An Empirical Study of LLM-as-a-Judge for LLM Evaluation: Fine-tuned Judge Models are Task-specific Classifiers

Anonymous ACL submission

Abstract

Recently, there has been a growing trend of utilizing Large Language Model (LLM) to evaluate the quality of other LLMs. Many studies have employed proprietary close-source models, especially GPT4, as the evaluator. Alternatively, other works have fine-tuned judge models based on open-source LLMs as the evaluator. In this study, we conduct an empirical study of different judge models on their evaluation capability. Our findings indicate that although the fine-tuned judge models achieve high accuracy on in-domain test sets, even surpassing GPT4, they are inherently task-specific classifiers, and their generalizability and fairness severely underperform GPT4.

1 Introduction

Recently, the evaluation for Large-scale Language Models (LLMs) has drawn considerate attention of research community (Liang et al., 2022; Chang et al., 2023). As the capabilities of LLMs continue to develop across various tasks, it is essential to evaluate them from a comprehensive perspective (Qin et al., 2023). However, traditional evaluation metrics for generative models, such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), only capture limited aspects of a model's performance.

Some research has proposed LLM-as-a-Judge (Li et al., 2023b; Zheng et al., 2023), namley utilizing proprietary LLMs, especially GPT4 (Achiam et al., 2023), to evaluate the LLM's response. By defining evaluation schemes in the prompt template, LLMs can leverage their instruction-following ability to provide reliable evaluation. For example, Li et al. (2023b) constructed a test set containing 805 questions and used the win rate compared with text-davinci-003 as the evaluation result, which is determined by GPT4. Zheng et al. (2023) developed 80 multi-round test questions covering eight common areas, and then automatically scored the model's answers using GPT4. Their results reveal that strong LLM-based evaluators can achieve a high agreement rate among human experts, establishing a foundation for LLM-as-a-Judge.

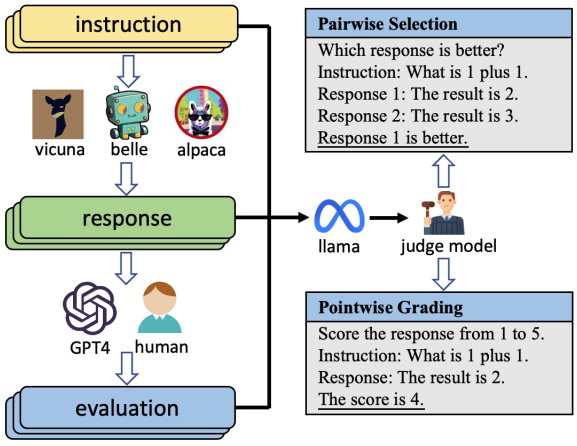

Figure 1: The general training and inference procedure of fine-tuned judge models.

However, relying on external API for evaluation may introduce consideration about privacy leakage, and the opacity of API models also challenges the evaluation reproducibility. To address these issues, several fine-tuned judge models are proposed, relying on open-source foundation models and data constructed from either GPT4 or human annotation, as shown in Figure 1. For instance, PandaLM (Wang et al., 2024) constructs data based on Alpaca instructions and GPT3.5 annotation, and then fine-tunes LLaMA-7B (Touvron et al., 2023) as a judge model. JudgeLM (Zhu et al., 2023b) constructs data from GPT4 annotations and fine-tunes a scalable judge model. Auto-J (Li et al., 2023a) constructs judgement data upon multiple scenarios to train a generative judge model, which can provide both judgement and critic. Prometheus (Kim et al., 2023) defines thousands of evaluation criteria and finetunes a fine-grained judge model.

