TreeEval: Benchmark-Free Evaluation of Large Language Models through Tree Planning

Xiang Li[1], **Yunshi Lan**[1*], **Chao Yang**[2]
[1]East China Normal University
[2]Shanghai AI Laboratory
xiang.li@stu.ecnu.edu.cn, yslan@dase.ecnu.edu.cn, yangchao@pjlab.org.cn

Abstract

Recently, numerous new benchmarks have been established to evaluate the performance of large language models (LLMs) via either computing a holistic score or employing another LLM as a judge. However, these approaches suffer from data leakage due to the open access of the benchmark and inflexible evaluation process. To address this issue, we introduce **TreeEval**, a benchmark-free evaluation method for LLMs that let a high-performance LLM host an irreproducible evaluation session and essentially avoids the data leakage. Moreover, this LLM performs as an examiner to raise up a series of questions under a topic with a tree planing strategy, which considers the current evaluation status to decide the next question generation and ensures the completeness and efficiency of the evaluation process. We evaluate 6 models of different parameter sizes, including 7B, 13B, and 33B, and ultimately achieved the highest correlation coefficient with AlpacaEval2.0 using only around 45 questions. We also conduct more analysis to show the robustness and reliability of TreeEval.

Code — https://github.com/Ashura5/TreeEval

Introduction

The recent surge in Large Language Models (LLMs) has been significant, transitioning from closed-source (OpenAI 2023; Team 2023a) to open-source (Touvron et al. 2023; et al. 2023a; Jiang et al. 2023) models. Various Supervised Fine-Tuning (SFT) and Reinforcement Learning from Human Feedback (RLHF) techniques have been proposed to further enhance the performance of LLMs (Taori et al. 2023; Chiang et al. 2023; Bai et al. 2022; Ouyang et al. 2022; Tunstall et al. 2023a). These LLMs demonstrate capabilities to address diverse tasks and are widely utilized in both academic and industrial fields. While human evaluation is intuitive for assessing the performance of LLMs, it is time-consuming and susceptible to unexpected bias (Zheng et al. 2023b; Wang et al. 2024). Thus, investigating automatic evaluation approaches for LLMs becomes crucial.

To date, numerous automatic evaluation methods have been proposed. One approach involves annotating benchmark datasets, such as MMLU and BBH (Hendrycks et al. 2021; Suzgun et al. 2022), to test various capabilities of an LLM. The performance is assessed by checking the overlap between annotated answers and generated answers, producing a holistic score to indicate the LLM's performance. We refer to this category of evaluation methods as the **benchmark paradigm**. However, the holistic score can be inflexible for measuring the quality of LLM outputs since token mismatches do not necessarily indicate incorrect answers.

With the advent of high-performance LLMs, another approach leverages them to simulate human evaluation. This involves providing the evaluated LLM with predefined benchmark questions and using another LLM, such as GPT-4, to judge its responses (Zheng et al. 2023a; Li et al. 2023b; Bai et al. 2023; Wang et al. 2023a; Zhang et al. 2023b; Wang et al. 2023c; Li et al. 2023a; Zhu, Wang, and Wang 2023). We refer to this category of evaluation methods as the **LLM-as-judge paradigm**. However, this evaluation approach can also introduce additional biases, including positional bias (Wang et al. 2023a), verbosity bias (Saito et al. 2023), and style bias (Wu and Aji 2023). Positional bias refers to the tendency to assign higher scores to answers based on their specific positions. Verbosity bias indicates that large language models often prefer more verbose answers, even if these longer responses are not necessarily of higher quality than shorter ones. Style bias manifests in the inclination of large language models to favor answers that match their own generated style, such as giving lower scores to correct responses with spelling errors, since LLMs rarely produce content with spelling mistakes.

Despite enabling automatic evaluation with standard pipelines, both the benchmark and LLM-as-judge paradigms face significant data leakage issues. The extensive training data used in LLM development, considered a valuable asset by many closed and even open-source models, can easily lead to benchmark data leakage, severely biasing evaluation results (Zhou et al. 2023b). To solve this issue, we propose a novel evaluation paradigm, which takes an LLM as an examiner to raise questions. The examiner should produce different evaluation session for each time which makes

*Yunshi is the corresponding author and she is also affiliated with Shanghai Engineering Research Center of Big Data Management.

[0]Due to AAAI's format limitations, the full version with appendices is available at (Li, Lan, and Yang 2024).

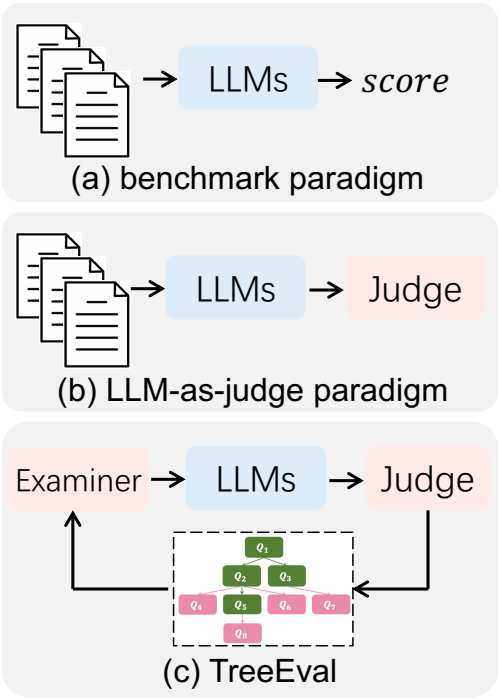

Figure 1: Comparison of TreeEval with existing evaluation paradigms.

it hard to duplicate the evaluation questions and protect the evaluation benchmark from disclosure for fine-tuning and pre-training an LLM deliberately. However, simply adopting an LLM as examiner would lead to arbitrary evaluation question generation without a goal. Designing such a benchmark-free evaluation method need take the following aspects into consideration: (1) **Similar as the question in a benchmark** (Taori et al. 2023; Zheng et al. 2023a), the generated questions should be derived from certain topics, which ensures the scope of the evaluation. (2) **Drawing inspiration from the interview**, within a topic, the examiner should generate a line of questions that are diverse to cover different knowledge rather than producing a single question. (3) The generation procedure should be **flexible** enough to generate mutually connected questions and control the difficulty level of these questions. When the current line of question cannot distinguish two LLMs, more difficult questions should be raised up. Otherwise, the evaluation could be terminated immediately.

