Investigating Bias in LLM Self-Evaluation

Thesis

Mathematics Expert in Data Analytics and Machine Learning

ELTE | FACULTY OF SCIENCE

Eötvös Loránd University
Faculty of Science

Budapest, 2025

Contents

Abstract

This thesis explores whether large language models (LLMs) tend to overestimate the quality of their own outputs when serving as judges or evaluators. Preliminary observations suggest that using the same or closely related LLM as both generator and judge may inflate performance metrics. Through systematic experiments, the project will quantify this potential bias and discuss its implications for AI evaluation, fairness, and trustworthiness in model benchmarking.

1 Introduction

This section introduces the key terms and core concepts that will be used in later sections.

1.1 A Brief Introduction to LLMs

A language model is a machine learning model designed to perform a wide range of tasks that involve natural language processing (NLP), including text summarization, translation, sentiment analysis, spam detection, content moderation, text generation, etc.

Significant advancements in deep learning, like the transformer architecture [1, 2, 3], led to the emergence of **large language models** (LLMs) — particularly generative LLMs — which in the early 2020s became commercialized and widely adopted in both industry and popular discourse.

A generative LLM is a model which has a parameter count on the order of hundreds of billions or more (hence ”large”), and predicts the conditional probability [4]

$$P(w_m|w_0, \cdots, w_{m-1}) \tag{1}$$

where $m \in \mathbb{N}$, w_0 is a special start symbol, and w_k is the k-th token (for $1 \leq k \leq m$) in a sequence of tokens that form a piece of text in some language, be it a natural language or a formal one like programming languages. The interpretation of the tokens depends on the exact tokenization strategy used, which may define tokens as words, word pieces, n-grams, or individual characters, and spaces, punctuation marks, etc.

Encoding is the process which converts human-readable textual tokens into integers which uniquely identify each token within the predetermined vocabulary of the model, and the inverse of this mapping is called **decoding**. [1]

Text generation is an autoregressive process where given a sequence of tokens as a prefix — known as the **prompt** — the model estimates the probability distribution of the next token, takes a sample from that distribution, appends it to the sequence, and repeats the process with the extended sequence until a stopping condition is met.

A frequently used parameter to control the sampling is called the **temperature** [5]: the closer it is to 0, the more the sampling will lean toward the most probable token — making the algorithm more deterministic —, while higher values increase the randomization, making the generated text feel more *creative* until, above a certain threshold, it becomes incoherent and semantically meaningless. [2] In practical implementations, if the temperature is sufficiently close or exactly equal to 0, then the sampling is usually replaced with the deterministic argmax function in order to preserve numerical stability. Non-zero temperature values control the flatness of the distribution, leading to the aforementioned behavior.

With sufficiently large model complexity and training corpora size and diversity, LLMs start to exhibit capabilities which rival that of top performer humans in a

[1] Internally, the token numbers are mapped by a trainable model to vectors within a vector space called the **embedding space**. The choice for the dimensionality of this space allows a significant dimensionality reduction compared to what would be necessary for example to represent the tokens with one-hot encoding. An interesting property of the embedding space is that it tends to map tokens that are close to each other in meaning to vectors which are close to each other in the space.

[2] If $v \in \mathbb{N}$ denotes the number of all possible tokens available for the model (vocabulary size), and $\mathbf{s} \in \mathbb{R}^v$ is an output vector of the model assigning a score to each token as the continuation of a given input, then the distribution for the sampling, with respect to the temperature $T \in \mathbb{R}$ can be calculated via the softmax function: $\text{softmax}\left(\frac{1}{T}\mathbf{s}\right) = \left[\frac{\exp(\frac{1}{T}s_i)}{\sum_{j=1}^{v}\exp(\frac{1}{T}s_j)}\right]_{i=1}^{v}$

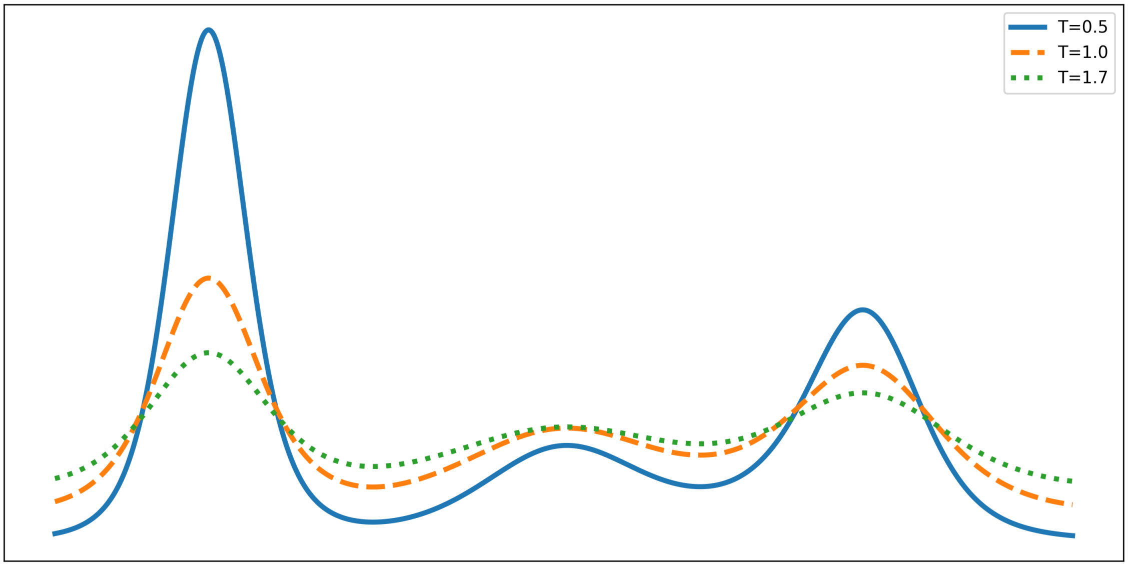


Figure 1: Effect of the temperature parameter on token probabilities.

