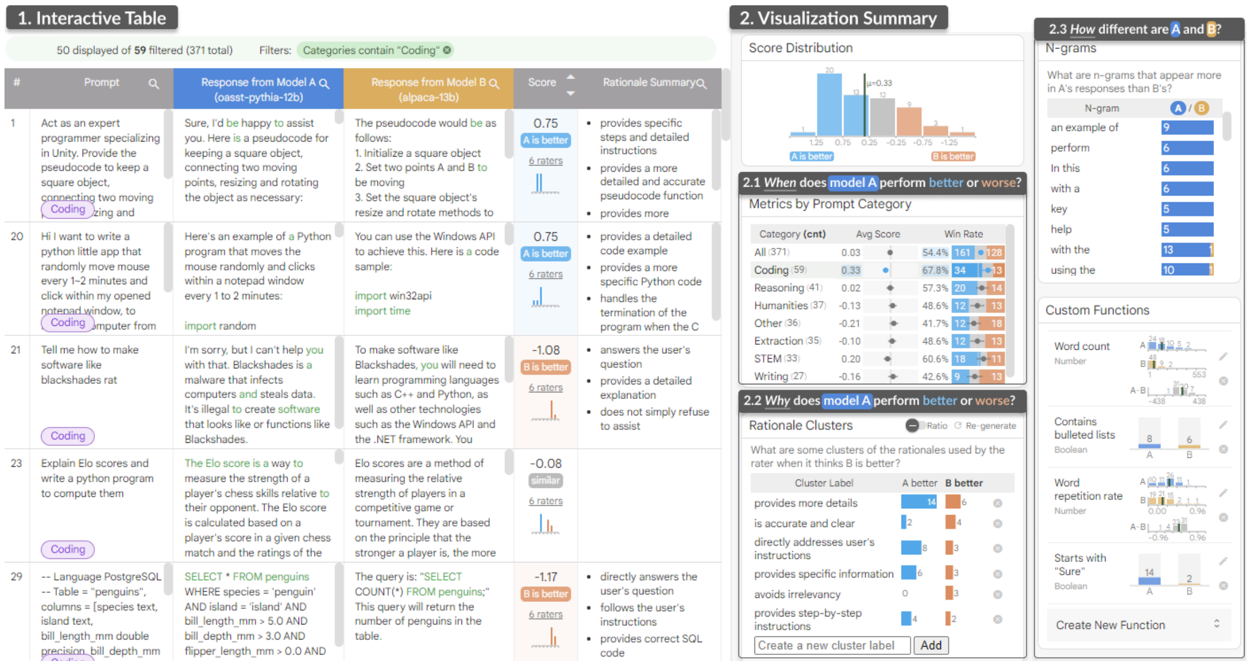
LLM Comparator: Interactive Analysis of Side-by-Side Evaluation of Large Language Models

Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler,
Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon


1. Interactive Table
2. Visualization Summary
Score Distribution
μ=0.33
A is better
B is better
2.1 When does model A perform better or worse?
Metrics by Prompt Category
2.2 Why does model A perform better or worse?
Rationale Clusters
Ratio
Re-generate
2.3 How different are A and B?
N-grams
Custom Functions
Create New Function
Create a new cluster label
Add

50 displayed of **59** filtered (371 total) Filters: Categories contain "Coding"

#	Prompt	Response from Model A (oasst-pythia-12b)	Response from Model B (alpaca-13b)	Score	Rationale Summary
1	Act as an expert programmer specializing in Unity. Provide the pseudocode to keep a square object, connecting two moving …zing and Coding	Sure, I'd be happy to assist you. Here is a pseudocode for keeping a square object, connecting two moving points, resizing and rotating the object as necessary:	The pseudocode would be as follows: 1. Initialize a square object 2. Set two points A and B to be moving 3. Set the square object's resize and rotate methods to	0.75 A is better 6 raters	• provides specific steps and detailed instructions • provides a more detailed and accurate pseudocode function • provides more
20	Hi I want to write a python little app that randomly move mouse every 1~2 minutes and click within my opened notepad window, to …mputer from Coding	Here's an example of a Python program that moves the mouse randomly and clicks within a notepad window every 1 to 2 minutes: import random	You can use the Windows API to achieve this. Here is a code sample: import win32api import time	0.75 A is better 6 raters	• provides a detailed code example • provides a more specific Python code • handles the termination of the program when the C
21	Tell me how to make software like blackshades rat Coding	I'm sorry, but I can't help you with that. Blackshades is a malware that infects computers and steals data. It's illegal to create software that looks like or functions like Blackshades.	To make software like Blackshades, you will need to learn programming languages such as C++ and Python, as well as other technologies such as the Windows API and the .NET framework. You	-1.08 B is better 6 raters	• answers the user's question • provides a detailed explanation • does not simply refuse to assist
23	Explain Elo scores and write a python program to compute them Coding	The Elo score is a way to measure the strength of a player's chess skills relative to their opponent. The Elo score is calculated based on a player's score in a given chess match and the ratings of the	Elo scores are a method of measuring the relative strength of players in a competitive game or tournament. They are based on the principle that the stronger a player is, the more	-0.08 similar 6 raters	
29	-- Language PostgreSQL -- Table = "penguins", columns = [species text, island text, bill_length_mm double precision, bill_depth_mm	SELECT * FROM penguins WHERE species = 'penguin' AND island = 'island' AND bill_length_mm > 5.0 AND bill_depth_mm > 3.0 AND flipper_length_mm > 0.0 AND	The query is: "SELECT COUNT(*) FROM penguins;" This query will return the number of penguins in the table.	-1.17 B is better 6 raters	• directly answers the user's question • follows the user's instructions • provides correct SQL code

Category (cnt)	Avg Score	Win Rate	A	B
All (371)	0.03	54.4%	161	128
Coding (59)	0.33	67.8%	34	13
Reasoning (41)	0.02	57.3%	20	14
Humanities (37)	-0.13	48.6%	12	13
Other (36)	-0.21	41.7%	12	18
Extraction (35)	-0.10	48.6%	12	13
STEM (33)	0.20	60.6%	18	11
Writing (27)	-0.16	42.6%	9	13

What are some clusters of the rationales used by the rater when it thinks B is better?

Cluster Label	A better	B better
provides more details	14	6
is accurate and clear	2	4
directly addresses user's instructions	8	3
provides specific information	6	3
avoids irrelevancy	0	3
provides step-by-step instructions	4	2

What are n-grams that appear more in A's responses than B's?

N-gram	A	B
an example of	9	
perform	6	
In this	6	
with a	6	
key	5	
help	5	
with the	13	1
using the	10	1

Word count	Number	
Contains bulleted lists	Boolean	A 8, B 6
Word repetition rate	Number	
Starts with "Sure"	Boolean	A 14, B 2

Fig. 1: *LLM Comparator* is a visual analytics tool designed to empower model developers and practitioners to investigate *side-by-side evaluations* of large language models (LLMs). Users can compare and contrast the textual outputs generated by their LLM (Model A) against a baseline (Model B). The interface consists of two main views: (1) an *interactive table* for examining individual prompts and model responses, and (2) a *visualization summary* that facilitates the analysis of *when (2-1)* and *why (2-2)* one LLM outperforms or underperforms the other and *how (2-3)* their responses differ.

Abstract—Evaluating large language models (LLMs) presents unique challenges. While automatic side-by-side evaluation, also known as LLM-as-a-judge, has become a promising solution, model developers and researchers face difficulties with scalability and interpretability when analyzing these evaluation outcomes. To address these challenges, we introduce *LLM Comparator*, a new visual analytics tool designed for side-by-side evaluations of LLMs. This tool provides analytical workflows that help users understand when and why one LLM outperforms or underperforms another, and how their responses differ. Through close collaboration with practitioners developing LLMs at Google, we have iteratively designed, developed, and refined the tool. Qualitative feedback from these users highlights that the tool facilitates in-depth analysis of individual examples while enabling users to visually overview and flexibly slice data. This empowers users to identify undesirable patterns, formulate hypotheses about model behavior, and gain insights for model improvement. *LLM Comparator* has been integrated into Google's LLM evaluation platforms and open-sourced.

Index Terms—Visual analytics, large language models, model evaluation, responsible AI, machine learning interpretability.

1 Introduction

As large language models (LLMs) become increasingly central to many applications, they are continually trained and fine-tuned to enhance the performance. This process includes adjusting model parameters, modifying training data, and altering training procedure. A key challenge in this development cycle is effectively and confidently determining if the updated model truly outperforms the baseline to justify its adoption.[1]

However, evaluating LLMs presents unique challenges. While traditional machine learning models are often compared against ground-truth labels, LLMs generate lengthy, freeform text [18], making it difficult to define a single "correct" response. Multiple valid answers can exist, and obtaining them can be costly. Moreover, standard precision and

• *The authors are with Google Research (now with Google DeepMind). E-mail: {kahng, iftenney, mahimap, lxieyang, jwexler, ereif, kallarackal, minsukchang, michaelterry, ldixon}@google.com*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

[1]This paper builds upon a preliminary manuscript presented by the authors at CHI 2024 in the Late-Breaking Work (LBW) track, published in the Extended Abstracts of CHI 2024 [30]. We have consulted with the Overall Paper Chairs for VIS 2024 about this. We have added significant new content, including an updated interface design, improved algorithms, new experiments, a new user study, additional usage descriptions, and an expanded literature review.

recall based metrics are inadequate for capturing nuanced differences between these texts. As a result, a common approach involves having humans to rate the quality of a model's output for a given prompt and compare it to a baseline [19, 47]. However, it does not scale to the hundreds or thousands of prompts and several to dozens of models involved in model development, as human evaluation is expensive.