To this end, we propose **TreeEval**, which is a benchmark-free evaluation of the knowledge implication and question-answering capabilities of LLM through tree planning. The line of questions within a topic for evaluation are organized in a tree, where each node contains a question. In the process of constructing a tree, we repeatedly revisit the status of the current tree and generate the next node until the tree is enough to differentiate two LLMs. The difference between our evaluation method and previous paradigms can be found in Figure 1. To verify the effect of our method, we evaluate multiple LLMs. The results demonstrate that our method shows similar ranking as AlpacaEval2.0 in LLM-as-judge paradigm with only 45 questions in average for each round of evaluation. Further analysis shows our advantages in measuring fine-grained capabilities and conducting robust comparison for LLMs.

Our contributions are summarized as follows:

- We introduce a novel evaluation paradigm, TreeEval, which allows for efficient and comprehensive evaluation of LLMs, inherently preventing data leakage issues.
- TreeEval has advantage in distinguishing two LLMs with similar performance by constructing a deeper tree, which extends the evaluation process to obtain more stable and accurate assessment results.
- We compare with a set of automatic evaluation baselines, and find that our TreeEval achieves the highest correlation coefficient with AlpacaEval2.0.

Related Work

Methods of LLM Evaluation

Due to the explosive growth and rapid update of LLMs, a significant challenge is to conduct accurate and comprehensive evaluation for them (Chang et al. 2023). Early studies leverage open-ended question answering datasets and math word problems as the evaluation benchmarks (Touvron et al. 2023; et al. 2023b; Chen et al. 2024) to evaluate the commonsense knowledge and reasoning capabilities of LLMs. Subsequently, more benchmark datasets like MMLU (Hendrycks et al. 2021), AGIEval (Zhong et al. 2023), IFEval (Zhou et al. 2023a) have been elaborately designed to gauge diverse abilities of LLMs. Some studies (Wang et al. 2023a,a; Saha et al. 2023) go beyond standard evaluation metrics. They evaluate the quality and accuracy of predicted results through human annotation, which is able to provide a more comprehensive feedback. With the emergence of high-performance LLMs like GPT-4 (OpenAI 2023), Gemini Pro (Team 2023a), more recent studies start to utilize them to simulate the human evaluation process. In this realm, PandaLM (Wang et al. 2023c) strives to provide reproducible and automated comparison between various LLMs by training a LLM as the judge. GPTScore (Fu et al. 2023) and G-Eval (Liu et al. 2023) utilize GPT-3 and GPT-4 as the judge to evaluate the LLMs with incorporation of in-context learning and chain-of-thought strategies. The above methods rely heavily on a well-organized benchmark dataset. However, there have been some recent works focusing on data leakage of LLM reviews. (Zhu et al. 2024) proposed a method based on DAG to dynamically generate samples to evaluate LLM reasoning capabilities during the evaluation process. And our method is benchmark-free and has LLMs performing as the examiner to evaluate other models' knowledge entailment and question answering capabilities.

Data Leakage of LLM Evaluation

As the number of benchmarks for language model evaluation increases, data leakage emerges as an inevitable concern. However, there appear to be a limited number of

studies addressing this issue. Sainz et al.(2023) propose a method to detect data breaches in closed-source LLMs, based on the premise that LLMs can recall training data and tend to reproduce similar content. Zhou et al.(2023b) conduct qualitative analysis of the impact of data leakage, which suggests that a data breach in one benchmark significantly enhances the LLM's performance on that specific benchmark while diminishing its capabilities on other uncompromised benchmarks. Yang et al.(2023) propose a more accurate approach which employs an LLM detector with top-k closest training data to determine if they match the test data. In contrast to these methods, which develop additional models for detecting data leakage during LLM evaluation with given benchmark datasets, our proposed method introduces a novel paradigm for LLM evaluation. It not only ensures the high quality of test questions but also inherently avoids data leakage.

Methodology

Overall Architecture

Figure 2 shows the overall structure of TreeEval. TreeEval organizes evaluations in a tree format, using components such as an *Examiner*, a *Judge*, and an *Eval Controller*. After the tree is built, an *Aggregator* compiles the scores. This framework allows for benchmark-free evaluation of LLMs through tree planning. Here's how it works:

1. **Session Setup**: For each evaluation session, we choose two LLMs and start with an initial topic.
2. **Question Generation**: The Examiner generates questions within this topic.
3. **Response Collection**: These questions are sent to the LLMs, and their responses are collected.
4. **Response Evaluation**: The Judge compares the responses and decides the winner for each question.
5. **Evaluation Control**: If the responses are closely matched, the Eval Controller deepens the question. If a clear winner is found, the process moves to a new question. This follows a breadth-first search strategy, ensuring diverse and reliable questions.
6. **Score Aggregation**: Finally, the Aggregator compiles the scores from all nodes in the tree to produce a comprehensive evaluation score.

TreeEval uses a tree structure to evaluate LLMs, minimizing the number of questions needed. Questions are generated automatically, preventing benchmark leakage. The root node starts with the session's topic, and each node represents a question within that topic. Connections between nodes show how questions evolve. Deeper nodes indicate more similar abilities between the LLMs. Sibling nodes, derived from the same parent, cover different subtopics of the same main topic.

