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 performing a comprehensive analysis of the quality of RAG system output. 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 community[1]. A **live instance** of the platform is available at `https://ibm.biz/InspectorRAGet`.

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).

There has been increased recognition of the importance of RAG evaluation (Longpre et al., 2024). This consists of 1) designing, 2) running, and 3) analyzing experiments (see Section 2). The research community has mainly limited analyzing to aggregate metrics via benchmark datasets (Liu et al., 2023; Chen et al., 2023), evaluation metrics (Es et al., 2023) as well as evaluation frameworks (such as RAGAs (Es et al., 2023) and ARES (Saad-Falcon et al., 2023)).

For any input (e.g., a question), a RAG system runs a retriever and passes the input and retrieved passages to a generator. This *RAG output* can then be evaluated on a variety of metrics. The output is affected by variations in retriever and generator models as well as different model configurations. Improving system performance requires actionable analysis of these variations, specific to RAG. We present a platform for performing a comprehensive analysis of the quality of *RAG output*:

Holistic analysis: End-to-end analysis of RAG system performance involves aggregate evaluations along several dimensions and configurations, enabling continuous benchmarking of models and datasets (Gehrmann et al., 2022). A comprehensive set of RAG evaluation metrics comprises algorithmic scores, LLM judges (Es et al., 2023) and human judgements, and our platform enables comparison and correlation analysis to create a full picture of performance.

Quality analysis of all aspects of a RAG experiment: While most evaluation works focus on the quality of model outputs, it is important to also analyze the quality of all other aspects of a RAG experiment. These include (a) the annotator behavior (e.g. inter-annotator agreement and underperforming annotators) (b) the effectiveness of evaluation metrics at modeling the desired functionality (e.g., faithfulness to context(s)), and (c) the RAG-specific properties of the employed dataset (e.g., relevant passages and question attributes). This analysis can be used to improve the experiment as well as providing context for the interpretation of the results.

Error analysis through instance-level inspection: Actionable error analysis for RAG requires going beyond the typical aggregate-level statistics into inspection of individual instances. This is essential for developers to identify the source of undesirable output, correct erroneous reference answers, clarify ambiguous instances, and form hypotheses for model improvement.

Currently, these analyses can only be done piecemeal, requiring the manual examination of eval-

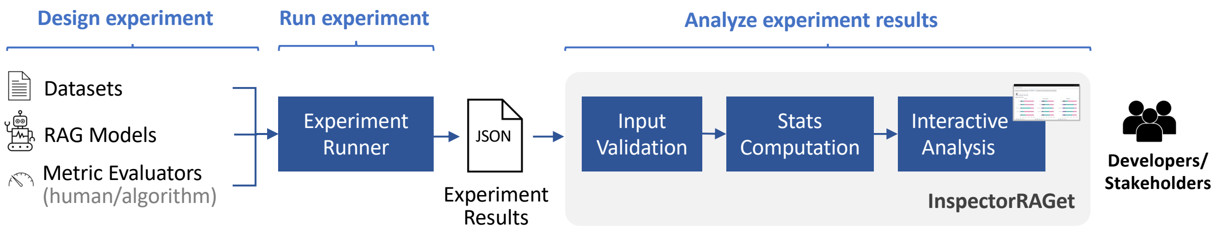


Figure 1: RAG evaluation life cycle. Evaluations of the RAG output are analyzed using INSPECTORRAGET.

uation output using various ad-hoc data processing scripts and spreadsheet tools which are not sustainable, lack re-usability, and are difficult to interpret. We provide the first platform with end-to-end capabilities for thorough RAG analysis, an undersupported aspect of RAG system development intended for researchers, model developers, and stakeholders. Our contributions are as follows:

1. We present INSPECTORRAGET, a rich interactive platform for performing a comprehensive analysis of RAG system output quality.
2. We evaluate INSPECTORRAGET on its ability to yield concrete and actionable insights on two use cases, on both new and existing datasets.
3. We open source the platform to the community[1] and host it on HuggingFace[2].

