Bayesian Calibration of Win Rate Estimation with LLM Evaluators

Yicheng Gao[*1] **Gonghan Xu**[*1] **Zhe Wang**[1] **Arman Cohan**[1]
[1]Yale University
{charlie.gao, gonghan.xu, zhe.wang.zw439, arman.cohan}@yale.edu

Abstract

Recent advances in large language models (LLMs) show the potential of using LLMs as evaluators for assessing the quality of text generations from LLMs. However, applying LLM evaluators naively to compare different systems can lead to unreliable results due to the inaccuracy and intrinsic bias of LLM evaluators. In order to mitigate this problem, we propose two calibration methods, Bayesian Win-Rate Sampling (BWRS) and Bayesian Dawid-Skene, both of which leverage Bayesian inference to more accurately infer the true win rate of generative language models. We empirically validate our methods on six datasets covering story generation, summarization, and instruction following tasks. We show that both our methods are effective in improving the accuracy of win rate estimation using LLMs as evaluators, offering a promising direction for reliable automatic text quality evaluation.

1 Introduction

Evaluating the quality of AI-generated text has been a longstanding and evolving challenge in NLP. In recent years, this challenge has become increasingly crucial due to the growing interest in the field of generative AI. While human judgment is still considered the most reliable form of assessment, common automatic approaches to evaluating quality of AI-generated text include heuristic-based evaluation metrics (Papineni et al., 2002; Lin, 2004; Pillutla et al., 2021), model-based evaluation metrics (Zhang et al., 2019; Fabbri et al., 2022; Zha et al., 2023; Chen and Eger, 2023), and recently, LLM-based evaluations (Kim et al., 2024a,b; Wang et al., 2024). Due to their low cost and high correlation with human preferences, LLM-based evaluations are receiving an increasing amount of attention. Most previous studies that apply LLM evaluators (Chiang and Lee, 2023a,b; Dubois et al., 2024; Kim et al., 2024a,b; Wang et al., 2024) attempt to improve the agreement between LLM evaluators and human preference by training expert models for evaluation or improving prompting strategies. However, such methods often either require compute-expensive finetuning, or suffer from common problems of LLM evaluators such as position bias (Wang et al., 2023b), self-preference, and more (Koo et al., 2023). Besides, as we will discuss in Section 3.2, directly applying a non-perfect LLM evaluator will result in a bias problem in the estimation of win rate.

In this paper, we attempt to address these challenges by proposing two methods, BWRS and Bayesian Dawid-Skene. Our methods leverage Bayesian inference to infer the true win rate of one text generator against another using evaluation results of LLM evaluators and incorporating optional prior knowledge about human preferences. By employing these methodologies, we observe a closer alignment between LLM-generated evaluations and human judgment. [1]

The contribution of this paper is threefold:

- We identify the bias problem in win rate estimation with LLM evaluators.
- We conduct exploratory study on mitigating this bias with Bayesian inference. Specifically, we propose BWRS and Bayesian Dawid-Skene, both of which are shown effective in calibrating win rate estimation given LLM evaluation results, and optionally, some human evaluation results.
- We publish our LLM evaluation annotations to facilitate future study in LLM-based evaluation.

[*]Equal contribution

[1]The code and data used in our experiments are available at https://github.com/yale-nlp/bay-calibration-llm-evaluators under Apache 2.0 license.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 4757–4769
November 12-16, 2024

2 Related work

LLM as evaluators A line of research in LLM-based evaluation evaluated the performance of LLM evaluators and proposed methods to improve them. Some works applied various prompting techniques to improve the accuracy of LLM evaluation, including chain of thought (Liu et al., 2023a), evaluation with explanation (Chiang and Lee, 2023b), multi-LLM discussion (Chan et al., 2023; Li et al., 2023), and calibration with human expert (Liu et al., 2023b). Some other works (Wang et al., 2024; Kim et al., 2024a,b) trained expert models in evaluation. As for evaluating the general capability of LLM evaluators, most previous studies (Liu et al., 2023a; Chiang and Lee, 2023a,b; Dubois et al., 2024) used correlation coefficients such as Pearson's correlation or Kendall's tau to measure the preference of different LLM evaluators compared with human evaluators.

On the application side, LLM evaluators are often applied to build LLM rankings. (Dubois et al., 2024) proposed a simple LLM evaluation framework by looking at the win rate decided by GPT-4 evaluators on a large number of texts generated by the two generators under the same generation prompts. Auto-Arena(Zhao et al., 2024) used LLM judge agents to determine the winner of each LLM pair. However, as we'll discuss in Section 3.2, these methods can lead to biased win rate estimations, especially when the LLM evaluators do not align well enough with human preferences.

Annotation models In the field of crowdsourced annotations, a line of research focuses on simultaneously modeling the accuracy of individual annotators and determining the true labels of tasks. These works mostly target aggregating crowdsourced data and improving data quality in case of non-expert or adversarial annotators. Dawid-Skene (Dawid and Skene, 1979) is the first model proposed to consider individual annotator error rates by using maximum likelihood estimation to infer true labels from annotators with different accuracies. Since then, many other models (Albert and Dodd, 2004; Carpenter, 2008; Whitehill et al., 2009; Kim and Ghahramani, 2012; Hovy et al., 2013; Passonneau and Carpenter, 2014; Zhang et al., 2016) were developed to improve performance and efficiency. These methods were originally proposed to model the accuracy of human annotators, in our paper we instead apply them to model LLM evaluators.

3 Methods

In this section, we first formalize the problem of applying LLMs as evaluators. We then point out the bias problem associated with directly applying LLM evaluator results, and then propose our methods to address this problem.

3.1 Problem formalization

3.1.1 True win rate and observed win rate

Consider two LLMs as text generators (LLM generators) G_0 and G_1. Let Σ be the set of all possible inputs to the text generators, and let Ω be the set of all possible outputs given the inputs from Σ. We can then define the LLMs as two functions $G_0 : \Sigma \to \Omega$ and $G_1 : \Sigma \to \Omega$. Additionally, let P_Σ be a probability distribution on Σ that denotes the possibility of each input to appear, let $\sigma \sim P_\Sigma$ be a random input.

