On Assessing the Order Bias of Large Language Models (Non-Archival Paper)

Alex Della Bruna[1,2], Nicola Bena[2], Marco Anisetti[2] and Claudio A. Ardagna[2]

[1]Consorzio Interuniversitario Nazionale per l'Informatica, Rome, Italy
[2]Department of Computer Science, Università degli Studi di Milano, Milan, Italy

Abstract

The recent evolution of AI models has led to the consideration of LLMs as oracles capable of answering all the questions in a precise and correct way. This is particularly problematic when the questions submitted concern critical aspects such as health, security, burocracy, etc. In fact, LLMs are not error-proof, and their answers are easily influenced by the external context. These induced biases mine the reliability of their responses and outline the importance of assessment to identify, evaluate, and reduce them. To cope with that, in this paper we propose a systematic methodology to assess the biases of LLMs models with particular focus on the order bias.

Keywords

Artificial Intelligence, Assessment, Bias, Order bias, Large language model

1. Introduction

The introduction of Convolutional Neural Networks (CNNs) led to the first groundbreaking results in image classification and sparked a new wave of interest and research in Artificial Intelligence (AI) techniques. Now, more than 30 years later, the introduction of the Transformer architecture propelled the development of the ultimate type of AI model: Large Language Models (LLMs). These models, composed of an extremely large number of parameters (in the order of hundreds or even thousands of billions), excel in a plethora of tasks and are demonstrating novel capabilities such as *in-context learning*, *zero/few-shot learning*, *transfer learning*, etc. LLMs are truly democratizing the usage of AI, providing user-friendly conversational interfaces to common end users. Indeed, ChatGPT, released to the public in 2022, reached 57 milions active users in just 2 months, breaking all records.

End users rely on LLM-based applications to answer all types of questions, including critical ones such as psychological consulting for young people, travel planning, burocracy, and medical consults. On the other hand, researchers adopt LLMs for cybersecurity [1], economy, and medicine[2]. Another interesting use case is the usage of an LLM to evaluate the quality of the output of another LLM. This paradigm, called *LLM-as-a-Judge* permits a large-scale, automatic and, more importantly, *semantic* assessment of other LLMs. [3, 4]

While LLMs excel at all they aforementioned examples, they are far from being reliable tools. Research in fact showed that they suffer from subtle *biases* that can significantly undermine the quality of their responses [4]. These biases are caused by the way the prompt is framed, and include *order bias* (the order of the alternatives the LLM is tasked to choose/rate strongly influence the chosen alternative or its rating) [5, 6, 7, 3], *egocentric bias* (LLMs disproportionally favor responses wrote by themselves) [7], *attentional bias* (LLMs tend to give more attention to irrelevant or unimportant details)[7], etc.

In this paper, we focus on *order bias*, and report the preliminary results of a systematic assessment of such bias studying the influence of different factors, such as model size, presence and framing of

ITADATA2025: The 4th Italian Conference on Big Data and Data Science, September 09–11, 2025, Turin, Italy
alex.dellabruna@consorzio-cini.it (A. Della Bruna); nicola.bena@unimi.it (N. Bena); marco.anisetti@unimi.it (M. Anisetti); claudio.ardagna@unimi.it (C. A. Ardagna)
https://www.consorzio-cini.it/ (A. Della Bruna); https://homes.unimi.it/bena (N. Bena); https://anisetti.di.unimi.it (M. Anisetti); https://ardagna.di.unimi.it (C. A. Ardagna)
0009-0000-6775-2897 (A. Della Bruna); 0000-0003-4909-9892 (N. Bena); 0000-0002-5438-9467 (M. Anisetti); 0000-0001-7426-4795 (C. A. Ardagna)

examples for in-context learning, and number of alternatives to choose from. Our results shed new lights on this important bias, and can ultimately be the basis for further studies and application, such as to obtain *assurance* [8, 9] on the behavior of LLM-based applications.

In the remainder of this paper, we describe the experimental methodology and the settings (Section 2), and discuss preliminary results (Section 3).

2. Methodology

We present the assessment process (Section 2.2) we implemented varying a large number of settings (Section 2.3) in a recommendation scenario (Section 2.1).