2 The RAG Evaluation Life Cycle

We briefly describe 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, which consists of *Models, Datasets, Metrics* and *Metric Evaluators*, as defined below.
(2) Run the evaluation experiment and compute the evaluation scores. This involves (a) generating the model responses/output and (b) passing the responses to the metric evaluators (algorithms or humans) to produce instance-level scores.
(3) Analyze the experiment results 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, users upload the standardized experiment results JSON file which 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 model responses were evaluated. These include metric name, type (i.e., *algorithmic*, LLM-as-a-judge and *human*, such as crowd workers), and scale (yes/no, Likert scale, numeric, etc.).
• *Model output and Evaluation scores:* The model responses and evaluation scores for the models along each metric on every data instance [3].

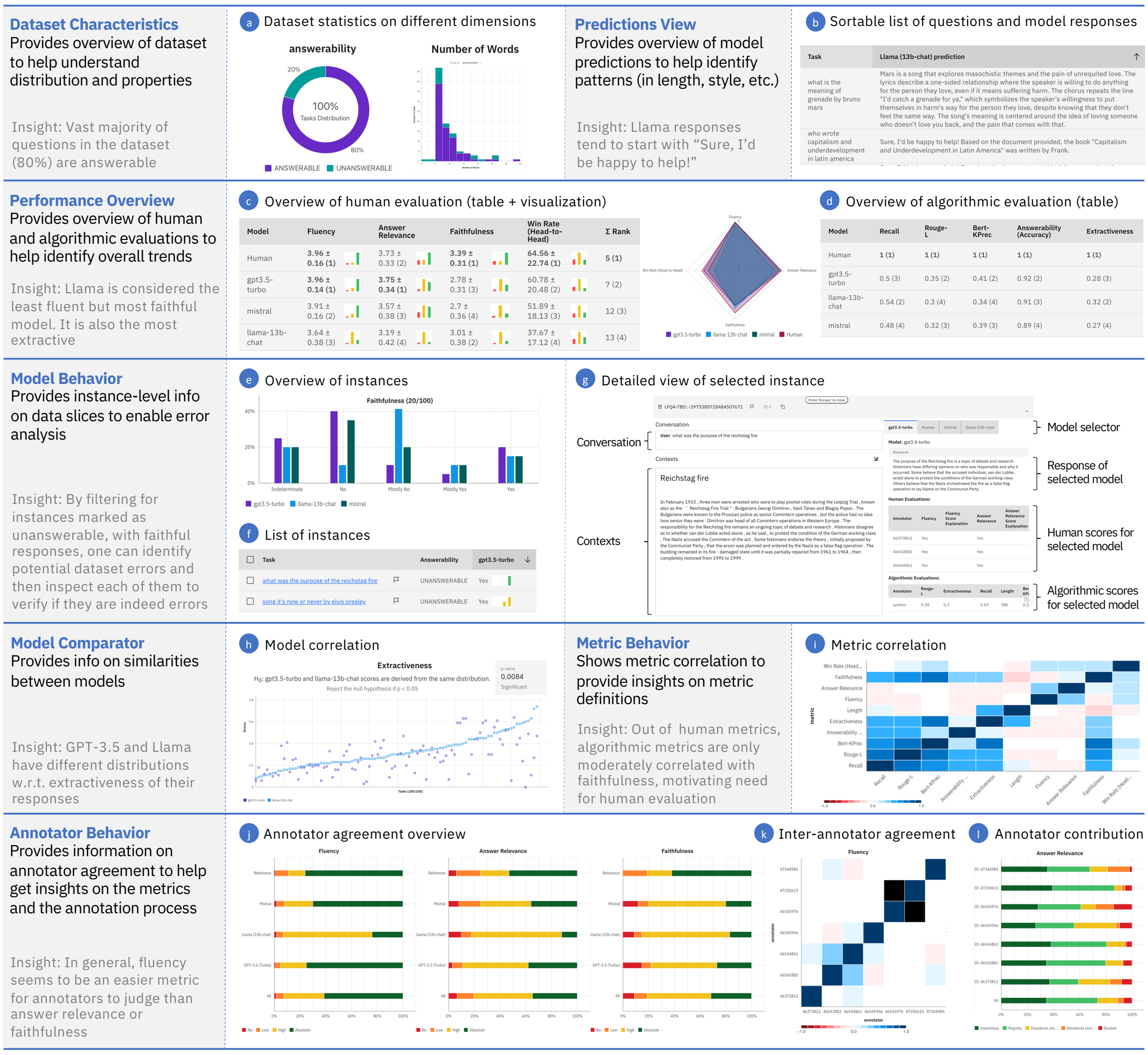
The platform and the experiment results file format have been designed in a model-agnostic and metric-agnostic way to support analysis of diverse evaluation experiments (see Appendix A).