In this study, we make an empirical study of the evaluation capabilities of judge models. We conduct extrapolated evaluations among available

Model	Foundation	Instruction	Response	Annotation	Evaluation Scheme	Testset
JudgeLM (Zhu et al., 2023b)	Vicuna	Instruct Datasets (Alpaca-GPT4, Dolly-15K...)	11 models (Alpaca,Vicuna...)	GPT4	Pairwise Grading	GPT4
PandaLM (Wang et al., 2024)	LLaMA	Alpaca 52K	5 models (LLaMA, Bloom...)	GPT3.5	Pairwise Selection	Human
Auto-J (Li et al., 2023a)	LLaMA2-chat	Preference Datasets (Chatbot Arena, OpenAI WebGPT...)	Preference Datasets	Human	Pairwise Selection Pointwise Grading	Human
Prometheus (Kim et al., 2023)	LLaMA2-chat	GPT4 Generated	GPT4 Generated	GPT4	Pointwise Grading	GPT4

Table 1: Detailed statistics of the four fine-tuned judge models, which is the foundation of our cross-validation. All the four models are open-source, with their training and test data also publicly released.

judge models and meta-evaluation testsets. Experiment results indicate that while the fine-tuned judge models achieve superior accuracy on their respective in-domain test sets, they still exhibit limitations in the following aspects:

- The fine-tuned judge model is inherently a classification model;
- The fine-tuned judge model is overfitted to specific evaluation schemes;
- The fine-tuned judge model is biased towards superficial quality;

To draw a conclusion, the fine-tuned judge models should be used only in similar evaluation scenarios, and can not serve as a general substitution for GPT4 in terms of LLM evaluation.

2 How Far can Fine-tuned Judges Go?

The typical process for finetuning a judge model consists of the following three steps:

Step 1: Data Collection. The training data generally comprises three components: instructions, responses and evaluations. The instructions are typically obtained from instruction datasets, with the responses generated by various representative models, and the evaluations can be derived from either GPT4 or human annotation;

Step 2: Prompt Designing. The prompt template can be structured in various ways depending on the evaluation scheme, such as pairwise selection (which aims to select the better one from a pair of responses), pointwise grading (which aims to score a single reference), etc.

Step 3: Model Fine-tuning. Using the designed prompt and collected data, the training process of the judge model typically follows the instruction fine-tuning paradigm (Ouyang et al., 2022). The model is fed with a instruction alongside answer(s) for yielding output comprising evaluation results.

While current judge models mostly self-claim exceeding the evaluation capability of GPT4, in this work, we make an extrapolated validation based on four representative works as listed in Table 1. Our findings are presented in the following sections.

2.1 Inherently a Classification Model

If we do not consider critic generation (which is seldom used in system-level evaluation), then LLM evaluation is inherently a classification (or regression) task. While current judge models are all trained in a generation-style, in this study, we train four classification (regression) models based the four groups of data in Table 1[1]. We also train four classification models based on DeBERTaV3-large (He et al., 2023) on the same data, which is 20 times smaller than the 7B version of LLaMA.

As shown in Table 2, the classification model performs equally well as the generation model for both pairwise selection and pointwise grading. The powerful generation ability of LLMs brings no gain to the evaluation accuracy, as they are trained on the same data for the same objective. Moreover, the DeBERTa-based evaluator achieves comparable performance with the LLM-based evaluators[2], which might be due to the encoder-only architecture is more suitable for classification.

We also analyze the correlation between different predictions made by different evaluators. As

[1]The training settings and prompt templates are presented in Appendix A.1 due to space limitation.

[2]The only exception is on Auto-J-test, which is possibly due to a large proportion of the test data exceeds 512, which is the maximum context length of DeBERTa.

Model	Train	JudgeLM-test		PandaLM-test		Auto-J-test	Prometheus-test	
		accuracy	F1	accuracy	F1	agreement	pearson-ind	pearson-ood
GPT 3.5		73.83	52.85	62.96	58.20	42.7	0.636	0.563
GPT 4-0613		**85.28**	**76.87**	**78.68**	**73.24**	**56.3**	0.742	0.743
Released Models†		79.02	71.87	67.57	57.49	54.6	**0.864**	**0.869**
Vicuna-7B	generation‡	**82.44**	**71.77**	**72.37**	**60.78**	**47.6**	0.826	0.815
Vicuna-7B	classification‡	82.16	70.07	70.87	60.34	46.8	**0.846**	**0.831**
DeBERTa	classification‡	81.30	68.34	72.27	51.75	31.7	0.835	0.813

Table 2: Comparison of evaluators trained with different architectures. Results with † are from evaluating the four publicly released models on their respective testsets, and results with ‡ are from evaluating models trained by us. Notice all our LLM-based evaluators are trained from Vicuna-7B (Chiang et al., 2023).