Let $H : \Omega \times \Omega \to \{0, 1\}$ be the **average human evaluator function**, which assesses the relative quality of two outputs. $H(y_0, y_1) = 0$ indicates that the output y_0 is preferred over y_1 by an average human expert, and $H(y_0, y_1) = 1$ indicates the opposite. Let $T_e : \Omega \times \Omega \to \{0, 1\}$ be the **LLM evaluator function**, which represents the preference of a certain LLM evaluator e. Let P be a probability measure that encapsulates the stochastic nature of σ, G_1, G_2, H, and T_e.

Given the notations above, we define the following variables:

Definition 1 (True win rate)**.** *The true win rate p is defined as:*

$$p \triangleq P\left(H(G_0(\sigma), G_1(\sigma)) = 0\right) \tag{1}$$

Definition 2 (Observed win rate)**.** *The observed win rate k of an LLM evaluator e is defined as:*

$$k_e \triangleq P\left(T_e(G_0(\sigma), G_1(\sigma)) = 0\right) \tag{2}$$

Intuitively, the true win rate p is the probability that G_0 will generate a "truly better" output than G_1 when they are given the same, arbitrary input, where "truly better" means being regarded as "better" by a human expert on average. Similarly, the observed win rate k is the probability that G_0 will be evaluated by an LLM evaluator as generating a better output than G_1 when they are given the same, arbitrary input.

Due to the complexity of the stochasticity in p and k_e, it is unrealistic to derive them analytically. However, given a large number of input-output pairs evaluated by human and LLM evaluators, we can approximate p and k_e empirically. We formalize it as follows.

Assume n is a large number. Then for n outputs $y_i^{(0)}$ ($i \in [n]$) generated by G_0 and n outputs $y_i^{(1)}$ ($i \in [n]$) generated by G_1 given the same n inputs of interest, we let a human evaluator h and the LLM evaluator e of interest carry out n comparison tasks, where the i-th comparison task is between $y_i^{(0)}$ and $y_i^{(1)}$. Then the true win rate p and the observed win rate k_e can be empirically approximated with

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n}\left[1 - H_h(y_i^{(0)}, y_i^{(1)})\right] \tag{3}$$

$$\hat{k}_e = \frac{1}{n}\sum_{i=1}^{n}\left[1 - T_e(y_i^{(0)}, y_i^{(1)})\right] \tag{4}$$

where $H_h : \Omega \times \Omega \to \{0, 1\}$ is the human evaluator function of a specific human evaluator h (or an aggregation of multiple human evaluators). Note that in our experiments, in order to make sure that $\hat{p}$ is an accurate estimator of p, we assume that the preference of h is representative of an average human expert evaluator.

3.1.2 Evaluator accuracy

We also define two variables q_0^e (**true positive evaluation accuracy**) and q_1^e (**true negative evaluation accuracy**) associated with an LLM evaluator e[2]. Given two arbitrary outputs generated under the same arbitrary input where the first output is evaluated as "better" than the second one by an average human expert, q_0^e is defined as the conditional probability that e will give the same evaluation as an average human expert. In other words, we have

$$\begin{aligned} q_0^e \triangleq P(T_e(G_0(\sigma), G_1(\sigma)) = 0 \mid \\ H(G_0(\sigma), G_1(\sigma)) = 0) \end{aligned} \tag{5}$$

where the random element $\sigma \in \Sigma$ and probability measure P follow the same notions as in the definitions of p and k. Similarly, we have

$$\begin{aligned} q_1^e \triangleq P(T_e(G_0(\sigma), G_1(\sigma)) = 1 \mid \\ H(G_0(\sigma), G_1(\sigma)) = 1) \end{aligned} \tag{6}$$

Empirically, we can approximate q_0^e and q_1^e with

$$\hat{q_0^e} = \frac{\sum_{i=1}^{n} \mathbb{1}\left[T_e(y_i^{(0)}, y_i^{(1)}) = H_h(y_i^{(0)}, y_i^{(1)}) = 0\right]}{\sum_{i=1}^{n} \mathbb{1}(H_h(y_i^{(0)}, y_i^{(1)}) = 0)} \tag{7}$$

where $\mathbb{1}(\cdot)$ is the indicator function. Similarly, we have

$$\hat{q_1^e} = \frac{\sum_{i=1}^{n} \mathbb{1}\left[T_e(y_i^{(0)}, y_i^{(1)}) = H_h(y_i^{(0)}, y_i^{(1)}) = 1\right]}{\sum_{i=1}^{n} \mathbb{1}(H_h(y_i^{(0)}, y_i^{(1)}) = 1)} \tag{8}$$

3.1.3 Win rate estimation

As we discussed in Section 2, the true win rate p can be used as a metric to compare various generative LLMs. Specifically, for two generative LLMs G_0 and G_1, G_0 outperforms G_1 when $p > 0.5$. Conversely, G_1 outperforms G_0 when $p < 0.5$. Furthermore, the absolute value of p signifies the degree of superiority of one LLM to another. Given a list of LLMs $\Gamma = [G_a, G_b, ...]$ of interest and a certain baseline generative LLM G, we can use the p values of G with respect to each generator in Γ to compare the LLMs in Γ (1 vs. n comparison). Therefore, it is a meaningful question to derive an accurate estimation of p. This is the essential goal of this paper.

3.2 Estimation by observed win rate

A simple approach employed by prior work (Dubois et al., 2024) to approximate p is to directly apply the observed win rate k_e. Here we show that this approach suffers from a bias problem when the evaluator accuracies are not high enough.

By the Law of Total Probability we have

$$\begin{aligned} k_e =& P\left(T_e(G_0(\sigma), G_1(\sigma)) = 0\right) \\ =& P(H(G_0(\sigma), G_1(\sigma)) = 0) \cdot q_0^e + \\ & P(H(G_0(\sigma), G_1(\sigma)) = 1) \cdot (1 - q_1^e) \\ =& p q_0^e + (1-p)(1-q_1^e) \end{aligned} \tag{9}$$

Therefore, k_e has the following value of bias:

$$\begin{aligned} |k_e - p| =& |p q_0^e + (1-p)(1-q_1^e) - p| \\ =& |p q_0^e + p q_1^e - 2p - q_1^e + 1| \end{aligned} \tag{10}$$

We can see that $k_e = p$ if (though not only if) $q_0^e = q_1^e = 1$, which is typically not the case for any non-perfect LLM evaluator. In order to fix this bias problem, we propose the following two methods to improve the accuracy in the estimation of p.