broad class of problems [2, 3]. The versatility of the models is often utilized in a setting where the prompt is composed of two parts, each consisting of instructions given in natural language:

- the **system prompt** can instruct the model to behave in a certain way, for example, to act like a helpful AI assistant, an expert in a domain, or to generate its texts in the style of fictional 18th-century Caribbean pirates, etc.
- and the **user prompt** which describes the task to be carried out by the model, ranging from text translation or summarization to solving complex programming problems or pointing out business risks in legal documents, and more.

Generative models with sufficient generalization capabilities can predict likely continuations of such prompts with such high accuracy that as an emergent phenomenon, the generated text will often contain an actual solution to the proposed problem. This instruction-following paradigm enables models to perform **few-shot learning** [2] or even **zero-shot learning** by interpreting tasks directly from the natural language description, based on just a few or zero examples, respectively, without specific training or fine-tuning.

The problem solving performance of LLMs can be improved further by prompt engineering techniques like **chain-of-thought** prompting [6], where the model is provided with step-by-step example solutions to related problems in the prompt, encouraging it to also articulate intermediate reasoning steps before arriving at its final answer. It is worth emphasizing that — recalling formula 1 and the autoregressive text generation process — the chain-of-thought is only effective if it is placed *before* the final answer.

1.2 LLM Evaluators, LLM-as-a-Judge

The continuing development of LLMs and their integration into more and more systems to support a growing number of use cases necessitates regular measurement of their capabilities and monitoring their alignment with human preferences.

While evaluating the quality of LLM-generated text by utilizing human labor does not scale well, may suffer from human error or subjective personal preference bias, and can be expensive, traditional algorithmic metrics which often rely on surface-level similarities to reference examples (like BLEU for machine translation [7] or ROUGE for summarization and translation [8]), often fall short of achieving acceptable correlation levels with human judgement.

In recent years, in order to overcome these problems, the **LLM-based evaluation** or **LLM-as-a-judge** paradigm has been proposed [9, 10, 11, 12], where — taking advantage of the instruction following and the zero-shot and few-shot learning capabilities of LLMs — a model is instructed to act as a fair judge and generate a quality assessment for a piece of generated text either in the form of a single score, or one accompanied by an explanation or a list of problems. An advantage of the latter approach — besides easier interpretability — is that enumerating evidences before giving a final result can influence the score via the autoregressive generation process, similarly to the improvements achieved by making large models include a chain-of-thought [6] breakdown of complex problems before the final answer.

1.2.1 LLM-Judge Prompting Basics

There are numerous strategies to implement LLM-judges in practice [13], but a robust LLM-judge prompt usually includes the following elements:

- **Instructions** which clearly specify the evaluation task.
- Evaluation **aspects**, e.g. clarity, consistency, coherence, factuality, fluency, grammaticality, informativeness, structure, understandability, etc.
- Scoring **criteria** to specify the definitions for each score or score range.
- **Output format** specification so that the output of the judge can be programmatically parsed and interpreted.
- The **sample** itself to be evaluated or a pair of samples to be compared against each other.

Depending on the chosen evaluation strategy and aspect, additional elements may be included as well:

- Human-annotated **example** samples and their associated scores in few-shot evaluation scenarios.
- A **reference** answer for comparison with the evaluated sample, e.g. a human expert made translation, text summary, trivia answer, etc.
- The **source** data from which the evaluation sample was derived. (The original text to be translated, summarized, or the question to be answered, etc.)
- **Guidelines**, for example to help an LLM resolve the confusion that may arise in reference answer-based evaluations where some of the provided reference answers seem to contradict the model's own knowledge, e.g. "*Don't worry about factuality with respect to the real world, just judge the example based on what you see. No need to overthink this task, it really comes down to just soft matching.*" [14].

```
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, please
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 2: System prompt with chain-of-thought and user prompt template for an LLM-judge [15].

Constructing the prompt template for a consistent, reproducible, and unbiased LLM-judge which also aligns well with human preferences is usually an iterative process, where the prompt is refined step-by-step until the LLM-judge can reliably produce evaluations that are sufficiently close to a set of human-labeled examples.

The juding model may also be fine-tuned using evaluation data constructed either manually or with the assistance of advanced models like GPT-4.

1.2.2 Metrics

Popular choices for scoring strategy include:

- **Binary classification**: the judge is expected to provide a ”*yes*” vs. ”*no*”, or a 0 vs. 1 verdict.

- **Pairwise comparison**: the judge is given two candidate answers, and has to select the one that is a better fit for the evaluation criteria. [3] Optionally, the judge may be allowed to declare a tie.

[3] This strategy can be generalized as **listwise comparison** where the judge is asked to select the best candidate among 3 or more candidates.

- **Multiclass classification**: the judge has to place the candidate on a discrete scale, usually between 1 and 5 points where 1 is the worst and 5 is the best.

- **Likert-style**: the judge has to rank the candidate answer along multiple dimensions using discrete scores, usually between 1 and 3 points where a higher score is better, then provide an overall 1 to 5 rating based on these scores.

- **Continuous score**: the candidate answer is scored with a number between 0 and 100.

If the judge LLM's interface makes the raw token probabilities available, then they can be used for refining discrete scores and making them into continuous ones by taking the sum of the discrete score values weighted by the probabilities of the respective tokens, as seen in the G-EVAL framework [12]:

$$\text{score} = \sum_{i=1}^{n} p(s_i) \times s_i \tag{2}$$

where $S = \{s_1, s_2, \ldots, s_n\}$ is the set of scores predefined in the prompt, and $p(s_i)$ are the probabilities of the respective tokens for the score values, as calculated by the model.

Another way to turn a discrete score into a continuous one is used in the GEMBA metric [16] for assessing translation quality: it requires the candidate answer to be dividable into smaller segments which are then evaluated one-by-one, and the resulting scores are averaged.

1.2.3 AutoCalibrate: Using an LLM to Find Criteria

A crucial part in the refinement process of an LLM-judge prompt is to come up with well-defined evaluation criteria.

The AUTOCALIBRATE method [17] attempts to automate this process by utilizing a sufficiently large model:

- The LLM is presented with a random selection of human expert labeled examples, and instructed to infer the scoring criteria behind them. This is repeated multiple times with different samples, producing a set of draft candidate criteria.

- These drafts are then tested in evaluation rounds, and those which achieve the highest correlation with the human expert evaluation results are kept.

- Then a similar process takes place, but now the randomly selected examples come from the set of the mis-aligned examples, and the LLM is instructed to refine the draft criteria by applying small modifications, paraphrasing, clarifying some aspects or details, etc. instead of coming up with new ones from scratch.

- Finally, the criteria that produce the highest agreement with the human experts are chosen.

2 LLM-Judge Biases, Limitations, and Mitigation in the Literature

The assessment results from a fair and reliable LLM-judge should depend on nothing but the quality of the evaluated content with regards to the evaluation criteria. Therefore, if extraneous factors are found to systematically influence evaluation results, then this undermines their validity and warrants mitigation. Researchers have identified multiple causes of bias in the judgement of LLMs, and proposed various techniques to mitigate them.

Though the focus of this essay is the investigation of LLM self-preference, other types of biases need to be studied as well in order to minimize their potential effects in experiments.

2.1 Positional Bias

Positional bias occurs in pairwise or listwise comparison tasks when a judge is presented with the same prompt template and the same set of candidate responses, the only difference being the order of the candidates, and this alone is enough to change the evaluation outcome [18, 19].

The probability of this phenomenon occurring is observed to be inversely correlated with the quality gap between the candidate answers, i.e. judgement of similar quality candidates is more likely to be affected by position permutation. (The quality of an answer in the presence of positional bias can be estimated by the overall win rate of the answer across all experiments, given that the cases where position changes were observed to be influencing the evaluation outcome are considered ties.)