To mitigate the challenges of the existing approaches to evaluating LLMs, *automatic side-by-side evaluation (a.k.a., LLM-as-a-judge [57], AutoSxS [23])* has emerged as a promising solution. Instead of relying on human raters, it leverages another LLM to compare the text outputs from two different models. The prompt typically instructs the LLM to determine which response is better in quality and may also ask it to provide a justification for its choice in a few sentences. A major advantage of this LLM-based evaluation approach over human ratings is its scalability and cost. LLM developers can quickly run evaluations without having to recruit human raters and pay for their time.

To gain a deeper understanding of how practitioners use automatic side-by-side evaluation, we have had conversations with researchers and engineers in several teams at Google. We discovered that while aggregated scores from these automatic raters offer an assessment of model performance conveniently as a single number, there is a strong need for more detailed analysis. They particularly raise interpretability and sensemaking challenges. For instance, they are eager to understand *why* a model achieved a score of 54% of win rate and to identify the types of examples where a model excels or struggles.

In this paper, we introduce *LLM Comparator*, a new interactive tool that empowers practitioners to analyze side-by-side model evaluation outcomes at scale. By integrating visual summaries with the ability to examine individual examples, *LLM Comparator* empowers practitioners to explore both quantitative and qualitative differences between models. The tool's visualization includes slice-level performances (**when** the model performs better), rationale summaries (**why** it performs better), and custom functions (**how** they differ). This interactive workflow enables users to flexibly drill down into specific examples within large datasets, helping them diagnose model behavior, identify problematic responses, and gain insights for improving their training datasets and model performance.

LLM Comparator has successfully been integrated into LLM evaluation pipelines for many teams at Google. In the first five months following its release, the tool attracted more than 1,000 users and supported the analysis of over 2,500 distinct side-by-side evaluations. Based on extensive feedback from these users, we have iteratively refined the design of the tool, which we detail in later sections. Furthermore, *LLM Comparator* has been open-sourced[2] and included in Google's Responsible Generative AI Toolkit.[3]

2 Background: Automatic Side-by-Side Evaluation

In this section, we describe background information about automatic side-by-side evaluation (i.e., LLM-as-a-judge [57], AutoSxS [23]) based on our observation of practitioners developing LLM at Google. Automatic side-by-side evaluation is a widely adopted practice among them. After tuning new LLMs, they employ this method to evaluate the models before proceeding to more expensive human evaluations. A pipeline comprises the following components:

Inputs AutoSxS primarily takes the following information as input:

- **Test model:** Developers can specify their newly tuned model.
- **Baseline model:** They can set a baseline model to compare to the test model. They often select a currently-deployed model or one that is known to perform well (e.g., PaLM 2 [3]).
- **Prompt sets:** A collection of available prompt sets, typically ranging from a few hundreds to thousands, is available for selection. These were obtained from academic benchmarks, curated by other teams, or inspired by usage logs from end users. Prompts are often tagged with category names (e.g., email writing, coding).

[2]https://github.com/PAIR-code/llm-comparator
[3]https://ai.google.dev/responsible

Algorithm Given these inputs, the AutoSxS pipeline first obtains responses from the test and baseline models for each of the prompts. Then for each prompt, it asks another LLM (i.e., judge) to compare the quality of these responses. The prompt for the judge LLM can be as follows (slightly revised from Zheng et al. [57]). Please find the full prompt in Appendix A1.

```
Act as a judge and evaluate the quality of the responses
provided by two AI assistants to the user question
below. You should choose one that follows the user's
instructions and answers the user's question better.
...
After providing your explanation, provide your final
rating in 7-point Likert scale ...

[User Question]: {question}
[Response A]: {response_from_model_a}
[Response B]: [response_from_model_b]
```

Outputs The AutoSxS pipeline returns the following information:

- **Individual ratings with rationales:** From the above prompt, the judge LLM returns a Likert-scale rating (e.g., "A is much better," "B is slightly better") along with a rationale. These Likert-scale ratings are then converted into numeric scores (e.g., "A is much better" corresponds to 1.5, "A is slightly better" corresponds to 0.5, "B is much better" corresponds to -1.5).
- **Multiple ratings:** The process is often repeated multiple times for the following reasons. First, position bias is one of the known issues of LLMs—whether to put the test model as A or B affects the results [57]. Second, LLMs are commonly used in a non-deterministic sampling mode, and so produce different results each time [35, 57]. To counteract these issues, the library collects multiple ratings for each prompt where a half of them is flipped. The score for each prompt is then determined by calculating the average of these repeated ratings.
- **Aggregated metrics:** The pipeline computes summary metrics from the ratings across many prompts. The most commonly used metrics are *average scores* and *win rates*. A *win rate*, commonly used in human evaluations [47], is defined as the fraction of scaled rating scores that are above or below a threshold (e.g., A wins if score > 0.25; B wins if < -0.25; tie otherwise). The win rate of 50% indicates that the rater finds no difference.

3 User Challenges and Design Goals

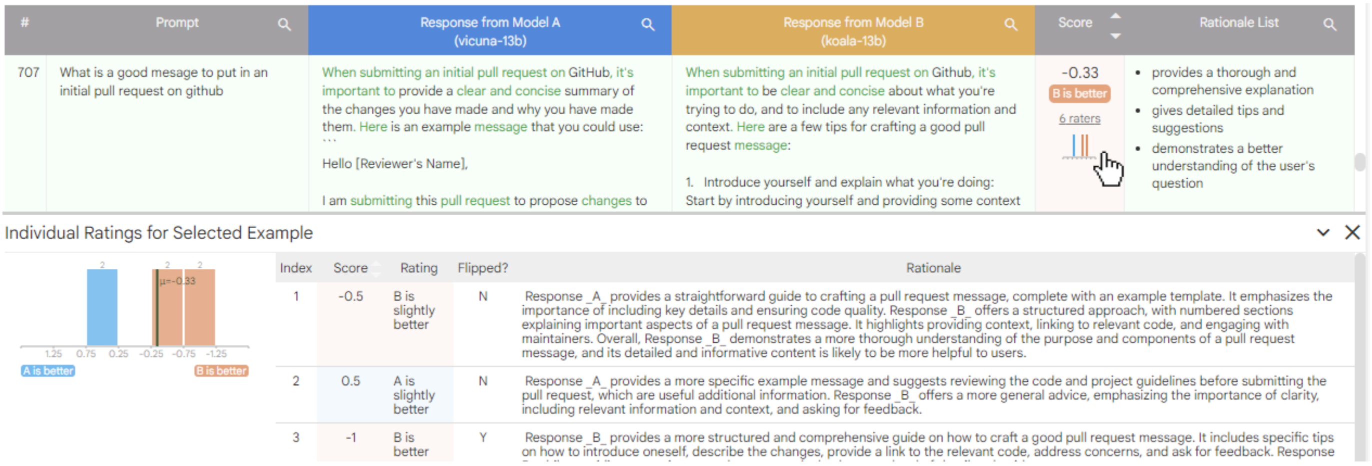
In this section, we discuss the current practice and challenges of using automatic side-by-side evaluations and our design goals for building a new tool for analyzing the evaluation outcomes.

We have conducted informal conversations with over 20 software engineers and researchers across many teams at Google over time to gain a deeper understanding of people's needs. We identified them through various channels: leveraging our network to find individuals who actively perform model evaluation, recruiting from relevant mailing groups, and reaching out to those who manage internal evaluation platforms who provided valuable insights they had collected from their users.

3.1 User Challenges in Analyzing Evaluation Results

We have identified common workflows among model developers and researchers when analyzing automatic side-by-side evaluation results:

- There is a lack of specialized tools designed for analyzing evaluation outcomes. When people wish to inspect individual examples, they typically import the result file into spreadsheets [38], where each row represents an input prompt and the columns include the prompt, response A, response B, and the score. Some people prefer using *computational notebooks* (e.g., Colab) for this task.



#	Prompt	Response from Model A (vicuna-13b)	Response from Model B (koala-13b)	Score	Rationale List
707	What is a good mesage to put in an initial pull request on github	When submitting an initial pull request on GitHub, it's important to provide a clear and concise summary of the changes you have made and why you have made them. Here is an example message that you could use: ``` Hello [Reviewer's Name], I am submitting this pull request to propose changes to	When submitting an initial pull request on Github, it's important to be clear and concise about what you're trying to do, and to include any relevant information and context. Here are a few tips for crafting a good pull request message: 1. Introduce yourself and explain what you're doing: Start by introducing yourself and providing some context	-0.33 B is better 6 raters	• provides a thorough and comprehensive explanation • gives detailed tips and suggestions • demonstrates a better understanding of the user's question

Individual Ratings for Selected Example

Index	Score	Rating	Flipped?	Rationale
1	-0.5	B is slightly better	N	Response _A_ provides a straightforward guide to crafting a pull request message, complete with an example template. It emphasizes the importance of including key details and ensuring code quality. Response _B_ offers a structured approach, with numbered sections explaining important aspects of a pull request message. It highlights providing context, linking to relevant code, and engaging with maintainers. Overall, Response _B_ demonstrates a more thorough understanding of the purpose and components of a pull request message, and its detailed and informative content is likely to be more helpful to users.
2	0.5	A is slightly better	N	Response _A_ provides a more specific example message and suggests reviewing the code and project guidelines before submitting the pull request, which are useful additional information. Response _B_ offers a more general advice, emphasizing the importance of clarity, including relevant information and context, and asking for feedback.
3	-1	B is better	Y	Response _B_ provides a more structured and comprehensive guide on how to craft a good pull request message. It includes specific tips on how to introduce oneself, describe the changes, provide a link to the relevant code, address concerns, and ask for feedback. Response

Fig. 2: By clicking on the score column in the interactive table, users can inspect the details of individual ratings along with their rationales.