[2]For simplicity, we will use "evaluator accuracies" when we refer to q_0^e and q_1^e together.

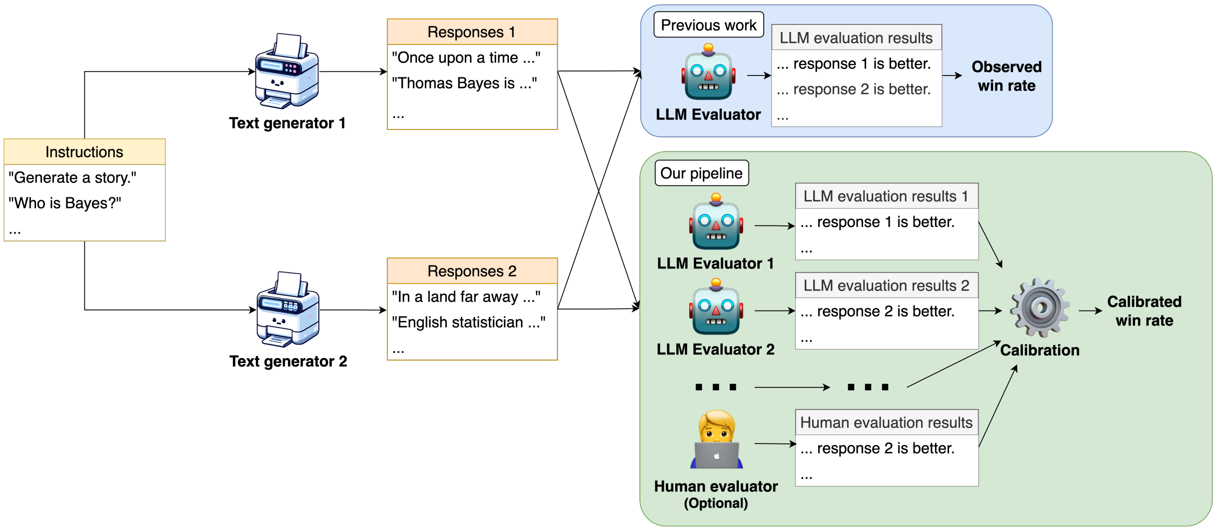


Figure 1: Illustration of our pipeline and previous work. The "calibration" part of our pipeline indicates one of BWRS or Bayesian Dawid-Skene.

3.3 Bayesian Win Rate Sampling

First, we propose a sampling-based algorithm, Bayesian Win Rate Sampling (BWRS), which is shown in Algorithm 1. The intuition of the BWRS algorithm is that, given an LLM evaluator e and a dataset $D = \{(y_i^{(0)}, y_i^{(1)}), i \in [n]\}$ containing outputs generated by G_0 and G_1 with respect to the same set of inputs, we first apply e to generate its annotations $\{T_e(y_i^{(0)}, y_i^{(1)}), i \in [n]\}$ on D, and apply Equation 4 to approximate k_e. Next, assume we have access to some human annotations, either on a small fraction of D or on a similar dataset F, then we are able to approximate q_0^e and q_1^e using Equation 7 and 8. Finally, we apply the following equation rearranged from Equation 9:

$$p = \frac{k_e + q_1^e - 1}{q_0^e + q_1^e - 1} \tag{11}$$

given the assumption that $q_0^e + q_1^e \neq 1$. [3] We can use the approximated values of k_e, q_0^e, and q_1^e to infer one sample of p, which characterizes the relative performance between G_0 and G_1.

Note that there is still one key difference between the intuition above and our actual implementation described in Algorithm 1. In our implementation, instead of estimating k_e, q_0^e, q_1^e directly using Equations 4, 7, 8, we use Bayesian inference and apply Beta-Bernoulli models to estimate the posterior distributions for k_e, q_0^e, and q_1^e. We then obtain N (10000 in our case) samples of p from these distributions using Equation 11 and apply Kernel Density Estimation (KDE) on all the p samples to approximate the distribution of p, and estimate the value of p using the mean $\hat{p}_{mean}$ or mode $\hat{p}_{mode}$ of this distribution. The purpose of applying a Bayesian setting is to incorporate the uncertainty of k_e, q_0^e, q_1^e into consideration, and also facilitate the usage of prior knowledge on evaluator accuracies, which will be discussed in Section 4.3.

[3] In practice, though this assumption is satisfied under most cases, some values of evaluator accuracies might cause sampling failure. Please refer to Limitations for details.

3.4 Bayesian Dawid-Skene model

The vanilla Dawid-Skene model (Dawid and Skene, 1979) is optimized with the Expectation-Maximization (EM) algorithm. Following (Paun et al., 2018), we instead use a Bayesian Dawid-Skene model with E evaluators. The pseudocode of our model is shown in Model 1. The parameters in this model include $\alpha_p, \beta_p, \alpha_{q_0}, \beta_{q_0}, \alpha_{q_1}$, and$\beta_{q_1}$. We initialize the distribution of p with a uniform distribution, and thus α_p, β_p are initialized as 1. The initialization of the other parameters will be discussed in Section 4.3. We apply the evaluation results of LLM evaluator e as observations t_i^e, and use Hamiltonian Monte Carlo (HMC) sampling to fit the model and sample from the posterior distribution of p. Similar to BWRS, we use the posterior mean ($\hat{p}_{mean}$) and posterior mode ($\hat{p}_{mode}$) as two estimators of p. In order to improve sampling efficiency, we employ NUTS sampler (Hoffman and Gelman, 2011) and the Binary Gibbs-Metropolis sampler implemented in PyMC (Oriol et al., 2023). We tune and sample from the model with 4 chains, with 10000 tuning steps and 10000 sampling steps