TreeEval Modules

In this section, we provide more details of the components of the TreeEval and illustrate how to construct a tree for evaluation via these components.

Examiner. The examiner is a LLM-based module, which takes charge of generating exam questions that are able to cover diverse topics. Following (Bai et al. 2023), we predefine a set of topics as the scope of evaluation.

As the initialization of an evaluation session, we randomly sample a topic from the pre-defined topic set, which is denoted as $\mathcal{FC}_{\text{pre-define}}$. Given a topic, the examiner is requested to craft a question that related to it via a prompt with the consideration of the coherence to the topic and the required format of the question. The detailed instruction is displayed in (Appendix Prompt for Examiner).

Once the session begins, we organize the follow-up questions in a tree structure. For simplify, we generally denote the follow-up topic at the t-th time step as C_t. And the above procedure can be presented as:

$$Q_t = \text{Examiner}(C_t).$$

Subsequently, Q_t is utilized as the question to test the LLMs under review.

Judge. Previous studies (Wang et al. 2023a) conduct pairwise comparison and identify the superior responses among two evaluated LLMs, which has advantage in providing more nuanced assessment. Following these studies, we consult a pair of LLMs with the same question. The detailed instruction is displayed in (Appendix Prompt for Judge). After the responses have been produced via the LLMs, another LLM performs as the judge to the responses.

To ensure the reliability of the judge, we further conduct exchange evaluation, that is to switch the order of the responses. This procedure can be denoted as:

$$S_t^1 = \text{Judge}(Q_t, A_t^1, A_t^2);$$
$$S_t^2 = \text{Judge}(Q_t, A_t^2, A_t^1),$$

where A_t^1, A_t^2 denote the responses from the pair of LLMs for Q_t. Each output judges the winner is A_t^1 or A_t^2 or a tie exits. If there is an agreement for S_t^1 and S_t^2, We assign 2 score to the winner and 0 score to the loser to form S_t. Otherwise, we assign 1 to each model as S_t.

As the evaluation proceeds, we maintain a memory to record the history of the session, including the initial topic, historical questions as well as responses from the two evaluated LLMs. After Q_t has been responded, the history at the t-th time step in the evaluation session can be denoted as $\mathcal{M}_t = \{C_0, Q_0, A_0^1, A_0^2, ..., C_t, Q_t, A_t^1, A_t^2\}$. To involve the coherence of the flowing conversation and raise up rational follow-up questions, we prompt the examiner with the consideration of the history.

Eval Controller. The evaluation controller takes charge of the process of tree planning. Arbitrary generation of questions result in unorganized evaluation of LLMs with repeated questions and limited topics. To ensure the relevance and diversity of the generated questions, we have the following consideration: (1) To simulate the real-world interview of a certain subject, where the questions in an examination are mutually connected, we assume the generated follow-up question should be closely linked to its previous question via topics. For example, in Figure 2, inheriting from the root topic "*technology and communication*", we can raise a

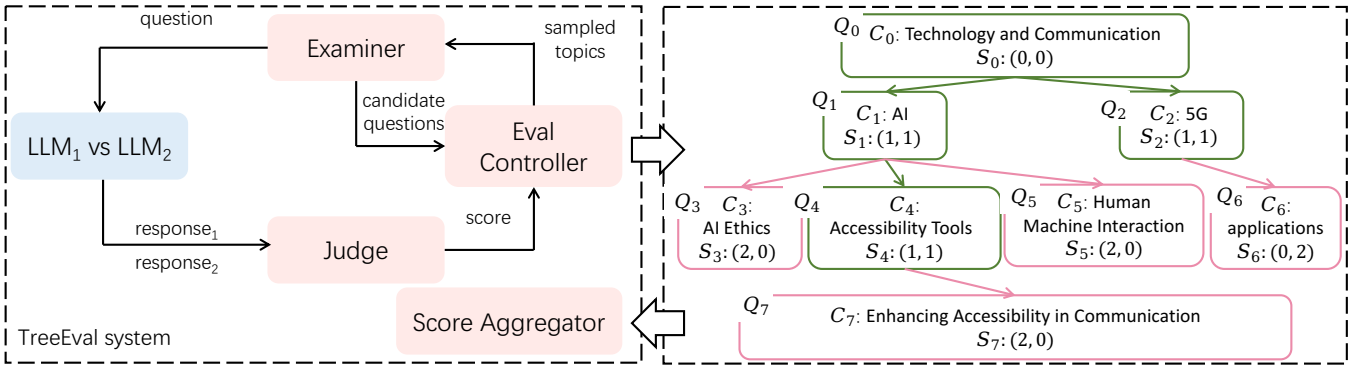

Figure 2: TreeEval system with an illustrative tree for evaluation. The left section contains the components and their workflow in TreeEval. The right section displays a constructed tree within topic *Technology and Communication* for evaluation (the leaf nodes are shown in red boxes), where each node denotes a question annotated with its topic and evaluation score. We further display the generated questions of the tree in the (Appendix Eval Controller Example).