Once provided with the input file, INSPECTORRAGET validates it for errors (e.g., missing evaluation scores) and augments it with additional statistics (e.g. inter-annotator agreement for human evaluations). The augmented results power the platform's frontend: a visual analytics application that enables users to interactively analyze and gain insights on different aspects of the experiment.

Finally, to help users employ the platform as part of the broader RAG evaluation life cycle we also provide sample experiment runners, showing how to run experiments using popular evaluation frameworks and output the results in the format expected by INSPECTORRAGET for analysis of experiment results. Our experiment runners showcase integrations with the Language Model Evaluation Harness (Gao et al., 2023) and RAGAs (Es et al., 2023) evaluation frameworks, as well as HuggingFace (Wolf et al., 2020) (see Appendix B).

[1]https://github.com/IBM/InspectorRAGet
[2]https://ibm.biz/InspectorRAGet

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



Dataset Characteristics
Provides overview of dataset to help understand distribution and properties

Insight: Vast majority of questions in the dataset (80%) are unanswerable

Predictions View
Provides overview of model predictions to help identify patterns (in length, style, etc.)

Insight: Llama responses tend to start with "Sure, I'd be happy to help!"

Task	Llama (13b-chat) prediction
what is the meaning of grenade by bruno mars	Mars is a song that explores masochistic themes and the pain of unrequited love. The lyrics describe a one-sided relationship where the speaker is willing to do anything for the person they love, even if it means suffering harm. The chorus repeats the line "I'd catch a grenade for ya," which symbolizes the speaker's willingness to put themselves in harm's way for the person they love, despite knowing that they don't feel the same way. The song's meaning is centered around the idea of loving someone who doesn't love you back, and the pain that comes with that.
who wrote capitalism and underdevelopment in latin america	Sure, I'd be happy to help! Based on the document provided, the book "Capitalism and Underdevelopment in Latin America" was written by Frank.

Performance Overview
Provides overview of human and algorithmic evaluations to help identify overall trends

Insight: Llama is considered the least fluent but most faithful model. It is also the most extractive

Model	Fluency	Answer Relevance	Faithfulness	Win Rate (Head-to-Head)	Σ Rank
Human	**3.96 ± 0.16 (1)**	3.73 ± 0.33 (2)	**3.39 ± 0.31 (1)**	**64.56 ± 22.74 (1)**	**5 (1)**
gpt3.5-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)

Model	Recall	Rouge-L	Bert-KPrec	Answerability (Accuracy)	Extractiveness
Human	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**
gpt3.5-turbo	0.5 (3)	0.35 (2)	0.41 (2)	0.92 (2)	0.28 (3)
llama-13b-chat	0.54 (2)	0.3 (4)	0.34 (4)	0.91 (3)	0.32 (2)
mistral	0.48 (4)	0.32 (3)	0.39 (3)	0.89 (4)	0.27 (4)

Model Behavior
Provides instance-level info on data slices to enable error analysis

Insight: By filtering for instances marked as unanswerable, with faithful responses, one can identify potential dataset errors and then inspect each of them to verify if they are indeed errors