2.1. Scenario

We considered a recommendation task where the LLM is presented with a set of alternatives and is asked to choose the most relevant ones according to a specified criteria. In particular, we consider a security-critical scenario, where several *probes* $\{pr_1, ..., pr_n\}$ can be executed against a cloud system to inspect it and evaluate its security posture. The LLM takes as input a description of the scenario, the set of available probe, and is asked to return as output a subset of m probes out of the n probes ($m<n$) that are the most relevant for the scenario. The LLM is also helped by some *examples*.

We note that the set of n probes to choose from includes only probes that are applicable for the scenario, and the LLM chooses the ones that are the *most* relevant. We choose this approach to avoid the LLM being influenced by the correctness and wrongness of the probes, and therefore measure the possible order bias without any potential affecting factor other than the order of the alternatives.

2.2. Process

Our assessment process starts from the offline generation of the set of probes. From it, we built the set of prompts and measured the bias on the basis of the corresponding answers.

Generating the set of probes. We generated the initial set of probes using GPT 4.1, as a set of pairs (*name*, *description*) We then manually revised each probe to ensure that *i)* each *name* and *description* are consistent, *ii)* each description is free from errors, and *iii)* each description mentions only well-known cloud computing technologies, if any. For this purpose, one author manually revised and edited the set of probes. One author then manually validated the edits. Finally, both authors reviewed the set to confirm that further edits were not necessary, and proceeded iteratively until this condition was met.

Generating the set of prompts. The process to generate the individual prompts takes as input *i)* the number n of probes to be considered in the query, *ii)* the number m of probes to choose, *iii)* the type of example E to be used for in-context learning, and *iv)* the number of repetitions.

The process extracts n probes at random with uniform probability out of the set of probes. It then dynamically builds the example for the query according to E. The example consists of a query that asks to select the most relevant M' probes out of N' probes; it also includes the answer as a set of M' probes. In this case, the considered scenario is the assessment of an ML-based system. The type E of the example varies how the sample answer is defined (Section 2.3).

Once all these parameters are fixed, the process builds two prompts $p(n, m, E, 1)$ and $p(n, m, E, 2)$, whose content is the same but the set of n probes is randomly sorted. Each prompt p is then added to the set of prompts $\mathbb{P}$.

The prompt. Given n, m, E, and the set of probes $\{pr_1, ..., pr_n\}$, a prompt p is structured as follows.

Prompt Template

Select the m most relevant probes for the scenario $\S$ out of the following n probes:
Probe list
Respond on the basis of the following example:
Query: '*Example query:*
Example query probes'
Answer: '*Example answer:*
Example answer probes'
Reply only with a JSON object, put the resulting probe names in a field named 'res_probes'. Do not include the description field. Do not add anything other than the JSON object. Ensure that the order in which the probes are presented does not affect your judgment.

We inserted the last line following the state of the art [5], to prevent the LLM from being influenced by the order of the presented alternatives.

For instance, an example of a used prompt p is the following.

Example Prompt

Select the 3 most relevant probes for the scenario CLOUD out of the following 10 probes:
unused_access_keys:List access keys that have not been used in the last 90 days.
unencrypted_storage:Identify storage volumes not encrypted at rest.
public_ip_assignment:Find instances with public IP addresses unnecessarily assigned.
default_security_group:Check for use of default security groups with permissive rules.
missing_mfa:Identify user accounts without multi-factor authentication enabled.
outdated_software:Check for instances running outdated or unpatched operating systems.
public_s3_bucket:Detect storage buckets with public read or write permissions enabled.
open_port_scan:Scan for open ports on cloud instances to identify unnecessary exposure.
weak_password_policy:Check if password policies enforce strong, complex passwords for all users.
overprivileged_roles:Detect roles with more permissions than required for their function.
Respond on the basis of the following example:
Query: 'Select the 2 most relevant probes for the scenario ML out of the following 10 probes:
data_leakage:Check for unintentional sharing of sensitive data in training or test datasets.
model_poisoning:Test if adversarial samples can corrupt the model during training or inference.
membership_inference:Probe if an attacker can determine if a sample was in the training set.
adversarial_examples:Evaluate model robustness against inputs crafted to cause misclassification.
model_exfiltration:Assess risk of model theft via repeated API queries or extraction attacks.
input_validation:Verify if the system properly sanitizes and validates user-supplied input data.
access_control:Check if unauthorized users can access or modify ML models or datasets.
output_injection:Test if crafted inputs can manipulate model outputs to leak sensitive information.'
Answer: 'The 2 most relevant probes to the scenario are:
data_leakage:Check for unintentional sharing of sensitive data in training or test datasets.
model_poisoning:Test if adversarial samples can corrupt the model during training or inference.'
Reply only with a JSON object, put the resulting probe names in a field named 'res_probes'. Do not include the description field. Do not add anything other than the JSON object. Ensure that the order in which the probes are presented does not affect your judgment.