F1 score	Vicuna-generation	Vicuna-classification	DeBERTa-classification	GPT4
Vicuna-generation	100	83.27	82.74	64.96
Vicuna-classification	83.27	100	84.51	64.29
DeBERTa-classification	82.74	84.51	100	65.03
GPT4	64.96	64.29	65.03	100

Figure 2: The F1 score between the predictions of different evaluators on JudgeLM testset.

pearson	Vicuna-generation	Vicuna-classification	DeBERTa-classification	GPT4
Vicuna-generation	1.0	0.961	0.954	0.630
Vicuna-classification	0.961	1.0	0.977	0.627
DeBERTa-classification	0.954	0.977	1.0	0.623
GPT4	0.630	0.627	0.623	1.0

Figure 3: The pearson coefficient between the predictions of different evaluators on Prometheus testset.

shown in Figure 2 and 3, the correlation among different classification models is much closer than their correlation with GPT4. Although with different architectures, all three models are inherently classifiers fitting to the same set of supervision, leading to similar evaluation outcomes.

2.2 Overfitting to Evaluation Scheme

One of the most appealing attribute of LLMs is their generalization ability, enabling them to execute various task defined by various instructions (Zhu et al., 2023a). Under the case of LLM evaluation, the instruction can also be formed in various schemes: pairwise selection, pointwise grading, etc. While different judge models are fine-tuned on different schemes, we would like to verify their evaluation capability under the scheme defined by others. Therefore, we cross-validate the judge models on each other's testset[3].

As shown in Table 3 and 4, all four models performs the best on their respective testsets, with results comparable with GPT4. However, if we employ a model on evaluation schemes where it is not trained on[4], the evaluation performance would drop by a large margin. On the contrary, GPT4 consistently exhibits superior performance across various evaluation schemes.

We also validate the judge models on MT-bench (Zheng et al., 2023), which is a multi-turn meta-evaluation dataset. As can be seen in Table 5, while the four models are all trained for single-turn evaluation, they underperforms GPT4 on MT-bench by a large margin. This demonstrates that the finetuned judge models are overfitted to their respective evaluation schemes and lack generalibility.

2.3 Biased Towards Superficial Quality

Recently, there has been a lot of research about the bias of LLM-based evaluators, namely the evaluator would favor more verbose answers, or answers with similar format (Wang et al., 2023; Saito et al., 2023). Targeted at this problem, Zeng et al. (2023) proposed LLMBar as a testbed for the fairness of evaluators. It comprises one natural testset (Natural) and four adversarial testsets (Neighbor, Manual, GPTOut, GPTInst), and the adversarial testsets consist of paired outputs with a correct answer and a smooth, coherent but inconsistent answer. We evaluate the judge models on LLMBar[3], and the results are shown in Table 6.

[3]We applying their publicly released checkpoints with predefined prompts. For details please refer to Appendix A.2.

[4]For example, using a pairwise model (such as PandaLM or JudgeLM) for pointwise grading (such as Promethues testset), or using a pointwise model (such as Promethues) for pairwise selection (such as PandaLM or JudgeLM testsets).

Model	JudgeLM-test		PandaLM-test		Auto-J-test	Average
	accuracy	F1	accuracy	F1	agreement	
GPT 3.5	73.83	52.85	62.96	58.20	42.7	59.83
GPT 4-0613	85.28	76.87	78.68	73.24	56.3	73.42
JudgeLM-7B	**79.02**	**71.87**	70.97	**67.59**	46.6	65.53
PandaLM-7B	65.24	47.42	67.57	57.49	40.0	57.61
Auto-J-13B	72.86	57.60	**71.47**	61.01	**54.6**	**66.31**
Prometheus-13B	54.24	50.04	45.25	43.58	47.8	49.10
w/o trans	24.58	23.39	29.03	27.92	16.2	23.26

Table 3: Results of evaluators on pairwise selection. Notice Prometheus can be transformed for pairwise selection by grading two answers twice and compare the scores, therefore we release both results with and without transformation.

