Self-Evaluation of Large Language Model based on Glass-box Features

Hui Huang[1], Yingqi Qu[2], Jing Liu[2], Muyun Yang[1]✉, Bing Xu[1]✉,
Tiejun Zhao[1], Wenpeng Lu[3]
[1]Faculty of Computing, Harbin Institute of Technology, Harbin, China
[2]Baidu Inc., Beijing, China
[3]Key Laboratory of Computing Power Network and Information Security, Ministry of Education,
Shandong Computer Science Center, Qilu University of Technology, Jinan, China
huanghui@stu.hit.edu.cn, {quyingqi, liujing46}@baidu.com,
{yangmuyun, hitxb, tjzhao}@hit.edu.cn, lwp@qlu.edu.cn;

Abstract

The proliferation of open-source Large Language Models (LLMs) underscores the pressing need for evaluation methods. Existing works primarily rely on external evaluators, focusing on training and prompting strategies. However, a crucial aspect – model-aware glass-box features – is overlooked. In this study, we explore the utility of glass-box features under the scenario of self-evaluation, namely applying an LLM to evaluate its own output. We investigate various glass-box feature groups and discovered that the softmax distribution serves as a reliable quality indicator for self-evaluation. Experimental results on public benchmarks validate the feasibility of self-evaluation of LLMs using glass-box features[1].

1 Introduction

Recently, as the Large Language Models (LLMs) brings a storm to the area of artificial intelligence, evaluating the quality of LLM outputs also draws a lot of research concentration. As the ability of LLMs develop beyond limitation, it is essential to evaluate them from a comprehensive and scalable perspective. However, traditional evaluation metrics for generative models, such as BLEU and ROUGE, only capture limited aspects of a model's performance (Mathur et al., 2020).

Some research has proposed language model-based evaluation (Li et al., 2023b; Zheng et al., 2023), leveraging proprietary LLMs such as GPT-3.5 or GPT4 (Achiam et al., 2023), to evaluate the LLM's outputs. However, relying on external API for evaluation may introduce consideration about privacy leakage, and the opacity of API models also challenges the evaluation reproducibility. Other works propose to fine-tune open-source models as specialized evaluators (Wang et al., 2024; Zhu et al., 2023; Li et al., 2023a). However, constrained by the capability of foundation model, these fine-tuned evaluators severely underperform GPT4 on generalizability and fairness (Huang et al., 2024).

In this work, we take a novel approach to LLM evaluation: Is LLM capable of self-evaluation? To answer this, we delve into the utility of glass-box features, namely the useful information that can be extracted from the model as a by-product of generation. Concretely, we explore three groups of glass-box features: 1) softmax distribution, 2) uncertainty estimation and 3) attention distribution. Our findings reveal that manipulating the softmax distribution by calculating its entropy and variance exhibits a strong correlation with annotated evaluation results. Furthermore, we propose two strategies to enhance the evaluation by incorporating features derived from references.

We conduct our experiments on two widely used LLM evaluation benchmarks, MT-Bench (Zheng et al., 2023) and Vicuna-Bench (Chiang et al., 2023). Experimental results notify that the LLM is capable of providing accurate self-evaluation, surpassing proprietary evaluators such as GPT-3.5 and Auto-J. The self-evaluation capability of LLMs holds promise for various applications, ranging from self-reflection to reward modeling.

2 Glass-box Features for Self-Evaluation

We assume an LLM architecture based on Transformer networks (Vaswani et al., 2017), which is currently the mainstream LLM architecture. In this section, we introduce the three groups of glass-box features for self-evaluation.

2.1 Softmax Distribution

Given an instruction x, the probability of generating response y can be factorized as:

$$\text{SentProb} = \prod_{t=1}^{T} p(y_t|y_{<t}, x, \theta)$$

[1]Codes are openly available at https://github.com/HuihuiChyan/SelfEval

Findings of the Association for Computational Linguistics: EACL 2024, pages 5813–5820

where θ represents model parameters and T is the response length. However, the model would always select from the most probable tokens during decoding regardless of their quality, leading to biased evaluation. Therefore, we propose two metrics to exploit the softmax distribution for evaluation.