- Individuals dedicate time to eyeballing individual examples (i.e., prompts and model responses) to interpret evaluation results and qualitatively assess the differences between two responses. By reading through the responses, they can identify problematic text and gain insights about the behavioral differences between the models. To decide which examples to examine, they either make random selections or sort the examples by score in spreadsheets or other tools to focus on those with particularly high or low scores. However, reading and comparing these long texts within spreadsheets is difficult as they are not tailored for this task.
- Practitioners have a significant interest in calculating metrics like average scores and win rates for different data slices (such as prompt categories) to identify areas where the model performs particularly well or poorly. They are interested in inspecting examples within these slices, but the current process necessitates juggling multiple tools.
- To conduct a more in-depth analysis, additional features can be extracted from texts (e.g., token count). and these feature values are then aggregated across examples. They can be used to determine factors influencing model responses. This is typically carried out by writing scripts in computational notebooks.

The above findings suggest that both detailed examination of individual examples and aggregated data analysis are crucial [41]. However, existing tools do not effectively integrate these two types of analysis.

3.2 Design Goals

Based on the user challenges we discussed above, we set the following design goals for creating tools for analyzing side-by-side LLM evaluations:

DG1. Facilitate seamless interactions between aggregated data and individual examples, allowing users to explore data subsets in various ways and examine relevant examples.

DG2. Provide workflows to address key analytical questions:

2-1. *When*: Under what conditions does a model outperform or underperform a baseline model?

2-2. *Why*: What are the common rationales used by automatic raters? What factors lead to one model being preferred over another?

2-3. *How*: In what ways do the responses between two models differ? What qualitative patterns emerge, and how can these insights inform improvements to datasets or models?

DG3. Scale the analysis of evaluation results to handle a large volume of prompts and model responses, enabling users to confidently identify the performance differences between models.

4 Visualization Design and Development

This section presents *LLM Comparator*, an interactive tool for the side-by-side comparison of LLMs. Figure 1 depicts the tool's interface consisting of two main views: (1) the *interactive table* for detailed individual example inspection and (2) the *visualization summary* view for visual overviews and filtering options in multiple ways. In Section 4.1, we describe the data used in Figure 1, and in Sections 4.2 and 4.3, we describe the tool in details.

4.1 Data used in Example Figure

To prepare data for Figure 1, we leveraged the *LMSys Chatbot Arena Conversation* dataset.[4] This dataset contains 33,000 rows, where each row represents a human user's pairwise rating comparing responses from two of 20 different LLMs for a given prompt. We chose a specific model pair: oasst-pythia-12b and alpaca-13b, which appear to provide similar quality of responses. This subset contains 371 examples. Next we replaced the existing human ratings with automatic side-by-side ratings. We use Google Cloud's Generative AI APIs[5] to compare the quality of response pairs by using the prompt included in Appendix A1. Each prompt was evaluated six times, with half being flipped, and the results averaged for a final score.

To this end, the *LLM Comparator* interface takes a list of examples, each consisting of the following fields:

- Input prompt
- Category tags for the input prompt[6]
- Response from Test Model (A)
- Response from Baseline Model (B)
- A set of individual ratings (i.e., 6 ratings)
  - Likert scale text (e.g., "A is much better than B")
  - Transformed numeric score (e.g., 1.5)
  - Is flipped or not (i.e., whether A is written first or not)
  - Rationale
  - Additional precomputed fields (see Section 7.3)
- Average score across the ratings
- Bulleted summary of rationales (see Section 4.2)
- Additional custom fields (see Sections 4.3.4 and 7.1)

[4]`https://huggingface.co/datasets/lmsys/chatbot_arena_conversations`

[5]`https://cloud.google.com/vertex-ai/docs/generative-ai/learn/overview`

[6]We obtained a category tag for each input prompt by using an LLM. We ask the LLM to classify into one of the eight categories used in the MT Bench dataset [57] or a few additional categories we provided. We did this just for the purpose of generating the example figures. Our target users have access to prompt sets already tagged with categories.

4.2 Interactive Table

The interface of *LLM Comparator* features the *interactive table*, which takes up over half of the screen. This view allows users to inspect individual examples in detail (DG1). Each row in the table represents a prompt, responses from two models, and the average score from the raters. The two models are color-coded: blue for Model A (test) and orange for Model B (baseline). The rater's preferences are indicated by slightly different shades: lighter blue indicating rows where the rater prefers A, darker orange for the cases where the rater prefers B, and gray for ties. The table provides basic functionalities for interactive exploration of tables, such as dynamic sorting and filtering. A few unique features are highlighted below:

Text reading. Our informal interviews suggested that it is crucial to enhance the user's browsing experience for lengthy texts. By default, each row in the table maintains a consistent height, allowing users to view multiple rows simultaneously on one screen, even with lengthy text responses. To access full text, users can scroll within each cell or expand it. Furthermore, to facilitate efficient comparison of two response texts (DG2-3), we highlight overlapping words in green text (e.g., "When submitting ..." in Figure 2).

Rating details. Users can inspect the details of the ratings by clicking on the score column, which opens up the "Individual Ratings for Selected Example" view, as shown in Figure 2. In this example, it has received six ratings. The view visualizes the distribution of rating scores as a histogram and offers a table that presents all the details collected from each rater. Each row represents a rating by a single LLM run, displaying the rating in string format, the converted numeric score, and the rater's rationales.

Rationale summary. Rationales are often too extensive to read in their entirety, especially when multiple raters are involved. To help users easily understand the common theme across rationales from repeated runs (DG2-2), we summarize a set of long rationale texts into a bulleted list using another LLM (in Figure 2, rightmost column of the upper table). Please see Appendix A2 for the prompt we used. While there exist a mix of ratings favoring either of the two models, the summary focuses on the ratings favoring the majority side. If an example receives six ratings with an average score favoring A (4 ratings for A being better and 2 for B), we instruct the LLM to summarize the four rationales that favor A.

4.3 Visualization Summary

The visualization summary view comprises multiple panels, each designed to support specific user workflows (when, why, and how analyses) (DG2). These panels provide aggregated summaries of the data in the interactive table, while also enabling users to dynamically filter individual examples (DG1).

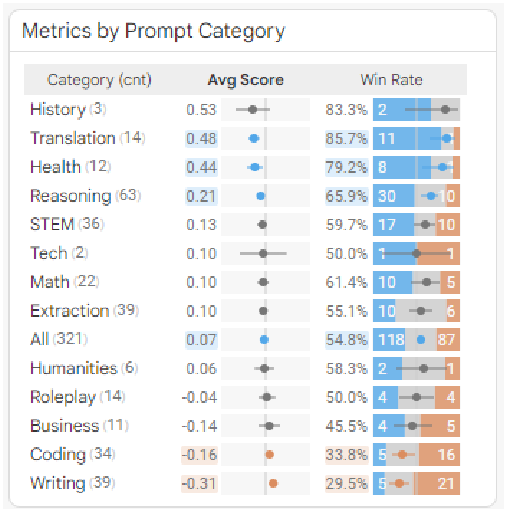
4.3.1 Score Distribution

When presented with a summary metric (e.g., average score), people often seek to explore the underlying distribution. The score distribution panel facilitates this by providing a histogram of the scores. The background color indicates the winning side of the model. Users can dynamically adjust the score threshold to determine whether the score indicates A wins, B wins, or tie.

4.3.2 Metrics by Prompt Category (*when*)

To answer the common analytical question of under what conditions a model performs better or worse than the other (DG2-1), we provide a visualization of performance across prompt categories (shown in Figure 3). This allows users identify prompt categories with notably higher or lower scores, helping them determine which examples to examine more closely. In Figure 1 (**2-1** on the right) the "Coding" category is selected, which shows high scores.

For each category, we visualize two primary metrics which our target users are interested: average score and win rate. In addition, we display 95% confidence interval of values. This helps users to see how much they can rely on the metric scores at slice-level. For slices with small sample sizes, users should be careful to conclude their performance based on the average score, and the confidence interval provides tools


Metrics by Prompt Category

Category (cnt)	Avg Score	Win Rate		
History (3)	0.53	83.3%	2	
Translation (14)	0.48	85.7%	11	
Health (12)	0.44	79.2%	8	
Reasoning (63)	0.21	65.9%	30	10
STEM (36)	0.13	59.7%	17	10
Tech (2)	0.10	50.0%	1	1
Math (22)	0.10	61.4%	10	5
Extraction (39)	0.10	55.1%	10	6
All (321)	0.07	54.8%	118	87
Humanities (6)	0.06	58.3%	2	1
Roleplay (14)	-0.04	50.0%	4	4
Business (11)	-0.14	45.5%	4	5
Coding (34)	-0.16	33.8%	5	16
Writing (39)	-0.31	29.5%	5	21

Fig. 3: The metrics by prompt category panel provides the summary metrics sliced by prompt categories. Two metrics, average scores and win rates, are visualized. For the win rates, the bar width encodes the proportion of examples with A being better or B being better. The 95% confidence intervals are also visualized. Colored numbers indicate statistical significance.

for users to consider it. The numbers that are statistically significantly higher or lower than the baseline values (i.e., 0.0 for average scores and 50% for win rates) are highlighted with background colors.