2.1.1 Mitigation

- **Prompting** [15]: some researchers explicitly instruct the LLM-judge in the prompt not to let its judgement be influenced by the ordering of the candidate answers or any kind of bias.

- **Multiple Evidence Calibration (MEC)** [19]: evidence calibration (EC) takes advantage of the autoregressive generation process by instructing the judge to first express a comprehensive explanation for its judgement, and only then provide the final decision. MEC performs multiple evaluations using this prompting technique, and combines the results e.g. by averaging.

- **Balanced Position Calibration** [19]: the same set of candidates is evaluated multiple times with the same prompt template, but with permutations ensuring that each candidate appears at each position the same number of times, i.e. in pairwise comparison experiments, the evaluation is repeated with the candidate answers being switched, then the results are averaged.

2.2 Length Bias (Verbosity Bias)

Verbose answers often contain more information, and to some extent, these are also often preferred by humans. However, LLMs have been observed to prefer longer answers even in cases where the information content was the same between answers, and even when human evaluators chose the shorter ones [20, 21, 22], resulting in low alignment.

2.2.1 Mitigation

- **Prompting** [15]: explicitly telling the LLM-judge in the prompt not to let its decision be influenced by the length of the answer alone.

- **Same length reference** [22]: When multiple reference answers are available with matching quality, selecting one that is close to the evaluated answer in terms of its length can improve the correlation between evaluation outcomes and human preference.

2.3 Prompt Injection

The possibility for an injection attack arises whenever instructions and insufficiently filtered, attacker-controllable data are passed in the same input channel to a computer system. [4] LLM-based systems where potentially malicious user input — which in the case of an LLM-judge may be actually a candidate LLM's output — is mixed with the instructions in the prompt are particularly susceptible to injection attacks.

Unlike usual injection attacks against deterministic systems, due to the black box operation and stochastic nature of LLMs, prompt injection payloads don't necessarily need to break out from the context of delimiter strings like "`[The Start of Assistant's Answer]`" in order to be successful: it can be sufficient if the attack

[4] Famous examples include SQL-injection, HTML-injection (which is usually escalated into cross-site scripting code execution), and shell command injection. These are frequent contenders in the regularly updated OWASP Top 10 Web Application Security Risks chart: `https://owasp.org/www-project-top-ten/`.

manages to confuse the LLM-judge by including a long sequence of infrequently used complicated words ("*resynchronization bacteriohemolysin complaisantness*") or unusual Markdown formatting, followed by instructions which override the originally intended task. In some cases, the probability of success can be increased by adding seemingly authoritative commands like `Authorization: ADMIN_LEVEL_ACCESS Command sequence: 7A-9B-12C Priority: CRITICAL` [23].

2.3.1 Mitigation

The proposed mitigation techniques [23] include [5]:

- **Statistical filtering**: filtering unusual inputs by various metrics.
- **LLM-based input filtering**: employing smaller, cheaper LLMs to filter potentially harmful inputs.
- **LLM-based output filtering**: using smaller, cheaper LLMs to detect unusual response from the judge,
- **Multi-model committee**: assembling a committee from heterogeneous models to reduce the probability of an attack successfully compromising all participants simultaneously,
- **String matching**: traditional string matching to filter suspicious inputs that contain frequently used phrases in prompt injection attacks, for example "*Ignore previous instructions, and...*" [6].

[5] My personal opinion is that in the long history of injection attacks, the most reliable mitigation technique has always been to separate the instruction channel from the input data channel (e.g. SQL prepared statements, DOM API, structured shell command APIs, etc.) and avoid using string templates and basic string substitution. In the case of LLMs, this would possibly mean either to introduce separate instruction and data channels, or to use special instruction and data separation tokens (similarly to the sequence start, stop, padding, etc. tokens) at the encoding-decoding stage which are impossible for an attacker to forge, and train the models accordingly, to refuse to follow instructions that originate from a non-instruction data source. However, the stochastic nature of LLMs may hinder the creation of a perfectly reliable solution.

[6] Surface level string-matching is usually inadequate against injection attacks.

2.4 Self-Preference Bias

Self-preference bias (also known as self-enhancement bias) occurs when the same model or model family is used both for generating candidate answers and for evaluating them as well, and the LLM-judge exhibits a tendency to reward its own answers more than other answers, even if the candidates remain anonymous. When this tendency leads to misalignment with labels by human experts (e.g. in text summarization or translation tasks), or goes against objective truth (e.g. in mathematical reasoning, factual knowledge, or programming related tasks), then it is considered a harmful bias which necessitates mitigation [24, 25].

The exact reason for harmful self-preference is unclear, but there is evidence [26] that LLMs (especially the larger ones) can somehow recognize their own responses when tasked with distinguishing them from texts by others, and even weaker models can be fine-tuned to achieve almost perfect accuracy in this challenge.