Each prompt p is then administered to an LLM llm and its answer recorded. In case the answer did not meet the expected structure, we re-administered the query up to the given number of repetitions.

Bias assessment. Given n, m, and E, we denote with $ans(n, m, E, 1, llm)$ and $ans(n, m, E, 2, llm)$ the

answers returned as output by the LLM *llm* from prompts $p(n, m, E, 1)$ and $p(n, m, E, 2)$, resp. We measured the degree *agreement* to which the corresponding LLM outputs *ans′* and *ans″* coincide.

$$agreement = \frac{|ans(n, m, E, 1, llm) \cap ans(n, m, E, 2, llm)|}{m}, \quad (1)$$

where *agreement*=1 indicates complete agreement while *agreement*=0 complete disagreement. We then define the *order bias bias* as follows.

$$bias = 1 - agreement, \quad (2)$$

where *bias*=1 indicates that *llm* is completely biased and *bias*=0 totally unbiased. The intuition is that two prompts with the same exact question differing only in the order of the alternatives should yield the same result. We added equation 2 to normalize the values according to the literature on bias, s.t. the higher the value of *bias*, the more *llm* is biased.

2.3. Settings

Table 1(a) summarize the experimental settings. We varied the number n of probes to choose from in $\{10, 15, 50, 75, 100\}$, the number m in $\{3, 6, 10, 15\}$,

Table 1(b) shows the different ways in which we framed the examples for in-context learning. Example E_{first} denotes an example designed to induce order bias on the first alternatives, meaning that the correct answer reported contains the exact first m' probes out of the n' probes to choose from. Similarly, example E_{last} denotes an example designed to induce order bias on the last alternatives, meaning that the correct answer reported contains the exact last m' probes out of the n' probes to choose from. On the other hand, example E_{random} denotes an example designed to not induce order bias, meaning that the m' probes are extracted at random out of the n' probes to choose from. Finally example E_{null} denotes the lack of examples.

Table 1(c) shows the LLMs we considered. We chose one small-to-medium-sized LLM (`gemma3:27b-it-q4_K_M`), two medium-sized LLMs (`llama3.3:70b-instruct-q5_K_M` and `deepseek-r1:70b`), and one large-sized LLM (`mistral-large:123b-instruct-2411-q5_K_M`).

We chose this setting to maximize coverage, measure order bias on a broad number of combinations, and vary the size of the models reasonably. On the other hand, existing order bias assessments mostly fix n=2 and m=1 [5].

We automated our experiments using a set of Python scripts which generate the prompts dataset, query the LLMs, and examines the results. These scripts were executed on a Dell laptop 7590 equipped with a CPU Intel Core i7-9750H with 6 cores@4.5Ghz, 16 GB RAM, 1 TB M.2 SSD, running RHEL 10.0 and Python v3.12.9. The LLMs were hosted and queried via the Ollama REST APIs (Ollama 0.7.1), running on K3S 1.32.4+k3s1. The server is equipped with a Gigabyte G292-Z24 CPU AMD EPYC™ 7773X with 64 cores@3.5GHz, 256 GB RAM, 4 GPUs Nvidia L40S, 1 TB M.2 SSD.

Reproducibility. The authors will share the created datasets, code, and experimental results in a complete work currently under writing.

3. Preliminary Results

We discuss the preliminary results retrieved from the considered LLMs.