First, we compute the entropy of softmax output distribution over target vocabulary of size V at each decoding step to obtain a sentence-level measure:

$$\text{Softmax-Ent} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{v=1}^{V} p(y_t^v)\log p(y_t^v)$$

where $p(y_t)$ represents the conditional distribution $p(y_t|x, y_{<t}, \theta)$. If the majority of the probability mass is concentrated on a limited number of vocabulary words, it indicates that the model is confident and the response is more likely to be accurate. Conversely, if the softmax probabilities resemble a uniform distribution, where selecting any word from the vocabulary is equally probable, the quality of the resulting response is expected to be low.

Second, we hypothesize that the dispersion of probabilities of individual words might provide useful information that is inevitably lost when taking an average. To formalize this intuition we compute the variance of word-level log-probabilities:

$$\text{Softmax-Var} = \mathbb{E}[\text{P}^2] - (\mathbb{E}[\text{P}])^2$$

where $\text{P} = p(y_1), ..., p(y_T)$ represents word-level probabilities for a given response.

2.2 Uncertainty Estimation

Uncertainty estimation aims to assess the confidence of a mapping across various inputs (Xiao and Wang, 2019). In this work, we propose to employ ensemble-based uncertainty estimation (Malinin and Gales, 2021) during LLM inference as a quality indicator. More specifically, we perform several random forward passes through the model and collect posterior probabilities. Intuitively, if the model is confident with the generation, the sampled distributions should be concentrated and the diversity among them should be small.

Given that LLMs are typically trained without dropout, we propose two strategies to introduce randomness to the forward-passes:

1. Decoding-based Ensemble: Adopt random top-k decoding (Fan et al., 2018) for inference.

2. Prompt-based Ensemble: Randomly choose a system prompt from a pre-designed prompt pool[2] for each inference.

Subsequently, expectation and variance of the resulting probability distributions can be used to quantify uncertainty:

$$\text{Unt-Exp} = \frac{1}{N}\sum_{n=1}^{N} \text{SP}_{T^n}$$

$$\text{Unt-Var} = \mathbb{E}[\text{SP}_{T^n}^2] - (\mathbb{E}[\text{SP}_{T^n}])^2$$

where SP denotes the sentence level probability, and N is the forward-pass number.

2.3 Attention Distribution

Attention weights represent the strength of connection between source and target tokens, which may be indicative of response quality (Rikters and Fishel, 2017). One way to measure it is to compute the entropy of the attention distribution:

$$\text{AttnEnt} = -\frac{1}{I}\sum_{i=1}^{I}\sum_{j=1}^{J} \alpha_{ji}\log\alpha_{ji}$$

where α represents attention weights, I and J are the token numbers of instruction and response.

Since LLMs typically employ a multi-layer and multi-head self-attention architecture, we calculate attention entropy for each head (H) and layer (L) of the decoder in this study. As it is not clear which combination would give the best results for quality prediction, to summarize the information from different heads and layers, we propose to choose the minimum value or compute the average:

$$\text{AttnEnt-Min} = min_{hl}(\text{AttnEnt}_{hl})$$

$$\text{AttnEnt-Avg} = \frac{1}{H \times L}\sum_{h=1}^{H}\sum_{l=1}^{L} \text{AttnEnt}_{hl}$$

3 Self-Evaluation with Reference

When evaluating the LLM's response to an instruction, a reference answer might be available. In this work, we introduce two strategies to effectively utilize the references for self-evaluation.

3.1 In-Context Illustration

Previous research shows that a few annotated samples can improve the performance of LLM via In-

[2]The prompt pool is presented in Appendix A.1.

