Table-To-Text Using Pre-trained Large Language Model and LoRA

Hidekazu Yanagimoto
Graduate School of Informatics
Osaka Metropolitan University
Osaka, Japan
hidekazu@omu.ac.jp

Iroha Kisaku
School of Knowledge and Information Systems
Osaka Prefecture University
Osaka, Japan
sfa00090@st.osakafu-u.ac.jp

Kiyota Hashimoto
Faculty of Information Science
Shunan University
Yamaguchi, Japan
hash@g.shunan-u.ac.jp

2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI) | 979-8-3503-7790-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/IIAI-AAI63651.2024.00026

***Abstract*—This paper propose a Table-To-Text system that generates text explaining the contents of a table from the table itself. To accurately capture the information in the table, it is represented in JSON format. The use of JSON format allows for handling not only table data but also various other types of information, such as sensor data in IoT. To develop this system, we fine-tune an existing pre-trained large language model with a Table-To-Text dataset. Furthermore, to reduce the computational cost of fine-tuning, we utilize Low-Rank Adaptation (LoRA). Our evaluation experiments show that while large language models without fine-tuning is not able to solve the Table-To-Text tasks, our proposed system is able to generate text based on tables. We have confirmed that fine-tuning with LoRA can extend large language models to new tasks that were not anticipated during pre-training. As future work, we will develop Table-To-Text system using other large language model.**

***Index Terms*—Table-To-Text, large language model, natural language processing, deep learning**

I. Introduction

Dialog systems like ChatGPT [1], which use large language models built on extensive training data as foundational models, are attracting significant attention because they can solve various problems by asking question [2]. As a future development of large language models, the ability to process information that differs from natural language, such as images and programming codes, is hilighted. If a large language model can handle various formats of data, it can be used for various tasks such as data analysis and data mining. For example, if data from sensors obtained via IoT can be converted into text, an observer can be easier to understand the system's status.

In this paper, we address the TableToText [3]–[5], specifically representing tables in JSON format [6] and generating a text based on the content of the table. Fig. 1 shows an example of Table-To-Text. A Table-To-Text system accepts data representing a table and elements that should be highlighed, and generates text that describes the content of the table. Additionally, by using the JSON format, it is also possible to apply the proposed system to a broader range of Data-To-Text application beyond just Table-To-Text. For implementing Table-To-Text systems, researchers employ Sequence-To-Sequence models [3], [8], VTM, which is modified Variational AutoEncoder [4], and prompt-based approach [5].

In this paper, we tackle with Table-To-Text using an approach that fine-tunes a pre-trained large language model. Additionally, to reduce the cost for fine-tuning, we employ Low-rank Adaptation (LoRA) [9] to decrease the number of trainable parameters, thereby lowering computational costs.

Our contributions of this paper are as floows.

- We develop a Table-To-Text system using exiting pre-trained large language model.
- We reduce the computational cost of fine-tuning of a pre-trained large language model using LoRA.

The organization of this paper is as follows: Section II describes the proposed system: 2.A introduces a large language models; 2.B describes an Alpaca format; 2.C explains LoRA. Section III reports experiments: 3.A reports dataset for the experiment; 3.B explain OPT-350m in detail; 3.C describes hyper-parameter settings; 3.D discusses a evaluation metric, ROUGE; 3.E describes evaluation of text generation.

II. Fine-tuning Large Language Model for Table-To-Text with LoRA

The proposed system realizes Table-To-Text by fine-tuning the existing pre-trained large language model. Specially, instead of modifying the pre-trained large language model itself, we achieves fine-tuning for Table-To-Text using LoRA, which fixes the pre-trained large language model and add trainable rank decomposition matrices. Fig. 2 shows the architecture of our proposed system.

The proposed method represents tabular data in JSON format and passes it to the large language model for text generation in Table-To-text. The text generation module includes a pre-trained large larnguage model and a LoRA module tailored to Table-To-Text tasks. The LoRA module is trained with Table-To-Text specific training data.

