

INSPECTORRAGET: An Introspection Platform for RAG Evaluation

Kshitij Fadnis

kpfadnis@us.ibm.com

Siva Sankalp Patel

siva.sankalp.patel@ibm.com

Odellia Boni

odelliab@il.ibm.com

Yannis Katsis

yannis.katsis@ibm.com

Sara Rosenthal

sjrosenthal@us.ibm.com

Benjamin Sznajder

benjams@il.ibm.com

Marina Danilevsky

mdanile@us.ibm.com

IBM Research - AI

Abstract

Large Language Models (LLM) have become a popular approach for implementing Retrieval-Augmented Generation (RAG) systems, and a significant amount of effort has been spent on building good models and metrics. In spite of increased recognition of the need for rigorous evaluation of RAG systems, few tools exist that go beyond the creation of model output and automatic calculation. We present INSPECTORRAGET, an introspection platform for RAG evaluation. INSPECTORRAGET allows the user to analyze aggregate and instance-level performance of RAG systems, using both human and algorithmic metrics as well as annotator quality. INSPECTORRAGET is suitable for multiple use cases and is available publicly to the community¹. The demo video is available at <https://youtu.be/MJhe8QIXcEc>

1 Introduction

The recent advances in Large Language Models (LLMs) have led to an explosion of research on Retrieval-Augmented Generation (RAG): combining generative LLMs with data retrieval to provide responses grounded on authoritative document collections (Lewis et al., 2020). RAG systems have been deployed in diverse domains (see (Gao et al., 2024) for a recent survey).

The development of all RAG systems have one thing in common: the need for *rigorous evaluation*. Evaluating such systems requires continuously designing experiments including datasets, models, and metrics; running evaluations; and analyzing the results to address shortcomings and make appropriate deployment decisions. Businesses wishing to deploy a RAG system must evaluate it on their specific use cases (not only academic benchmarks), often comparing multiple architectures and models.

Recognizing the importance of evaluation, the research community has been creating evaluation benchmark datasets (Liu et al., 2023; Chen et al.,

2023), evaluation metrics (Es et al., 2023), and noise robustness (Chen et al., 2023), as well as evaluation frameworks (such as RAGAs (Es et al., 2023), and ARES (Saad-Falcon et al., 2023)). Leveraging these assets, practitioners can automatically compute aggregate evaluation scores for their RAG systems. However, this provides only a very limited lens of a system’s performance. We suggest that a comprehensive evaluation of RAG systems should include the following:

Aggregate Performance: Overall evaluation is important for continuous benchmarking of models and dataset performance (Gehrmann et al., 2022).

Instance-level Analysis: Aggregate metrics alone do not offer much insight into the RAG system, particularly in identifying the source of undesirable output. A flexible, feature-rich workflow for detecting and inspecting related individual instances empowers the researcher to perform actionable error analysis.

Mixed Metrics: Algorithmic metrics such as BLEU and ROUGE do not necessarily align with human judgements (Reiter, 2018; Blagec et al., 2021); nor are automatic evaluations using LLMs (Es et al., 2023) a perfect substitute. However, a comprehensive analysis of all types of metrics can yield a rich array of insights.

Annotator Qualification: Even human judgements of RAG systems are imperfect (Chiang and Lee, 2023). A thorough understanding of annotator behavior allows the researcher to identify and improve ambiguous guidelines, complex data points, and underperforming annotators, resulting in higher quality evaluations.

Dataset Characterization: The dataset itself should be subjected to a thorough inspection during evaluation. Fixing erroneous reference answers, clarifying ambiguous instances, or even identifying bias in the content can improve the overall evaluation outcome by providing much needed context for the observed quantitative results.

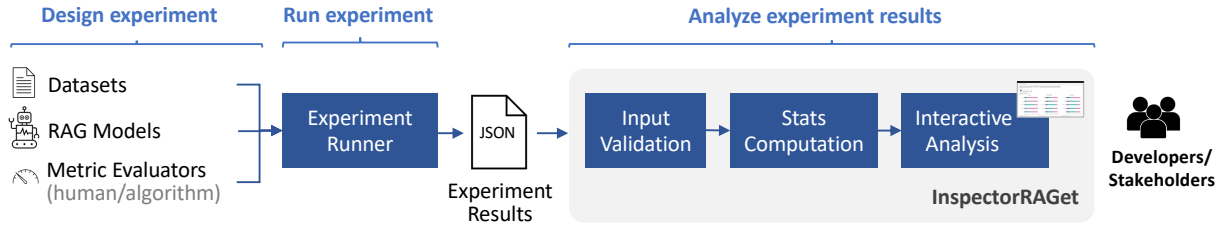


Figure 1: RAG evaluation life cycle

Achieving such a multi-pronged evaluation would currently require researchers to manually examine their evaluation output, perhaps creating various ad-hoc data processing scripts and visualizations in spreadsheet tools. Our goal is to significantly lower the required effort for this important and undersupported aspect of RAG evaluation. Users should be able to easily drill down from aggregate performance overviews to instance-level views of data, models, metrics and annotators, in order to perform detailed error analysis and identify actionable insights.

Our contributions are as follows:

1. We present INSPECTORRAGET, an interactive evaluation platform with a rich feature set that addresses each of the desired aspects of a comprehensive RAG evaluation: performance benchmarking, a combined aggregate and instance level analysis, a holistic view of results via a mix of metrics, annotator qualification, and dataset characterization.
2. We use INSPECTORRAGET to yield concrete and actionable insights on two use cases, on both new and existing datasets.
3. We open source the platform for use by the community¹.

2 The RAG Evaluation Life Cycle

We begin by briefly describing the life cycle of evaluating RAG systems, and situating INSPECTORRAGET within it, as shown in Figure 1. Evaluating RAG systems involves three main steps:

(1) Design the evaluation experiment: One starts by designing the *evaluation experiment* (or simply *experiment*), which consists of *Models*, *Datasets*, *Metrics* and *Metric Evaluators*, as defined below.

(2) Run the evaluation experiment: Once the experiment has been designed, the next step is to

compute the evaluation scores. This involves (a) computing the model responses and (b) passing the responses to the metric evaluators (algorithms or humans) to produce instance-level scores.