Model	Method	MT-Bench			Vicuna-Bench			Average
		Pearson	Kendall	Spearman	Pearson	Kendall	Spearman	
LLaMA2-7B-Chat	Auto-J	0.5024	0.4092	0.5112	0.5233	0.3403	0.3773	0.5129
	GPT-3.5-Turbo	0.4342	0.3982	0.5033	**0.6695**	0.3858	0.4330	0.5519
	Self-Generation	0.1492	0.0992	0.1976	0.1415	0.1600	0.1023	0.1454
	SentProb	0.2034	0.2099	0.3067	0.4970	0.2304	0.3192	0.3502
	Softmax-Ent	0.4666	0.4395	0.5978	0.3730	0.2441	0.3223	0.4198
	Softmax-Var	0.5612	**0.4695**	**0.6239**	0.6506	**0.4534**	**0.5894**	**0.6059**
	Softmax-combo	**0.5879**	0.4638	0.6222	0.6209	0.4352	0.5650	0.6044
	DecEnsem-Exp	0.4105	0.3877	0.3526	0.5606	0.3545	0.5051	0.4856
	DecEnsem-Var	0.3535	0.3215	0.3023	0.2504	0.1615	0.3197	0.3020
	PrmEnsem-Exp	0.4242	0.3054	0.3692	0.5381	0.3359	0.4675	0.4812
	PrmtEnsem-Var	0.1695	0.1692	0.2441	0.1321	0.0895	0.1250	0.1508
	Attn-Ent-Min	0.1444	0.1150	0.1563	0.1182	0.1161	0.1584	0.1313
	Attn-Ent-Avg	0.1361	0.1094	0.1475	0.0977	0.1181	0.1600	0.1169
Vicuna-7B	Auto-J	**0.5673**	**0.4302**	**0.5405**	0.6003	0.4818	0.5345	0.5838
	GPT3.5-Turbo	0.4812	0.4358	0.5353	**0.6946**	**0.5901**	**0.6733**	0.5879
	Self-Generation	0.1636	0.1753	0.1498	0.1985	0.1319	0.1709	0.1811
	SentProb	0.2350	0.1706	0.2547	0.4606	0.2545	0.3051	0.3478
	Softmax-Ent	0.4690	0.3003	0.4291	0.6379	0.2851	0.3855	0.5535
	Softmax-Var	0.4632	0.3171	0.4403	0.6146	0.2879	0.3772	**0.5389**
	Softmax-combo	0.5358	0.3578	0.5022	0.6856	0.3276	0.4270	0.6107
	DecEnsem-Exp	0.4518	0.2645	0.2168	0.5099	0.5137	0.6548	0.4809
	DecEnsem-Var	0.2091	0.1476	0.1622	0.3246	0.2931	0.3014	0.2669
	PrmEnsem-Exp	0.4414	0.2677	0.2229	0.4897	0.4873	0.6312	0.4656
	PrmEnsem-Var	0.2353	0.0875	0.1239	0.2028	0.1860	0.3201	0.2191
	Attn-Ent-Avg	0.0355	0.0234	0.0043	0.2254	0.1981	0.2567	0.1305
	Attn-Ent-Min	0.0463	0.0307	0.0530	0.2161	0.1888	0.2459	0.1312

Table 1: Experiment results of different groups of methods for LLM self-evaluation. Softmax-combo denotes the normalized summation of Softmax-Ent and Softmax-Var.

context Learning (Wang et al., 2023a). Therefore, we extend the prompt with the instruction and its reference as in-context demonstration[3].

By prefixing the reference, the model tends to generate a similar softmax distribution (Wang et al., 2023a). Therefore, when the model is subsequently forced to generate the current answer, the resulting softmax distribution could represent the difference between the current answer and the golden answer, effectively indicating the quality.

3.2 Probability Calibration

There has been a lot of research about the bias of the evaluator (Huang et al., 2023; Wang et al., 2023b), where the evaluator would predict on superficial quality, such as complexity. As the reference should always be assigned with a maximum score, we can quantify the bias of the model by calculate the log-probability of reference answer:

$$\text{SentProb-Ref} = -\frac{1}{T}\sum_{t=1}^{T} p(\tilde{y}_t)\log p(\tilde{y}_t)$$

where $\tilde{y}_t$ denotes the log-probability assigned by the model to the reference. Based on that, the bias of the self-evaluation can be mitigated by subtracting the result with SentProb-Ref, thereby improving the evaluation accuracy.