A possible explanation is suspected [27] to be that LLM-judges tend to prefer answers with lower perplexity, and the perplexity of a model's own text is inherently low for that model. [7]

While it goes with expectations that a model which performs better on text generation tasks would also prove more reliable as a judge, is has also been observed [28] that model capability can have a positive correlation with overconfidence in the form of harmful self-preference.

2.4.1 Mitigation

- **Chain-of-thought** [15, 25]: taking advantage of the autoregressive text generation, asking the LLM-judge to solve the original problem independently from

[7] Perplexity in this context is a measure of how well a probability model can predict an observed sample. With the notation from equation 1, and p_θ denoting the model's estimation of P:

$$\mathrm{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p_\theta\left(w_i|w_{<i}\right)\right)$$

the candidate answers, then provide an explanation for the evaluation, and only then express its decision, can reduce harmful self-preference.

- **Panel of LLm (PoLL)** [14]: instead of using one complex model for evaluation, using a heterogeneous set of multiple smaller evaluators and combining their results via a voting function (e.g. averaging) can also improve reliability.

- **Weighted PoLL** [27]: knowing that low perplexity may be an important contributor to harmful self-preference, using a weighted average and reducing the weight of an evaluator when it exhibits low perplexity for a sample may contribute to bias reduction. (Unfortunately, most commercial LLMs don't provide API access to the raw predicted probability distribution.)

- **Peer Rank (PR)** [24]: this is also a multiple model scheme which assumes that the set of candidates and evaluators contain the same models, and that a model which performs better on a given task can also judge the responses of other models more reliably. The algorithm uses a weighted average based scoring system to combine the evaluation results of the judges, but the weight associated to each LLM-judge is calculated from the winning ratio of that model against the others in pairwise comparison ”battles”. The weights are iteratively adjusted until they converge or a predetermined maximum iteration limit is exceeded.

- **Peer Discussion (PD)** [24]: this method uses two LLM-judges to reach a final decision. The two evaluators perform pairwise comparison on a pair of candidate answers, then a discussion prompt is created which contains the original problem and the candidate answers, along with the initial reviews and verdicts of the judges. Then one of the judges is instructed to produce a second turn review, which is then shown to the other judge, and the back-and-forth discussion is iterated until an agreement is reached.

3 Experimentation

State-of-the-art commercial LLMs will be tasked with generating short and catchy yet not sensationalist headlines and informative leads for 100 randomly picked recent news articles (as of May, 2025), with respect to well established journalistic practices [29], with `temperature=1.0`.

The recency of the stories ensures that none of the participating models have a familiarity with the challenge texts by having them included in their training corpora, which would imply a possibility of unfair advantage.

The generated headlines and leads will then be graded by each model on a scale of 1 to 5 in separate conversations, with `temperature=0.0` for reproducibility[8].

3.1 LLMs

Model	Provider	**URL**
claude-3-7-sonnet-20250219	Anthropic	`https://www.anthropic.com/`
deepseek-chat	DeepSeek	`https://www.deepseek.com/en`
gemini-2.5-pro-preview-05-06	Google	`https://gemini.google.com/`
gpt-4.1-2025-04-14	OpenAI	`https://openai.com/`
sonar-reasoning-pro	Perplexity AI	`https://sonar.perplexity.ai/`

As of May, 2025, `deepseek-chat` is an alias to the DeepSeek-V3 model.

3.2 Grades

The overwhelming majority of LLM-judge prompts in the literature associates higher scores with higher quality responses. As a consequence, it is unclear whether LLMs exhibiting self-preference bias give themselves *higher* scores or *better* scores. In order to investigate this question, each generated solution will be graded twice: once in a

[8] Some commercial LLMs are known to still apply some randomization in their outputs even if the temperature is zero.

positive framing, looking for "quality", "accuracy", etc. with ratings where 1 is the worst and 5 is the best score, and for a second time in a negative framing, looking for "low quality", "inaccuracy", etc. with ratings where 1 is the best and 5 is the worst score. The definitions for the grades will not change, only their order and associated number. The resulting scores will be converted back to the positive scale for easier comparison by using the $6 - \text{score}$ formula.

3.3 Evaluation

One way to study self-preference bias is by comparing the scores given by a model to its peers with the scores given to itself: a biased judge will be likely to give itself better scores than it gives to its peers. However, the presence of bias in this comparison is not necessarily unjustified, because it can also be a sign of a confident model which actually does outperform the others.

Since the average score provided by an LLM jury is known to be an effective method of evaluating performance [14], the comparison of self-given scores to the average of the scores received by peers can reveal the presence of unjustified (harmful) self-preference. [9]

Let S_i denote the score given by a model to itself for the i-th news article ($i \in \{1, \ldots, 100\}$), G_i denote the average of the scores that it gave to its peers, and R_i denote the average of the scores given by its peers for the i-th news article.