(3) Analyze the experiment results: Finally, the experiment results need to be analyzed to gain actionable insights about the models, datasets, metrics, annotation quality, etc.

2.1 INSPECTORRAGET for RAG Evaluation

INSPECTORRAGET focuses on the third step of the RAG evaluation life cycle. To analyze an experiment using the platform, model developers or stakeholders upload a standardized JSON file summarizing the experiment results. This summary contains the following information:

- *Datasets*: The set of data instances included in the experiment, each containing user input (e.g., question/conversation), contexts (e.g., passages), and, optionally, reference responses.
- *Model metadata*: Name and description of the RAG models that were evaluated.
- *Metric metadata*: Metadata about the *metrics* on which the model responses were evaluated. These include the metric name, type (i.e., *algorithmic*, if it was evaluated by an algorithm or an LLM used as a judge and *human*, if it was evaluated by humans, such as crowd workers), and scale (yes/no, Likert scale, numeric, etc.).
- *Evaluation scores*: The evaluation scores of models along each metric on every data instance².

Both the platform and the experiment results file format (described in more detail in Appendix A) have been designed in a model-agnostic and metric-agnostic way, in order to support analysis of diverse evaluation experiments.

Once provided with the input file, INSPECTORRAGET validates it for errors (e.g., missing evaluation scores) and augments it with additional statis-

¹<https://github.com/IBM/InspectorRAGet>

²In case of multiple annotators per metric, such as when multiple humans annotate a single response, multiple evaluation scores are included: one per annotator.



Figure 2: Illustration of INSPECTORRAGET’s core views and corresponding visualizations

tics (such as inter-annotator agreement for human evaluations). The augmented experiment results are then used to power the platform’s frontend; a visual analytics application that enables developers and stakeholders to interactively analyze and gain insights on different aspects of the experiment, as described in the next section.

Finally, while INSPECTORRAGET focuses on the analysis of the experiment results, to help users employ the platform as part of the broader RAG evaluation life cycle, we have also implemented a sample experiment runner in a Python notebook, showing how to run experiments using Hugging-Face assets (Wolf et al., 2020) (e.g. datasets, models, and metric evaluators) and output the results in the format expected by INSPECTORRAGET (see Appendix B for details).

3 INSPECTORRAGET Platform

INSPECTORRAGET’s frontend offers a series of views (presented as separate tabs), each tailored to a different aspect of the evaluation process for analysis (see Appendix C for implementation details). Excerpts of these views are shown in Figure 2. We describe each view along with the analyses it enables. In Section 4, we then demonstrate how these views can be used to extract actionable insights for two different use cases.

3.1 Predictions Table

When analyzing an experiment, it helps to first see a few examples of instances. To this end, INSPECTORRAGET’s predictions view shows a table of all questions with the corresponding model responses. This view not only gets developers acquainted with

the experiment but also helps them spot patterns in the model responses (e.g., length, repetitions of text, etc.) that should be investigated further.

3.2 Performance Overview

After getting acquainted with the experiment, developers and stakeholders can get an overview of the experiment results through the overall performance view. This view shows the aggregate score and ranking of each model for each evaluation metric. This information is rendered both in tabular form, as well as through a Radar chart with separate tables/visualizations for human and algorithmic metrics (Figures 2b and 2c).

While designing this view, special attention was given to quantifying the uncertainty present in human evaluations to avoid misinterpretation of the results. Aggregate evaluation scores for human metrics are shown along with (a) their standard deviation and (b) a visualization of inter-annotator agreement (displayed through sparkline charts).

Despite its resemblance to leaderboards (Zheng et al., 2023; Hendrycks et al., 2021), the goal of this view is not only to declare winners but to also make initial observations that need to be explored in more depth.

3.3 Model Behavior

After obtaining an overview of aggregate model performance, one can drill down and inspect model responses for individual instances through the model behavior view. Users start the analysis by filtering instances based on criteria, such as dataset domain, whether a question is answerable, etc. The view then shows a histogram of model scores for all instances satisfying the filter (Figure 2d), as well as a sortable table of the actual instances (Figure 2e). Upon selecting an instance, users see a detailed view of the instance, including the conversation/question, context, the model responses and their respective evaluation scores (Figure 2f). Instance-level analysis is crucial for conducting error analysis and understanding the root causes of issues observed in other views, as well as identifying “I know it when I see it” types of issues. The view also provides users with the functionality to easily copy, flag or comment on instances.

3.4 Model Comparator

In addition to inspecting individual model responses, one can also analyze entire models. This is enabled by the model comparator view, which for a

chosen pair of models and a metric, shows whether the scores of the two models for the selected metric are derived from the same distribution (Figure 2g). This is shown both through a scatter plot, depicting the scores of the two models for the metric, as well as through the result of a statistical significance test, computed using Fisher’s randomization method (Smucker et al., 2007). Continuing the support for instance-level analysis, the view also allows one to drill down and inspect the instances where two models received very similar/dissimilar evaluation scores. This view can be useful for spotting similarities and differences between models.

3.5 Metric Behavior

Similarly to comparing models, one can also compare metrics. This is accomplished through the metric behavior view, which shows the Spearman correlation scores for each pair of metrics. This allows developers to gain insights on metrics, such as identifying whether an automatic metric correlates well with human judgements or whether supposedly orthogonal metrics correlate with each other (see fluency and answer relevance in Figure 2h), hinting at issues with metric definitions.

3.6 Annotator Behavior

Finally, when human evaluation is involved, it is imperative to also analyze its quality. This is enabled by the annotator behavior view, which contains two types of visualizations related to annotation quality: (a) *Model-level visualizations* that show the annotator agreement when evaluating each model (Figure 2i). These can help provide insights on challenges that certain models pose to human annotators (e.g., if humans had trouble reaching agreement when evaluating certain models). (b) *Annotator-level visualizations* depicting individual annotators’ performance. These include a visualization of inter-annotator agreement (Figure 2j), computed using Cohen’s kappa (Cohen, 1960), annotator contribution (i.e., how often an annotator agreed with the majority) (Figure 2k), and annotation time (not shown for space reasons). Insights drawn from this view can drive additional analyses (e.g., to understand why annotators had trouble annotating the responses of certain models) or changes to the experiment setting (e.g., give feedback to individual annotators or improve the annotation guidelines).