4 Experiments

4.1 Set-up

Benchmark. We carry out our experiments on two widely used LLM evaluation benchmarks: MT-Bench (Zheng et al., 2023) and Vicuna-Bench (Chiang et al., 2023). Both of the benchmarks encompass 80 questions covering diverse areas, and we derive the responses for different models following the default settings of MT-Bench. To mitigate the cost of human annotations, we follow the official evaluators according to each benchmark, namely GPT-4 (Achiam et al., 2023).

Model. Our experiments are based on two popular models, namely Vicuna-7B (Chiang et al., 2023) and LLaMA2-7B-chat (Touvron et al., 2023), both are enabled with instruction-following ability.

Metric. Pearson Correlation Coefficient between the prediction and the annotation is taken as the

[3]Detailed prompt template is presented in Appendix A.2.

major metric. We also report Spearman's Ranking Correlation Coefficient and Kendall's Tau Ranking Correlation Coefficient[4] as extra reference.

Baseline. We mainly compare with two evaluators, GPT-3.5-Turbo[5] and Auto-J (Li et al., 2023a). The former is a close-source LLM carefully prompted for LLM evaluation, and the latter is an open-source LLM specifically fine-tuned for LLM evaluation. These are the two mainstream methods for LLM evaluation. We also compare with Self-Generation, namely prompting the model as an evaluator to evaluate its own response[6].

4.2 Main Results

As shown in Table 1, among the three groups of glass-box features, softmax distribution based features perform the best, which achieves a high correlation with annotations and outperforms Auto-J and GPT-3.5-Turbo. This verifies the strong association between the confidence represented by the softmax distribution and the response quality. When the instruction exceeds the LLM's capability scope, the model would generate the response with low confidence, resulting in a sparse log-probability distribution. Furthermore, combining both entropy and variance by simply adding them together can achieve further improvement.

The uncertainty-based methods underperform, particularly those relying on variance. This might be because we set forward-pass number as 10, which is inadequate to quantify model uncertainty. Considering multiple forward-passes is overly time-consuming, we think the ensembled uncertainty estimation is less practical for self-evaluation.

The attention-based methods exhibit a poor correlation. Despite its effectiveness in unsupervised translation evaluation (Fomicheva et al., 2020), LLM-based response generation differs significantly from encoder-decoder-based machine translation, which typically produces translations for one or two tokens at each step. Therefore, a dispersed attention across multiple positions, leading to a low attention entropy, does not necessarily indicate a poor response.

The self-generation also exhibits a poor correlation. This is because the model fails to comprehend the instruction to generate a score between 1 to 10 in most cases. This underscores the superiority of our proposed method, which is not limited by the capabilities of the evaluated model and can be effectively applied to 7B-sized models.

Method	Vicuna-Bench		
	Pearson	Kendall	Spearman
GPT-3.5-Turbo	0.6695	0.3858	0.4330
Results on LLaMA2-7B-Chat			
Softmax-Ent	0.3730	0.2441	0.3223
+illustration	0.3678	0.2429	0.3370
+calibration	**0.3930**	**0.2631**	**0.3490**
Softmax-Var	0.6506	**0.4534**	**0.5894**
+illustration	0.6580	0.4472	0.5865
+calibration	**0.6617**	0.4360	0.5625
Results on Vicuna-7B			
Softmax-Ent	0.6379	0.2851	0.3855
+illustration	0.6375	0.2982	0.3960
+calibration	**0.6441**	**0.3177**	**0.4240**
Softmax-Var	0.6146	0.2879	0.3772
+illustration	0.6247	0.2976	0.3903
+calibration	**0.6526**	**0.3099**	**0.4133**

Table 2: Experiment results of different reference augmentation strategies for self-evaluation.