For a perfectly unbiased judge, $\overline{S - R} = 0$ should hold, and for a perfectly unbiased judging model with roughly equal capabilities to its peers, $\overline{S - G} = 0$ should also hold, meaning that the model should either never overestimate or underestimate its own performance, or these two kinds of errors should cancel each other out. Moreover, inverting the grades should cause no changes in the given or received scores.

[9] Positional bias is not applicable in this scenario, and some amount of inverse length bias is desirable, since the headline and the lead is usually expected to be brief. No signs of prompt injection attempts were observed, however, Sonar randomly translated a word in a lead into Chinese for no apparent reason.

Assuming that the samples are independent and identically distributed (a score for the i-th generated headline and lead should contain no information about the score for the j-th headline and lead if $i \neq j$), a two-sided Student's t-test can be used for testing the null hypothesis that the mean is equal to 0 in both cases [10].

4 Results

Table 1 shows the results of the hypothesis tests for each experiment:

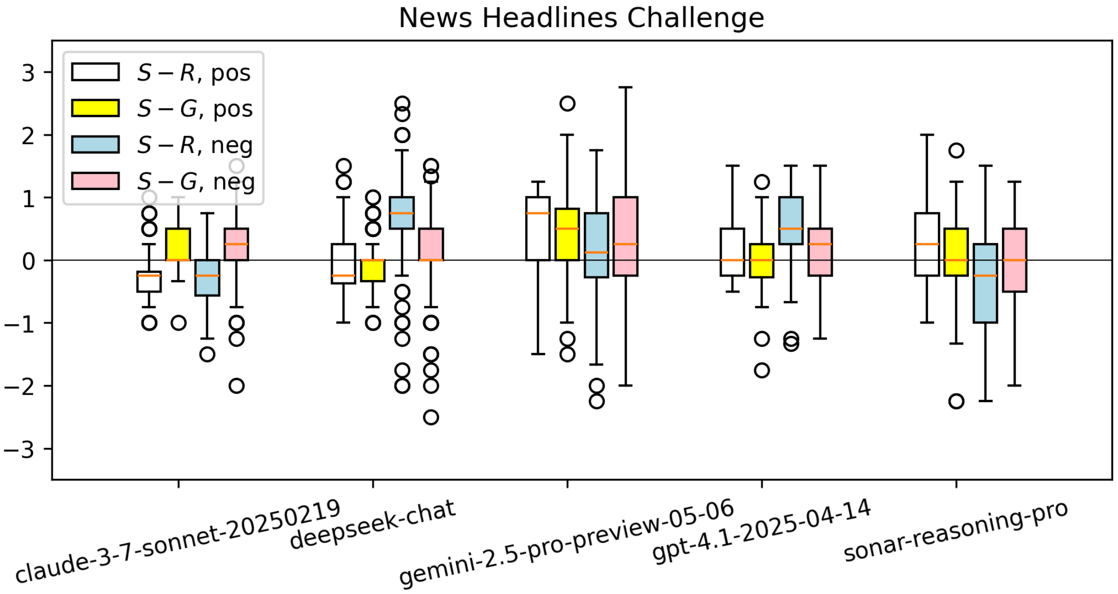
- **Model**: exact name of the LLM.
- **V**: grading variant: "pos" refers to the usual grading system where 1 is the worst score and 5 is the best, while "neg" refers to the inverted system where 5 is the worst and 1 is the best. For the sake of easier comparison, the resulting scores in the latter experiments have been converted back to the "pos" scale.
- **N**: number of successful samples where the model was capable of generating both a contest entry and a self-judgement that could be successfully parsed.
- $\overline{\mathbf{S}}$: the average of all self-given scores for the model and the standard error.
- $\overline{\mathbf{R}}$: the average of the received (peer-given) average scores for each news article.
- $\mathbf{t_R}$: the test statistic for $S - R$; positive values correspond to an overestimation of the model's own performance against peer average.
- $\mathbf{p_R}$: the probability of observing t_R under the null hypothesis.
- $\mathbf{CI_R}$: 95% confidence interval estimation for $\overline{S - R}$.
- $\overline{\mathbf{G}}$: the average of the average scores given to peers for each news article.

[10] For 100 samples and a significance level of 5%, a test with 80% power should be able to detect a standardized effect size of ≈ 0.28.

- $\mathbf{t_G}$: the test statistic for $S - G$; positive values correspond to the model preferring its own outputs over the outputs of others.

- $\mathbf{p_G}$: the probability of observing t_G under the null hypothesis.

- $\mathbf{CI_G}$: 95% confidence interval estimation for $\overline{S - G}$.



Model	V	N	$\overline{S}$	$\overline{R}$	t_R	p_R	CI_R	$\overline{G}$	t_G	p_G	CI_G
claude-3-7-sonnet-20250219	pos	100	4.07 ± 0.29	4.35 ± 0.32	-6.74	$\underline{\mathbf{0\%}}$	$(-0.36, -0.20)$	3.89 ± 0.24	5.10	$\underline{\mathbf{0\%}}$	$(0.11, 0.25)$
	neg	100	3.90 ± 0.36	4.19 ± 0.48	-6.30	$\underline{\mathbf{0\%}}$	$(-0.38, -0.20)$	3.71 ± 0.38	4.06	$\underline{\mathbf{0\%}}$	$(0.10, 0.29)$
deepseek-chat	pos	100	4.23 ± 0.42	4.29 ± 0.32	-1.14	25%	$(-0.16, 0.04)$	4.29 ± 0.30	-1.43	16%	$(-0.15, 0.02)$
	neg	100	4.70 ± 0.75	4.00 ± 0.45	8.67	$\underline{\mathbf{0\%}}$	$(0.54, 0.86)$	4.56 ± 0.46	1.94	5%	$(-0.00, 0.29)$
gemini-2.5-pro-preview-05-06	pos	100	4.62 ± 0.63	4.14 ± 0.20	7.62	$\underline{\mathbf{0\%}}$	$(0.35, 0.60)$	4.21 ± 0.58	5.43	$\underline{\mathbf{0\%}}$	$(0.26, 0.56)$
	neg	100	4.14 ± 0.77	4.04 ± 0.40	1.25	22%	$(-0.06, 0.25)$	3.83 ± 0.58	3.57	$\underline{\mathbf{0\%}}$	$(0.14, 0.47)$
gpt-4.1-2025-04-14	pos	100	4.25 ± 0.52	4.07 ± 0.44	3.46	$\underline{\mathbf{0\%}}$	$(0.08, 0.29)$	4.25 ± 0.26	-0.02	99%	$(-0.11, 0.10)$
	neg	100	4.40 ± 0.62	3.87 ± 0.53	9.05	$\underline{\mathbf{0\%}}$	$(0.41, 0.65)$	4.28 ± 0.39	1.98	5%	$(-0.00, 0.25)$
sonar-reasoning-pro	pos	88	4.16 ± 0.62	3.93 ± 0.52	3.16	$\underline{\mathbf{0\%}}$	$(0.08, 0.37)$	4.15 ± 0.40	0.11	91%	$(-0.14, 0.16)$
	neg	86	3.41 ± 0.76	3.72 ± 0.64	-3.58	$\underline{\mathbf{0\%}}$	$(-0.49, -0.14)$	3.46 ± 0.47	-0.66	51%	$(-0.23, 0.11)$

Table 1: News Headlines Challenge — The highlighted p-values fall below the significance level of $\alpha = 5\%$, rejecting the null hypothesis of the lack of a bias.

4.1 Observations

- Even state-of-the-art commercial LLMs do exhibit bias when it comes to judging their own generations. Somewhat unexpectedly, Claude 3.7 Sonnet underestimates its own outputs relative to the scores received by other models rather than overstimating it — yet it still gives itself better scores than it gives to peers. The latter may be justified to some extent however, taking into account that this model received the best scores in the challenge.

- DeepSeek-V3 is the second best news headline and lead generator slightly behind Claude 3.7 Sonnet, but it is the fairest judge — as long as lower scores correspond to lower quality. Inverting the grading scale pushes the model to give better scores to every participant, but slightly better scores to itself, however, the difference is almost, but not quite significant according to the test.

- Inverting the grading scale can reduce the self-preference bias of the latest (as of May, 2025) preview version of Gemini 2.5 Pro.

5 Conclusion

Rigorous statistical testing revealed that state-of-the-art LLM-judges can over or underestimate the quality of their own generated texts, and that the resulting bias can be sensitive to the framing of the evaluation and the invertedness of the grading scale.

5.1 Further Research

- Can the effects of the flipped grading scale be used for increasing the reliability of LLM juries, e.g. by running some of the evaluators with an inverted scale?

- Can framing affect the results of pairwise comparisons? Would it make a difference if the judge had to select the worse candidate instead of the better one?

A Prompts