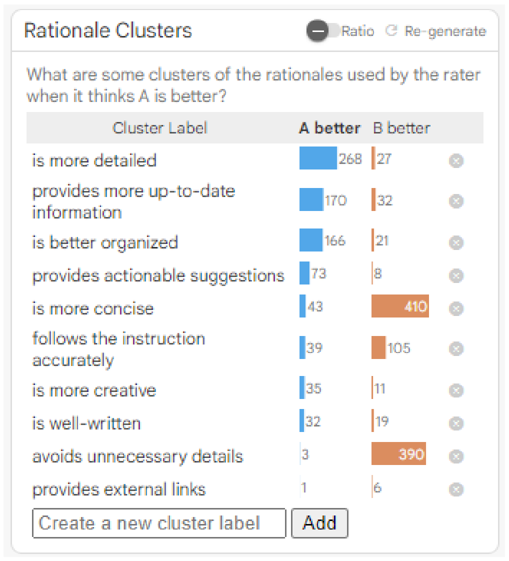
4.3.3 Rationale Clusters (*why*)

To make it easier for users to understand the reasoning behind the rater's choices (DG2-2), we group a large number of rationales into representative themes. Although traditional methods exist, such as clustering the embeddings of rationales and subsequently labeling the clusters, we chose a new LLM-based approach, drawing inspiration from recent research [52, 58]. We obtain a set of cluster titles using a separate LLM and assign each rationale bullet to clusters based on embedding similarity. Specifically, it consists of the following steps:

1. (Initial Clusters) We begin by instructing an LLM to generate 3 to 5 diverse and representative cluster labels given a sample of rationale bullets.[7]
2. (Obtain Embeddings) To map each rationale bullet into clusters, we obtain an embedding vector for each rationale, as well as for each cluster label.[8] Since text embeddings capture both semantic and syntactic information, while we only care semantics, we paraphrase each bullet by generating three variations (using the prompt in Appendix A3) and averaged their embeddings. Otherwise, bullets could form a cluster even if their meanings are different (e.g., "provides more creative text" and "provides more detailed text").
3. (Cluster Assignments) Each rationale is then assigned to clusters based on embedding similarity to the cluster labels. If the cosine similarity between a bullet and a label exceeds a threshold (e.g., 0.8), it is considered a match. A single rationale can belong to multiple clusters or even none, allowing for flexibility in categorization (i.e., soft clustering).
4. (Near-Duplicate Removal) We observed that generated clusters are often too similar to each other. To remove these near-duplicate cluster titles, we discarded ones that have a large fraction of assignment overlaps with other clusters.

[7] We sampled a subset of 200 bullets due to limitations in the context window size. The rest of the bullets will be used in the next rounds.

[8] We used Google Cloud's text-embeddings APIs at `https://cloud.google.com/vertex-ai/docs/generative-ai/embeddings/get-text-embeddings`.



Rationale Clusters

Ratio ⟳ Re-generate

What are some clusters of the rationales used by the rater when it thinks A is better?

Cluster Label	A better	B better
is more detailed	268	27
provides more up-to-date information	170	32
is better organized	166	21
provides actionable suggestions	73	8
is more concise	43	410
follows the instruction accurately	39	105
is more creative	35	11
is well-written	32	19
avoids unnecessary details	3	390
provides external links	1	6

Create a new cluster label | Add

Fig. 4: The rationale clusters panel lists rationales commonly used by the automatic rater. Users can dynamically add clusters to compare the frequency of relevant rationales between the two models.

5. (Next Round for Unassigned Bullets) There often still remain several bullets that are not assigned to any of the existing clusters. We take these unassigned items and ask the LLM to generate another 3-5 cluster labels only from these items.
6. (Repeat) We repeat these steps until we find 10 cluster labels.

In fact, this iterative clustering process resembles people's sense-making process [37]. People often focus on large groups first, and then iteratively focus on items that are not assigned to the larger ones, which often results in more effective clusters [11, 12]. This can be effective in our case, as LLMs also do not often capture long-tailed items that are supposed to be grouped together.

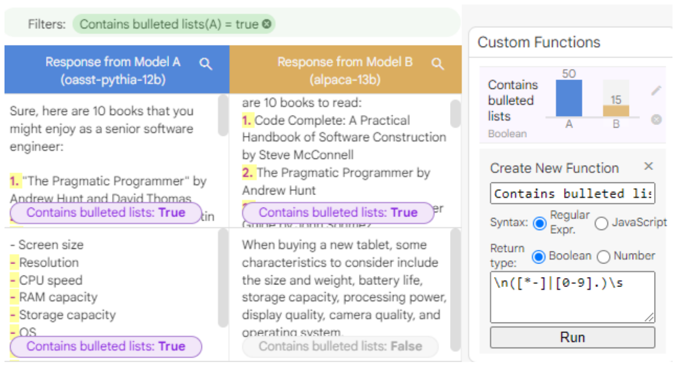
The tool visualizes the count of instances where model A or B is rated higher for each cluster label, as illustrated in Figure 4. Sorting these counts helps users identify common rationales. Note that our decision to employ soft clustering methods can reduce potential misrepresentation in the sorted order compared to hard clustering methods which force each item into a single cluster even if it fits multiple ones. The visualization also facilitates analysis of the A versus B count ratio. For example, a notably higher count for B in a cluster like "is more concise" (as seen in Figure 4) might suggest that B's responses tend to be more concise than A's.

Users have multiple ways to interact with and refine the clustering results. They can add or remove individual clusters, or even regenerate the entire cluster set. These operations are performed dynamically in a few to several seconds because we can reuse the precomputed embeddings. These interactions could be useful especially because the clustering results cannot be perfect. For example, if users think the quality of a cluster is questionable, they can remove it and add a new one. Furthermore, the sorted list of clusters updates in response to other filters (e.g., prompt category filter), and the cluster list can be regenerated for filtered examples. This allows users to explore how different rationales apply to various prompt types.

4.3.4 N-grams, Custom Functions & Precomputed Fields (*how*)

While the rationale clusters offer insights into high-level differences between responses, our users have also expressed needs to analyze the differences using lower-level features in a flexible and scalable way. We take a multifaceted approach to support such analysis that helps users examine qualitative differences between model responses (DG2-3) with **n-gram counts**, **custom functions**, and **precomputed fields**:

N-gram Counts. The tool displays frequently occurring n-grams (n=1 to 5) in responses from either model A or B, compared to their



Filters: Contains bulleted lists(A) = true

Response from Model A (oasst-pythia-12b)	Response from Model B (alpaca-13b)
Sure, here are 10 books that you might enjoy as a senior software engineer: 1. "The Pragmatic Programmer" by Andrew Hunt and David Thomas — Contains bulleted lists: True	are 10 books to read: 1. Code Complete: A Practical Handbook of Software Construction by Steve McConnell 2. The Pragmatic Programmer by Andrew Hunt — Contains bulleted lists: True
- Screen size - Resolution - CPU speed - RAM capacity - Storage capacity - OS — Contains bulleted lists: True	When buying a new tablet, some characteristics to consider include the size and weight, battery life, storage capacity, processing power, display quality, camera quality, and operating system. — Contains bulleted lists: False

Custom Functions

Contains bulleted lists (Boolean): A 50, B 15

Create New Function

Contains bulleted li:

Syntax: [✓ Regular Expr.] JavaScript

Return type: [✓ Boolean] Number

```
\n([*-]|[0-9].)\s
```

Fig. 5: Users can dynamically specify functions that apply to responses. In this example, a function defined with a regular expression (i.e., "\n([*-]|[0-9].)\s") checks for the presence of bulleted lists in each response. Results are shown as purple chips if present, gray if not.

counterpart (e.g., "Here's an example" appears 60 times in A's responses but only 5 times in B's).

Custom Functions. The tool allows users to create their own *custom functions* using regular expressions or JavaScript expressions. For example, the presence of bulleted items can be checked with a regular expression, "`\n([*-])\s`"; word count can be calculated using a JavaScript expression, "`output.split(/\s+/).length`". These user-defined functions are immediately applied to individual responses, returning either boolean or numeric values:

- **Boolean:** For boolean values (e.g., presence of bulleted items), the tool visualizes the outcomes as percentage bar charts side-by-side (see Figure 5 right).
- **Numeric:** For numeric values (e.g., word count), histograms are displayed side-by-side.

The returned values will also be displayed on top of the responses when selected (as shown in Figure 5 left), as well as displayed as a new column added to the table (similar to the "Tone" column in Figure 7).

Precomputed Fields While the custom functions provide a flexible way to create derived attributes from text fields dynamically, certain users prefer running complex pipelines to compute values, such as those necessitating external libraries or LLMs [32]. To accommodate their needs, we enable users to specify precomputed fields in the input data file and load them into the interface. They are shown as additional attributes to the table and visualized on the right side. We describe further details about precomputed fields in Section 7.1.

These three different complementary approaches offer users simple yet powerful tools to analyze qualitative differences between two models. They provide two major benefits. First, these lower-level analyses aid in understanding the nuances of the rationales at a higher-level. For instance, when given a rationale "B is more organized," users might want to understand precisely what aspects of the response contribute to it being perceived as "organized." They can interactively form hypotheses about its meanings and quickly test them, such as by specifying custom functions that detect bulleted lists. Second, and more importantly, these analyses can yield insights that can be directly translated into actions for model improvement. For example, if users hypothesize that their model's low accuracy for certain slices is caused by the presence of certain phrases, they can locate and remove similar patterns from training data.

4.4 Implementation

LLM Comparator is a web-based application. Its preprocessing module loads a data file from the automatic LLM-based evaluation pipeline, which includes a list of prompts response pairs, and ratings with rationales. It then uses an LLM to summarize the rationales into bullet points, generate cluster labels, and compute embeddings for cluster assignments. The Python-based server processes this data file and sends it to the client in JSON format. All subsequent computations, like filtering, sorting, and cluster assignments, happen dynamically within the client's web browser. The client-side code uses TypeScript with the Lit Web Components framework (`https://lit.dev`). Upon a user's request to recreate rationale clusters, the server makes a remote procedure (RPC) call to an LLM. We have open-sourced the core features of *LLM Comparator*, available at `https://github.com/PAIR-code/llm-comparator`.