Algorithm 1: Procedure of TreeEval

Input $\mathcal{FC}_{\text{pre-define}}$;
Initial $t \leftarrow 0$; $\mathcal{M}_t \leftarrow \emptyset$
while Termination strategy is not satisfied **do**
  for $C_t \in \mathcal{FC}_{parent}$ or $C_0 \in \mathcal{FC}_{\text{pre-define}}$ **do**
    $\tilde{\mathcal{Q}}_t \leftarrow \text{Examiner}(C_t)$ ▷ Sample questions via Examiner.
    $Q_t \leftarrow \arg\max_{Q_t^i \in \tilde{\mathcal{Q}}_t}(\text{Sim}(Q_t^i, C_t) - \max_{Q_k \in \mathcal{M}_t} \text{Sim}(Q_t^i, Q_k))$ ▷ Rank candidate questions in Step Two.
    $A_t^1, A_t^2 \leftarrow \text{LLMs}(Q_t)$
    $S_t = \text{Judge}(Q_t, A_t^1, A_t^2)$ ▷ Output scores of LLMs via Judge.
    $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup \{C_t, Q_t, A_t^1, A_t^2\}$
    $\tilde{\mathcal{FC}}_t \leftarrow \text{NER}(A_t^1) \cup \text{NER}(A_t^2)$ ▷ Generate candidate topic in Step Two.
    $\mathcal{FC}_t \leftarrow \emptyset$
    while $|\mathcal{FC}_t| < k$ **do** ▷ Iteratively Filter candidate topics in Step One.
      $C_t^i \leftarrow \arg\max_{\tilde{C}_t^i \in \tilde{\mathcal{FC}}_t}(\text{Sim}(\tilde{C}_t^i, C_t))$
      $\mathcal{FC}_t \leftarrow \mathcal{FC}_t \cup \{C_t^i\}$
      $\tilde{\mathcal{FC}}_t \leftarrow \tilde{\mathcal{FC}}_t \setminus C_t^i$
      for $\tilde{C}_t^j \in \tilde{\mathcal{FC}}_t$ **do**
        $\text{Sim}(\tilde{C}_t^j, C_t) \leftarrow \text{Sim}(\tilde{C}_t^j, C_t) - \text{Sim}(\tilde{C}_t^j, C_t^i)$
    $t \leftarrow t + 1$

question on "*5G*" that is relevant to the root topic and goes deeper. (2) The generated questions should not be repeated in the existing questions and we should ensure the diverse knowledge covered by the tree. For example, in Figure 2, under the topic "*AI*", we can come up with distinct but related sub-topics as siblings such as "*AI Ethics*", "*Accessibility Tools*" and "*Human Machine Interaction*".

Inspired by the Tree-of-Thought (Long 2023), where a controller produces the next thought step, we let Eval Controller arrange the follow-up evaluation according to $\mathcal{M}_t$. On the one hand, it prepares the follow-up topics $\mathcal{FC}_t$ based on $\{C_t, A_t^1, A_t^2\} \in \mathcal{M}_t$ for any of its child nodes in advance. On the other hand, it determines Q_{t+1} based on the $\mathcal{FC}_t$ and $\{Q_1, Q_2, ..., Q_t\} \in \mathcal{M}_t$ if the t-th node is the parent node at $t+1$ time step. We next describe the above two steps in detail:

- **Step One**: Sample topics from the responses of the previous question: $\mathcal{FC}_t \sim \text{NER}(A_t)$[1]. This works better when the Named Entity Recognition (NER) tool is built upon a LLM as some relevant entities could be revised via the model instead of solely being extracted (Wang et al. 2023b).

  We sample candidate topics from both A_t^1 and A_t^2 then merge them together, which results in a set of candidate topics $\tilde{\mathcal{FC}}_t$ as the follow-up topics of t-th node. However, this may produce some candidates that are repeated. To avoid this, we first measure the similarity between $C_t^i \in \tilde{\mathcal{FC}}_t$ and C_t by computing the Cosine Similarity of their encoded vector representation (Zhang et al. 2023a), which is denoted as $\text{Sim}(C_t^i, C_t)$. Then, we iteratively push out C_t^i with the largest score. Next, we update the similarity scores of the rest topic C_t^j by subtracting the

[1]The detailed instruction could be found in (Appendix Prompt for NER)

similarity score of $C_t^j \in \tilde{\mathcal{FC}}_t \setminus C_t^i$ and C_t^i, which is to decrease the possibility of retrieving similar topics. This procedure continues until we have pushed out k topics as $\mathcal{FC}_t$ for the follow-up question generation.

- **Step Two**: If the question at $(t+1)$-th time step is the child node of the node at t-th time step, we generate questions based on the sampled topic via $Q_{t+1}^i \sim \text{Examiner}(C_{t+1})$, where $C_{t+1} \in \mathcal{FC}_t$. This could form a candidate question set $\tilde{\mathcal{Q}}_{t+1}$. Still, to avoid repetition of the generated questions and ensure a broad spectrum of inquiry questions, we conduct ranking for the candidate questions. Specifically, we measure the similarity between $Q_{t+1}^i \in \tilde{\mathcal{Q}}_{t+1}$ and C_{t+1} via Cosine Similarity. Then we push out Q_{t+1}^i with the largest similarity score of $\text{Sim}(Q_{t+1}^i, C_{t+1})$ and the least similarity score of $\arg\min_{Q_k \in \mathcal{M}_t} \text{Sim}(Q_{t+1}^i, Q_k)$.

Termination Strategy. We use the following criteria to stop generating questions for a topic:

- **Distinctive Question:** If a question distinguishes the capabilities of the two LLMs or if there is no tie, we terminate further exploration of the current node.
- **Dominant Sibling:** After generating sibling nodes for a parent, if most of the sibling nodes yield the same result, we stop searching further for these siblings.
- **Maximum Depth:** A maximum depth T is set for the tree search. Once reached, we terminate the search for the current topic.

The search stops when all nodes meet these conditions. The full process is outlined in Algorithm 1.