Use Case	Dataset	Models	Evaluation Metrics
RAG Model Evaluation	CLAPNQ	Llama-13B (Touvron et al., 2023), GPT-3.5 Turbo (Brown et al., 2020), Mistral (Jiang et al., 2023)	- Human: fluency, answer relevance, faithfulness, win-rate - Algorithmic: Recall, Rouge, Bert-KPrec, Answerability, Extractiveness, Length
LLM-as-a-Judge Evaluation	MT-Bench	Alpaca-13B (Taori et al., 2023), Claude-v1 (cla, 2023), GPT-3.5 (Brown et al., 2020), GPT-4 (OpenAI et al., 2024), Vicuna-13B (Chiang et al., 2023)	- Human: Win-Rate - Algorithmic: LLM-Judge GPT-4

Table 1: Evaluation settings for the use cases presented in this paper

4 Finding Insights

To showcase the value of INSPECTORRAGET, we next describe how it can be used to get insights on two use cases: (1) *RAG Model Evaluation*: We collected human judgements of model responses for a RAG dataset. This allows us to show the full scope of our platform for comparing human and algorithmic metrics as well as multiple annotators. (2) *LLM-as-a-Judge Evaluation*: We use annotations comparing human and LLM-as-a-judge on model output for multi-turn question answering.

The experiment setting for each use case (incl. dataset, models, and evaluation metrics) is summarized in Table 1. For each use case we describe the discovered insights along with the *Source* views used to identify them and propose possible **Actions** for improving the RAG life cycle. We provide platform screenshots of the findings in Appendix D.

4.1 RAG Model Evaluation Use Case

To illustrate the full capabilities of our platform on RAG we performed our own manual evaluation on CLAPNQ (Rosenthal et al., 2024), a long form question answering dataset. We also explored Fine-Grained ASQA (Stelmakh et al., 2022; Wu et al., 2023) and InstructQA (Adlakha et al., 2023) as alternative RAG model evaluations but both of these evaluations did not provide enough details suitable for analysis (e.g., missing model information and incomplete human annotations).

CLAPNQ is built on the portion of the Natural Questions dataset (Kwiatkowski et al., 2019) that only has a long answer (gold passage) without an extractive short answer. The responses in CLAPNQ are grounded on the gold passage and must be concise and complete. We ensure that every question is evaluated by the same algorithmic metrics and the same number of human evaluators (3 per question). The human evaluation annotation task was completed on Appen³, a crowdsourcing platform

for collecting high-quality annotations. Each task has a question, grounding passage, and multiple randomly shown model responses for the annotator to provide their evaluations. We asked annotators to evaluate the answers on three metrics: *fluency*, *answer relevance*, and *faithfulness* as commonly used in the literature (Es et al., 2023; Chiang and Lee, 2023). In addition, we also asked them to perform a head-to-head comparison of all models for *win-rate*. These annotations allow us to use the platform to compare and draw insights from the models, data, metrics, and annotators. We next describe key insights we found using the platform.

Low Performance: It is clear that Llama is the model least preferred by the annotators based on win-rate. Its answers are somewhat faithful but not relevant. It is also the only one that has lower fluency for 29/100 tasks. Annotators did not like the phrase “Sure I’d be happy to help” which was used frequently by Llama. Llama answers are also the longest and most extractive (which correlates with high faithfulness). *Source: Predictions, Performance Overview, Model Behavior, Model Comparator* **Action:** Propose to stakeholders to reconsider Llama as model of choice for this use case.

Algorithmic vs Human: Based on algorithmic metrics Mistral is the worst, but it was considerably preferred over Llama by human evaluators. We suspect that this is because its responses are slightly shorter, which biases Recall. *Source: Performance Overview.* **Action:** Continue including human evaluation in future evaluation rounds, as it provides different insights than algorithmic metrics.

Dataset Inconsistencies: During the human evaluation the reference responses (labeled Reference) was rated highest by the annotators which shows that the dataset is of good quality. However, the annotators disagree most on these responses. Filtering to see examples of disagreement, shows that the most disagreement is 25 cases of Answer relevance. Opening up an indeterminate example

³<https://www.appen.com/>

where there was no agreement, shows a case of a tricky question+answer which explains the disagreement.

Source: *Performance Overview, Annotator Behavior, Model Behavior*. **Action:** Propose revisiting the experiment design via data analysis and improvement.

Data Types: It is also interesting to explore the answerable and unanswerable splits separately. Examples of insights include finding examples where an unanswerable question was answered by the models and is faithful, which may indicate that the question is actually answerable. One may also search for unanswerable examples that are not faithful to show that the model is coming up with an answer from its own knowledge or is hallucinating. Source: *Performance Overview, Model Behavior*. **Action:** Performance can differ due to question type, domain and other characteristics. Provide model feedback to stakeholders per dimension.

Metric correlations: Win rate is correlated mostly with human metrics. This highlights the importance of human evaluation. The algorithmic metrics do not indicate which responses are preferred. Extractiveness and BertK-Precision is correlated with faithfulness. This is expected as a faithful model will have information extracted from the passage either directly or reworded. The metric correlation can be used to inspect why there are cases with low extractiveness/BertK-Precision and high faithfulness. This may highlight annotator confusion. Source: *Metric Behavior*. **Action:** Perform a human evaluation for accurate model preference. Investigate possible annotator confusion.

4.2 LLM-as-a-Judge Evaluation Use Case

There has been a significant uptick in the popularity of LLMs as judges to evaluate the quality of LLM responses in RAG settings, complemented by active research in the efficacy of these judges (Chiang and Lee, 2023; Shen et al., 2023). Our platform also seamlessly supports analyzing the performance of LLM-as-a-judge approaches, and we illustrate this use case using MT-Bench. MT-Bench (Zheng et al., 2023) is a multi-turn question answering dataset of 80 high quality multi-turn questions. The paper introducing this dataset explores using LLM-as-a-judge as a means of evaluating benchmarks by comparing it to human evaluation by 58 expert annotators. To this end, they release human and algorithmic GPT-4 judgements on the MT-bench dataset.