A. Large Language Model

Large language models are foundational models pre-trained with various kinds of vast training data. Existing large language models typically consist of multiple stacked Transformers. For example, in the open-source LLaMA2 offers pre-trained models ranging from 7 billion to 70 billion parameters. These pre-trained model have acquired the vocabulary and grammatical knowledge necessary for text generation. However, since they are generally designed to handle plain text

979-8-3503-7790-3/24/$31.00 ©2024 IEEE
DOI 10.1109/IIAI-AAI63651.2024.00026

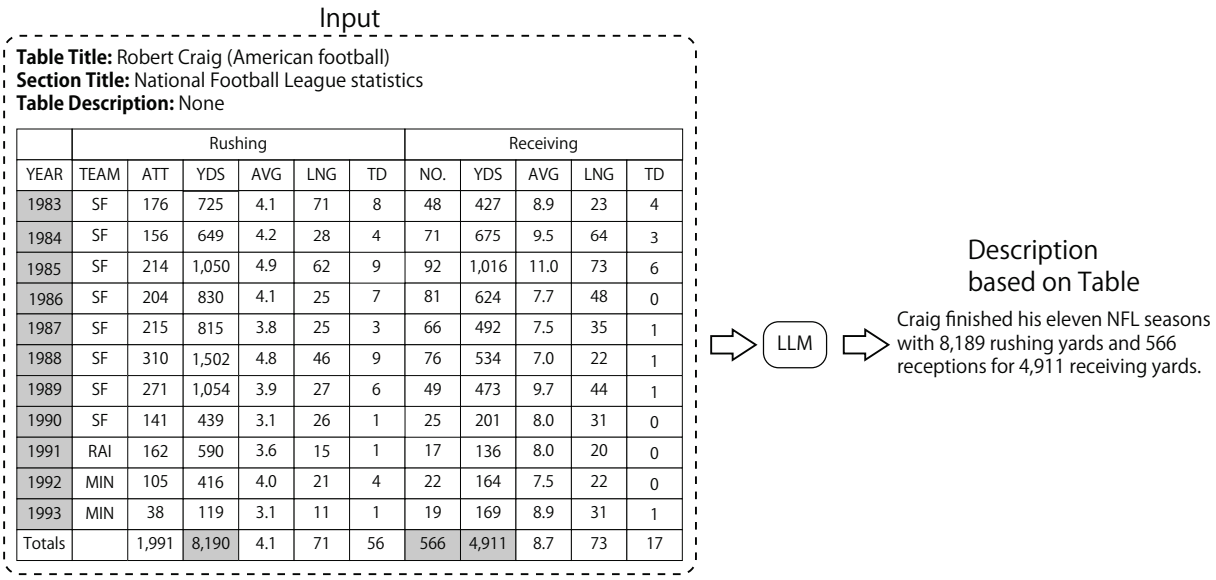

Fig. 1. An example of Table to Text task. In this task, description text is generated by specifying elements to focus on, along with information such as a table represented in JSON format and a table title.

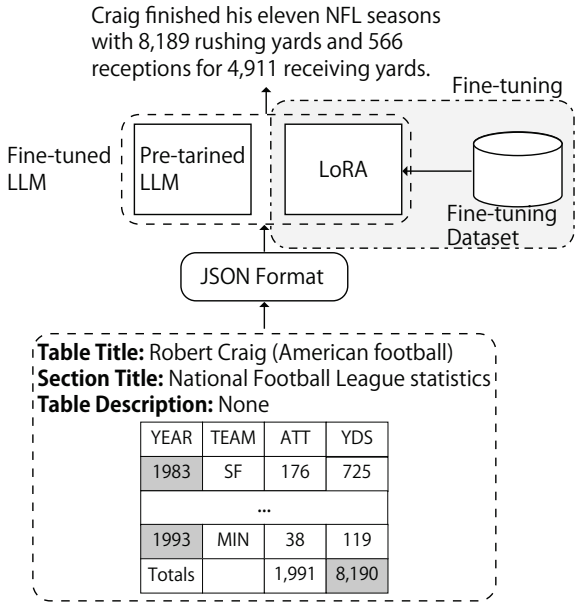

Fig. 2. Architecture of the proposed system for Table-To-Text including the LoRA module.

inputs, directly applying them to Table-To-Text tasks can be hard.