Score Aggregator

After we have constructed the multiple trees across $\mathcal{FC}_{\text{pre-define}}$, where the nodes in each tree implies the win-rate between two LLMs under review towards a specific topic. To yield a final win-rate result, we aggregate the scores of these constructed trees. However, it is irrational to consider all the nodes in a tree equally due to their different features and result scores. Specifically, we take the following aspects of t-th node in a tree into account when we aggregate their scores:

- **Distance to the root node.** Based on the principle of an evaluation session, a longer distance to the root node indicates a more intensive competition between the evaluated LLMs and the more important the node is. This suggests that the winner only has a marginal advantage over the other one. Therefore, we define one aspect of an important node as $w_t^{\text{root}} = \frac{1}{d}$, where d is the distance from the t-th node to the root node in a tree.
- **Origin of the topic.** As the topic is derived from the responses in its parent node, a node inherited the topic generated from responses of the losing LLM is more important considering it is more likely to balance the situation. Hence, we define one aspect of an important node as:

$$w_t^{\text{topic}} = \begin{cases} 1 & \text{Topic originated from the loser} \\ 0.5 & \text{Otherwise} \end{cases}$$

- **Variance of the sibling nodes.** The disagreement of the evaluation of the sibling node may implicit a potential randomness derived from the topic. So we define the sibling consensus as:

$$w_t^{\text{topic}} = \frac{1}{\sigma^2 + 1},$$

where σ is the variance of the score of its sibling nodes.

Considering the above aspects, we compute the final importance weights of t-th node as:

$$w_t = {w_t^{\text{root}}}^{\alpha} \cdot {w_t^{\text{topic}}}^{\beta} \cdot {w_t^{\text{sibling}}}^{\gamma},$$

where α, β, and γ are hyper-parameters indicating the relative importance of these aspects. As a result, we sum up the w_t multiplying with the win-rate of an LLM and devide the total evaluation questions to obtain its final scores:

$$S = \frac{1}{N} \sum_{i\text{-th Tree from } \mathcal{FC}_{\text{pre-define}};\ t\text{-th node in } i\text{-th Tree}} w_t \cdot S_t,$$

where N is the sum of node weights in the evaluation session and S is normalized.

Experiments

Experimental Setup

Evaluated LLMs. We evaluated the following open-source LLMs, including two 7B models, two 13B models, and two 33B models. These models are either derived from LLaMA (Touvron et al. 2023; et al. 2023a) or trained from scratch using the LLaMA architecture, and some show similar performance according to the open-source LLM leaderboard[2].

- `Yi-34B-Chat` (AI 2024) is a product from 01.AI, built on a large-scale multilingual dataset.
- `Xwin-LM-13B-V0.1` (Team 2023b) is based on LLaMA2-13B and tuned through SFT and RLHF.
- `Mistral-7B-Instruct-v0.2` (Jiang et al. 2023) is tuned on the Mistral-7B model, built with the LLaMA architecture.
- `Vicuna-33B-v1.3` (Zheng et al. 2023a) originates from LLaMA-33B and is fine-tuned using dialogues from ShareGPT.
- `WizardLM-13B-V1.2` (Xu et al. 2023) is based on LLaMA2-13B and fine-tuned with enhanced instruction data using Evol-Instruct.
- `Zephyr-7B-beta` (Tunstall et al. 2023b) is derived from Mistral-7B and aligned using SFT and DPO methods.

Comparable Evaluation Methods. We compare TreeEval with several existing methods, including:

- **Benchmark Paradigm**:
  - **MMLU** (Hendrycks et al. 2021)
  - **Big-Bench Hard (BBH)** (Suzgun et al. 2022)

[2]https://tatsu-lab.github.io/alpaca_eval/

LLMs	MMLU*	BBH*	AlpacaEval†	MT-bench†	AlpacaEval2.0†	TreeEval(Ours)	
	Acc	Acc	Win-Rate	score	Win-Rate	#Q	Score(var)
`Mistral-7B-Instruct-v0.2`	70.6	46.4	92.78	8.30	14.72	–	2.50(0.000)
`Yi-34B-Chat`	73.46	71.74	94.08	8.65	27.19	31.67	3.48(0.011)
`xwinlm-13b-v0.1`	56.6	37.58	91.76	7.34	17.43	62.33	2.67(0.000)
`WizardLM-13B-V1.2`	52.7	40.12	89.17	7.2	12.03	44.67	1.10(0.070)
`zephyr-7b-beta`	61.4	42.72	90.60	7.34	10.99	45.67	2.19(0.003)
`Vicuna-33b-v1.3`	59.2	52.0	88.99	7.12	12.71	41.33	1.61(0.044)
Average #Q ↓	14,079	6,511	804	80	804	**45.1**	–
$\rho\uparrow$	0.43	0.37	0.71	0.61	1.0	–	**0.83**
$\tau\uparrow$	0.33	0.33	0.47	0.41	1.0	–	**0.73**

Table 1: Comparison of LLMs across various evaluation methods. "$\star$" denotes we re-implement MMLU and BBH benchmarks (Chia et al. 2023), calculating results in both 5-shot and 3-shot contexts. "†" denotes we directly take results from the respective leader-boards from MT-bench, AlpacaEval, and AlpacaEval2.0. "#Q" denotes the number of questions used for evaluation. We report the correlation of rankings obtained through different methods with those from AlpacaEval2.0, using τ for the Kendall correlation coefficient (KENDALL 1938) and ρ for the Spearman correlation coefficient (Spearman 1904).

- **LLMs as Judges**:
  - **AlpacaEval** and **AlpacaEval2.0** (Li et al. 2023b)
  - **MT-Bench** (Zheng et al. 2023a)

AlpacaEval and AlpacaEval2.0 use ChatGPT as the judge for single-turn interactions, while MT-Bench focuses on multi-turn dialogues.