Task	Answerability	gpt3.5-turbo
what was the purpose of the reichstag fire	UNANSWERABLE	Yes
song it's now or never by elvis presley	UNANSWERABLE	Yes

Model Comparator
Provides info on similarities between models

Insight: GPT-3.5 and Llama have different distributions w.r.t. extractiveness of their responses

H₀: gpt3.5-turbo and llama-13b-chat scores are derived from the same distribution. Reject the null hypothesis if $p < 0.05$

p-value 0.0084 Significant

Metric Behavior
Shows metric correlation to provide insights on metric definitions

Insight: Out of human metrics, algorithmic metrics are only moderately correlated with faithfulness, motivating need for human evaluation

Annotator Behavior
Provides information on annotator agreement to help get insights on the metrics and the annotation process

Insight: In general, fluency seems to be an easier metric for annotators to judge than answer relevance or faithfulness

Figure 2: Illustration of InspectorRAGet's core views and corresponding visualizations. Screenshots are drawn from the RAG model performance use case, described in Section 4.1.

3 InspectorRAGet Platform

InspectorRAGet is a React web application built with NextJS 13 framework[4]. We use the Carbon Design System[5] for the user interface. The experiment results are provided as a json input file that is loaded on the platform. This enables our platform to be lightweight; it can easily be run on virtual machines or even personal laptops/desktops. To enable privacy, InspectorRAGet is a stateless application and does not retain any uploaded datasets. InspectorRAGet's frontend offers a series of views (presented as separate tabs), each tailored to a different aspect of the analysis process. Excerpts of these views are shown in Figure 2. We describe each view along with hypotheses that can be formed, analyses it enables, and insights which may require further investigation.

3.1 Dataset Characteristics

The Dataset Characteristics view (Figure 2a) provides details regarding characteristics of the dataset such as answerability (e.g. answerable, unanswerable, partial), question type (e.g. factoid, comparative, explanation), and question length. These details can provide a quick snapshot of dataset trends, outliers, and potential biases.

3.2 Predictions Table

When analyzing an experiment, it helps to first see a few examples of instances. InspectorRAGet's predictions view shows a table of all questions with

[4] https://react.dev, https://nextjs.org
[5] https://carbondesignsystem.com

the respective model responses (Figure 2b). This view not only gets developers acquainted with the experiment but also helps them spot patterns in model responses (e.g., length, repetitions of text).

3.3 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 2c and 2d).

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 (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 further explored.

3.4 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 2e), as well as a sortable table of the actual instances (Figure 2f). Upon selecting an instance, users see a detailed view of the instance, including the conversation/question, context(s), the model responses and their respective evaluation scores (Figure 2g). 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.

3.5 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 2h). 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.

3.6 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 2i), hinting at issues with metric definitions.

3.7 Annotator Behavior

Finally, when human evaluation is performed, 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 2j). 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 2k), computed using Cohen’s kappa (Cohen, 1960)), annotator contribution (i.e., how often an annotator agreed with the majority) (Figure 2l), 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 Performance	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 Performance	MT-Bench	Alpaca-13B (Taori et al., 2023), Claude-v1 (Anthropic, 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 Evaluating InspectorRAGet

To showcase the value of InspectorRAGet, we evaluate its ability to yield actionable insights on two use cases: (1) *Analyzing RAG Model Performance:* 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) *Analyzing LLM-as-a-Judge Performance:* 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[6] (e.g. 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 experience.

4.1 Analyzing RAG Model Performance

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 Appen[7], 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) were 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 specific instances of disagreement,

[6]The use cases are at ibm.biz/InspectorRAGet

[7]https://www.appen.com/

shows that the most disagreement is 25 cases of Answer relevance. Opening up an example 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:** Investigate whether data inconsistencies should be improved or removed, and if further annotator training is needed.