Algorithm 1 Bayesian Win Rate Sampling (BWRS) algorithm

Input: Dataset without human annotation: $D = \{(y_i^{(0)}, y_i^{(1)}), i \in [n]\}$; similar dataset with human annotation (e.g. the OOD set): $F = \{(z_i^{(0)}, z_i^{(1)}), i \in [m]\}$; annotation by LLM evaluator e on D: $D_e = \{T_e(y_i^{(0)}, y_i^{(1)}), i \in [n]\}$; annotation by LLM evaluator e on F: $F_e = \{T_e(z_i^{(0)}, z_i^{(1)}), i \in [m]\}$; annotation by human evaluator h on F: $F_h = \{H_h(z_i^{(0)}, z_i^{(1)}), i \in [m]\}$; Number of samples drawn for Bayesian inference: N
Output: An estimation of the true win rate p
▷ Total number of data points on F with either human evaluation result (0 or 1)
$n_0 = |\{(z_i^{(0)}, z_i^{(1)}) \in F : H_h(z_i^{(0)}, z_i^{(1)}) = 0\}|$
$n_1 = |\{(z_i^{(0)}, z_i^{(1)}) \in F : H_h(z_i^{(0)}, z_i^{(1)}) = 1\}|$
▷ Number of correct judgements by e on F
$s_0 = |\{(z_i^{(0)}, z_i^{(1)}) \in F : H_h(z_i^{(0)}, z_i^{(1)}) = T_e(z_i^{(0)}, z_i^{(1)}) = 0\}|$
$s_1 = |\{(z_i^{(0)}, z_i^{(1)}) \in F : H_h(z_i^{(0)}, z_i^{(1)}) = T_e(z_i^{(0)}, z_i^{(1)}) = 1\}|$
$n_k = |D|$
$s_k = |\{(y_i^{(0)}, y_i^{(1)}) \in D : T_e(y_i^{(0)}, y_i^{(1)}) = 0\}|$
for $i = 1, 2, ..., N$ **do**
    ▷ Estimated evaluator accuracies
    Draw $q_0^e \sim \text{Beta}(s_0 + 1, n_0 - s_0 + 1)$
    Draw $q_1^e \sim \text{Beta}(s_1 + 1, n_1 - s_1 + 1)$
    ▷ Observed win rate
    Draw $k_e \sim \text{Beta}(s_k + 1, n_k - s_k + 1)$
    Derive the i-th sample $p_i = \frac{k_e + q_1^e - 1}{q_0^e + q_1^e - 1}$, append to sample list
end for
return mean ($\hat{p}_{mean}$) or mode ($\hat{p}_{mode}$) of KDE($\{p_1, p_2, ..., p_N\}$)

Model 1 Bayesian Dawid-Skene model for two-class problems

▷ Prior class prevalence
Draw $p \sim \text{Beta}(\alpha_p, \beta_p)$
for $e = 1$ **to** E **do**
    ▷ Evaluator accuracies
    Draw $q_0^e \sim \text{Beta}(\alpha_{q_0}, \beta_{q_0})$
    Draw $q_1^e \sim \text{Beta}(\alpha_{q_1}, \beta_{q_1})$
end for
for $i = 1$ **to** n **do**
    ▷ Ground truth labels
    Draw $h_i \sim \text{Bernoulli}(p)$
    for $e = 1$ **to** E **do**
        ▷ Predicted labels
        if $h_i = 1$ **then**
            Draw $t_i^e \sim \text{Bernoulli}(q_1^e)$
        else
            Draw $t_i^e \sim \text{Bernoulli}(1 - q_0^e)$
        end if
    end for
end for

on each chain. On an AMD EPYC 7763 processor, comparing each generator pair takes around 10 minutes.

4 Experiment Settings

4.1 Datasets

The datasets we use in the experiments are HANNA (Chhun et al., 2022), OpenMEVA-MANS (Guan et al., 2021), SummEval (Fabbri et al., 2021), LLMBar (Zeng et al., 2024), MT-Bench (Zheng et al., 2023), and LLMEval^2 (Zhang et al., 2023), covering tasks of story generation (HANNA, OpenMEVA-MANS), summarization (SummEval), and instruction following (the other three). All of them provide machine-generated content with human annotations. For MT-Bench and LLMEval^2, we used the smaller, curated versions prepared by the authors of the LLMBar paper (Zeng et al., 2024). For the three instruction following datasets, since they are presented as a list of (input, output1, output2, human preference) tuples without specifying or fixing the output generators, we simulate two generators based on these datasets by randomly attributing 80% of the human-preferred outputs to the first (simulative) generator and rest 20% to the second such that the true win rate between them is 80%. The choice of the 80%-20% ratio is arbitrary.

A detailed description about each dataset can be found in Appendix A.

4.2 Evaluator settings

For HANNA, OpenMEVA-MANS, and SummEval, we prompt a set of LLM evaluators to compare the outputs of generator models in the datasets. Specifically, we employ GPT-3.5-turbo-0125 (OpenAI, 2023) and Gemini-1.0-Pro (Team, 2024) as the evaluator models for our experiments. GPT-3.5 has been proved to have positive correlation with human annotations (Chiang and Lee, 2023a; Wang et al., 2023a), while Gemini-1.0-Pro's performance on meta-evaluation have not yet been widely studied in previous works. For each output pair, we prompted each LLM evaluator to rate the two outputs that are based on the same input and generated by two different generator models. For each LLM evaluator, we used three prompting strategies including Score-only, Rate-explain, and Analyze-rate following (Chiang and Lee, 2023b). For LLMBar, MT-Bench, LLMEval[2], the LLM evaluation work has already been carried out in (Zeng et al., 2024). For these three datasets, we selected the best LLM evaluators among the many ones used, including evaluators based on GPT-4, PaLM 2, etc. for our experiments. More details regarding the specific LLM evaluator modes used for each dataset can be found in Appendix B.

4.3 Win rate estimation

After obtaining the human evaluation and LLM evaluation data, we apply BWRS (Section 3.3) and Bayesian Dawid-Skene model (Section 3.4) to each dataset described above. Additionally, we calculate the observed win rate (k) using Equation 4 averaged over the results of all LLM evaluators combined. The error of estimating p with the observed win rate, i.e. $|k-p|$, acts as a baseline that shows the aggregated performance of the LLM evaluators applied without any calibration.

In order to further study the effectiveness of each estimation method, we also explore their performance given the following three different sources of human evaluation results. For simplicity, we refer to these human evaluation results as **prior**s, since they act as prior knowledge of human preferences in our methods.