This evaluation is not centered around comparing model performance, but rather judge performance.

They discuss several limitations of LLM-as-a-judge: positional bias, verbosity bias, self-enhancement bias (judge favors its own model’s answers), and limited reasoning ability (low reasoning and math capability). They show that GPT-4 judge matches human evaluation at over 80%. We expand on their insights and introduce additional ones.

Self-Enhancement Bias: The expert annotators tend to agree with each other and match GPT-4 closely on all models, but GPT-4 seems to prefer responses from its own model. Source: *Annotator Behavior, Model Behavior*. **Action:** Inform stakeholders of the slight bias of GPT-4.

Verbosity Bias: Answer length is strongly correlated with win-rate. This is true for LLM-as-a-judge and human annotators. Claude-v1, GPT-3.5, and GPT-4 usually have the longer response and win, while Llama and Alpaca rarely have the longer response and usually lose. Source: *Model Behavior*. **Action:** Analyze data and share with stakeholders. Do the answers need to be long for these questions? What is missing in the short responses?

Positional Bias: As hinted in the paper, the first answer being favored is not as much of an issue for GPT-4. We believe it also may be model dependent. For instance Llama is always shown to GPT-4 first but also almost always loses and Claude-v1 is always shown second and always wins. Source: *Model Behavior*. **Action:** Revisit evaluation design. Investigate if there may be some unintentional bias because these two models were never randomized.

5 Conclusion

We present INSPECTORRAGET, a publicly available⁴ platform to empower researchers, developers and stakeholders to gain a deeper understanding of the strengths and limitations of RAG systems. It includes aggregate-level and instance-level views as well as capabilities to explore human and algorithmic metrics and annotator behavior, allowing for more holistic analysis. We describe two pertinent use cases - RAG model evaluation and LLM-as-a-Judge - that showcase and highlight the value of these features, and publicly release our use-case input files and notebooks. In the future, we will explore adding additional capabilities to INSPEC-

⁴<https://github.com/IBM/InspectorRAGet>

TORRAGET to facilitate comprehensive pattern discovery, as well as extending analytical capabilities beyond RAG to other popular LLM tasks such as summarization and code generation.

6 Ethical Considerations

A key claim in our paper is to include human annotations in the evaluation process. Human evaluation is subjective and prone to errors. We expect even the best annotators to make mistakes. We address mitigating this by including multiple annotators per question, but biases still may occur.

All proposed actions are based on evidence provided by the tool and considered to be our own opinions for possible areas of improvement. Any biases in the datasets may impact the analysis. The platform is meant to be used as a means of drawing conclusions and insights of RAG systems; it does not create its own conclusions or insights.

We acknowledge that there are accessibility limitations in the platform and we plan on providing additional features to improve these limitations.

References

2023. [Introducing claude](#).
- Vaibhav Adlakha, Parishad BehnamGhader, Xing Han Lu, Nicholas Meade, and Siva Reddy. 2023. [Evaluating correctness and faithfulness of instruction-following models for question answering](#). *ArXiv*, abs/2307.16877.
- Kathrin Blagec, Georg Dorffner, Milad Moradi, and Matthias Samwald. 2021. [A critical analysis of metrics used for measuring progress in artificial intelligence](#). *ArXiv*, abs/2008.02577.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#).
- Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2023. [Benchmarking large language models in retrieval-augmented generation](#). *ArXiv*, abs/2309.01431.
- Cheng-Han Chiang and Hung-yi Lee. 2023. [Can large language models be an alternative to human evaluations?](#) In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15607–15631, Toronto, Canada. Association for Computational Linguistics.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. [Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality](#).
- Jacob Cohen. 1960. [A coefficient of agreement for nominal scales](#). *Educational and psychological measurement*, 20(1):37–46.
- Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. [RAGAS: Automated evaluation of retrieval augmented generation](#). *ArXiv*, abs/2309.15217.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. [Retrieval-augmented generation for large language models: A survey](#). *ArXiv*, abs/2312.10997.
- Sebastian Gehrmann, Abhik Bhattacharjee, Abinaya Mahendiran, Alex Wang, Alexandros Papangelis, Aman Madaan, Angelina Mcmillan-major, Anna Shvets, Ashish Upadhyay, Bernd Bohnet, Bingsheng Yao, Bryan Wilie, Chandra Bhagavatula, Chaobin You, Craig Thomson, Cristina Garbacea, Dakuo Wang, Daniel Deutsch, Deyi Xiong, Di Jin, Dimitra Gkatzia, Dragomir Radev, Elizabeth Clark, Esin Durmus, Faisal Ladhak, Filip Ginter, Genta Indra Winata, Hendrik Strobelt, Hiroaki Hayashi, Jekaterina Novikova, Jenna Kanerva, Jenny Chim, Jiawei Zhou, Jordan Clive, Joshua Maynez, João Sedoc, Juraj Juraska, Kaustubh Dhole, Khyathi Raghavi Chandu, Laura Perez Beltrachini, Leonardo F. R. Ribeiro, Lewis Tunstall, Li Zhang, Mahim Pushkarna, Mathias Creutz, Michael White, Mihir Sanjay Kale, Moussa Kamal Eddine, Nico Daheim, Nishant Subramani, Ondrej Dusek, Paul Pu Liang, Pawan Sasanka Ammanamanchi, Qi Zhu, Ratish Puduppully, Reno Kriz, Rifat Shahriyar, Ronald Cardenas, Saad Mahamood, Salomey Osei, Samuel Cahyawijaya, Sanja Štajner, Sebastien Montella, Shailza Jolly, Simon Mille, Tahmid Hasan, Tianhao Shen, Tosin Adewumi, Vikas Raunak, Vipul Raheja, Vitaly Nikolaev, Vivian Tsai, Yacine Jernite, Ying Xu, Yisi Sang, Yixin Liu, and Yufang Hou. 2022. [GEMv2: Multilingual NLG benchmarking in a single line of code](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 266–281, Abu Dhabi, UAE. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#).
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego

- de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. 2023. [Mistral 7b](#).
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: A benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K  ttler, Mike Lewis, Wen-tau Yih, Tim Rock-t  schel, Sebastian Riedel, and Douwe Kiela. 2020. [Retrieval-augmented generation for knowledge-intensive nlp tasks](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.
- Yi Liu, Lianzhe Huang, Shicheng Li, Sishuo Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023. [RECALL: A benchmark for LLMs robustness against external counterfactual knowledge](#). *ArXiv*, abs/2311.08147.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Sim  n Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David M  ly, Ashvin Nair, Reiichiro Nakano, Rameev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambatista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cer  n Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. [Gpt-4 technical report](#).
- Ehud Reiter. 2018. [A structured review of the validity of BLEU](#). *Computational Linguistics*, 44(3):393–401.
- Sara Rosenthal, Avirup Sil, Radu Florian, and Salim Roukos. 2024. [CLAPNQ: Cohesive long-form answers from passages in natural questions for rag systems](#). *ArXiv*, abs/2404.02103.
- Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2023. [ARES: An automated evalua-](#)

tion framework for retrieval-augmented generation systems. *ArXiv*, abs/2311.09476.

Chenhui Shen, Liying Cheng, Xuan-Phi Nguyen, Yang You, and Lidong Bing. 2023. [Large language models are not yet human-level evaluators for abstractive summarization](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4215–4233, Singapore. Association for Computational Linguistics.

Mark D. Smucker, James Allan, and Ben Carterette. 2007. [A comparison of statistical significance tests for information retrieval evaluation](#). In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM '07*, page 623–632, New York, NY, USA. Association for Computing Machinery.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. [ASQA: Factoid questions meet long-form answers](#). In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8273–8288, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. [Llama 2: Open foundation and fine-tuned chat models](#).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame,

Quentin Lhoest, and Alexander M. Rush. 2020. [Huggingface’s transformers: State-of-the-art natural language processing](#).

Zequ Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. [Fine-grained human feedback gives better rewards for language model training](#). *ArXiv*, abs/2306.01693.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. [Judging llm-as-a-judge with mt-bench and chatbot arena](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

A Experiment Results File Format

As INSPECTORRAGET is a web application, we naturally gravitated towards adopting JSON as an input format. Our prescribed structure for an experiment results file is intuitive and strives to minimize repetition of information.

The experiment result file can be broadly split into six sections along the functional boundaries. The first section captures general details about the experiment in *name*, *description* and *timestamp* fields. The second and third sections describe the sets of models and metrics used in the experiment via the *models* and *metrics* fields, respectively. The last three sections cover the dataset and the outcome of evaluation experiment in the form of *documents*, *tasks* and *evaluations* fields. Figure 3 describes the requirements associated with each section in detail.

B Sample Experiment Runner

To help users employ INSPECTORRAGET as part of the broader RAG evaluation life cycle, we will also release a sample experiment runner in the form of a Python notebook. The notebook demonstrates how to run experiments using Hugging Face⁵ assets (datasets, models, and metric evaluators) and output the results in the experiment result JSON format expected by the platform. For our example, we use the HuggingFaceH4/ultrafeedback_binarized dataset, run inference using the zephyr-7b-sft-lora and Mistral-7B-v0.1 models, and calculate Bert score, RougeL, and BLEU metrics. The notebook can be easily modified by changing the dataset (or picking one from Hugging Face datasets), adding

⁵<https://huggingface.co>

<pre>"models": [{ "model_id": "model_1", "name": "Model 1" }, { "model_id": "model_2", "name": "Model 2" }]</pre>	<pre>"metrics": [{ "name": "metric_a", "display_name": "Metric A", "description": "", "author": "algorithm human", "type": "numerical", "aggregator": "average", "range": [0, 1, 0.1] }, { "name": "metric_b", "display_name": "Metric B", "description": "", "author": "algorithm human", "type": "categorical", "aggregator": "majority average", "values": [{ "value": "value_a", "display_value": "A", "numeric_value": 1 }, { "value": "value_b", "display_value": "B", "numeric_value": 0 }] }, { "name": "metric_c", "display_name": "Metric C", "description": "", "author": "algorithm human", "type": "text" }]</pre>	<pre>"documents": [{ "document_id": "GUID 1", "text": "documen text 1", "title": "documen title 1" }, { "document_id": "GUID 2", "text": "documen text 2", "title": "documen title 2" }, { "document_id": "GUID 3", "text": "documen text 3", "title": "documen title 3" }]</pre>
1. Each model must have a unique model_id and name.	1. Each metric must have a unique name. 2. Metric can be of "numerical", "categorical" or "text" type. 3. Numerical type metrics must specify "range" field in [start, end, bin_size] format. 4. Categorical type metrics must specify "values" field where each value must have "value" and "numeric_value" fields. 5. Text type metric are only accessible in instance level view and not used in any experiment level aggregate statistics and visual elements.	1. Each document must have a unique document_id field. 2. Each document must have a "text" field.

<pre>"tasks": [{ "task_id": "GUID X", "task_type": "question_answering conversation", "input": [{ "speaker": "user agent", "text": "" }], "contexts": [{ "document_id": "GUID 1" }, { "document_id": "GUID 2" }], "target": { "text": "Gold Reference", "enrichments": { "dataset": "A" } } }]</pre>	<pre>"evaluations": [{ "task_id": "GUID X GUID Y", "model_id": "model_1 model_2", "model_response": "", "annotations": { "metric_a": { "system": { "value": 0 } }, "metric_b": { "system": { "value": "value_a value_b" } }, "metric_c": { "system": { "value": "text" } } } }]</pre>
1. Each task must have a unique task_id. 2. Task type can be of "question_answering" or "conversation" type. 3. "input" is an array of utterance. An utterance's speaker could be either "user" or "agent". Each utterance must have "text" field. 3. "contexts" field represents a subset of documents from the "documents" field relevant to the "input" and is available to the generative models. 4. "target" field indicates expected gold or reference text. 5. The "enrichments" field is an optional field to capture additional metadata like dataset, question category, etc. Specified as a dictionary where values could be "string" or array of "strings".	1. "evaluations" field must contain evaluation for every model in "models" field on every task from "tasks" field. Thus, total number of evaluations is equal to number of models (M) X number of tasks (T) = M X T 2. Each evaluation must be associated to single task and single model. 3. Each evaluation must have model prediction on task captured in "model_response" field. 4. "annotations" field capture ratings on the model on give task for every metric specified in "metrics" field. 5. Each metric annotation is a dictionary containing worker ids as keys. In above example, "system" is a worker id. 6. Annotation from any worker on all metrics must be in the form of dictionary. At minimum, such dictionary contains "value" key capturing model's rating for the metric metric by the worker.