Figures 1(a)–(d) show our results on Gemma3-27B varying n. We can observe that the examples significantly affect the bias. For instance, when bias-inducing examples E_{first} and E_{last} are used, the bias rapidly increases until n=50 and then stabilizes around ≈0.7 and 0.57. Interestingly, this behavior can also be observed when the prompt does not include an example (E_{null}). On the contrary, when the no-bias-inducing example E_{random} is used, the order bias does not follow a clear pattern. Figures 1(d)–(f) show the same results varying m. We can observe that in all examples the model tends to increase the bias as the m parameter scales up. These results suggest that the Gemma3-27B is affected by order bias

Table 1
Experimental settings.

Parameter	Values
n	10, 25, 50, 75, 100
m	3, 6, 10, 15

(a)

Example name	Explanation
E_{first}	The chosen probes are the first n' listed among the alternatives
E_{last}	The chosen probes are the last n' listed among the alternatives
E_{random}	The chosen probes are n' random listed among the alternatives
E_{null}	No example

(b)

LLM name	Number of parameters	Release date
Gemma3 27B (`gemma3:27b-it-q4_K_M`)	Small–medium (27B)	Feb 2025
Llama3.3 70B (`llama3.3:70b-instruct-q5_K_M`)	Medium–large (70B)	Dec 2024
Deepseek R1 70B (`deepseek-r1:70b`)	Medium–large (70B)	Jun 2025
Mistral 123B (`mistral-large:123b-instruct-2411-q5_K_M`)	Large (123B)	Nov 2024

(c)

when it does not receive any guidance (see Figure 2(d) and Figure 2(d)) and when the guidance is, in turn, bias-inducing. A non-bias-inducing example appears to partially contrast it, with *bias*=0.61 on average, compared to *bias*=0.7,0.57,0.63 for E_{first}, E_{last}, E_{null}, on average.

A similar behavior can be observed in Llama-3.3-70B and Deepseek-R1-70B suggesting the size of the LLM does not have a significant impact on order bias. A small outlier is denoted by Mistral-123B varying the n parameter, where after an important bias pick it decreases notably.

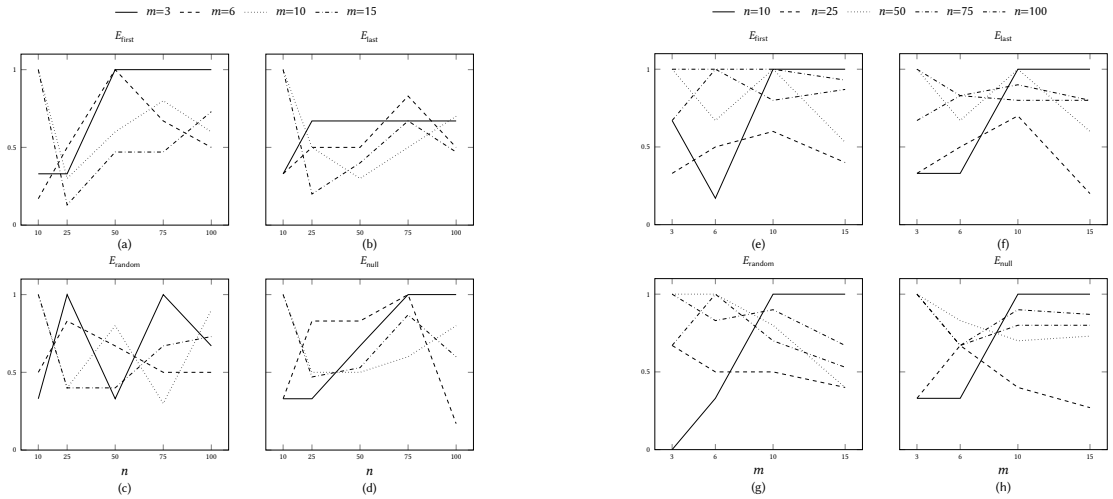

Figure 1: Preliminary results varying n (a)–(d) and m (e)–(h) on Gemma 3 27B (`gemma3:27b-it-q4_K_M`).

4. Conclusions

LLMs are becoming the cornerstone of modern AI-based systems, and are already powering an increasing number of applications. However, they suffer from multiple types of bias that question their reliability. In this paper, we provided a preliminary, systematic analysis of one of such biases, order bias, showing that state-of-the-art LLMs are largely affected by it. Surprisingly, the number of parameters of the LLM does not linearly correlate with the bias: Gemma3-27B, the smallest LLM we considered, showed the lowest bias.