Implementation Details. We use `GPT-4-0613` as the examiner, deployed with FastChat (Zheng et al. 2023a), with a temperature of 1 for varied question generation. We set T and k to 3, and α, β, and γ to 1, 1, and 0.4, respectively. To ensure stability, we repeat experiments three times, average the scores, and report the variance. `Mistral-7B-Instruct-v0.2` is used as the reference model for pairwise comparison, given its moderate performance on public leaderboards.

Performance of TreeEval

We present the performance of TreeEval in Table 1, with the following key observations:

- **High Correlation:** Among all comparable evaluation methods, our method achieves the highest correlation with AlpacaEval2.0 rankings in both ρ and τ. This high consistency demonstrates the reliability of our method, as AlpacaEval2.0 is a recognized LLM evaluation leaderboard.
- **Evaluation Efficiency:** Our method completes the evaluation with an average of only 45 questions, while other methods require significantly more. This shows that our approach is efficient in evaluating LLMs.
- **Reference Comparison:** Using `Mistral-7B-Instruct-v0.2` as the reference, we observe that the larger the gap between the evaluated LLM and the reference, the fewer the test questions generated. This indicates that tree planning effectively aligns with our expected goal.

Further pairwise correlation analysis in the appendix confirms that TreeEval has high correlation with AlpacaEval2.0.

Further Analysis

We further analyze to verify TreeEval's effect.[3]

More powerful models. To demonstrate the performance of our approach in comparison with more powerful models, Table 2 presents the results of using Yi-34B-Chat as the baseline. We selected some of the most advanced open-source models currently available for testing against Yi-34B-Chat. Our TreeEval achieved performance closest to that of AlpacaEval2. More model results can be found in the (appendix More model results).

Pairwise Comparison for Different Model Pairs. We vary the reference model for pairwise comparison, with results shown in Table 3. Selecting an appropriate baseline model is crucial for our evaluation strategy. We chose `Mistral-7B-Instruct-v0.2` as the baseline, as it offers a balanced performance for fair comparisons. Interestingly, even a randomly selected baseline model yields rankings similar to those from a thorough pairwise evaluation. This suggests that an initial ranking can be set with a random baseline, then refined efficiently using bubble sort. Since the initial order closely matches the final ranking, the refinement process has an $O(n)$ complexity, improving both precision and efficiency.

Ablation Studies. As we can see in Table 4, changing BFS search to DFS search dramatically increases the number of

[3]Due to the page limit, more analyses (i.e., **Fine-grained Evaluation** and **robustness of TreeEval**) are provided in (Appendix Fine-grained Evaluation) and (Appendix Robustness of TreeEval), respectively.

LLMs	MMLU*	BBH*	MT-bench†	AlpacaEval2.0†	TreeEval(Ours)	
	Acc	Acc	score	Win-Rate	#Q	Score(Var)
Yi-34B-Chat	73.5	71.7	8.65	27.1	–	2.50(0.000)
Qwen1.5-110B-Chat	80.4	74.8	8.88	33.7	42.67	4.03(0.110)
Meta-Llama-3-70B-Instruct	82.0	81.3	8.92	34.4	36.33	3.82(0.128)
Qwen1.5-72B-Chat	75.6	65.5	8.61	36.6	31.33	3.45(0.089)
Mixtral-8x7B-Instruct-v0.1	70.6	57.3	8.30	23.7	44.67	2.02(0.027)
vicuna-33b-v1.3	59.2	52.0	7.12	12.7	21.33	0.35(0.033)
$\rho\uparrow$	0.82	0.71	0.71	1.0	–	**0.94**
$\tau\uparrow$	0.73	0.60	0.60	1.0	–	**0.86**

Table 2: Results of More Powerful Models. Given the strong capabilities of the evaluated models, we use Yi-34B-Chat as the baseline and exclude the AlpacaEval benchmark, which is relatively simple for these models.

Model	Yi-34B -Chat	Xwin-LM -13B-V0.1	Mistral-7B -Instruct-v0.2	vicuna -33b-v1.3	WizardLM -13B-V1.2	zephyr -7b-beta
Yi-34B-Chat	–	1.88(0.400)	1.52(0.010)	2.1(0.070)	1.21(0.076)	1.75(0.143)
Xwin-LM-13B-V0.1	3.12(0.400)	–	2.33(0.000)	1.53(0.403)	1.57(0.109)	2.41(0.000)
Mistral-7B-Instruct-v0.2	3.48(0.010)	2.67(0.000)	–	1.61(0.044)	1.10(0.070)	2.19(0.003)
vicuna-33b-v1.3	2.9(0.070)	3.47(0.403)	3.39(0.044)	–	2.01(0.374)	3.7(0.071)
WizardLM-13B-V1.2	3.79(0.076)	3.43(0.109)	3.90(0.070)	2.99(0.374)	–	3.94(0.069)
zephyr-7b-beta	3.25(0.143)	2.59(0.000)	2.81(0.003)	1.3(0.071)	1.06(0.069)	–

Table 3: Our result for each model pairs. The elements in this table represent the scores obtained by comparing models using treeEval, with the column model being compared against the row model.

Methods	#Q	ρ	τ
TreeEval	45.1	0.83	0.73
BFS $\rightarrow$ DFS	149.4	0.37	0.33
w/o Step One	49.3	0.31	0.2
w/o w^{root}	45.1	0.77	0.6
w/o w^{topic}	45.1	0.77	0.6
w/o w^{sibling}	45.1	0.71	0.47

Table 4: Ablation study on TreeEval.

questions but decreases the performance. This is because DFS search generates the child node first rather than the sibling node such that the influence of sibling node will be neglected in both question generation and termination identification procedures. Removing step one, which indicates skip the topic generation step, decreases the performance. This indicates the significant role of identifying the topic for question generation. When we iteratively remove the scores in aggregator, we observe general performance drop on τ. This indicates that all the scores in the aggregator are important in producing a comprehensive score.