No prior[4]. We assume no prior knowledge of q, and only depend on the Dawid-Skene model to estimate the accuracy of each evaluator. In this case, we initialize the parameters of evaluator accuracies in Model 1 with $\alpha_{q_0} = \alpha_{q_1} = 2, \beta_{q_0} = \beta_{q_1} = 1$, which is a beta distribution skewed towards higher q_0 and q_1 values, because we expect our evaluators to generally perform better than random guessing such that $q_0 > 0.5$ and $q_1 > 0.5$.

In-distribution prior. We assume that we have access to human evaluations on a subset of all output pairs generated by the two generators of interest. In BWRS, these human evaluation results are used as F_h in Algorithm 1 to obtain an estimate of each LLM evaluator's accuracies q_0, q_1. In the Bayesian Dawid-Skene model, they are instead used as observations (h_i in Model 1), while α_{q_0}, β_{q_0}, α_{q_1}, and β_{q_1} are initialized in the same way as in the no prior setting. We refer to the ratio of human-evaluated output pairs over the entire dataset as **prior data ratio**. In our experiments, we try 10 different values of prior data ratio (0.1, 0.2, ..., 1.0) and compare the results.

Out-of-distribution (OOD) prior. We assume that we have access to human evaluations on some other datasets beyond comparing the two generators of interest. These human evaluation results are also used to calculate priors for q_0 and q_1. In our experiments, we use the generator pair in the dataset that has the closest observed win rate with the compared generators. For BWRS, these priors are used as F_e and F_h in Algorithm 1. For the Bayesian Dawid-Skene model, with the in-distribution prior setting, the priors are used as observations of ground truth labels h_i in Model 1. For the OOD prior setting, they are instead used to derive a prior distribution of the evaluator accuracies so that the model won't be affected as much by the distribution shift of evaluator accuracies on different generator models. Specifically, we use a Beta-Bernoulli model similar to the ones we used in BWRS. The only difference is that we normalize the Beta parameters to have a mean value of 1 in order to prevent over-confident priors. Concretely, we initialize the distributions of q_0^e and q_1^e in Model 1 for each evaluator e as follows:

$$
\begin{aligned}
n_0 =& |\{(z_i^{(0)}, z_i^{(1)}) \in \text{OOD}: \ H_h(z_i^{(0)}, z_i^{(1)}) = 0\}| \\
n_1 =& |\{(z_i^{(0)}, z_i^{(1)}) \in \text{OOD}: \ H_h(z_i^{(0)}, z_i^{(1)}) = 1\}| \\
s_0 =& |\{(z_i^{(0)}, z_i^{(1)}) \in \text{OOD}: \\
& \quad H_h(z_i^{(0)}, z_i^{(1)}) = T_e(z_i^{(0)}, z_i^{(1)}) = 0\}| \\
s_1 =& |\{(z_i^{(0)}, z_i^{(1)}) \in \text{OOD}: \\
& \quad H_h(z_i^{(0)}, z_i^{(1)}) = T_e(z_i^{(0)}, z_i^{(1)}) = 1\}| \\
q_0^e \sim& \text{Beta}(\frac{2s_0+2}{n_0+2}, \frac{2n_0-2s_0+2}{n_0+2}) \qquad (12) \\
q_1^e \sim& \text{Beta}(\frac{2s_1+2}{n_1+2}, \frac{2n_1-2s_1+2}{n_1+2}) \qquad (13)
\end{aligned}
$$

where OOD is the OOD set (dataset F) we use, the term $n_0 + 2$ and $n_1 + 2$ on the denominator of Equation 12 and 13 are both normalization terms as described above.

[4]The no prior setting is not applicable for BWRS, since BWRS requires informative priors of evaluator accuracies to be accurate.

Evaluator model	Prompt template	q_0	q_1	$\lvert q_0 - q_1 \rvert$	Overall Accuracy
Gemini-1.0-Pro	Score-only	0.782	0.526	0.256	0.649
	Analyze-rate	**0.802**	0.428	0.374	0.607
	Rate-explain	0.760	0.512	0.248	0.631
GPT-3.5	Score-only	0.700	0.653	0.047	**0.676**
	Analyze-rate	0.657	**0.677**	0.020	0.667
	Rate-explain	0.699	0.655	0.044	**0.676**

Table 1: Overall average evaluator accuracies across all pair-wise comparisons across all of HANNA, OpenMEVA-MANS, and SummEval. Best performance on each column is marked with bold font.

Dataset	Method	Prior setting	$\lvert \hat{p}_{mean} - p \rvert$	$\lvert \hat{p}_{mode} - p \rvert$
HANNA	Observed win rate (baseline)	/	**0.079**	**0.079**
	Bayesian Dawid-Skene	No prior	0.129	0.132
	Bayesian Dawid-Skene	OOD prior	0.084	0.081
	BWRS	OOD prior	0.129	0.095
OpenMEVA-MANS	Observed win rate (baseline)	/	0.065	0.065
	Bayesian Dawid-Skene	No prior	0.065	0.065
	Bayesian Dawid-Skene	OOD prior	0.034	**0.033**
	BWRS	OOD prior	0.064	0.102
SummEval	Observed win rate (baseline)	/	0.167	0.167
	Bayesian Dawid-Skene	No prior	0.125	0.123
	Bayesian Dawid-Skene	OOD prior	0.115	**0.110**
	BWRS	OOD prior	0.112	0.112

Table 2: Results of win rate estimation with no prior on HANNA, OpenMEVA-MANS, and SummEval. All results are averaged over ten repetitive runs over all six evaluator modes given in Table 1. The best estimator for each dataset is marked with bold font.

5 Results

In this section, we first analyze the evaluator accuracies on our datasets, and then list the results of our experiments, including win rate estimation with no prior, OOD prior, and in-distribution prior. We show that both our methods are able to effectively calibrate the estimation of win rate given good estimations of evaluator accuracies. We also show that even with no or OOD knowledge of human preference, our methods are still able to perform well overall.