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*, *Dataset Characteristics*. **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 Analyzing LLM-as-a-Judge Performance

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 (Zheng et al., 2023), a dataset of 80 high quality multi-turn questions. The MT-Bench authors compare LLM-as-a-judge with humans by releasing human judgments (provided by 58 experts) and algorithmic GPT-4 judgments on the MT-Bench dataset.

This INSPECTORRAGET use case is not centered around comparing model performance, but rather judge performance. The MT-Bench authors 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 available[8] 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 a more holistic analysis. Our evaluation shows INSPECTORRAGET's ability to yield concrete and actionable insights on two pertinent use cases - analyzing RAG model performance and LLM-as-a-judge performance. We publicly release our platform, input files used in this paper, and notebooks. In the future, we will explore augmenting INSPECTORRAGET to facilitate comprehensive pattern

[8]https://github.com/IBM/InspectorRAGet

discovery, as well as extending its 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 suggest 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

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.

Anthropic. 2023. Introducing Claude.

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.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

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, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée 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 Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, 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.

Shayne Longpre, Stella Biderman, Alon Albalak, Hailey Schoelkopf, Daniel McDuff, Sayash Kapoor, Kevin Klyman, Kyle Lo, Gabriel Ilharco, Nay San, Maribeth Rauh, Aviya Skowron, Bertie Vidgen, Laura Weidinger, Arvind Narayanan, Victor Sanh, David Adelani, Percy Liang, Rishi Bommasani, Peter Henderson, Sasha Luccioni, Yacine Jernite, and Luca Soldaini. 2024. The responsible foundation model development cheatsheet: A review of tools resources.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, 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, Simón 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 Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista 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 Cerón 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.

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 evaluation 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.

Zeqiu 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 the 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 their functional boundaries. The first section captures general details about the *experiment*, including its name, description, and timestamp. The second and third sections describe the sets of *models* and *metrics* used in the experiment, respectively. The fourth and fifth sections cover the dataset, in the form of a list of *documents/passages* that are used in the experiment and a list of *tasks*, each representing a data instance, which is composed of the user input (e.g., question/conversation), references to the corresponding documents/passages, and, optionally, a list of reference responses. Finally, the sixth section includes information about the *outcome of the evaluation*, in the form of the scores of the different evaluation metrics for each task.

Note, that as part of the experiment results file, one is not providing the implementation of the metrics and models used, but rather high-level *metadata* about them (e.g., the name of a metric and its scale - e.g., yes/no, numeric - and the name of a model) along with their *outputs* (i.e., the resulting evaluation scores and model responses). This separation of the evaluation experiment implementation and run from the analysis of the results, allows INSPECTORRAGET to be agnostic of the specific models or metrics used, thus allowing the analysis of diverse evaluation experiments.

A detailed description of the input format can be found on our GitHub repository [9].

B Sample Experiment Runners

To help users employ INSPECTORRAGET as part of the broader RAG evaluation life cycle, we have also released sample experiment runners, showcasing how to use INSPECTORRAGET in combination with popular evaluation frameworks. Each experiment runner is provided in the form of a Python notebook, which demonstrates how to use the corresponding evaluation framework to run an evaluation experiment and transform its output to the input format expected by INSPECTORRAGET for an analysis of the evaluation results. As of this writing, we have released notebooks demonstrating integrations of our platform with the following popular frameworks:

- *Language Model Evaluation Harness* [10] (Gao et al., 2023); a popular evaluation framework used to evaluate LLMs on different tasks.
- *RAGAs* [11] (Es et al., 2023); a popular evaluation framework specifically designed for the evaluation of RAG systems through LLM-as-a-judge techniques.
- *HuggingFace* [12] (Wolf et al., 2020), which offers libraries and assets (incl. datasets, models, and metric evaluators) that can be used to both create and evaluate RAG systems.

All experiment runners are available on the platform's GitHub repository.

[9] https://github.com/IBM/InspectorRAGet

[10] https://github.com/EleutherAI/lm-evaluation-harness

[11] https://github.com/explodinggradients/ragas

[12] https://huggingface.co