Figure 3: Format of the Experiments Results input file

other models for comparison, and/or adding more metrics. Once the model predictions and metrics scores are generated, we convert them to the format outlined in Appendix A and generate a file that can be imported into INSPECTORRAGET.

C INSPECTORRAGET Implementation

INSPECTORRAGET is a React⁶ web application built with NextJS 13 framework⁷. We extensively use the Carbon Design System⁸, an open-source design system with a wide range of assets including react and web components, styling guidelines, custom icons, and others. Adopting such a design system grants us the freedom to concentrate our efforts on user experience (UX) over user interface (UI).

As mentioned in Section 2, since INSPECTORRAGET, does not run the evaluation experiment (relying instead on the experiment results being provided to it as input), it does not require GPUs to run. This allows the platform to be deployed on lightweight virtual machines or even on personal laptops/desktops.

Finally, INSPECTORRAGET is built as a stateless application and does not retain any uploaded datasets. This choice was made with privacy in mind. Employing a stateless approach allows organizations to deploy a single hosted instance of the platform for multiple teams/members without the risk of sensitive data or evaluation results being inadvertently shared to inappropriate parties.

D Use Case Screenshots

We provide screenshots showcasing several of the findings we describe in Section 4 in Figures 4-15.

⁶<https://react.dev>

⁷<https://nextjs.org>

⁸<https://carbondesignsystem.com>

Task	Target	GPT-3.5 (Turbo) prediction	Llama (13b-chat) prediction	Mistral prediction	Reference prediction
which method of forecasting uses averages to predict future weather	Seasonality, in which data experiences regular and predictable changes which recur every calendar year, is the method of forecasting, where averages are used to calculate the seasonal index of the season.	The document does not provide information about the specific method of forecasting that uses averages to predict future weather.	Sure, I'd be happy to help! Based on the provided document, the method of forecasting that uses averages to predict future weather is called "Moving Average" or "Exponential Smoothing." These methods use historical data to calculate the average demand for a given season, and then use that average to predict future demand. An index higher than 1 indicates that demand is higher than average, while an index less than 1 indicates that demand is less than the average.	The Moving Average method is a simple statistical forecasting technique which uses the average of historical data as a predictor of future values. In the context of seasonality, it can be used to calculate the average demand of a particular season (day, week, month, or year) by calculating the average of historical data corresponding to that season. However, it is important to note that Moving Average is a time series model that does not account for trends or seasonality explicitly. Other models such as ARIMA, SARIMA, and Exponential Smoothing State Space Models are more advanced and can handle trends and seasonality in a more accurate manner. In conclusion, while Moving Average can be used to calculate seasonal indices, it is not the most suitable method for forecasting seasonal time series data.	Seasonality, in which data experiences regular and predictable changes which recur every calendar year, is the method of forecasting, where averages are used to calculate the seasonal index of the season.

Figure 4: Predictions View: Prediction Example from CLAPNQ. This highlights an example of Llama starting with “Sure, I’d be happy to help!”

Algorithmic Evaluations (100/100)							
Model	Recall	Rouge-L	Bert-KF1	Answerability (Accuracy)	Extractiveness	Length	Σ Rank
Reference	0.8 (1)	0.8 (1)	0.34 (3)	1 (1)	0.34 (1)	203.97 (4)	11 (1)
GPT-3.5 (Turbo)	0.5 (3)	0.35 (2)	0.41 (1)	0.92 (2)	0.28 (3)	415.78 (2)	13 (2)
Llama (13b-chat)	0.54 (2)	0.3 (4)	0.34 (3)	0.91 (3)	0.32 (2)	485.4 (1)	15 (3)
Mistral	0.48 (4)	0.32 (3)	0.39 (2)	0.89 (4)	0.27 (4)	410.24 (3)	20 (4)

* (rank) indicates model's comparative position w.r.t to other models for a given metric

Figure 5: Performance Overview: Algorithmic Metrics Overview of CLAPNQ. Compare with Human Metrics Overview in Figure 6

Human Evaluations (100/100)						
Model	Fluency	Answer Relevance	Faithfulness	Win Rate (Head-to-Head)	Σ Rank	
Reference	3.96 ± 0.16 (1)	3.73 ± 0.33 (2)	3.39 ± 0.31 (1)	64.56 ± 22.74 (1)	5 (1)	
GPT-3.5 (Turbo)	3.96 ± 0.14 (1)	3.75 ± 0.34 (1)	2.78 ± 0.31 (3)	60.78 ± 20.48 (2)	7 (2)	
Mistral	3.91 ± 0.16 (2)	3.57 ± 0.38 (3)	2.7 ± 0.36 (4)	51.89 ± 18.13 (3)	12 (3)	
Llama (13b-chat)	3.64 ± 0.38 (3)	3.19 ± 0.42 (4)	3.01 ± 0.38 (2)	37.67 ± 17.12 (4)	13 (4)	

*** (rank)** indicates model's comparative position w.r.t to other models for a given metric

*** values ± std** shows averages of aggregate values and standard deviation across all tasks

***** reflects confidence level on the aggregate values.

of tasks where where minority rating is far from majority rating,

of tasks where where minority rating is similar to majority rating and

of tasks where where all annotators chose same rating

Figure 6: Performance Overview: Human Metrics Overview of CLAPNQ. This highlights that Llama is preferred least by humans. It also shows that humans rank Mistral answers higher than algorithmic metrics (Figure 5)

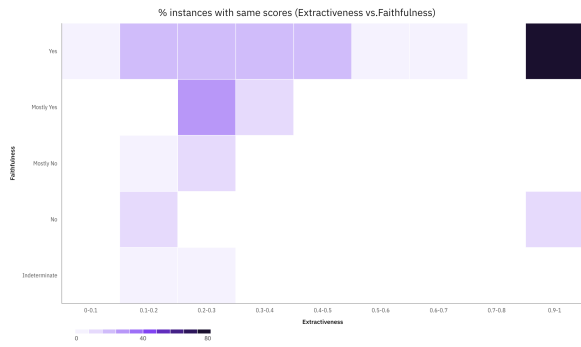


Figure 7: Metric Behavior: The majority of instances receive high scores for both Extractiveness and Faithfulness in CLAPNQ.