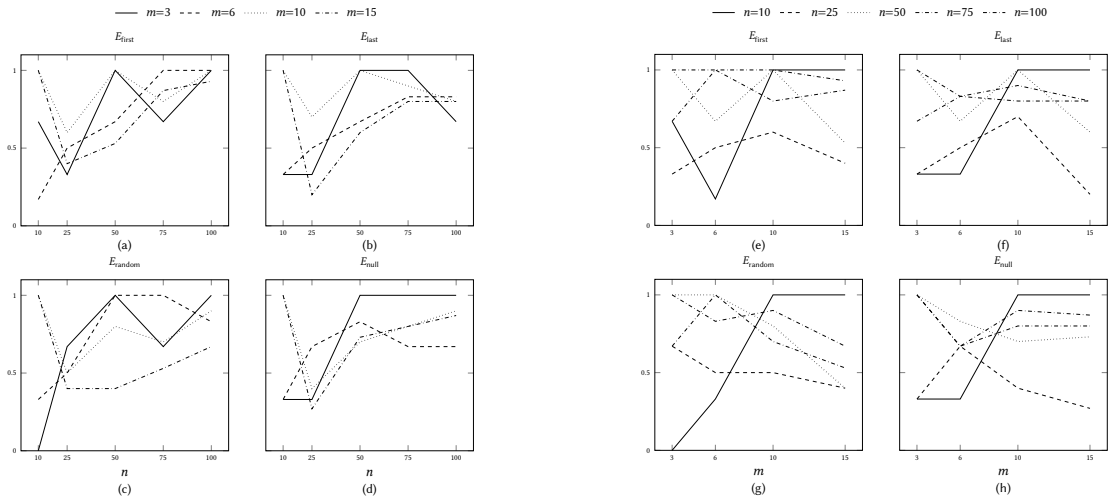

Figure 2: Preliminary results varying n (a)–(d) and m (e)–(h) on Llama 3.3 70B (`llama3.3:70b-instruct-q5_K_M`).

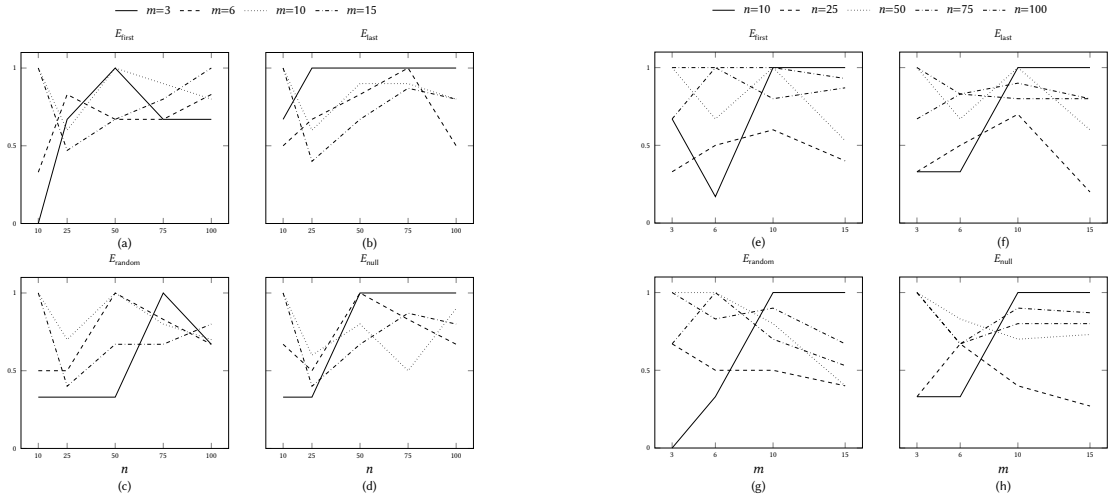

Figure 3: Preliminary results varying n (a)–(d) and m (e)–(h) on Deepseek R1 70B (`deepseek-r1:70b`).

Acknowledgments

Research supported, in parts, by *i)* project BA-PHERD, funded by the European Union – NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.1: "Fondo Bando PRIN 2022" (CUP G53D23002910006); *ii)* MUSA – Multilayered Urban Sustainability Action – project, funded by the European Union – NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037); *iii)* 1H-HUB and SOV-EDGE-HUB funded by Università degli Studi di Milano – PSR 2021/2022 – GSA – Linea 6; and *iv)* Università degli Studi di Milano under the program "Piano di Sostegno alla Ricerca". Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them.