5.1 Evaluator accuracies

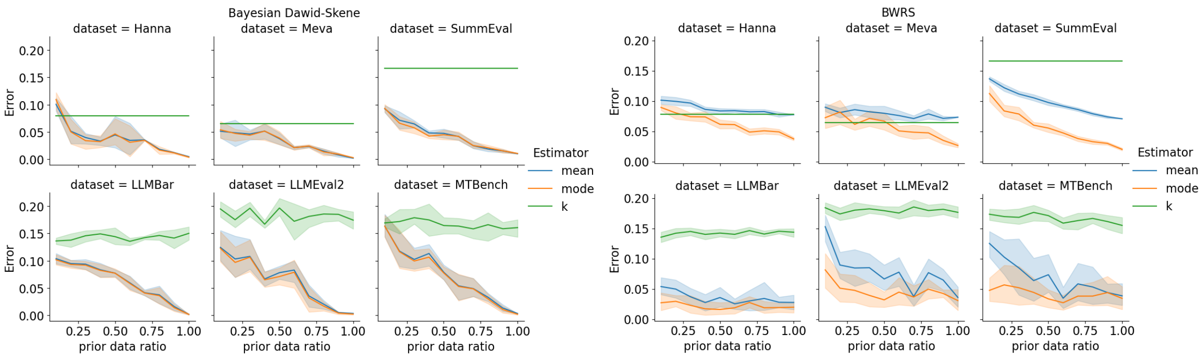
For the three non-instruction following datasets (HANNA, OpenMEVA-MANS, SummEval) on which we carry out LLM evaluation by ourselves, the average accuracies of LLM evaluators are shown in Table 1. The overall accuracy is defined as the proportion of all pair-wise comparisons where the LLM evaluation aligns with human evaluation. For "all pairwise comparisons," we actually mean the "1 vs. n" comparisons where the GPT-2 text generator is compared to all the other text generators in the dataset. We employ this "1 vs. n" comparison strategy because the corresponding "n vs. n" strategy is much more costly in terms of computation time and budget. We can see that:

- In terms of overall accuracy, there is not a significant difference (>5%) between the three prompt templates.
- There is a significant difference between q_0 and q_1 even though we applied the swap-and-sum strategy (see Appendix A). This can be attributed to the correlation between evaluator accuracy and the difference between the generators' capabilities. When one generator is significantly better than the other, it is easier for the LLM evaluator to identify cases where the better generator does better, and harder when the better generator does worse. Also, Gemini-1.0-Pro evaluators suffer from this problem more significantly than GPT-3.5 evaluators. This shows the necessity of modeling q_0 and q_1 separately for each evaluator when comparing two generators.

For the instruction following datasets (LLMBar, LLMEval[2], MT-Bench), the overall evaluator accuracies are given in the LLMBar paper (Zeng et al., 2024), where the overall evaluation accuracies are generally above 70% for the evaluator modes we use.

Dataset	Method	Prior setting	$\lvert\hat{p}_{mean} - p\rvert$	$\lvert\hat{p}_{mode} - p\rvert$
LLMBar	Observed win rate (baseline)	/	0.142	0.142
	Bayesian Dawid-Skene	No prior	0.140	**0.138**
LLMEval[2]	Observed win rate (baseline)	/	0.178	0.178
	Bayesian Dawid-Skene	No prior	0.157	**0.156**
MT-Bench	Observed win rate (baseline)	/	**0.162**	**0.162**
	Bayesian Dawid-Skene	No prior	0.190	0.188

Table 3: Results of win rate estimation with no prior on the three instruction following datasets. All results are averaged over ten repetitive runs over all evaluator modes. The variance of all runs are insignificant ($< 10^{-2}$). The best estimator for each dataset is marked with bold font.



(a) Bayesian Dawid-Skene

(b) BWRS

Figure 2: Win rate estimation error with various proportions of the original data used for in-distribution prior measurement. The results are averaged over all generator pairs over all evaluator modes. The mean and variance of all results are calculated over ten repetitive runs. The variance of k values in the three instruction following datasets are results of randomly assigning outputs to two simulative generators, as described in Section 4.1

5.2 Win rate estimation results

The results of win rate estimation with no prior and OOD prior on HANNA, OpenMEVA-MANS, and SummEval are shown in Table 2. We can observe that:

- The mode estimator in Bayesian Dawid-Skene with OOD prior is the overall best estimator. In this setting, estimation of p is more accurate than baseline (k) in all datasets except HANNA.
- The Bayesian Dawid-Skene model with OOD prior is more accurate than the model with no prior. This shows that the OOD prior is able to provide some useful information on the accuracy of each evaluator, which helps the Bayesian model converge to a better result.

The results of win rate estimation with no prior on LLMBar, LLMEval[2], and MT-Bench are shown in Table 3. Note that OOD prior is not applicable for these instruction following datasets due to the absence of relevant data to act as the OOD set. We can see that the mode estimator in Bayesian Dawid-Skene with no prior outperforms the baseline in all datasets except MT-Bench.

The results of BWRS and Bayesian Dawid-Skene with in-distribution prior are shown in Figure 2. We can observe the following:

- As prior data ratio increases, win rate estimation accuracy of both BWRS and Bayesian Dawid-Skene improves. This enhancement arises because having more human annotations for in-distribution data allows for a more precise assessment of evaluator accuracies and consequently leads to a more accurate estimation of the true win rate p. This shows that our methods will indeed offer a more accurate estimation of the true win rate p if we have good estimations of q_0 and q_1.
- The mode estimator shows consistently better performance compared with the mean estimator and k.
- The proportion of human evaluation data needed to ensure improvement of the true win

rate estimation varies for each dataset due to the internal variance of evaluator accuracies. Generally, a prior data ratio of 30% would be sufficient for both Bayesian Dawid-Skene and BWRS, with one exception (BWRS for OpenMEVA-MANS).

6 Conclusion

In this paper, we identified the bias problem in win rate estimation using non-perfect LLM evaluators, and proposed two methods, BWRS and Bayesian Dawid-Skene, in order to address this issue. We then obtained LLM evaluation results on six datasets, and used these results to examine the effectiveness of our methods empirically. Our results show that both BWRS and Bayesian Dawid-Skene can effectively reduce the error in win rate estimation, especially given good approximations on evaluator accuracies. We also showed that even without in-distribution prior knowledge of human preferences, our methods are still able to effectively calibrate the estimation of win rate under most cases. The effectiveness of our methods manifests the possibility to calibrate the estimation of win rate in a post-hoc manner after LLM evaluations are completed, and also enlightens future study on applying annotation models for accurate win rate estimation using LLM evaluators.