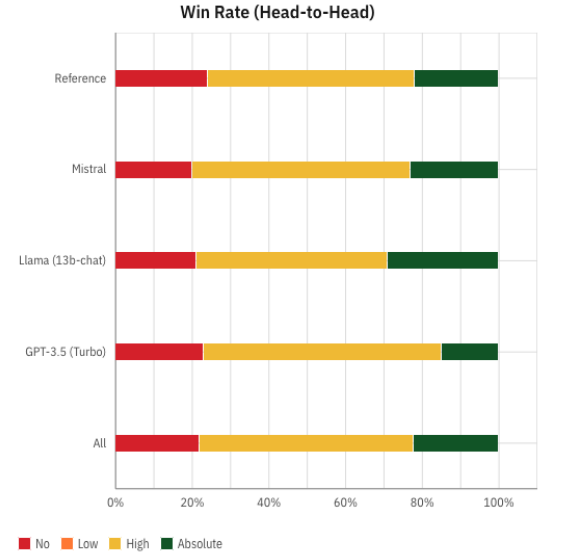


Figure 8: Annotator Behavior: Agreement for CLAPNQ on model outputs. Unexpectedly, disagreement between annotators with regard to reference responses are similar to those of generated responses.

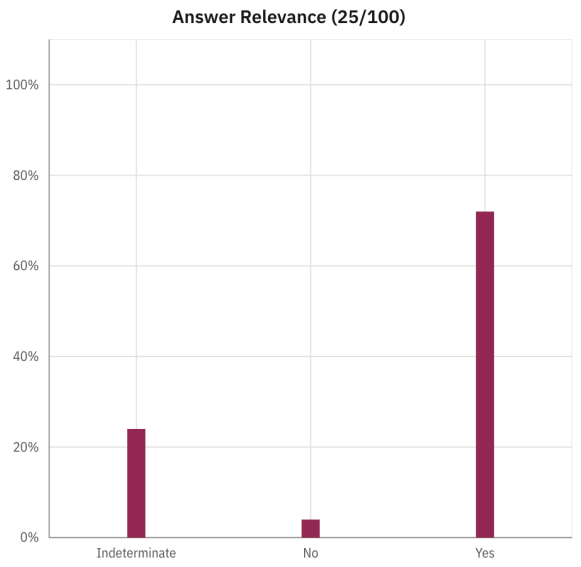


Figure 9: Model Behavior: Low Agreement on Answer Relevance for reference responses of CLAPNQ.

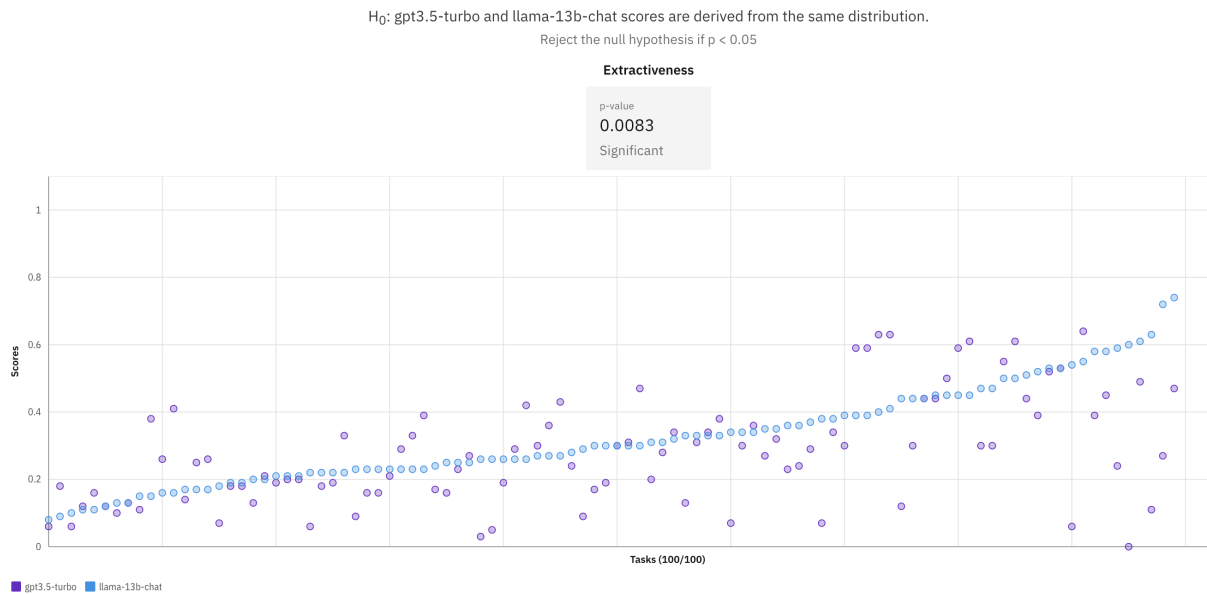


Figure 10: Model Comparator: Llama is significantly more extractive than GPT 3.5 for CLAPNQ.

LFQA-TBD::3630668852329184957

Press "Escape" to close

Conversation

User: who takes photos of planes in the air

Contexts

Aviation photography

Air - to - air photography is the art of photographing aircraft in the air , while using another aircraft as a photo platform . The subject aircraft is photographed while both aircraft are in flight . This allows the photographer to position the subject in specific locations and angles to get the most desirable shot . Some things that must be considered to achieve best results are lighting and background . Proper lighting is achieved through correct placement of the aircraft relative to the sun , and is accomplished flying only at certain times of the day and / or by flying at a heading that lines the sun up on the subject aircraft properly . The background can highlight or distract from the subject and must be carefully considered when taking shots . Air - to - air photography uses include commercial use and advertising . Remote

Llama (13b-chat ...)