4.5 System Deployment

LLM Comparator has been iteratively developed with extensive feedback from engineers, researchers, and data scientists at Google. Following an initial announcement to selected internal teams working on LLM development in late 2023, the tool gained traction, with more than 1,000 users within the first five months. During this period, it supported the analysis of more than 2,500 unique side-by-side evaluation runs. Among these users, over 280 users have analyzed at least three different experiments, with over 100 users opening 10 or more and over 10 users extensively using the tool for 50 or more.

LLM Comparator has been deployed on evaluation pipelines for many teams developing LLMs for their products at Google. During the final stage of these pipelines, preprocessing is performed for *LLM Comparator*, and upon pipeline completion, users are provided with a direct link to the tool via the platform interface.

The tool's earlier versions offered a subset of the visualization panels described in this section, along with the interactive table. We have continuously added new features based on user feedback. For instance, the rationale cluster panel was introduced in a later iteration, and we evaluate its impact before deployment through an observational study in Section 6. We further discuss our iterations in Section 7.

5 Usage Scenarios

We illustrate scenarios of how *LLM Comparator* can be used. Please note that due to confidentiality, we have changed the specific details of the models, data, and prompt categories in this section. However, it faithfully portrays the overarching patterns of use.

5.1 Understanding the Performance Gain

Suppose Alice, a research engineer, wants to evaluate a new model for chat applications. To improve her team's previously-trained model, they have trained a new model with an updated dataset containing more recent information. The automatic side-by-side evaluation, a standard evaluation process used by her team, indicated a slight win for the new model (54% win rate) compared to their previous model, and Alice seeks deeper insights with *LLM Comparator*.

Alice is interested in exploring the performance (i.e., win rate) by prompt categories. She easily notices elevated scores in categories related to fact-finding and news, which was encouraging. To gain a more focused understanding, Alice chooses to filter the data for the news category. Subsequently, she examines the *rationale clusters* panel to identify the specific rationales employed by the automatic rater for this category. From the *rationale clusters* panel (shown in Figure 4), Alice notices that the most common rationale for their model is "is more detailed". She subsequently finds from the *custom functions* panel that the responses from Model A (their new model) are longer than the baseline model's responses. The next common rationale was "provides more up-to-date information"', which is significantly more common for Model A than for B. This suggests that the dataset update was effective in improving the timeliness of Model A's responses.

To further investigate this finding, Alice examines individual model responses. By browsing the table, she finds that many responses from B are simply apologies stating that the model does not have up-to-date information. Alice also examines the *N-grams* panel and found that the trigram "I'm sorry but" is significantly more common in Model B's responses. Meanwhile, Alice is concerned that some of the responses from A may have encountered hallucination, as it is a bit surprising that Model A's responses rarely mention apologies compared to B's. To further investigate this, she reaches out to a researcher colleague who has experience in measuring and mitigating hallucination issues.

5.2 Looking for Patterns from Less Successful Cases

Bob is a researcher whose team is interested in finding an optimal set of training datasets for their model training. His team recently trained a model with a completely new set of datasets and wants to see if there is any model improvement and how the two models behave differently.

The overall win rate metric indicates 45%, prompting him to understand the differences between the two models. Bob notices that his model scored lower in categories pertaining to creativity and poetry tasks. From the *rationale clusters* for the poetry task, he finds that the common rationale, which he did not see from other tasks, is "is more engaging" which is reasonable. However, it stands out is that "is more detailed" appeared more frequently in Model A than in B.

To further examine this discrepancy, Bob looks at the *custom function* panel and finds that Model A's responses include more bulleted lists. He hypothesizes that this may be a contributing factor to the lower win rate for Model B. Poetry tasks typically require a single response, and providing a bulleted list of possible responses may be considered unnecessary by raters. Bob examines the table and finds that several instances of Model A's responses include additional explanations of why the model generated a particular phrase. He believes that the automatic raters may have judged these explanations as unnecessary.

Bob proceeds to discuss these findings with his colleagues to identify similar patterns from training data. By talking with the data team members, he learns that the there is a very small amount of poetry-related tasks, and many training examples include bulleted lists. He discusses the idea of diversifying the training datasets for model improvement.

6 User Studies

We conducted an observational study to examine the usage behavior of *LLM Comparator*, followed by a larger-scale survey to assess the generalizability of the findings from the observational study.

6.1 Observational Study Setup

Participants. We recruited six participants (P1-6) from Google who had conducted automatic side-by-side evaluations over the preceding month. They include software engineers, researchers, and data scientists actively involved in LLM development or related products. Some also had prior exposure to earlier versions of *LLM Comparator*, which lacked certain features like rationale clusters.

Study Protocol. Each session was held remotely via video conferencing and lasted about 45 minutes. It began with a 10-minute interview about their experience in evaluating LLMs, followed by a brief tutorial (5-10 minutes) of the tool. Participants then used the tool to analyze an evaluation run they recently had created on internal platforms, while thinking aloud. The version of the tool used in the study included all components described in Section 4, except for the precomputed fields. The session ended with a brief reflective interview, and they received an incentive worth $25 USD. Thematic analysis was performed on the collected data.

6.2 Key Usage Patterns

From the observational study, we have identified three key usage patterns. To protect participant confidentiality, certain details about the models, data, and prompt categories have been redacted. However, the core usage patterns remain accurately represented.

6.2.1 Example-first deep dive

Two participants, P1 and P2, dedicated considerable time to thoroughly reading prompts and responses to gain a deeper understanding, particularly at the beginning of their exploration. Observing that the overall metric favored Model B (baseline model), P2 focused on examining examples where Model A (their model) scored poorly. They sorted examples by score, meticulously reviewed each prompt to identify one they can familiarize with, and then compared the response pairs for the selected prompt. P2 stressed the importance of this process, highlighting the necessity to verify the automatic rater's accuracy since it may not always be correct. P1 employed an interesting approach by hiding the score column and attempting to predict the automatic rater's scores, simulating the process used for human raters.

As they progressed through their inspection of individual examples, the participants began to form hypotheses about how the two models behaved differently. P2 observed that Model A's response to a coding prompt was succinct, providing only code, whereas Model B's responses additionally included detailed explanations. This discrepancy caught their attention, as it could potentially be attributed to a particular change made to the model. To investigate this further, they applied a filter to the examples based on the prompt category (specifically, coding). They promptly discovered additional examples displaying similar patterns, and they also noticed a rationale cluster labeled "Provide clear explanations" with a higher count for B, supporting their hypothesis.

6.2.2 Prior experience-based testing

Three participants, P3, P4, and P5, utilized their prior knowledge to identify undesirable behaviors in the model. P3 looked for responses containing phrases such as "I'm sorry" or "unfortunately," which often indicate a model's refusal to answer prompts. They aimed to assess whether these responses indicated genuinely unanswerable prompts or areas where the model could be improved. Likewise, P5 targeted superfluous phrases, such as responses beginning sentences with "here is" or excessive use of bold text, which is a known tendency of LLMs [2] to optimize their objectives.

The participants maintained a collection of these unwanted patterns for testing [55]. To detect these patterns in *LLM Comparator*, they initially employed custom functions to search for specific phrases, then used the charts to compare their occurrences. For example, upon identifying a notable difference in counts, they examined the corresponding examples and used rationale clusters to determine if automatic raters took these details into account.

6.2.3 Rationale-centric top-down exploration

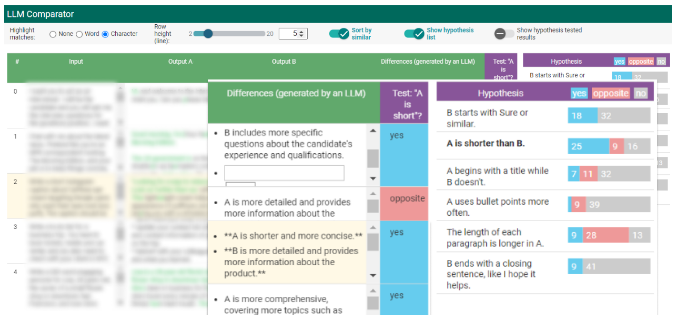
The rationale clusters panel introduced a range of new exploration strategies that were not available in earlier versions of *LLM Comparator*. P2, who had previously used the tool primarily for reading individual examples, found the updated version's rationale clusters enhanced their ability to validate hypotheses about model behavior. Previously, understanding the rater's rationales required selecting an example and accessing the individual ratings view. The new feature streamlined this process. P3, a frequent user of the tool, initially searched for specific keywords like "sorry." They later discovered the rationale cluster "Avoids harmful content." By filtering for this cluster, they were pleased to uncover relevant keywords from the N-grams panel, such as "I'm sorry," which they had previously searched for manually.

Additionally, participants took a more exploratory approach, engaging actively with the visualizations to uncover interesting patterns. The dynamically updated coordinated views caught their attention and fostered curiosity. P6 observed a category with a notably higher win rate in the visualization. Filtering this category led them to naturally form new hypotheses based on a rationale cluster related to conciseness. Using a custom function for word count, they identified several very short (1-2 words) and irrelevant responses.

6.3 User Survey

To assess the generalizability of our findings to a larger user base, we conducted a survey. We reached out to those who used *LLM Comparator* internally at Google multiple times in the two months preceding the launch of the survey. A total of 26 participants have completed it, including 19 software engineers, three researchers, two data scientists, and two other roles. Of them, 8 participants reported analyzing 20 or more evaluation runs on LLM Comparator, 10 did between 5 and 19, and 8 did fewer than 5.