4.3 Self-Evaluation with Reference

In this section, we evaluate the two reference augmentation methods proposed in Section 3, namely in-context illustration and probability calibration. As shown in Table 2, both the two strategies can enhance the evaluation accuracy by a large margin, from the perspective of in-context learning and bias calibration. This notifies the references is a pivotal information for response evaluation, and should not be neglected if available. Furthermore, among the two methods, the calibration-based method performs better, verifying calibration is an effective method to mitigate the bias introduced by the casual language modeling process, which matters a lot for more accurate self-evaluation.

5 Conclusion

In this work, we investigate the utility of glass-box features for LLM self-evaluation. Our findings verify that the confidence quantified by softmax distribution can be a reliable quality indicator. The self-evaluation ability of LLM is a promising pathway, for example, the LLM can rely on self-evaluation results to decide whether to perform self-reflection (Xie et al., 2024), or to select preferred data for reward modeling (Lee et al., 2024), and we leave these as the future work.

[4]Notice measuring our method with ranking coefficient is not fair, as our method can not predict tie.

[5]https://platform.openai.com/docs/models/gpt-3-5-turbo

[6]Detailed prompt template is presented in Appendix A.3.

Limitations

Our work still has some limitations: 1) Due to time and resource constraints, we primarily relied on GPT-4 for annotating golden labels during the meta-evaluation process. Including human evaluators would enhance the credibility of our proposed self-evaluation methods. 2) The experiments are primarily conducted on 7B version models. To conduct a more thorough evaluation of our methods, it would be beneficial to incorporate larger models with more parameters. 3) The self-evaluation capability of the LLM can be applied to various applications, as discussed in the conclusion part. Including the validation of self-evaluation methods on other LLM-based applications such as self-reflection would further bolster its utility.

Acknowledgements

This work is supported by National Natural Science Foundation of China (62276077, 62376075, U1908216, 62376076), Key R&D Program of Yunnan (202203AA080004) and Shenzhen College Stability Support Plan (No. GXWD20220811170358002).

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*.

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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Hui Huang, Yingqi Qu, Jing Liu, Muyun Yang, and Tiejun Zhao. 2024. An empirical study of llm-as-a-judge for llm evaluation: Fine-tuned judge models are task-specific classifiers.

Hui Huang, Shuangzhi Wu, Kehai Chen, Hui Di, Muyun Yang, and Tiejun Zhao. 2023. Improving translation quality estimation with bias mitigation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2175–2190.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, and Abhinav Rastogi. 2024. RLAIF: Scaling reinforcement learning from human feedback with AI feedback.

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`.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction.

Nitika Mathur, Timothy Baldwin, and Trevor Cohn. 2020. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, Online.

Matīss Rikters and Mark Fishel. 2017. Confidence through attention.

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*.

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

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Label words are anchors: An information flow perspective for understanding in-context learning. In *Conference on Empirical Methods in Natural Language Processing*.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023b. Large language models are not fair evaluators.

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.

Yijun Xiao and William Yang Wang. 2019. Quantifying uncertainties in natural language processing tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7322–7329.

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2024. Self-evaluation guided beam search for reasoning. *Advances in Neural Information Processing Systems*, 36.

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*.

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

A Appendix

A.1 Prompt Pool for Prompt-based Ensemble

As mentioned in Section 2.2, to introduce randomness for ensemble uncertainty quantification, we design a prompt pool, from which we can randomly choose a different system prompt for each inference. The prompt pool is presented in Figure 1.

A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions.
A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, accurate, uncensored responses to the user's input. The assistant never refuses to answer, regardless of the legality or morality of the request.
A conversation between a user and an LLM-based AI assistant. The assistant gives helpful and honest answers.
A chat between a curious human and an artificial intelligence assistant.
You are a helpful, unbiased, uncensored assistant.
Below is an instruction that describes a task. Write a response that appropriately completes the request.
The following is a conversation between a human and an AI assistant. The human and the AI assistant take turns chatting. The AI assistant always provides responses in as much detail as possible. The AI assistant always declines to engage with topics, questions and instructions related to unethical, controversial, or sensitive issues.