Model	Prometheus-test-ind			Prometheus-test-ood			Average
	pearson	kendalltau	spearman	pearson	kendalltau	spearman	
GPT 3.5	0.636	0.536	0.617	0.563	0.453	0.521	0.600
GPT 4-0613	0.742	0.659	0.747	0.743	0.660	0.747	0.743
Prometheus-13B	**0.864**	**0.788**	**0.863**	**0.869**	**0.789**	**0.869**	**0.867**
JudgeLM-7B	0.649	0.647	0.739	0.610	0.602	0.690	0.630
w/o trans	0.398	0.371	0.416	0.384	0.371	0.419	0.391
PandaLM-7B	0.417	0.368	0.423	0.386	0.333	0.383	0.402
Auto-J-13B	0.614	0.526	0.608	0.591	0.504	0.580	0.603

Table 4: Results of evaluators on pointwise grading. Notice JudgeLM can be transformed for pointwise grading by adding the reference as the first answer, therefore we release both results with and without transformation.

Model	MTBench			
	accuracy	precision	recall	F1
GPT 4-0613	**66.9**	**63.8**	**62.2**	**61.9**
JudgeLM-7B	48.7	52.0	49.7	48.7
PandaLM-7B	55.2	52.6	49.4	46.8
Auto-J-13B	51.7	50.2	46.8	43.7
Prometheus-13B	53.2	49.6	48.4	47.1

Table 5: Results of evaluators on multi-turn evaluation.

Model	LLMBar				
	Natu.	Neig.	GPTI.	GPTO.	Manu.
GPT 4-0613	**93.5**	**64.2**	**76.6**	**76.6**	**75.0**
JudgeLM-7B	62.0	23.1	26.1	46.8	28.3
PandaLM-7B	59.0	16.5	21.7	42.6	26.1
Auto-J-13B	70.0	20.9	21.7	46.8	23.9
Prometheus-7B	53.0	22.4	17.4	27.7	32.6
DeBERTa	62.0	26.9	42.4	55.3	34.8

Table 6: Accuracy of evaluators on bias evaluation.

As can be seen, the fine-tuned judge models are severely biased to superficial quality such as formality or verbosity, while neglecting crucial properties such as instruction following, leading to accuracy even worse than random guess on the adversarial testsets. On the other hand, GPT4 does not over-rely on the superficial features and achieves decent accuracy on the testsets. This also verifies that the fine-tuned judge models are inherently classifiers overfitted to the training data, relying on spurious statistical features for prediction.

It deserves noticing that the DeBERTa-based evaluator also outperforms the LLM-based evaluator by a large margin in terms of fairness. This inspires us that the bias of LLM-based evaluator may come from the casual language modeling process. While the model is trained to generate fluent and verbose responses, it also tends to prefer fluent and verbose response when employed for evaluation, even if it is not aligned with the instruction.

3 Conclusion

In this work, we make an empirical study of judge models for LLM evaluation. As revealed in our experiments, despite achieving superior evaluation performance on their in-domain testset, the fine-tuned judge models underperforms GPT4 in terms of generalibility and fairness by a large margin, which we believe cannot be amended by simply finetuning on more evaluation data.

Therefore, it is advisable to exercise caution when leveraging fine-tuned judge models for evaluation in real applications, depending on the overlap between the evaluation scenario and the training process. Nevertheless, the fine-tuned judge models are still not a general substitution for GPT4 in terms of LLM evaluation.

Limitations

Our work still has some limitations: 1) Due to the lack of related work, we primarily relied on cross-validation to assess the generalizability of the four fine-tuned judge models. To conduct a more thorough evaluation of the generalizability, it would be beneficial to incorporate additional independent testsets encompassing a broader range of evaluation schemes, such as reference augmentation and domain-specific evaluation. 2) The work of Zeng et al. (2023) is only a general assessment of evaluator bias, and we did not include fine-grained assessment for different biases, such as formality bias, verbosity bias, etc. 3) Due to time and resource constraints, we did not incorporate manual inspection into the meta-evaluation process. Including human evaluators would enhance the credibility of our claims.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. 2023. Prometheus: Inducing fine-grained evaluation capability in language models. *arXiv preprint arXiv:2310.08491*.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. 2023a. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. 2023. Verbosity bias in preference labeling by large language models. *arXiv preprint arXiv:2310.10076*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *ArXiv*, abs/2305.17926.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *arXiv preprint arXiv:2310.07641*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2023a. Promptbench: A unified library for evaluation of large language models. *arXiv preprint arXiv:2312.07910*.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023b. Judgelm: Fine-tuned large language models are scalable judges. *arXiv preprint arXiv:2310.17631*.