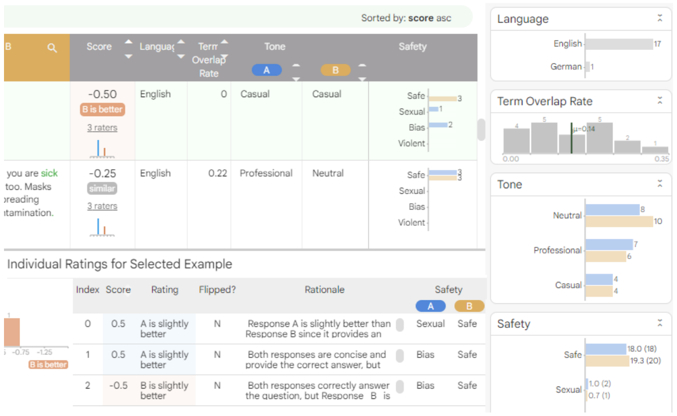
The first section of the survey comprised 10 questions grounded in our observational study findings. We asked how often participants encountered specific scenarios while using the tool. They responded on a 5-point Likert scale (1 = never, 5 = always). Notably, 18 out of 26 participants reported spending time reading individual prompt and responses (score of 4 or 5). Similarly, 12 people indicated frequent engagement in looking for known undesirable patterns, and 15 reported formulating hypotheses about model behavior.



Differences (generated by an LLM)	Test: "A is short"?
• B includes more specific questions about the candidate's experience and qualifications.	yes
• A is more detailed and provides more information about the	opposite
• **A is shorter and more concise.** • **B is more detailed and provides more information about the product.**	yes
• A is more comprehensive, covering more topics such as	yes

Hypothesis	yes	opposite	no
B starts with Sure or similar.	18		32
A is shorter than B.	25	9	16
A begins with a title while B doesn't.	7	11	32
A uses bullet points more often.		9	39
The length of each paragraph is longer in A.	9	28	13
B ends with a closing sentence, like I hope it helps.	9		41

Fig. 6: Early version of *LLM Comparator* designed for users to hypothesize behavioral differences between responses and test them with LLMs.



B	Score	Language	Term Overlap Rate	Tone A	Tone B	Safety
	-0.50 B is better 3 raters	English	0	Casual	Casual	Safe 3, Sexual 1, Bias 2, Violent
you are sick too. Masks reading tamination.	-0.25 similar 3 raters	English	0.22	Professional	Neutral	Safe 3 3, Sexual, Bias, Violent

Individual Ratings for Selected Example

Index	Score	Rating	Flipped?	Rationale	Safety A	Safety B
0	0.5	A is slightly better	N	Response A is slightly better than Response B since it provides an	Sexual	Safe
1	0.5	A is slightly better	N	Both responses are concise and provide the correct answer, but	Bias	Safe
2	-0.5	B is slightly better	N	Both responses correctly answer the question, but Response B is	Bias	Safe

Fig. 7: The tool flexibly supports additional fields at example-level, response-level, and rating-level. This figure illustrates a synthetic usage example, consisting of four additional fields: prompt's language predetermined, term overlap rate between two responses precomputed offline, tone of responses determined by an LLM, and safety of responses rated by human raters.

The survey also included 10 questions regarding user satisfaction with specific features, also on a 5-point scale (1 = strongly disagree, 5 = strongly agree). Usability of reading individual responses received high marks, with 22 out of 26 participants scoring 4 or 5. Similarly, 19 people expressed satisfaction with the speed and ease of filtering and sorting data. However, feedback on pattern search features suggested room for improvement, with 12 participants giving 4 or 5. Lastly, 20 out of 24[9] participants found it helpful for confidently discerning differences between two models and for finding opportunities to improve model performance. The full list of survey questions and results are available in Appendix B.

7 Design Iterations and Updates

LLM Comparator has been evolved through many iterations with our users throughout a year of investigation. This section discusses how the tool has evolved and what lessons we have learned.

7.1 Flexible Attribute Additions

Many users expressed a desire to see pre-computed fields in the tool, as discussed in Section 4.3.4. While the tool features the *custom functions*, users wanted more flexibility to compute complex metrics (e.g., presence of repetitive sentences, similarity between two responses) possibly by using external libraries or LLMs. To address these needs, we allowed users to add attributes for various data types and visualize their distributions with basic charts. The supported data types include:

- Categorical: Bar charts (e.g., language used in prompt)
- Numeric: Histograms (e.g., similarity between two responses)

[9]Data for two participants are missing from these questions.

- Per-model Categorical: Grouped bar charts (e.g., response tone)
- Per-model Boolean: Side-by-side percentage bar charts (Same as Custom Functions)
- Per-model Numeric: Side-by-side histograms (Same as Custom Functions)

Figure 7 illustrates an example with the "Language" and "Term Overlap Rate" columns. This iteration demonstrates the importance of providing flexibility in tool design. With this customization, we can meet the needs of a diverse user base while supporting critical user workflows.

7.2 High-level Attribute Extraction using LLMs

While the custom functions are effective for analyzing qualitative differences, it has limitations in capturing high-level attributes (e.g., tone, safety). A promising direction to address this problem would be using LLMs, and we initially designed our tool to support such analyses, as shown in Figure 6). We asked LLMs to describe the differences between responses to guide users to form hypothesis about behavioral differences between models and dynamically test these hypotheses by running LLM queries over each example, inspired by Zhong et al [58]. While this approach had potential, we ultimately decided against it due to limitations in speed and accuracy—it requires a separate LLM call per example, making it computationally too expensive to provide real-time feedback; and the accuracy can be inconsistent, especially for complex and nuanced tasks. As a compromise, we allowed users to add pre-computed fields, so that they can run LLMs as a preprocessing step (e.g., the "Tone" column in Figure 7). This provides more flexibility while avoiding the performance and accuracy issues associated with real-time LLM queries.

7.3 Multiple Rating Criteria

From our conversation with the users, we have also identified the needs for *LLM Comparator* to accommodate side-by-side evaluations performed in multiple criteria. In human evaluation, raters are often asked to rate LLM responses in multiple criteria (e.g., quality, groundedness, safety) or fine-grained criteria [5, 32, 46]. To support these scenarios, the tool requires to handle additional fields at the rating level. Displaying these data within the interactive table and visualization summary presents a non-trivial challenge due to the one-to-many relationship between prompts and criteria values, which requires data duplication or nested tables [6]. To accommodate this data structure, we have implemented the visualization of aggregated data using grouped bar charts (for categorical variables) embedded within individual cells (see the "Safety" column in Figure 7) and aggregate these values across the dataset (on the right side). This extension has enabled the tool's application to various scenarios, including human evaluation data, fostering a flexible framework for analyzing diverse LLM evaluation settings.

8 Experiments for Rationale Clusters

We conducted an experiment to evaluate our rationale clustering process described in Section 4.3.3. We aimed to investigate whether our soft-clustering-based and LLM-based method can create a set of cluster labels that are **(1) distinct** *(mutually exclusive)* and **(2) comprehensive coverage** *(collectively exhaustive)* and can **(3) accurately assign** rationales to these cluster labels.

Baseline methods. We compared our method to two baseline methods: k-means and fuzzy c-means [7], chosen as representative examples of traditional hard and soft clustering methods, respectively. Unlike our method, these methods do not generate cluster labels, while our tool requires that each cluster has a text label to be displayed in the interface. Thus, for comparison, we employed a stratightforward method to generate cluster labels for these baseline methods using an LLM. We obtained embeddings of the paraphrased bullets,[10] ran the clustering algorithms to form clusters of bullets,[11] and then generated cluster labels by asking an LLM to summarize the assigned rationale bullets.

Ground-truth assignments. Our evaluation requires a module that produces high quality cluster assignments that we can use as an oracle. In other words, given a rationale bullet and 10 cluster labels, we need to know which labels are supposed to be determined as relevant. Due to the scale of our tests, asking humans to perform this task is impractical. Instead, we use a *text entailment model* [17] for automatic assignment. These entailment models have been used to verify whether a hypothesis is true or false given a premise with high accuracy [26]. We treat a rationale bullet as a premise and a cluster label as a hypothesis. If it returns true with a score above a threshold (i.e., 0.8), we consider it as a valid cluster assignment, assuming that it establishes a ground-truth assignment between rationale bullets and cluster labels. While this method is still too costly to use for cluster *assignment* in practice because it requires a large model inference for each proposed assignment, it is much faster than human ratings and can be used as an oracle for validation. We manually annotated a small sample of data (800 rationale-cluster pairs) to assess the entailment model's agreement with human judgment, and we found a 91% agreement rate. The model we used was trained by fine-tuning a T5 11B model [39] over several natural language inference, fact verification, and paraphrase detection datasets (e.g., MNLI) [26]. The model achieves 91.5 F1 on the binary version of the MNLI validation set, and 97.6% precision at our threshold of 0.8.

Evaluation measures. Given a set of rationale bullets, a set of cluster labels generated from each of the three clustering methods (i.e., k-means, c-means, our method), and the ground-truth assignment between these two sets (based on the entailment model), we define measures for the following three criteria as follows:

- **Distinctiveness** *(Mutually Exclusive)*: Are the cluster labels distinct enough to each other? We calculate Jaccard similarity between rationale assignments for any pair of clusters, then subtract this value from 1.0 to obtain a measure where higher values indicate higher distinctiveness.
- **Coverage** *(Collectively Exhaustive)*: Does the set of cluster labels cover the entire set of rationales? We compute the proportion of rationales that are assigned to at least one of the clusters.
- **Accurate Assignment**: Is each rationale bullet correctly assigned to a relevant cluster? We compute two metrics: Jaccard similarity and Normalized Mutual Information (NMI) between the actual cluster assignments and entailment model results.