Prompt engineering [7] is gaining attention as a way to obtain appropriate response from large language models. A prompt [10] is a description of the task that the large language model should work, written in natural language. By using the prompts, it is believed that various task-specific problems can be solved without any fine-tuning. However, the number of tokens that large language models can accept is generally limited, and adding descriptions of a table for Table-To-Text tasks may often exceed the token limits. Furthermore, it is very challenging to create a prompt that allows the model to properly understand inputs in JSON format and generate an appropriate response. Fig. 3 shows an example of Table-To-Text using a pre-trained large language model without fine-tuning. In this case, the large language model was unable to

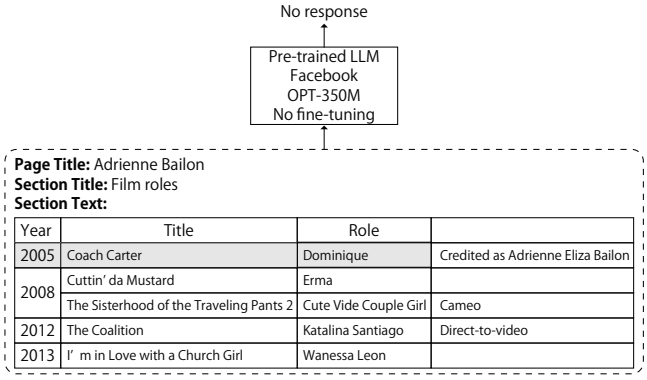

Fig. 3. Text generation using a large language model without fine-tuning. In this case, LLM cannot generate any text.

generate text describing the table. Here, we are using prompts in the Alpaca format, which will be explained next.

B. Alpaca Format as LLM Input

For Table-To-Text tasks, the necessary data is provided to a large language model as a prompt in Alpaca Format [11]. This format is used during both fine-tuning and prediction stages. Table I shows a prompt in Alpaca format which is generated

from input data for Table-To-Text. This prompt consists of

TABLE I
EXAMPLE OF ALPACA FORMAT FOR FINE-TUNING AN LARGE LANGUAGE MODEL

```
### Instruction:
Make text according to the following JSON data. Absolutely, the
following words are included in the response text.
{keywords}

### Input:
{Table Title}, {Section Title}, {Table Description}
{Table in JSON format}

### Response:
{Description based on Table}
```

three fields. First, the "### Instruct" field roughly outlines the task instruction, the "### Input" field provides the information necessary to solve the task, and the "### Response" field specifies what output is desired. In the prediction phase, the {Description based on Table} field is not added and remains blank. The {keyword} in Alpaca format consists of words and numerical values hilighted in the table.

The {Table in JSON format} is generated from the table, as shown in Fig.4. The JSON format is converted from the table and corresponds with the table, ensuring that no information is loss.

C. LoRA

To implement Table-To-Text using a pre-trained large language model, training data consisting of tables and their descriptions is prepared and we fine-tunes the large language model with it. However, directly modifying the parameters of the large language model can significantly needs much memory and computation power for fine-tuning. So, in this research, we use LoRA instead. Fig. 5 shows a fine-tuning mechanisms.

Not to modify the parameters of the large language model itself, a module with trainable parameters is added. However, to reduce the number of parameters in the module, the added module is a fully connected neural network that has a smaller hidden layer. The network structure means low-rank matrix composition. Currently, we consider the embedding dimensions of a word as d. In this case, since the input and the output of the added module are the dimensions of a word, it is represented as a $d \times d$ matrix. If all elements in the matrix are trainable, it would result in d^2 trainable parameters. On the other hand, if the network is designed as shown in Fig. 5, there exist two matrices of dimension $d \times r$ and the number of all trainable parameters becomes $2dr$. If r is small compared to d, this allows for a significant reduction.

The large language model incorporating LoRA operates as follows.

$$\mathbf{h} = W\mathbf{x} + \Delta W\mathbf{x} = W\mathbf{x} + BA\mathbf{x} \quad (1)$$

where, $\mathbf{h}$ represents the output of the Transformer in the large language model, $\mathbf{x}$ represents the vector generated from a token, W denotes the weights of the entire large language

Table

Year	Title	Role
1997	Eek! The Cat	Himself

JSON format

```
[
  [
    {   "column_span": 1,
        "is_header": true,
        "row_span": 1,
        "value": "Year"},
    {   "column_span": 1,
        "is_header": true,
        "row_span": 1,
        "value": "Title"},
    {   "column_span": 1,
        "is_header": true,
        "row_span": 1,
        "value": "Role"}
  ],
  [
    {   "column_span": 1,
        "is_header": false,
        "row_span": 1,
        "value": "1997"},
    {   "column_span": 1,
        "is_header": false,
        "row_span": 1,
        "value": "Eek! The Cat"},
    {   "column_span": 1,
        "is_header": false,
        "row_span": 1,
        "value": "Himself"}
  ]
]
```