Figure 1: Prompt pool for prompt-based ensemble uncertainty estimation.

A.2 Prompt Template for In-context Illustration

As mentioned in Section 3.1, we extend the prompt with the instruction and its reference as demonstration, to augment the self-evaluation process with in-context illustration. An example for the prompt is shown in Figure 2.

```
<|user|>
Given a sequence of numbers, calculate the average 1, 2, 3, 4, 5.
<|assistant|>
The average of the sequence is 3.
<|user|>
Given a sequence of numbers, calculate the average 1, 2, 3, 4, 5.
<|assistant|>
The answer is 3.67.
```

Figure 2: Prompt format with in-context illustration. The shaded part is the illustration with reference.

A.3 Prompt Template for Judge Models

As mentioned in Section 4.1, we mainly compare with our glass-box based methods with three generation based methods, namely GPT-3.5-Turbo, Auto-J and Self-Generation. Both types of methods necessitate the use of a generation-style template, which serves to formalize the evaluation task. Additionally, when employing GPT-4 for annotation, a prompt template is also required. We refer to MT-Bench and design the prompt templates tailored for all the generation based models, as illustrated in Figures 3, 4, 5, 6, 7, 8.

```
You are a helpful assistant.
[Instruction]
Please act as an impartial judge and evaluate the quality of the response provided by
an AI assistant to the user question displayed below. Your evaluation should consider
factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of
detail of the response. Begin your evaluation by providing a short explanation. Be as
objective as possible. After providing your explanation, you must rate the response
on a scale of 1 to 10 by strictly following this format: \"[[rating]]\", for example:
\"Rating: [[5]]\".

[Question]
{question}

[The Start of Assistant's Answer]
{answer}
[The End of Assistant's Answer]
```

Figure 3: Prompt template for GPT4 and GPT-3.5-Turbo applied for single-turn evaluation.

```
You are a helpful assistant.
[Instruction]
Please act as an impartial judge and evaluate the quality of the response provided by
an AI assistant to the user question displayed below. Your evaluation should consider
factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of
detail of the response. You evaluation should focus on the assistant's answer to the
second user question. Begin your evaluation by providing a short explanation. Be as
objective as possible. After providing your explanation, you must rate the response
on a scale of 1 to 10 by strictly following this format: \"[[rating]]\", for example:
\"Rating: [[5]]\".

<|The Start of Assistant A's Conversation with User|>

### User:
{question_1}

### Assistant A:
{answer_1}

### User:
{question_2}

### Assistant A:
{answer_2}

<|The End of Assistant A's Conversation with User|>
```

Figure 4: Prompt template for GPT4 and GPT-3.5-Turbo applied for multi-turn evaluation.

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

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

[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. Please rate the helpfulness, relevance,
accuracy, level of details of the response. 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 5: Prompt template for Auto-J applied for single-turn evaluation.

```
You are a helpful and precise assistant for checking the quality of the answer.
<|The Start of Assistant's Conversation with User|>

### User:
{question_1}

### Assistant:
{answer_1}

### User:
{question_2}

### Assistant:
{answer_2}

<|The End of Assistant's Conversation with User|>

[System]
We would like to request your feedback on the performance of the AI assistant in
response to the user question displayed above.
Please rate the helpfulness, relevance, accuracy, level of details of the responses.
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 Auto-J applied for multi-turn evaluation.

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

# [BEGIN DATA]
# ***
# [Query]: {question}
# ***
# [Response]: {answer}
# ***
# [END DATA]

# Write critiques for this response. 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 7: Prompt template for self-generation applied for single-turn evaluation.

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

# [BEGIN DATA]
# ***
# [Query 1]: {question_1}
# ***
# [Response 1]: {answer_1}
# ***
# [Query 2]: {question_2}
# ***
# [Response 2]: {answer_2}
# ***
# [END DATA]

# Write critiques for this response. 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 8: Prompt template for self-generation applied for multi-turn evaluation.