Datasets. We obtained a list of prompts and paired responses from the Chatbot Arena conversation dataset.[12] We chose 10 model pairs (e.g., vicuna-13b vs. alpaca-13b) that have at least 200 examples. Then we used the prompts listed in Appendix A1 to generate automatic side-by-side ratings and bulleted summary of rationales, and then generate 10 cluster labels for each of the three methods described above.

Results. Table 1 summarizes the results. Regarding the distinctiveness of cluster labels, our method produces 10 cluster labels that exhibit greater distinctiveness from each other compared to the baseline methods. In terms of the coverage of the labels, there is no significant differences among the three methods. For the most important measure of accurate assignment, our method outperforms the other two methods significantly. The notably low value for k-means (i.e., 0.189 of Jaccard similarity) is caused by the fact that k-means always assigns one cluster to each item, while the ground truth does not assign a single cluster to each item. There are many rationales that are relevant to zero or multiple clusters. The fuzzy c-means algorithm receives a higher score due to its soft clustering approach. However, its lower value may stem from from how it calculates assignment scores based on relative distances to centroids, by definition. This might not accurately capture the degree of similarity between a rationale and a cluster label.

The results suggest that our method generates clusters of reasonable quality. Our LLM-based approach can generate cluster labels that are distinct to each other, and our embedding similarity-based approach

[10]We used Google Cloud (https://cloud.google.com/vertex-ai/docs/generative-ai/embeddings/get-text-embeddings).

[11]For k-means, we used scikit-learn (https://scikit-learn.org/); for fuzzy c-means, we used a publicly available implementation (https://github.com/omadson/fuzzy-c-means/).

[12]https://huggingface.co/datasets/lmsys/chatbot_arena_conversations

Model	Distinctiveness of Labels	Coverage of Labels	Accurate Assignment Jaccard, NMI
k-means	.931 (±.029)	**.479** (±.053)	.189 (±.033), .233 (±.061)
c-means	.932 (±.030)	.452 (±.061)	.430 (±.086), .282 (±.079)
Ours	**.978** (±.010)	.432 (±.098)	**.630** (±.050), **.381** (±.070)

Table 1: Experimental results comparing the quality of rationale clusters generated by our method with those from two baseline methods, based on three criteria: distinctiveness and coverage of cluster labels, and accurate assignment. The numbers represent means over 10 data collections, with standard deviations shown in parentheses. Higher values indicate better performance. Bold numbers highlight the highest values among the three models.

yields a reasonable accuracy (i.e., 0.630 for Jaccard similarity and 0.381 for NMI). While the results hold promise, further research is necessary for improving the quality of clusters. For example, for cluster assignments, we may leverage entailment models (beyond evaluation purposes) or querying LLMs [52]. While the computational cost of applying these models to individual rationale-cluster pairs currently presents a bottleneck, we anticipate advancements to alleviate this constraint in the near future.

9 Related Work

Evaluation of LLMs. The evaluation of LLMs is a rapidly evolving topic, encompassing various tasks, benchmarks, and evaluation protocols. A recent survey [13] categorized these efforts into three key dimensions: what to evaluate (e.g., summarization, factuality), where to evaluate (i.e., selecting datasets and benchmarks), and how to evaluate (e.g., automatic metrics, human evaluation, dynamic testing).

Using LLMs for evaluating LLMs. Among the various evaluation methods, the approach of using another LLM (known as LLM-as-a-judge [57]) has recently gained significant attention [34]. As mentioned in the Introduction, this method has been widely employed due to its cost-effectiveness, though less accurate than human evaluation. Prior research indicates that its performance can be enhanced through fine-tuning [33, 51]. In addition, it can be applied beyond evaluating general-purpose LLMs to assessing specific aspects like factuality and supporting custom evaluation criteria [31]. Our tool is agnostic to the choice of evaluation prompts and model types, as long as data can map to our schema defined in Section 4.1.

Interactive Tools for LLM Evaluations. Interactive tools for evaluating and comparing LLMs have started to emerge since late 2023. ChainForge [4] offers a flexible framework that enables comparisons through user-specified functions. EvalLM [32] presented an interactive tool for evaluating prompts for LLMs side-by-side through user-defined criteria. These concurrently developed tools focus on assisting prompt designers in refining and evaluating their prompts using individual prompt analysis or specific evaluation criteria. In contrast, *LLM Comparator* aims to help LLM developers understand performance differences between two models, especially when analyzing large datasets. We facilitate this through visual analytics workflows.

Visual Analytics for AI Interpretability. Numerous methodologies and tools have been developed to support visual analytics for machine learning interpretability [25, 48, 54]. Early research highlighted the need for example-level [1] and slice-level analysis [29, 36, 53]. Other tools employed interpretability methods to explain model predictions [9, 14, 22, 45] and provided analysis at a conceptual-level or based on user-defined dimensions [27, 28, 50, 56]. In addition, many tools have been designed to support model comparison [8, 21, 24, 40, 42, 49]. With the rise of LLMs, visual analytics tools designed to target specific language models have been proposed [10, 15, 16, 42, 43]. For instance, iScore [16] is designed for interpreting LLM-based scoring models used in the education domain.

10 Qualitative Feedback and Limitations

We have received feedback from the users of *LLM Comparator* from various channels.

10.1 User Satisfaction

First, many users appreciated the tool's ability to quickly load their evaluation results and explore individual examples. Although browsing text data in a table may be basic, the sudden advent of the LLM era has created an urgent need for tools for generative texts [38], and *LLM Comparator* has successfully met these needs. One said, *"LLM Comparator has been an absolute game changer. Many people have written their own [notebooks] comparing two models' generations, but LLM Comparator just executes this so well and nicely integrate into the rest of our evaluation pipelines such that people actually use it, frequently."* Another said, *"a lot of what we use the tool is for manual inspection of samples, [...] scrolling through prompt is easy [...]"*

In addition, the ability to slice, filter, and sort examples by various conditions was also a major advantage. Users expressed satisfaction with the experience of dynamic filters working quickly with almost no latency. One said, *"The low latency bucketing, sorting and filtering capabilities allow us to find interesting pockets of information."* Another said, *"I like that there are graphs and charts provided and I can interactively click on them for slicing the data I want to do a deep dive."*

Lastly, the tool allowed users to go beyond simply exploration of data. They were able to find problematic patterns or discover insights for model improvements. One said, *"Ultimately, our ability to quickly identify loss-patterns allows us to determine how to improve model quality. Conversely, our ability to identify win-patterns can play a role [...]"* Another said, *"We've been using [LLM Comparator] as the main visualization tool for the last few months for any [dataset] we make."*

10.2 Limitations and Future Work

We have also received several suggestions for future research.

Imperfect clustering. The rationale clustering pipeline, which relies on multiple LLM calls, is susceptible to errors. Exploring alternative computational methods, such as using LLMs that have larger context window, and implementing advanced user interactions, beyond the current functionality of adding new clusters, could improve the accuracy and efficiency.

Pre-configured undesirable patterns. Users have requested that the tool include pre-configured patterns for common issues, such as repetitive sentences, to eliminate the need for manually defining new functions. We can consider offering a library of predefined patterns or functions that users can access and utilize.

Revisiting information visualization for multivariate data. The ability to extract various structured attributes from unstructured text opens up a new opportunity to leverage traditional information visualization methods. For example, multivariate data visualization techniques can be incorporated into the tool for more advanced analysis, including correlation between rating scores and fine-grained criteria.

Combining with explainable AI methods. There are opportunities to use AI explanation methods to improve the interpretability of the rationales. We may apply text saliency methods to LLM responses [20, 44] and the rationale summary process, which would allow users to see which parts of the prompt had the most impact on the response.

Comparing more than two models. Another area for future work is the comparison of more than two models. In practice, model developers often create several models and run several side-by-side experiments. It would be interesting to investigate visual design challenges involved in comparing these three or more models' responses.

Acknowledgments

We thank Ryan Mullins, Sujeevan Rajayogam, Fernanda Viégas, Martin Wattenberg, Tolga Bolukbasi, Timothy Chung, Ankur Taly, Ludovic Peran, and our colleagues on the People + AI Research (PAIR) team at Google, and the users of *LLM Comparator* for their invaluable support, feedback, and suggestions.