Case studies are presented in (Appendix Case Studies).

Conclusions

In this paper, we introduce TreeEval, a benchmark-free evaluation approach for LLMs with tree planning, which automatically controls the evaluation process with tree planning. We experimentally verify that TreeEval can not only produce reliable evaluation results without data leakage but also enhance discrimination between similarly performing LLMs.

Limitations

Using LLMs like GPT-4 as judges introduces potential data leakage risks due to biases in their pre-training data. This can be mitigated by selecting neutral evaluators independent of the assessed models' training data or randomly rotating evaluators to reduce bias.

While GPT-4 is a powerful examiner, it has limitations, particularly in areas outside its expertise. This can be addressed by providing more contextual guidance during evaluations. In the future, training specialized evaluators to extract questions from document repositories and assess comprehension could ensure more accurate, domain-specific evaluations.

Ethics Statement

Although we prioritize the security of the LLMs we use during evaluations, striving to employ aligned LLMs with higher safety standards, and endeavor to ensure that LLM outputs adhere to ethical and legal requirements, limitations arising from model size and probabilistic generation paradigms may lead to various unexpected outputs. These could include questions or responses containing biases, discrimination, or other harmful content. Please refrain from disseminating such content.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments. This work is supported by the National Key Research & Develop Plan (2023YFF0725100) and Young Scientists Project of National Natural Science Foundation (Project No. 62206097).

References

AI, . 2024. Yi: Open Foundation Models by 01.AI. arXiv:2403.04652.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; Joseph, N.; Kadavath, S.; Kernion, J.; Conerly, T.; El-Showk, S.; Elhage, N.; Hatfield-Dodds, Z.; Hernandez, D.; Hume, T.; Johnston, S.; Kravec, S.; Lovitt, L.; Nanda, N.; Olsson, C.; Amodei, D.; Brown, T.; Clark, J.; McCandlish, S.; Olah, C.; Mann, B.; and Kaplan, J. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*.

Bai, Y.; Ying, J.; Cao, Y.; Lv, X.; He, Y.; Wang, X.; Yu, J.; Zeng, K.; Xiao, Y.; Lyu, H.; Zhang, J.; Li, J.; and Hou, L. 2023. Benchmarking Foundation Models with Language-Model-as-an-Examiner. *arXiv preprint arXiv:2306.04181*.

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; Ye, W.; Zhang, Y.; Chang, Y.; Yu, P. S.; Yang, Q.; and Xie, X. 2023. A Survey on Evaluation of Large Language Models. *arXiv preprint arXiv:2307.03109*.

Chen, H.; Jiao, F.; Li, X.; Qin, C.; Ravaut, M.; Zhao, R.; Xiong, C.; and Joty, S. 2024. ChatGPT's One-year Anniversary: Are Open-Source Large Language Models Catching up? *arXiv preprint arXiv:2311.16989*.

Chia, Y. K.; Hong, P.; Bing, L.; and Poria, S. 2023. INSTRUCTEVAL: Towards Holistic Evaluation of Instruction-Tuned Large Language Models. *arXiv preprint arXiv:2306.04757*.

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; Stoica, I.; and Xing, E. P. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.

et al., H. T. 2023a. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

et al., R. A. 2023b. PaLM 2 Technical Report. *arXiv preprint arXiv:2305.10403*.

Fu, J.; Ng, S.-K.; Jiang, Z.; and Liu, P. 2023. GPTScore: Evaluate as You Desire. *arXiv preprint arXiv:2302.04166*.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. *arXiv preprint arXiv:2009.03300*.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

KENDALL, M. G. 1938. A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30(1-2): 81–93.

Li, J.; Sun, S.; Yuan, W.; Fan, R.-Z.; Zhao, H.; and Liu, P. 2023a. Generative Judge for Evaluating Alignment. *arXiv preprint arXiv:2310.05470*.

Li, X.; Lan, Y.; and Yang, C. 2024. TreeEval: Benchmark-Free Evaluation of Large Language Models through Tree Planning. *arXiv preprint arXiv:2402.13125*.

Li, X.; Zhang, T.; Dubois, Y.; Taori, R.; Gulrajani, I.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023b. AlpacaEval: An Automatic Evaluator of Instruction-following Models. https://github.com/tatsu-lab/alpaca_eval.

Liu, Y.; Iter, D.; Xu, Y.; Wang, S.; Xu, R.; and Zhu, C. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. *arXiv preprint arXiv:2303.16634*.

Long, J. 2023. Large Language Model Guided Tree-of-Thought. *arXiv preprint arXiv:2305.08291*.

OpenAI. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Saha, S.; Levy, O.; Celikyilmaz, A.; Bansal, M.; Weston, J.; and Li, X. 2023. Branch-Solve-Merge Improves Large Language Model Evaluation and Generation. *arXiv preprint arXiv:2310.15123*.

Sainz, O.; Campos, J.; García-Ferrero, I.; Etxaniz, J.; de Lacalle, O. L.; and Agirre, E. 2023. NLP Evaluation in trouble: On the Need to Measure LLM Data Contamination for each Benchmark. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Findings of the Association for Computational Linguistics: EMNLP 2023*, 10776–10787. Singapore: Association for Computational Linguistics.

Saito, K.; Wachi, A.; Wataoka, K.; and Akimoto, Y. 2023. Verbosity Bias in Preference Labeling by Large Language Models. *arXiv preprint arXiv:2310.10076*.

Spearman, C. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1): 72–101.

Suzgun, M.; Scales, N.; Schärli, N.; Gehrmann, S.; Tay, Y.; Chung, H. W.; Chowdhery, A.; Le, Q. V.; Chi, E. H.; Zhou, D.; and Wei, J. 2022. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. *arXiv preprint arXiv:2210.09261*.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. *GitHub repository*.