Reference

Mistral

GPT-3.5 (Turbo)

Model: Llama (13b-chat)

Response

Sure, I'd be happy to help! Based on the document you provided, air-to-air photography is the art of photographing aircraft in the air while using another aircraft as a photo platform. This type of photography is used to capture images of aircraft in specific locations and angles, and requires careful consideration of lighting and background to achieve the best results. Commercial use and advertising are some of the purposes of air-to-air photography.

Target

Air - to - air photography is the art of photographing aircraft in the air, while using another aircraft as a photo platform. The subject aircraft is photographed while both aircraft are in flight.

Human Evaluations:

Annotator	Fluency	Fluency Score Explanation	Answer Relevance	Answer Relevance Score Explanation	Faithfulness	Faithfulness Score Explanation	Win Rate (Head-to-Head)
46542882	Yes		Yes		Yes		33.33
46544861	Mostly Yes		Mostly Yes	Incorrect	Mostly Yes	Other	33.33
47200615	Yes		No	Incorrect	Yes		100

Algorithmic Evaluations:

Annotator	Rouge-L	Extractiveness	Recall	Length	Bert-KPrec	Answerability (Accuracy)
system	0.49	0.34	0.81	456	0.49	True Positive

Figure 11: Detailed View: Example of a question that is hard to interpret from CLAPNQ leading to disagreement among annotators with regard to answer relevance.

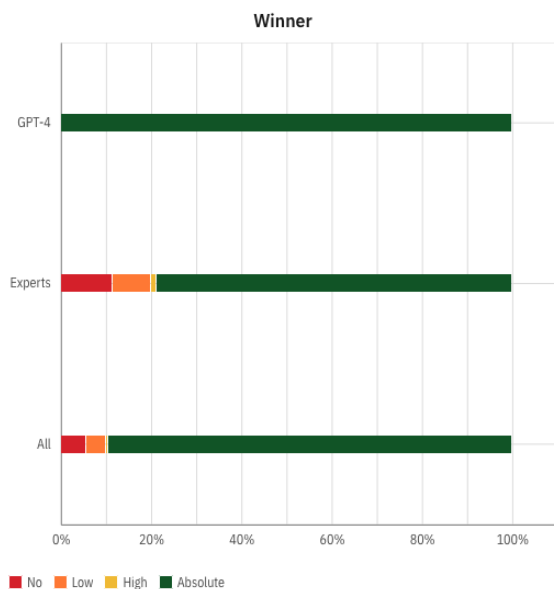


Figure 12: Annotator Behavior: The human annotators (Experts) tend to agree on head-to-head winner for MT-Bench.

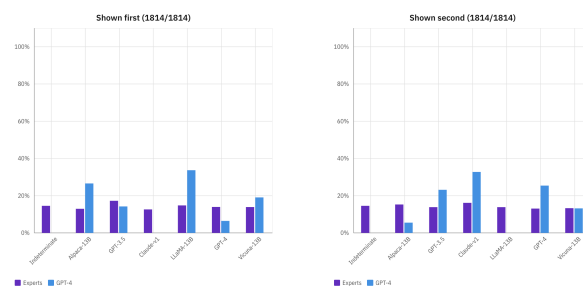


Figure 14: Model Behavior: while position distribution seem to be quite uniform among experts, GPT-4 was presented with models responses in a biased manner, with some models always occurring in the first or second position.

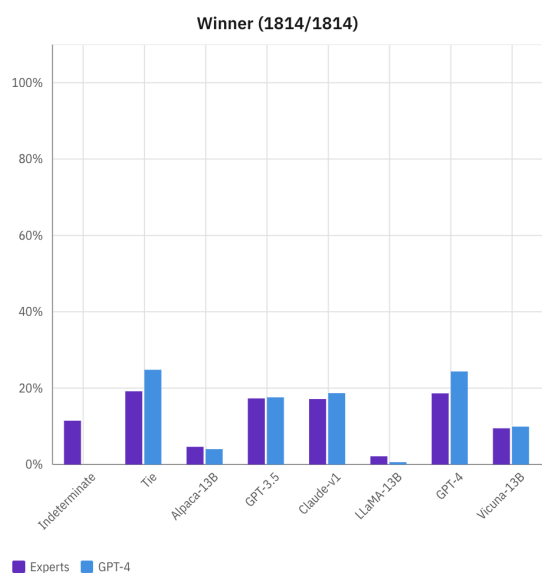


Figure 13: Model Behavior: GPT-4 responses win the most according to experts and GPT-4-judge. The experts and GPT-4-judge yield a very similar winning model distribution. They differ mostly in choosing GPT-4 as a winner (self-enhancement bias of GPT-4-judge) and declaring ties.

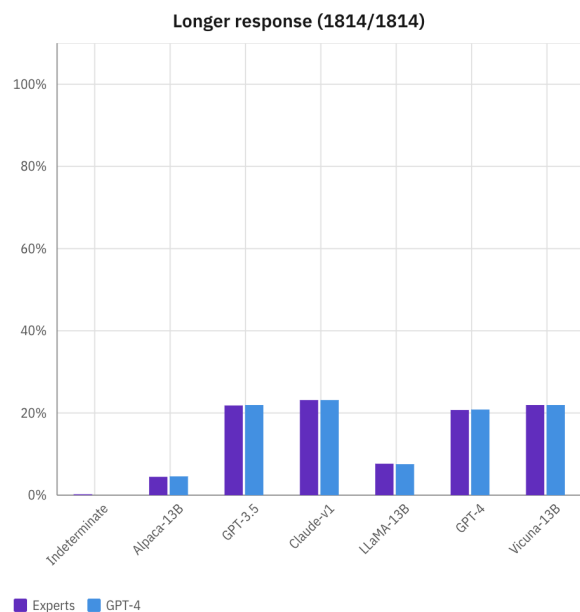


Figure 15: Model Behavior: There is a verbosity bias. Alpaca-13B and LLaMA-13B answers are the shortest and they lose the most as shown in Figure 13.