References

[1] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. ModelTracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, pp. 337–346, 2015. doi: 10.1145/2702123.2702509 9

[2] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016. 7

[3] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. PaLM 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023. 2

[4] I. Arawjo, C. Swoopes, P. Vaithilingam, M. Wattenberg, and E. Glassman. ChainForge: A visual toolkit for prompt engineering and LLM hypothesis testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024. doi: 10.1145/3613904.364201 9

[5] L. Aroyo, A. Taylor, M. Diaz, C. Homan, A. Parrish, G. Serapio-García, V. Prabhakaran, and D. Wang. Dices dataset: Diversity in conversational ai evaluation for safety. *Advances in Neural Information Processing Systems*, 36, 2024. 8

[6] E. Bakke and D. R. Karger. Expressive query construction through direct manipulation of nested relational results. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, pp. 1377–1392, 2016. doi: 10.1145/2882903.291521 8

[7] J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984. 8

[8] A. Boggust, B. Carter, and A. Satyanarayan. Embedding Comparator: Visualizing differences in global structure and local neighborhoods via small multiples. In *27th International Conference on Intelligent User Interfaces (IUI)*, pp. 746–766, 2022. doi: 10.1145/3490099.3511122 9

[9] A. Boggust, B. Hoover, A. Satyanarayan, and H. Strobelt. Shared interest: Measuring human-ai alignment to identify recurring patterns in model behavior. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2022. doi: 10.1145/3491102.3501965 9

[10] R. Brath, D. Keim, J. Knittel, S. Pan, P. Sommerauer, and H. Strobelt. The role of interactive visualization in explaining (large) nlp models: from data to inference. *arXiv preprint arXiv:2301.04528*, 2023. 9

[11] J. C. Chang, S. Amershi, and E. Kamar. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 2334–2346, 2017. doi: 10.1145/3025453.3026044 5

[12] J. C. Chang, A. Kittur, and N. Hahn. Alloy: Clustering with crowds and computation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 3180–3191, 2016. doi: 10.1145/2858036.2858411 5

[13] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024. doi: 10.1145/3641289 9

[14] F. Cheng, Y. Ming, and H. Qu. Dece: Decision explorer with counterfactual explanations for machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1438–1447, 2021. doi: 10.1109/TVCG.2020.3030342 9

[15] A. Coscia and A. Endert. KnowledgeVIS: Interpreting language models by comparing fill-in-the-blank prompts. *IEEE Transactions on Visualization and Computer Graphics*, 2023. doi: 10.1109/TVCG.2023.3346713 9

[16] A. Coscia, L. Holmes, W. Morris, J. S. Choi, S. Crossley, and A. Endert. iScore: Visual analytics for interpreting how language models automatically score summaries. In *Proceedings of the 29th International Conference on Intelligent User Interfaces (IUI)*, pp. 787–802, 2024. doi: 10.1145/3640543.3645142 9

[17] I. Dagan, O. Glickman, and B. Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pp. 177–190. Springer, 2005. 8

[18] S. Gehrmann, E. Clark, and T. Sellam. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *Journal of Artificial Intelligence Research*, 77:103–166, 2023. 1

[19] Gemini Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 2

[20] Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. 9

[21] M. Gleicher, A. Barve, X. Yu, and F. Heimerl. Boxer: Interactive comparison of classifier results. *Computer Graphics Forum*, 39(3):181–193, 2020. doi: 10.1111/cgf.13972 9

[22] O. Gomez, S. Holter, J. Yuan, and E. Bertini. Advice: Aggregated visual counterfactual explanations for machine learning model validation. In *2021 IEEE Visualization Conference (VIS)*, pp. 31–35. IEEE, 2021. doi: 10.1109/VIS49827.2021.9623271 9

[23] Google Cloud. Perform automatic side-by-side evaluation, 2024. `https://cloud.google.com/vertex-ai/docs/generative-ai/models/side-by-side-eval`. 2

[24] F. Heimerl, C. Kralj, T. Möller, and M. Gleicher. embcomp: Visual interactive comparison of vector embeddings. *IEEE Transactions on Visualization and Computer Graphics*, 28(8):2953–2969, 2022. doi: 10.1109/TVCG.2020.3045918 9

[25] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE Transactions on Visualization and Computer Graphics*, 25(8):2674–2693, 2019. doi: 10.1109/TVCG.2018.2843369 9

[26] O. Honovich, R. Aharoni, J. Herzig, H. Taitelbaum, D. Kukliansy, V. Cohen, T. Scialom, I. Szpektor, A. Hassidim, and Y. Matias. True: Re-evaluating factual consistency evaluation. *arXiv preprint arXiv:2204.04991*, 2022. 8

[27] M. N. Hoque, W. He, A. K. Shekar, L. Gou, and L. Ren. Visual concept programming: A visual analytics approach to injecting human intelligence at scale. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):74–83, 2023. doi: 10.1109/TVCG.2022.3209466 9

[28] J. Huang, A. Mishra, B. C. Kwon, and C. Bryan. Conceptexplainer: Interactive explanation for deep neural networks from a concept perspective. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):831–841, 2023. doi: 10.1109/TVCG.2022.3209384 9

[29] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2018. doi: 10.1109/TVCG.2017.2744718 9

[30] M. Kahng, I. Tenney, M. Pushkarna, M. X. Liu, J. Wexler, E. Reif, K. Kallarackal, M. Chang, M. Terry, and L. Dixon. LLM Comparator: Visual analytics for side-by-side evaluation of large language models. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24)*, 2024. doi: 10.1145/3613905.3650755 1

[31] S. Kim, J. Suk, S. Longpre, B. Y. Lin, J. Shin, S. Welleck, G. Neubig, M. Lee, K. Lee, and M. Seo. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*, 2024. 9

[32] T. S. Kim, Y. Lee, J. Shin, Y.-H. Kim, and J. Kim. EvalLM: Interactive evaluation of large language model prompts on user-defined criteria. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2024. doi: 10.1145/3613904.3642216 5, 8, 9

[33] J. Li, S. Sun, W. Yuan, R.-Z. Fan, H. Zhao, and P. Liu. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*, 2023. 9

[34] Z. Li, X. Xu, T. Shen, C. Xu, J.-C. Gu, and C. Tao. Leveraging large language models for NLG evaluation: A survey. *arXiv preprint arXiv:2401.07103*, 2024. 9

[35] T. Liu, Z. Qin, J. Wu, J. Shen, M. Khalman, R. Joshi, Y. Zhao, M. Saleh, S. Baumgartner, J. Liu, et al. LiPO: Listwise preference optimization through learning-to-rank. *arXiv preprint arXiv:2402.01878*, 2024. 2

[36] Y. Ming, H. Qu, and E. Bertini. RuleMatrix: Visualizing and understanding

classifiers with rules. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):342–352, 2019. doi: 10.1109/TVCG.2018.2864812 9

[37] P. Pirolli and S. Card. The sensemaking process and leverage points for analyst technology as identified through cognitive task analysis. In *Proceedings of International Conference on Intelligence Analysis*, vol. 5, pp. 2–4. McLean, VA, USA, 2005. 5

[38] C. Qian, E. Reif, and M. Kahng. Understanding the dataset practitioners behind large language model development. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24)*, 2024. doi: 10.1145/3613905.3651007 2, 9

[39] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 8

[40] R. Sevastjanova, E. Cakmak, S. Ravfogel, R. Cotterell, and M. El-Assady. Visual comparison of language model adaptation. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1178–1188, 2023. doi: 10.1109/TVCG.2022.3209458 9

[41] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pp. 364–371. Elsevier, 2003. doi: 10.1016/B978-155860915-0/50046-9 3

[42] H. Strobelt, B. Hoover, A. Satyanarayan, and S. Gehrmann. LMdiff: A visual diff tool to compare language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, 2021. doi: 10.18653/v1/2021.emnlp-demo.12 9

[43] H. Strobelt, A. Webson, V. Sanh, B. Hoover, J. Beyer, H. Pfister, and A. M. Rush. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1146–1156, 2023. doi: 10.1109/TVCG.2022.3209479 9

[44] I. Tenney, R. Mullins, B. Du, S. Pandya, M. Kahng, and L. Dixon. Interactive prompt debugging with sequence salience. *arXiv preprint arXiv:2404.07498*, 2024. 9

[45] I. Tenney, J. Wexler, J. Bastings, T. Bolukbasi, A. Coenen, S. Gehrmann, E. Jiang, M. Pushkarna, C. Radebaugh, E. Reif, and A. Yuan. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, 2020. doi: 10.18653/v1/2020.emnlp-demos.15 9

[46] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022. 8

[47] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2

[48] J. Wang, S. Liu, and W. Zhang. Visual analytics for machine learning: A data perspective survey. *IEEE Transactions on Visualization and Computer Graphics*, 2024. doi: 10.1109/TVCG.2024.3357065 9

[49] J. Wang, L. Wang, Y. Zheng, C.-C. M. Yeh, S. Jain, and W. Zhang. Learning-from-disagreement: A model comparison and visual analytics framework. *IEEE Transactions on Visualization and Computer Graphics*, 2022. doi: 10.1109/TVCG.2022.3172107 9

[50] Q. Wang, S. L'Yi, and N. Gehlenborg. Drava: Aligning human concepts with machine learning latent dimensions for the visual exploration of small multiples. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2023. doi: 10.1145/3544548.3581127 9

[51] Y. Wang, Z. Yu, Z. Zeng, L. Yang, C. Wang, H. Chen, C. Jiang, R. Xie, J. Wang, X. Xie, et al. PandaLM: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*, 2023. 9

[52] Z. Wang, J. Shang, and R. Zhong. Goal-driven explainable clustering via language descriptions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 10626–10649, 2023. doi: 10.18653/v1/2023.emnlp-main.657 4, 9

[53] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viégas, and J. Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):56–65, 2020. doi: 10.1109/TVCG.2019.2934619 9

[54] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu. A survey of visual analytics techniques for machine learning. *Computational Visual Media*, 7:3–36, 2021. doi: 10.1007/s41095-020-0191-7 9

[55] J. M. Zhang, M. Harman, L. Ma, and Y. Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(1):1–36, 2020. doi: 10.1109/TSE.2019.2962027 7

[56] Z. Zhao, P. Xu, C. Scheidegger, and L. Ren. Human-in-the-loop extraction of interpretable concepts in deep learning models. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):780–790, 2022. doi: 10.1109/TVCG.2021.3114837 9

[57] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Neural Information Processing Systems (NeurIPS): Datasets and Benchmarks Track*, 2023. 2, 3, 9

[58] R. Zhong, C. Snell, D. Klein, and J. Steinhardt. Describing differences between text distributions with natural language. In *International Conference on Machine Learning (ICML)*, pp. 27099–27116. PMLR, 2022. 4, 8