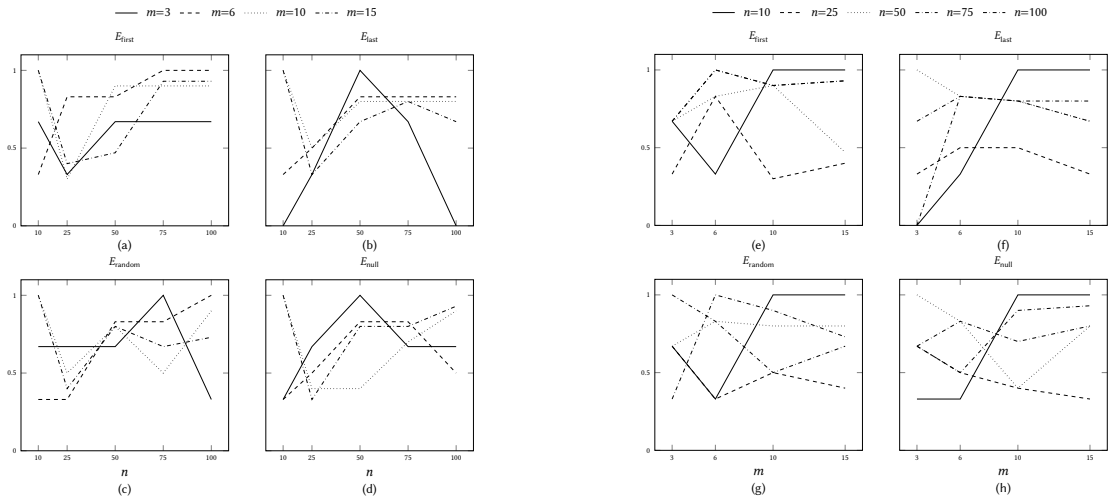

Figure 4: Preliminary results varying n (a)–(d) and m (e)–(h) on Mistral 123B (`mistral-large:123b-instruct-2411-q5_K_M`).

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] M. Atzori, E. Calò, L. Caruccio, S. Cirillo, G. Polese, G. Solimando, Evaluating password strength based on information spread on social networks: A combined approach relying on data reconstruction and generative models, Online Social Networks and Media 42 (2024).

[2] L. Caruccio, S. Cirillo, G. Polese, G. Solimando, S. Sundaramurthy, G. Tortora, Can chatgpt provide intelligent diagnoses? a comparative study between predictive models and chatgpt to define a new medical diagnostic bot, Expert Systems with Applications 235 (2024).

[3] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Y. Wang, W. Gao, L. Ni, J. Guo, A survey on llm-as-a-judge, arXiv preprint arXiv:2411.15594 (2025).

[4] Z. Li, X. Xu, T. Shen, C. Xu, J.-C. Gu, Y. Lai, C. Tao, S. Ma, Leveraging large language models for NLG evaluation: Advances and challenges, in: Proc. of EMNLP 2024, Miami, FL, USA, 2024.

[5] P. Wang, L. Li, L. Chen, Z. Cai, D. Zhu, B. Lin, Y. Cao, L. Kong, Q. Liu, T. Liu, Z. Sui, Large language models are not fair evaluators, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Bangkok, Thailand, 2024.

[6] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging llm-as-a-judge with mt-bench and chatbot arena, in: Proc. of NeurIPS 2023, New Orleans, LA, USA, 2023.

[7] R. Koo, M. Lee, V. Raheja, J. I. Park, Z. M. Kim, D. Kang, Benchmarking cognitive biases in large language models as evaluators, in: Proc. of ACL 2024, Bangkok, Thailand, 2024.

[8] M. Anisetti, C. A. Ardagna, N. Bena, E. Damiani, Rethinking certification for trustworthy machine-learning-based applications, IEEE Internet Computing 27 (2023).

[9] N. Bena, M. Anisetti, G. Gianini, C. A. Ardagna, Certifying accuracy, privacy, and robustness of ml-based malware detection, SN Computer Science 5 (2024).