Fig. 4. Conversion from table format to JSON format

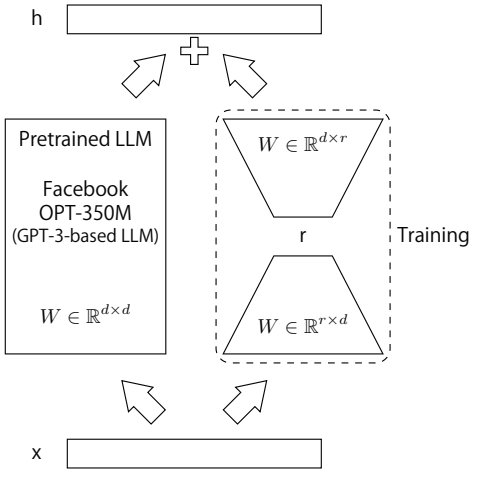

Fig. 5. Fine-tuning mechanism

model, and ΔW is the matrix representation of the module added by LoRA. ΔW is represented as the product of matrices A and B.

The matrices A and B are updated based on prediction errors like standard training of the language model. During the training, update amount is scaled by $\frac{\alpha}{r}$.

III. Experiments

We fine-tune a pre-trained large language model on a moderately sized dataset and evaluate the generated text using data not utilized in the fine-tuning process. For evaluation, we use ROUGE score [12] that compares the generated text with the correct text provided as a reference.

A. Dataset

In this experiment, we use ToTTo dataset [13], which is provided for Table-To-Text tasks by Google. The ToTTo dataset consists of 120,761 training data, 7,700 validation data, and 77,000 test data. Each entry includes information on a table and a description based on the table. The test data do not include correct description Therefore, in this experiment, we fine-tuned a large language model using the training data and evaluated the performance using the validation data. Table II shows the content of the ToTTo dataset.

TABLE II
THE CONTENT OF TOTTO DATASET

training data	120,761
validation data	7,700
test data	7,700

The ToTTo dataset is converted into the Alpaca format to use in both fine-tuning and prediction.

B. Large Language Model

In this experiment, we used the OPT-350m [14], a pre-trained large language model provided by Meta AI. OPT-350m is a GPT-3 based language model with 350 million parameters. It has been pre-trained on a large dataset, obtaining a substantial vocabulary and grammatical knowledge. Additionally, the we selected a language model with 350 million parameters and we did not reduce the precision of floating-point parameters during fine-tuning. Speaking concretely, we employed a double-precision floating-point number. Quantization [15], which converts double-precision floating-point number to single-precision floating-point number, is known widely but we do not use quantization here due to uncertain effect on the accuracy of Table-To-Text.

C. Parameter Settings

The proposed system includes some hyper-parameters and the hyper-parameters' settings is shown in Table III.

Since r is set to 8, ΔW becomes a low-rank matrix with a rank of 8. Since OPT-350m has a limitation number of tokens it can accept, it cannot use overly long table data. So, only 5,763 data from the validation data were used in this experiments. Handling tables that require more tokens is a future challenge. However, since larger language models can process more tokens than OPT-350m, they can deal with more complicated tables. Another approach is that selecting and refining input table can help reduce the number of necessary tokens.

TABLE III
HYPER-PARMETERS' SETTINGS FOR THE PROPOSED SYSTEM

Hyper-parameter	Value
α in LoRA	32
r in LoRA	8
The size of training data	120,761
The size of validation data	5,763
Max epoch	10

D. Evaluation

A generated text is evaluated with Rouge. The Rouge scores are used to evaluate text summarization and measure how many words are shared between a reference text and a generated text. Rouge-N is defined in Eq. (2) and Eq. (3)

$$\text{Precision} = \frac{\sum_{S\in\text{Reference}}\sum_{\text{gram}_N\in S}\text{count}_{\text{match}}(\text{gram}_N)}{\sum_{S\in\text{Prediction}}\sum_{\text{gram}_N\in S}\text{count}(\text{gram}_N)} \quad (2)$$

$$\text{Recall} = \frac{\sum_{S\in\text{Reference}}\sum_{\text{gram}_N\in S}\text{count}_{\text{match}}(\text{gram}_N)}{\sum_{S\in\text{Reference}}\sum_{\text{gram}_N\in S}\text{count}(\text{gram}_N)} \quad (3)$$