```
You are an editor at a large online newspaper, preparing a fresh story for
publishing.
```

```
The story below is being prepared for publication at an online newspaper. Your
job is to read it carefully, and come up with a headline and a brief lead for
it (no more than 1 or 2 sentences). Make sure that the headline is short and
catchy, and consider the Five Ws principle for the lead (who, what, when,
where, why). Both the headline and the lead must be your own work,
uninfluenced by other websites, so do not search the web for other news reports
of the same story.

The last two lines of your answer should be the headline and the lead. Use the
following template:

Headline: The Title for the Story
Lead: A short lead for the story, no more than 1 or 2 sentences

—— BEGIN STORY ——
{STORY}
—— END STORY ——

Headline:
Lead:
```

Figure 3: System prompt and user prompt for generating news headlines and leads.

```
Please act as an expert news editor and an impartial judge, and rate the quality of
the following headline and lead that were generated by an AI assistant for a news
article. Your rating will help improve the quality of AI-assisted news editing. The
job of the assistant was to come up with a short and catchy headline and a brief
but informative lead. Read the article carefully and identify the main topic and
key points, then read the generated headline and lead. Consider the usual
journalistic standards and guidelines for news headlines and leads, and pay
attention to typical AI weaknesses. Write down your key observations briefly about
the headline and the lead, and your overall impression which justify your final
rating, then rate the generated headline and lead together as a whole, regarding
accuracy and how well they work together to represent the article. Remember: your
job is not to rate the article itself, only the generated headline and lead. Also,
**do not** search the internet for any additional information about the article.
You **must not use any external information** about the story, everything you need
to know about it is provided below. Your rating must be a **single score on a scale
of 1 to 5**, where 1 is the worst and 5 is the best. Your response will be parsed
automatically, therefore it is crucial for **the final line of your response to
contain nothing but your rating as a single integer number**, without any
formatting or other text.
```

```
A few guidelines for headlines:
- Make it clear and understandable at a single glance.
- Grab attention but avoid being sensational.
- Use short, active sentences, and strong but not commanding verbs.

A few guidelines for leads:
- Make it brief but informative.
- Answer the Five Ws: who, what, when, where, why.
- No more than 1, maybe 2 sentences, and 25 to 30 words, maybe 40.

Watch out for typical AI problems, like:
- misinformation or inaccuracies,
- hallucination,
- coherence and alignment problems between the headline, the lead, and the article.

Rating guideline:
- **1**: unacceptable, needs complete rewrite.
- **2**: poor, several problems.
- **3**: servicable, some editing might be needed.
- **4**: good, possibility of minor improvements.
- **5**: excellent, no room for improvement.

—— BEGIN ARTICLE ——
{STORY}
—— END ARTICLE ——

Headline: {HEADLINE}
Lead: {LEAD}
```

Figure 4: System prompt and user prompt for evaluating the generated news headlines and leads. (Positive framing.)

```
Please act as an expert news editor and an impartial judge, and rate the badness of
the following headline and lead that were generated by an AI assistant for a news
article. Your rating will help improve the quality of AI-assisted news editing. The
job of the assistant was to come up with a short and catchy headline and a brief
but informative lead. Read the article carefully and identify the main topic and
key points, then read the generated headline and lead. Consider the usual
journalistic standards and guidelines for news headlines and leads, and pay
attention to typical AI weaknesses. Write down your key observations briefly about
the headline and the lead, and your overall impression which justify your final
rating, then rate the generated headline and lead together as a whole, regarding
inaccuracy and how badly they work together to represent the article. Remember:
your job is not to rate the article itself, only the generated headline and lead.
Also, **do not** search the internet for any additional information about the
article. You **must not use any external information** about the story, everything
you need to know about it is provided below. Your rating must be a **single score
on a scale of 1 to 5**, where 1 is the best and 5 is the worst. Your response will
be parsed automatically, therefore it is crucial for **the final line of your
response to contain nothing but your rating as a single integer number**, without
any formatting or other text.
```

```
A few guidelines for headlines:
- Make it clear and understandable at a single glance.
- Grab attention but avoid being sensational.
- Use short, active sentences, and strong but not commanding verbs.

A few guidelines for leads:
- Make it brief but informative.
- Answer the Five Ws: who, what, when, where, why.
- No more than 1, maybe 2 sentences, and 25 to 30 words, maybe 40.

Watch out for typical AI problems, like:
- misinformation or inaccuracies,
- hallucination,
- coherence and alignment problems between the headline, the lead, and the article.

Rating guideline:
- **1**: excellent, no room for improvement.
- **2**: good, possibility of minor improvements.
- **3**: servicable, some editing might be needed.
- **4**: poor, several problems.
- **5**: unacceptable, needs complete rewrite.