A Appendix

A.1 Training Settings

As mentioned in Section 2.1, we fine-tune our own judge models based on the four groups of data (JudgeLM (Zhu et al., 2023b), PandaLM (Wang et al., 2024), Auto-J (Li et al., 2023a), Prometheus (Kim et al., 2023)), both in generation-style and in classification-style, for the purpose of comparison. We train all the models on NVIDIA A100-80GB GPUs with Huggingface-transformers (Wolf et al., 2020) and DeepSpeed (Rasley et al., 2020). Detailed hyper-parameters are presented in Table 7. Notice when comparing generation and classification models, we adopt the same prompt template and same hyper-parameters, with the only difference lies in the prediction method. For generation model, the prediction head reused the pretrained language model head, and is trained akin to the process of language modeling. For classification (regression) model, the prediction head is newly initialized as a linear projection layer, and is decoupled from the language modeling process[5], as shown in Figure 4.

[5]Please refer to the class `AutoModelForSequenceClassification` in Huggingface library for more details.

A.2 Prompt Templates

As mentioned in Section 2.1, 2.2 and 2.3, we take the publicly released checkpoints of the four models and validate their performance. We use the same prompt templates as defined by the four open-source models, as presented from Figure 5 to Figure 12. For JudgeLM and PandaLM, their predefined prompts are in the form of pairwise selection, and we make slight modifications to apply them on pointwise grading. For Prometheus, the predefined prompt is in the form of pointwise grading, and we make slight modifications to apply it on pairwise selection. For Auto-J, they predefined prompts both for pairwise selection and pointwise grading.

Configuration	Vicuna-based	DeBERTa-based
base model	Vicuna-7B	DeBERTaV3-large
max length	2048	512
learning rate	2e-5	2e-5
learning rate schedule	cosine decay	cosine decay
optimizer	AdamW	AdamW
AdamW beta1	0.9	0.9
AdamW beta2	0.999	0.98
weight decay	0.0	0.0
GPU nums	8	2
training epochs	3	3
batch size	128	128
warmup ratio	0.003	0.003
numerical precision	bf16, tf32	fp16
ZeRO optimizer	stage 2	None

Table 7: Configurations of the judge models fine-tuned by us in Section 2.1. Both classification and generation judge models leverage the same group of configs based on their foundation model.

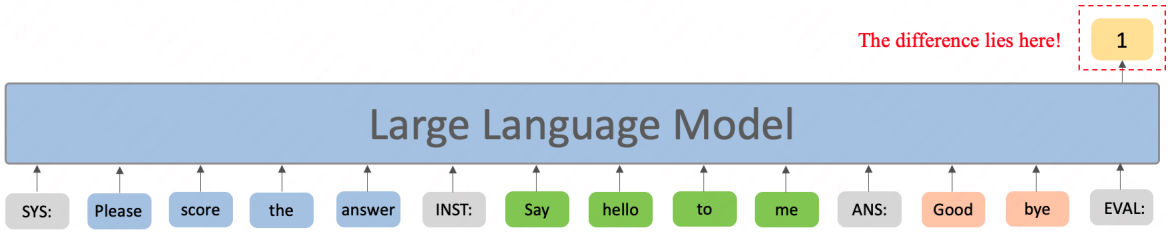

Figure 4: The architecture of classification-based judge model. The major difference lies in the prediction head, where a new classification (regression) head is initialized for predicting the result.

```
You are a helpful and precise assistant for checking the quality of the answer.
[Question]
{question_body}

[The Start of Assistant 1's Answer]
{answer1_body}

[The End of Assistant 1's Answer]

[The Start of Assistant 2's Answer]
{answer2_body}

[The End of Assistant 2's Answer]

[System]
We would like to request your feedback on the performance of two AI assistants in
response to the user question displayed above.
Please rate the helpfulness, relevance, accuracy, level of details of their responses.
Each assistant receives an overall score on a scale of 1 to 10, where a higher score
indicates better overall performance.
Please first output a single line containing only two values indicating the scores
for Assistant 1 and 2, respectively. The two scores are separated by a space. In the
subsequent line, please provide a comprehensive explanation of your evaluation,
avoiding any potential bias and ensuring that the order in which the responses were
presented does not affect your judgment.

### Response:
```