In a Rouge-N score, the score is based on the proportion of n-gram words shared between the reference text and the generated text. 1-gram represents a single word, while an n-gram represents a sequence of n consecutive words. $\text{count}_{\text{match}}(\text{gram}_\text{N})$ means the number of n-gram words shared between a reference text and a generated text. On the other hand, $\text{count}(\text{gram}_\text{N})$ means the number of n-gram words in text. Since this n-gram word is evaluated for all reference text, in Eq. (2), the sum are performed for all reference text.

Moreover, Rouge-L is defined in Eq. (4) and Eq. (5).

$$\text{Precision} = \frac{\text{LCS}(p,r)}{\text{The number of words in } p} \quad (4)$$

$$\text{Recall} = \frac{\text{LCS}(p,r)}{\text{The number of words in } r} \quad (5)$$

,where $\text{LCS}(p,r)$ denotes the number of longest common subsequences between a generated text, p, and a reference text, r. Rouge-L is calculated based on the longest common sequences between a generated text and a reference text. Additionally, we define an F-measure score which is a harmonic average.

$$\text{F}-\text{measure} = \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \quad (6)$$

E. Results

Fig. 6 shows the changes in error when the proposed method, OPT-350m with LoRA, is fine-tuned. As the fine-tuning progresses, the error decreases. It means that the learning is processing correctly. In the following experiment,

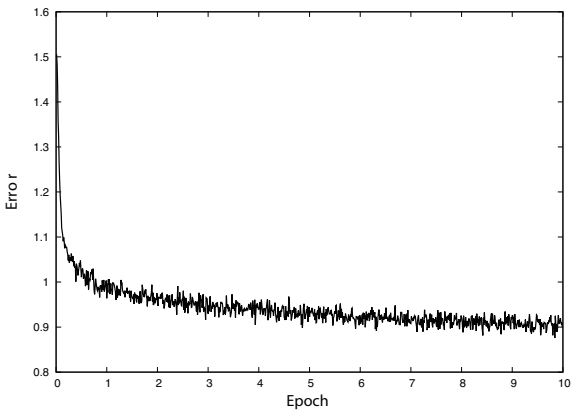

Fig. 6. Learning curve of the propose method

we used the model at the stage where 10 epochs have been completed.

Table IV presents Rouge scores to evaluate texts predicted with the proposed method and correct texts.

TABLE IV
GENERATED TEXT EVALUATION WITH ROUGE SCORE

	Score
Rouge-1 Precision	0.463
Rouge-1 Recall	0.393
Rouge-1 F-measure	0.389
Rouge-2 Precision	0.220
Rouge-2 Recall	0.189
Rouge-2 F-measure	0.185
Rouge-3 Precision	0.114
Rouge-3 Recall	0.099
Rouge-3 F-measure	0.096
Rouge-L Precision	0.380
Rouge-L Recall	0.317
Rouge-L F-measure	0.315

OPT-350m without fine-tuning cannot generate any sentences from all the validation data. Hence, the ROUGE scores could not be calculated for the original OPT-350m. This means that fine-tuning enables the model to solve Table-To-Text tasks and a simple prompting strategy is insufficient to solve Table-To-Text task. It has been confirmed that fine-tuning with LoRA is effective for extending large language models to different tasks. A comparison of Rouge-N in Table-To-Text tasks with other methods [16] has not been conducted and remains future work.

Since it is difficult to directly discuss Rouge-N scores, examples of the generated text are shown in Fig. 7 and Fig. 8.

In Fig. 7, appropriate text is generated according to the content and title of the table. OPT-350m, without fine-tuning, is unable to generate text. However, significant improvements are achieved with around 10 iterations of fine-tuning using LoRA. Furthermore, since the pre-training likely did not include training data related to Table-To-Text tasks, it appears

Table Title: The Weight of These Wings
Section Title: Awards
Table Description: None

Year	Association	Category	Result
2017	ACM Awards	Album of the Year	Won
	CMA Awards	Album of the Year	Nominated

Reference: The Weight of These Wings won Album of the Year at the 2017 ACM Awards.
Generation: The Weight of These Wings won the ACM Awards for Album of the Year.

