after abbreviation)

3.2 Better Feedback Mechanism

When reproducing CAAFE, we found that if the LLM generates non-executable feature engineering code, CAAFE retains these code in the chat history and attempts to have the LLM correct it. However, not all LLMs possess the same powerful capabilities as GPT-4. For relatively weaker models like Llama3-8B, correcting code is challenging, and the model continues to output different but similar erroneous codes.

Our solution is to retain only the correct codes previously generated in the chat history. When newly generated feature engineering code encounters an error in execution, we temporarily add this code to the history and let the LLM generate a different feature instead of correcting it. After generating the next code, we will remove the previous erroneous code from the chat history. With this optimization, LLM can generate other features after producing erroneous code, rather than making futile efforts to fix error. This significantly increases the probability of LLM generating feasible features.

3.3 Self-Instruct Data Generation

Fine-tuning refers to the process of further training a pre-trained LLM using a specific dataset to adapt the model to a particular task or domain. Based on CAAFE, we proposed a innovative method for automatically constructing high-quality fine-tuning datasets for feature engineering and successfully achieved much better performance through fine-tuning the base model. Specifically, within the CAAFE pipeline, newly generated feature engineering code by the LLM undergo cross-validation, and those that enhance AUC and accuracy are retained. Simultaneously, we apply stricter criteria than CAAFE to filter and record high-quality responses, which are then used as fine-tuning data.

Considering computing resource constraints, our fine-tuning data cannot be too lengthy and thus cannot include the entire chat history. However, the code for new features sometimes leverages model-generated features rather than just the original features. To ensure that the utilized features appear in the prompts, we integrate CAAFE prompts and chat history into a single prompt: its structure resembles the initial prompt, but the data types and samples reflect modifications made to the dataset during the CAAFE process. Ultimately, we generated a total of 293 high-quality responses of moderate length as fine-tuning data across 10 OpenML datasets.

4 Experiments

4.1 Fine-Tuning

After the construction of our Self-Instruct dataset, we utilized the LLaMA-Factory [4] and LoRA [2] method for fine-tuning the base model, Llama3-8B-Instruct. We completed the model training on an RTX 3090 within 3.5 hours, with a learning rate of 1e-4 and a batch size of 16. FP16 is enabled to achieve a balance between speed and precision, and saving GPU memory.

4.2 Evaluation on OpenML Datasets

The training data for the LLM originates from web sources, potentially encompassing open datasets. The pre-training corpus for Llama3-8B was finalized in March 2023. For our evaluation, we selected OpenML datasets consistent with those used in the paper of CAAFE [1], all of which were published before 2023 and widely recognized. This implies that Llama3-8B may have learned from these datasets during pre-training and leveraged this information in feature engineering. Each dataset includes a textual description and feature names, which we incorporated into the model prompts along with additional information.

We classified the feature-engineered data using three downstream classifiers and used AUC as the evaluation metric. After adapting Llama3-8B to CAAFE pipeline, the model demonstrated improvements on most of the 10 datasets compared to raw data without feature engineering, as shown in Table 1. The feature-engineered data generated by the fine-tuned model exhibited further improvements for each classifier, with classification performance increasing with the number of iterations, as illustrated in Table 2.

Table 1: The AUC results of TabPFN on 10 OpenML datasets, iteration time = 10, random seed = 42.

Dataset	**No Feat. Eng.**	**CAAFE with Llama3-8B**
balance-scale	1.0000	1.0000
breast-w	1.0000	1.0000
cmc	0.7672	0.7671
credit-g	0.7333	0.7467
diabetes	0.8639	0.8667
tic-tac-toe	0.3810	0.4762
eucalyptus	0.9282	0.9278
pc1	0.8669	0.8649
airlines	0.7324	0.7704
jungle-chess...	0.8438	0.8579
Average	0.8117	**0.8278**

Table 2: The mean AUC results of three different classifiers on 10 OpenML datasets. We changed the random seed and evaluated for twice, calculating the variance of the AUC. After fine-tuning, Llama3-8B achieves great improvement on all of three classifiers.

Method	TabPFN	Random Forest	XGBoost
No Feat. Eng.	0.8117	0.7836	0.7790
Llama3-8B (Iter=3)	0.8189 ± 0.0059	0.8219 ± 0.0068	0.7862 ± 0.0052
Llama3-8B (Iter=10)	0.8232 ± 0.0045	0.8167 ± 0.0115	0.8136 ± 0.0109
Llama3-8B-SFT (Iter=3)	0.8447 ± 0.0055	0.8403 ± 0.0019	0.8039 ± 0.0054
Llama3-8B-SFT (Iter=10)	**0.8595** ± 0.0093	**0.8561** ± 0.0038	**0.8221** ± 0.0093

5 Conclusion

Our study proposes an improved method of CAAFE by using a smaller open-source base model and further enhancing the performance by fine-tuning. By optimizing CAAFE pipeline, particularly through shortening the prompt templates and improving error feedback mechanism, we successfully adapted CAAFE to a smaller open-source base model, Llama3-8B-Instruct, without compromising the feature engineering performance. This adaptation significantly lowers the usage threshold and cost of CAAFE while increasing the speed. We also introduce a innovative method and pipeline for automatically accumulating experience to construct a high-quality fine-tuning dataset for feature engineering. Based on this, we fine-tuned Llama3-8B-SFT, elevating the performance of CAAFE to unprecedented levels. We hope our work can inspire future research on building more capable LLM agents for data science.

However, our current method has some limitations. Our preliminary exploration in experience generalization shows that about 1K instructions across 41 datasets is insufficient for Llama3-8B to generalize these feature engineering experiences to unseen datasets. Due to computing resources and experimental time constraints, we were unable to conduct larger-scale experience accumulation generalization experiments. We hypothesize that larger-scale experience accumulation from more datasets is required to meet the Scaling Law requirements and achieve emergent generalization capabilities, further exploration is needed in the future.

Acknowledgements

We thank Muning Wen for valuable discussions on our work. We thank the Meta Llama team for giving us access to their models, and open-source projects, including vLLM and LLaMA-Factory. We thank the Apex Lab for providing us computing resources.

References

[1] Noah Hollmann, Samuel Müller, and Frank Hutter. Large Language Models for Automated Data Science: Introducing CAAFE for Context-Aware Automated Feature Engineering. *arXiv preprint arXiv:2305.03403*, 2023.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022.

[3] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Model with Self Generated Instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[4] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. *arXiv preprint arXiv:2403.13372*, 2024.