Figure 5: Prompt template for JudgeLM applied for pairwise selection.

```
You are a helpful and precise assistant for checking the quality of the answer.
[Question]
{question_body}

[The Start of Assistant's Answer]
{answer_body}

[The End of Assistant's Answer]

[System]
We would like to request your feedback on the performance of the AI assistant in
response to the user question displayed above.
{rubric} The assistant receives an overall score on a scale of 1 to 10, where a
higher score indicates better overall performance.
Please first output a single line containing only one values indicating the score for
the Assistant. In the subsequent line, please provide a comprehensive explanation of
your evaluation, avoiding any potential bias.

### Response:
```

Figure 6: Prompt template for JudgeLM applied for pointwise grading.

```
Below are two responses for a given task. The task is defined by the Instruction.
Evaluate the responses and generate a reference answer for the task.

### Instruction:
{question_body}

### Response 1:
{answer1_body}

### Response 2:
{answer2_body}

### Evaluation:
```

Figure 7: Prompt template for PandaLM applied for pairwise selection.

```
Below are a response for a given task. The task is defined by the Instruction.
{rubric} Evaluate the response with an overall score on a scale of 1 to 10, and
generate a reference answer for the task.

### Instruction:
{question_body}

### Response:
{answer_body}

### Evaluation:
```

Figure 8: Prompt template for PandaLM applied for pointwise grading.

```
You are assessing two submitted responses on a given user's query and judging which
response is better or they are tied. Here is the data:

[BEGIN DATA]
***
[Query]: {question_body}
***
[Response 1]: {answer1_body}
***
[Response 2]: {answer2_body}
***
[END DATA]

Here are the instructions to assess and compare the two responses:

1. Pinpoint the key factors to distinguish these two responses.
2. Conclude your comparison by providing a final decision on which response is better,
or they are tied. Begin your final decision statement with "So, the final decision is
Response 1 / Response 2 / Tie". Ensure that your decision aligns coherently with the
comprehensive evaluation and comparison you've provided.
```

Figure 9: Prompt template for Auto-J applied for pairwise selection.

```
Write critiques for a submitted response on a given user's query, and grade the
response:

# [BEGIN DATA]
# ***
# [Query]: {question_body}
# ***
# [Response]: {answer_body}
# ***
# [END DATA]

# Write critiques for this response. {rubric} After that, you should give a final
rating for the response on a scale of 1 to 10 by strictly following this format:
"[[rating]]", for example: "Rating: [[5]]".
```

Figure 10: Prompt template for Auto-J applied for pointwise grading.

```
<<SYS>>\nYou are a fair evaluator language model.\n<</SYS>>

###Task Description:
An instruction (might include an Input inside it), two responses to evaluate, and a
score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the responses strictly based
on the given score rubric, not evaluating in general.
2. After writing a feedback, write two score that are integers between 1 and 5. You
should refer to the score rubric.
3. The output format should look as follows: \"Feedback: (write a feedback for
criteria) [RESULT] (two integer numbers between 1 and 5)\"
4. Please do not generate any other opening, closing, and explanations.

###The instruction to evaluate:
{question_body}

###Response1 to evaluate:
{answer1_body}

###Response2 to evaluate:
{answer2_body}

###Score Rubrics:
{rubric}

###Feedback:
```

Figure 11: Prompt template for Prometheus applied for pairwise selection.

```
<<SYS>>\nYou are a fair evaluator language model.\n<</SYS>>

###Task Description:
An instruction (might include an Input inside it), a response to evaluate, and a
score rubric representing a evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based
on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You
should refer to the score rubric.
3. The output format should look as follows: \"Feedback: (write a feedback for
criteria) [RESULT] (an integer number between 1 and 5)\"
4. Please do not generate any other opening, closing, and explanations.

###The instruction to evaluate:
{question_body}

###Response to evaluate:
{answer_body}

###Score Rubrics:
{rubric}

###Feedback:
```

Figure 12: Prompt template for Prometheus applied for pointwise grading.