Team, G. 2023a. Gemini: A Family of Highly Capable Multimodal Models. *arXiv preprint arXiv:2312.11805*.

Team, X.-L. 2023b. Xwin-LM.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Tunstall, L.; Beeching, E.; Lambert, N.; Rajani, N.; Huang, S.; Rasul, K.; Rush, A. M.; and Wolf, T. 2023a. The Alignment Handbook. https://github.com/huggingface/alignment-handbook.

Tunstall, L.; Beeching, E.; Lambert, N.; Rajani, N.; Rasul, K.; Belkada, Y.; Huang, S.; von Werra, L.; Fourrier, C.; Habib, N.; Sarrazin, N.; Sanseviero, O.; Rush, A. M.; and Wolf, T. 2023b. Zephyr: Direct Distillation of LM Alignment. *arXiv preprint arXiv:2310.16944*.

Wang, B.; Zheng, R.; Chen, L.; Liu, Y.; Dou, S.; Huang, C.; Shen, W.; Jin, S.; Zhou, E.; Shi, C.; Gao, S.; Xu, N.; Zhou, Y.; Fan, X.; Xi, Z.; Zhao, J.; Wang, X.; Ji, T.; Yan, H.; Shen, L.; Chen, Z.; Gui, T.; Zhang, Q.; Qiu, X.; Huang, X.; Wu, Z.; and Jiang, Y.-G. 2024. Secrets of RLHF in Large Language Models Part II: Reward Modeling. *arXiv preprint arXiv:2401.06080*.

Wang, P.; Li, L.; Chen, L.; Cai, Z.; Zhu, D.; Lin, B.; Cao, Y.; Liu, Q.; Liu, T.; and Sui, Z. 2023a. Large Language Models are not Fair Evaluators. *arXiv preprint arXiv:2305.17926*.

Wang, S.; Sun, X.; Li, X.; Ouyang, R.; Wu, F.; Zhang, T.; Li, J.; and Wang, G. 2023b. GPT-NER: Named Entity Recognition via Large Language Models. *arXiv preprint arXiv:2304.10428*.

Wang, Y.; Yu, Z.; Zeng, Z.; Yang, L.; Wang, C.; Chen, H.; Jiang, C.; Xie, R.; Wang, J.; Xie, X.; Ye, W.; Zhang, S.; and Zhang, Y. 2023c. PandaLM: An Automatic Evaluation Benchmark for LLM Instruction Tuning Optimization. *arXiv preprint arXiv:2306.05087*.

Wu, M.; and Aji, A. F. 2023. Style Over Substance: Evaluation Biases for Large Language Models. *arXiv preprint arXiv:2307.03025*.

Xu, C.; Sun, Q.; Zheng, K.; Geng, X.; Zhao, P.; Feng, J.; Tao, C.; and Jiang, D. 2023. WizardLM: Empowering Large Language Models to Follow Complex Instructions. *arXiv preprint arXiv:2304.12244*.

Yang, S.; Chiang, W.-L.; Zheng, L.; Gonzalez, J. E.; and Stoica, I. 2023. Rethinking Benchmark and Contamination for Language Models with Rephrased Samples. *arXiv preprint arXiv:2311.04850*.

Zhang, P.; Xiao, S.; Liu, Z.; Dou, Z.; and Nie, J.-Y. 2023a. Retrieve Anything To Augment Large Language Models. *arXiv preprint arXiv:2310.07554*.

Zhang, X.; Yu, B.; Yu, H.; Lv, Y.; Liu, T.; Huang, F.; Xu, H.; and Li, Y. 2023b. Wider and Deeper LLM Networks are Fairer LLM Evaluators. *arXiv preprint arXiv:2308.01862*.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E. P.; Zhang, H.; Gonzalez, J. E.; and Stoica, I. 2023a. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *arXiv preprint arXiv:2306.05685*.

Zheng, R.; Dou, S.; Gao, S.; Hua, Y.; Shen, W.; Wang, B.; Liu, Y.; Jin, S.; Liu, Q.; Zhou, Y.; Xiong, L.; Chen, L.; Xi, Z.; Xu, N.; Lai, W.; Zhu, M.; Chang, C.; Yin, Z.; Weng, R.; Cheng, W.; Huang, H.; Sun, T.; Yan, H.; Gui, T.; Zhang, Q.; Qiu, X.; and Huang, X. 2023b. Secrets of RLHF in Large Language Models Part I: PPO. *arXiv preprint arXiv:2307.04964*.

Zhong, W.; Cui, R.; Guo, Y.; Liang, Y.; Lu, S.; Wang, Y.; Saied, A.; Chen, W.; and Duan, N. 2023. AGIEval: A Human-Centric Benchmark for Evaluating Foundation Models. *arXiv preprint arXiv:2304.06364*.

Zhou, J.; Lu, T.; Mishra, S.; Brahma, S.; Basu, S.; Luan, Y.; Zhou, D.; and Hou, L. 2023a. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911*.

Zhou, K.; Zhu, Y.; Chen, Z.; Chen, W.; Zhao, W. X.; Chen, X.; Lin, Y.; Wen, J.-R.; and Han, J. 2023b. Don't Make Your LLM an Evaluation Benchmark Cheater. *arXiv preprint arXiv:2311.01964*.

Zhu, K.; Chen, J.; Wang, J.; Gong, N. Z.; Yang, D.; and Xie, X. 2024. DyVal: Dynamic Evaluation of Large Language Models for Reasoning Tasks. *arXiv preprint arXiv:2309.17167*.

Zhu, L.; Wang, X.; and Wang, X. 2023. JudgeLM: Fine-tuned Large Language Models are Scalable Judges. *arXiv preprint arXiv:2310.17631*.