Fig. 7. Examples of the correctly generate text based on tables.

Table Title: 1991 Washington Redskins season
Section Title: Passing
Table Description: None

Player	G	Comp.	Att.	Pct.	Yds.	TD	INT
Mark Rypien	16	249	421	59.1	3,564	28	11
Jeff Rutledge	16	11	22	50.0	189	1	0

Reference: Mark Rypien scored 3,564 passing yards in the 1991 Washington Redskins season.
Generation: Rypien had a career-high 27 receptions for 4,564 yards and 21 touchdowns.

Fig. 8. Examples of the incorrectly generate text based on tables

that the model has gained the ability to solve new tasks through fine-tuning with LoRA.

On the other hand, Fig. 8 shows that the model fails to generate correct text. Although the generated text is grammatically correct, the content is inaccurate, especially with incorrect numerical values. This indicates that the model still utilize the grammatical knowledge acquired during pre-training even after fine-tuning.

IV. CONCLUSIONS

In this paper, we solved the Table-To-Text by fine-tuning an existing pre-trained large language model using LoRA. By utilizing the ToTTo dataset, the proposed method enabled solving the Table-To-Text. The proposed method is effective, compared to a large language model without fine-tuning, which cannot solve the Table-To-Text. The evaluation experiment achieved a ROUGE-L F-measure score of 0.315 but we need further discussion regarding the Rouge scores. Upon reviewing the generated text with the proposed method, it is evident that fine-tuning with LoRA results in more appropriate text generation. However, for texts involving numbers, the generated content was not always accurate.

As a future works, we will evaluate the impact of quantization on Table-To-Text. By quantization, we can use larger language models and increases the number of pre-trained large language models available. In this paper, we used OPT-350m but we will conduct experiments using other pre-trained large

language models. Since the characteristics are different depending on the datasets used for pre-training and the structure of the large language models, performance differences are expected even among similar scaled language models.

REFERENCES

[1] OpenAI, "GPT-4 Technical Report", CoRR, 2023.

[2] C. D. Manning, "Human Language Understanding & Reasoning", Daedalus, Vol. 125, No. 2, pp.127–138, 2022.

[3] T. Liu, K. Wang, L. Sha, B. Chang, and Z. Sui, "Table-To-Text generation by Structure-Aware Seq2Seq Learning", Proc. of The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pp.4881–4888, 2018.

[4] R. Ye, W. Shi, H. Zhou, Z. Wei, and L. Li, "Variational Template Machine for Data-To-Text Generation", Proc. of ICLR2020, 2020.

[5] A. Aghajanyan, D. Okhonko, M. Lewis, M. Joshi, H. Xu, G. Ghosh, and L. Zettlemoyer, "HTLM: Hyper-Text Pre-Training and Prompting of Language Models", Proc. of ICLR2022, 2022.

[6] D. Crockford "The application/json Media Type for JavaScript Object Notation (JSON)", RFC4627, 2006.

[7] OpenAI, "Language Models are Unsupervised Multitask Learners", CoRR, 2019.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks", Proc. of the 27th International Conference on Neural Information Processing Systems, Vol. 2, pp.3104-3112, 2014.

[9] E. J. Hu, UY. SHen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models", Proc. of ICLR 2022, 2022.

[10] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models", Proc. of 36th Conference on Neural Information Processing Systems (NeuIPS 2022), 2022.

[11] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishizi, "Self-Instruct: Aligning Language Model with Self Generated Instructions", CoRR, 2022. : Low-Rank Adaptation of Large Language Models", Proc. of ICLR2022, 2022.

[12] C. Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries", Proc. of the Workshop on Text Summarization Branches Out (WAS2004), pp. 74–81, 2004.

[13] A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das, "ToTTo: A Controlled Table-To-Text Generation Dataset", Proc. of EMNLP2022, 2022.

[14] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "OPT: Open Pre-trained Tranformer Language Models", CoRR, 2022.

[15] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs", Proc. of 37th Conference on Neural Information Processing Systems (NeurIPS 2023), 2023.

[16] X. L. Li and P. Liang, "Prefix-Tuning: Optimizing Continuous Prompts for Generation", Proc. of the 59th Annual Meeting of the AAssociation for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pp.4582-4597, 2021.