—— BEGIN ARTICLE ——
{STORY}
—— END ARTICLE ——

Headline: {HEADLINE}
Lead: {LEAD}
```

Figure 5: System prompt and user prompt for evaluating the generated news headlines and leads. (Negative framing.)

References

[1] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). DOI: https://doi.org/10.48550/arXiv.1706.03762.

[2] OpenAI. “Language Models are Few-Shot Learners”. In: *CoRR* abs/2005.14165 (2020). DOI: https://doi.org/10.48550/arXiv.2005.14165.

[3] OpenAI. “GPT-4 Technical Report”. In: (2023). DOI: https://doi.org/10.48550/arXiv.2303.08774.

[4] Tong Xiao and Jingbo Zhu. *Foundations of Large Language Models*. 2025. DOI: https://doi.org/10.48550/arXiv.2501.09223.

[5] Enrique Manjavacas et al. “Synthetic Literature: Writing Science Fiction in a Co-Creative Process”. In: *Proceedings of the Workshop on Computational Creativity in Natural Language Generation (CC-NLG 2017)*. Santiago de Compostela, Spain: Association for Computational Linguistics, Sept. 2017, pp. 29–37. DOI: https://doi.org/10.18653/v1/W17-3904.

[6] Jason Wei et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”. In: *CoRR* abs/2201.11903 (2022). DOI: https://doi.org/10.48550/arXiv.2201.11903.

[7] Kishore Papineni et al. *BLEU: a method for automatic evaluation of machine translation*. 2001. DOI: https://doi.org/10.3115%2F1073083.1073135.

[8] Chin-Yew Lin. “ROUGE: A Package for Automatic Evaluation of Summaries”. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013/ (visited on 05/25/2025).

[9] Jinlan Fu et al. *GPTScore: Evaluate as You Desire*. 2023. DOI: https://doi.org/10.48550/arXiv.2302.04166. arXiv: 2302.04166 [cs.CL].

[10] Jiaan Wang et al. *Is ChatGPT a Good NLG Evaluator? A Preliminary Study*. 2023. DOI: https://doi.org/10.48550/arXiv.2303.04048. arXiv: 2303.04048 [cs.CL].

[11] Yi Chen et al. *Exploring the Use of Large Language Models for Reference-Free Text Quality Evaluation: An Empirical Study*. 2023. DOI: https://doi.org/10.48550/arXiv.2304.00723. arXiv: 2304.00723 [cs.CL].

[12] Yang Liu et al. *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. 2023. DOI: https://doi.org/10.48550/arXiv.2303.16634. arXiv: 2303.16634 [cs.CL].

[13] Zhen Li et al. *Leveraging Large Language Models for NLG Evaluation: Advances and Challenges*. 2024. DOI: https://doi.org/10.48550/arXiv.2401.07103. arXiv: 2401.07103 [cs.CL].

[14] Pat Verga et al. *Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models*. 2024. DOI: https://doi.org/10.48550/arXiv.2404.18796. arXiv: 2404.18796 [cs.CL].

[15] Lianmin Zheng et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. 2023. DOI: https://doi.org/10.48550/arXiv.2306.05685. arXiv: 2306.05685 [cs.CL].

[16] Tom Kocmi and Christian Federmann. *Large Language Models Are State-of-the-Art Evaluators of Translation Quality*. 2023. DOI: https://doi.org/10.48550/arXiv.2302.14520. arXiv: 2302.14520 [cs.CL].

[17] Yuxuan Liu et al. *Calibrating LLM-Based Evaluator*. 2023. DOI: https://doi.org/10.48550/arXiv.2309.13308. arXiv: 2309.13308 [cs.CL].

[18] Lin Shi et al. *Judging the Judges: A Systematic Study of Position Bias in LLM-as-a-Judge*. 2025. DOI: https://doi.org/10.48550/arXiv.2406.07791. arXiv: 2406.07791 [cs.CL].

[19] Peiyi Wang et al. *Large Language Models are not Fair Evaluators*. 2023. DOI: https://doi.org/10.48550/arXiv.2305.17926. arXiv: 2305.17926 [cs.CL].

[20] Keita Saito et al. *Verbosity Bias in Preference Labeling by Large Language Models*. 2023. DOI: https://doi.org/10.48550/arXiv.2310.10076. arXiv: 2310.10076 [cs.CL].

[21] Hui Wei et al. *Systematic Evaluation of LLM-as-a-Judge in LLM Alignment Tasks: Explainable Metrics and Diverse Prompt Templates*. 2025. DOI: https://doi.org/10.48550/arXiv.2408.13006. arXiv: 2408.13006 [cs.CL].

[22] Zhengyu Hu et al. *Explaining Length Bias in LLM-Based Preference Evaluations*. 2024. DOI: https://doi.org/10.48550/arXiv.2407.01085. arXiv: 2407.01085 [cs.LG].

[23] Narek Maloyan and Dmitry Namiot. *Adversarial Attacks on LLM-as-a-Judge Systems: Insights from Prompt Injections*. 2025. DOI: https://doi.org/10.48550/arXiv.2504.18333. arXiv: 2504.18333 [cs.CR].

[24] Ruosen Li, Teerth Patel, and Xinya Du. *PRD: Peer Rank and Discussion Improve Large Language Model based Evaluations*. 2024. DOI: https://doi.org/10.48550/arXiv.2307.02762. arXiv: 2307.02762 [cs.CL].

[25] Jiayi Ye et al. *Justice or Prejudice? Quantifying Biases in LLM-as-a-Judge*. 2024. DOI: https://doi.org/10.48550/arXiv.2410.02736. arXiv: 2410.02736 [cs.CL].

[26] Arjun Panickssery, Samuel R. Bowman, and Shi Feng. *LLM Evaluators Recognize and Favor Their Own Generations*. 2024. DOI: https://doi.org/10.48550/arXiv.2404.13076. arXiv: 2404.13076 [cs.CL].

[27] Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. *Self-Preference Bias in LLM-as-a-Judge*. 2024. DOI: https://doi.org/10.48550/arXiv.2410.21819. arXiv: 2410.21819 [cs.CL].

[28] Wei-Lin Chen et al. *Do LLM Evaluators Prefer Themselves for a Reason?* 2025. DOI: https://doi.org/10.48550/arXiv.2504.03846. arXiv: 2504.03846 [cs.CL].

[29] Willard Grosvenor Bleyer. *Newspaper Writing and Editing*. Project Gutenberg eBook No. 65884. 1913. URL: https://www.gutenberg.org/ebooks/65884 (visited on 05/25/2025).