Limitations

There are some limitations of our work. First, due to budget limit, for the non-instruction following datasets, we only examined our methods with GPT-3.5 and Gemini-1.0-Pro as LLM evaluators. Although we did incorporate more advanced LLM evaluators such as GPT-4 and PaLM 2 on the instruction following datasets, it would be illuminating to examine how more advanced evaluator models would affect our methods' performance on the non-instruction following datasets.

Second, the performance of both methods with OOD prior largely depends on the quality of OOD data. Specifically, when there is a large difference between evaluator accuracies on the OOD set and on the original dataset, our methods may produce highly-biased results. Therefore, in cases where human evaluation results on datasets with similar observed win-rates are absent, we would recommend against using OOD prior.

This paper is an exploratory study on adjusting bias of LLM evaluators. Besides resolving the limitations above, the exploration in this field could also be extended in the following aspects:

- Applying more complex annotator models. As discussed in Section 2, the Dawid-Skene model is the earliest annotator model proposed, and several improvements have been proposed since then. These improved methods can lead to potentially more accurate estimations of win rate.
- Introducing more robust methods. The performance of our proposed methods is contingent upon the accuracy of LLM evaluators. Concretely, from Equation 11 we know that

$$0 < p < 1 \Leftrightarrow \begin{cases} 1 - q_1^e < k_e < q_0^e, & q_0^e + q_1^e > 1 \\ q_0^e < k_e < 1 - q_1^e, & q_0^e + q_1^e < 1 \end{cases} \tag{14}$$

  We can see that, in order to make sure $p \in [0, 1]$, the evaluator accuracies q_0^e and q_1^e must satisfy one of the conditions in Equation 14. In cases where neither condition is satisfied, our methods can become unstable, and is prone to produce p distributions with high bias and/or variance. We leave it for future research to propose methods that work well for LLM evaluators with low or unstable accuracies.

References

Paul S Albert and Lori E Dodd. 2004. A cautionary note on the robustness of latent class models for estimating diagnostic error without a gold standard. *Biometrics*, 60(2):427–435.

Bob Carpenter. 2008. Multilevel bayesian models of categorical data annotation. *Unpublished manuscript*, 17(122):45–50.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. Chateval: Towards better llm-based evaluators through multi-agent debate.

Yanran Chen and Steffen Eger. 2023. Menli: Robust evaluation metrics from natural language inference.

Cyril Chhun, Pierre Colombo, Fabian M. Suchanek, and Chloé Clavel. 2022. Of human criteria and automatic metrics: A benchmark of the evaluation of story generation. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5794–5836, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Cheng-Han Chiang and Hung-yi Lee. 2023a. Can large language models be an alternative to human evaluations? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.

Cheng-Han Chiang and Hung-yi Lee. 2023b. A closer look into using large language models for automatic evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8928–8942, Singapore. Association for Computational Linguistics.

Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2024. Alpacafarm: A simulation framework for methods that learn from human feedback.

Alexander Fabbri, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. 2022. QAFactEval: Improved QA-based factual consistency evaluation for summarization. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2587–2601, Seattle, United States. Association for Computational Linguistics.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

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. Association for Computational Linguistics.

Jian Guan, Zhexin Zhang, Zhuoer Feng, Zitao Liu, Wenbiao Ding, Xiaoxi Mao, Changjie Fan, and Minlie Huang. 2021. OpenMEVA: A benchmark for evaluating open-ended story generation metrics. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6394–6407, Online. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1693–1701, Cambridge, MA, USA. MIT Press.

Matthew D. Hoffman and Andrew Gelman. 2011. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Hyun-Chul Kim and Zoubin Ghahramani. 2012. Bayesian classifier combination. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 619–627, La Palma, Canary Islands. PMLR.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. 2024a. Prometheus: Inducing fine-grained evaluation capability in language models.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024b. Prometheus 2: An open source language model specialized in evaluating other language models.

Ryan Koo, Minhwa Lee, Vipul Raheja, Jong Inn Park, Zae Myung Kim, and Dongyeop Kang. 2023. Benchmarking cognitive biases in large language models as evaluators.

Ruosen Li, Teerth Patel, and Xinya Du. 2023. Prd: Peer rank and discussion improve large language model based evaluations.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023a. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Yuxuan Liu, Tianchi Yang, Shaohan Huang, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, and Qi Zhang. 2023b. Calibrating llm-based evaluator.

OpenAI. 2023. Chatgpt. Large language model.

Abril-Pla Oriol, Andreani Virgile, Carroll Colin, Dong Larry, Fonnesbeck Christopher J., Kochurov Maxim, Kumar Ravin, Lao Jupeng, Luhmann Christian C., Martin Osvaldo A., Osthege Michael, Vieira Ricardo, Wiecki Thomas, and Zinkov Robert. 2023. Pymc:

A modern and comprehensive probabilistic programming framework in python. *PeerJ Computer Science*, 9:e1516.

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 on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Rebecca J. Passonneau and Bob Carpenter. 2014. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Silviu Paun, Bob Carpenter, Jon Chamberlain, Dirk Hovy, Udo Kruschwitz, and Massimo Poesio. 2018. Comparing Bayesian models of annotation. *Transactions of the Association for Computational Linguistics*, 6:571–585.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*, volume 34, pages 4816–4828. Curran Associates, Inc.

Gemini Team. 2024. Gemini: A family of highly capable multimodal models.

Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023a. Is ChatGPT a good NLG evaluator? a preliminary study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 1–11, Singapore. Association for Computational Linguistics.

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.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. 2009. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. AlignScore: Evaluating factual consistency with a unified alignment function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *ArXiv*, abs/1904.09675.

Xinghua Zhang, Bowen Yu, Haiyang Yu, Yangyu Lv, Tingwen Liu, Fei Huang, Hongbo Xu, and Yongbin Li. 2023. Wider and deeper llm networks are fairer llm evaluators.

Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I Jordan. 2016. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. *Journal of Machine Learning Research*, 17(102):1–44.

Ruochen Zhao, Wenxuan Zhang, Yew Ken Chia, Deli Zhao, and Lidong Bing. 2024. Auto arena of llms: Automating llm evaluations with agent peer-battles and committee discussions.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

A Dataset details

HANNA (Chhun et al., 2022) includes 1056 stories annotated by human raters with a 5-point Likert scale on 6 criteria: Relevance, Coherence, Empathy, Surprise, Engagement, and Complexity. These 1056 stories are based on 96 story prompts from the WritingPrompts (Fan et al., 2018) dataset. For each story prompt, HANNA collects 11 stories generated by 10 different generation models and a human, respectively. For our purpose of comparing automatic text generation systems, we did not use the stories written by humans in our experiments.

OpenMEVA-MANS (Guan et al., 2021) is a sub-dataset within the OpenMEVA dataset. It contains 1000 stories generated by 5 generation models based on 200 prompts from WritingPrompts (Fan et al., 2018). The overall quality of each story is rated by five humans on a 5-point Likert scale.

SummEval (Fabbri et al., 2021) includes 1600 summaries annotated by human expert annotators with a 5-point Likert scale on 4 criteria: coherence, consistency, fluency, and relevance. These 1600 summaries are based on 100 source articles from the CNN/DailyMail dataset (Hermann et al., 2015). For each source article, SummEval collects 16 summaries generated respectively by 16 different automatic summary generation systems. Each summary is scored by three human expert annotators.

LLMBar (Zeng et al., 2024) consists of 419 instances, each containing an instruction paired with two outputs: one that faithfully follows the instruction and another that deviates from it but may possess superficially appealing qualities. The dataset is divided into two main parts: the Natural set, which includes instances from existing human-preference datasets that have been filtered and modified to ensure objective preferences, and the Adversarial set, which contains outputs crafted to mislead evaluators by emphasizing superficial qualities. LLMBar aims to provide a more rigorous and objective evaluation of LLM evaluators compared to previous benchmarks, achieving a high human agreement rate of 94% (Zeng et al., 2024).

MT-Bench (Zheng et al., 2023) comprises 80 questions and answers to these questions generated by six models. For each question and each pair of models, an evaluation task was constructed, totaling 1200 tasks. The actual dataset that we used is a subset of the original MT-Bench dataset curated by the authors of (Zeng et al., 2024), to construct which they labelled a human-preferred answer for each task using majority vote, removed all the "tie" instances, and then randomly sampled 200 instances. We found five instances of this curated subset repeated themselves once, so we further removed these repeated ones and used the remaining 195 instances for our experiments.

LLMEval^2 (Zhang et al., 2023), similar to MT-Bench, is a question answering dataset where each instance comprises a question and two answers to that question. It consists of 2553 instances, each annotated with human preferences. The actual dataset that we used is a subset of the original LLMEval^2 dataset (Zhang et al., 2023) curated by the authors of (Zeng et al., 2024), to construct which they removed all the "tie" instances and then randomly sampled 200 instances.

For each dataset with multiple human evaluations on each piece of generated text, we average the human evaluation scores as the final human evaluation score for each piece of text.

B Evaluator setup details

We prepared prompt templates into which the input and the two outputs would be inserted. Specifically, we used the following three prompting strategies following (Chiang and Lee, 2023b).

The **Score-only** prompting strategy asks the LLM evaluator to only output the attribute scores of the generated texts without any further explanations.

The **Rate-explain** prompting strategy asks the LLM evaluator to rate the generated texts first and then provide an explanation for its ratings.

The **Analyze-rate** prompting strategy asks the LLM evaluator to first analyze the generated texts and then give the ratings for them.

Additionally, it has been reported that LLM evaluators suffer from position bias (Wang et al., 2023b), meaning that their decisions are often falsely correlated with the order of presenting the compared texts. In order to address this problem, we employ a straightforward **swap-and-sum** strategy inspired by the LLMBar paper (Zeng et al., 2024). For each pair of outputs to be compared, we query the LLM evaluator twice with the original and swapped ordering of the outputs. We then sum the scores given by the LLM evaluator in the two queries and choose the generated text with the higher total score as the LLM-evaluated winner. In cases where the total score is even for both outputs, we consider their quality to be equal, and randomly select one as the winner.

The details of the LLM evaluator modes used by our experiments can be found in Tables 4 and 5. For the prompting templates used for the three instruction following datasets shown in Table 5, please refer to the LLMBar paper (Zeng et al., 2024) for detailed explanations.

Dataset	Evaluator model	Prompt template
HANNA	GPT-3.5 Turbo	Score-only
		Rate-explain
		Analyze-rate
	Gemini-1.0-Pro	Score-only
		Rate-explain
		Analyze-rate
OpenMEVA-MANS	GPT-3.5 Turbo	Score-only
		Rate-explain
		Analyze-rate
	Gemini-1.0-Pro	Score-only
		Rate-explain
		Analyze-rate
SummEval	GPT-3.5 Turbo	Score-only
		Rate-explain
		Analyze-rate
	Gemini-1.0-Pro	Score-only
		Rate-explain
		Analyze-rate

Table 4: LLM evaluator modes used for the story generation and summarization datasets in our experiments.

Dataset	Evaluator model	Prompt template
LLMBar	GPT-4	CoT
		Metrics
		Metrics Reference
		Reference
		Swap
		Swap CoT
		Vanilla
		Vanilla NoRules
	PaLM 2	Metrics Reference
		Reference
		Swap
		Swap CoT
		Vanilla
		Vanilla NoRules
LLMEval[2]	ChatGPT	Metrics Reference
		Vanilla NoRules
	GPT-4	Metrics Reference
		Vanilla NoRules
	Llama 2	Metrics Reference
		Vanilla NoRules
	PaLM 2	Metrics Reference
		Vanilla NoRules
MT-Bench	ChatGPT	Metrics Reference
		Vanilla NoRules
	GPT-4	Metrics Reference
		Vanilla NoRules
	Llama 2	Metrics Reference
		Vanilla NoRules
	PaLM 2	Metrics Reference
		Vanilla NoRules

Table 5: LLM evaluator modes used for the instruction following datasets in our experiments.