ReScorer: An Aggregation and Alignment Technique for Building Trust into LLM Reasons

Jay Mohta*
Brian M. de Silva*
jaymoht@amazon.com
slvbria@amazon.com
Amazon
Seattle, Washington, USA

Sugumar Murugesan
Dantong Liu
musuguma@amazon.com
lidanton@amazon.com
Amazon
Sunnyvale, California, USA

Yan Xu
Mingwei Shen
yanxuml@amazon.com
mingweis@amazon.com
Amazon
Seattle, Washington, USA

ABSTRACT

Large language models (LLMs) offer substantial potential for automating labeling tasks, showcasing robust zero-shot performance across diverse classification tasks. The LLM-generated reasons that accompany these classifications contain signals about the quality of the classifications. Estimates of quality of these reasons can, in essence, be used to detect potentially incorrect predictions. Conventional metrics for scoring reasons such as ROUGE-L and BLEU scores depend on ground truth reference reasons, which are challenging and expensive to acquire, and are not available at inference time for new examples. In this paper, we use a product classification dataset to evaluate two reasoning scoring strategies that do not rely on reference reasons: one involving an LLM-based scorer and another using recently proposed ROSCOE metrics. Our analysis reveals that LLM-based approaches are computationally intensive, while aligning ROSCOE metrics with human judgment presents challenges. Consequently, we propose an extension to the ROSCOE framework called ReScorer, which achieves 7% better alignment with human judgment compared to LLM-based evaluation and 59% better than ROSCOE, while being 89% cheaper compared to LLM-based scoring.

1 INTRODUCTION

Large Language Models (LLMs) such as Claude [1], GPT-3 [2] and PaLM [3] are capable of generating fluent and realistic responses to a wide variety of prompts. Recent work has also demonstrated the use of LLMs as annotators [4, 5]. Wei et al. [6] showed that LLMs are capable of providing a reason for their decision along with the end classification.

In classification tasks, the quality of a reason generated by an LLM can be a reliable indicator of the quality of the classification itself [6]. For instance, in e-commerce, an LLM may predict that a product does not violate a safety rule, and provide a corresponding reason. An instance of a safety rule is sale of Sky Lanterns in dry regions, which can be fire hazards. If a scoring algorithm assigns a low score to the reason generated by LLM allowing the sale of the product under the rule, the product can be flagged for human audit before a safety violation happens. Hence, a metric measuring reason quality can help to identify potentially incorrect LLM classifications.

While traditional metrics like ROUGE-L [7] and BLEU [8] have been proposed for evaluating text generation models in summarization and machine translation, computing these metrics requires ground truth texts. This makes them unsuitable for inference time applications such as the Sky Lantern use case above. Consequently, there is a need for unsupervised metrics to evaluate the reasons generated by LLMs.

Zhang et al. [9] propose an unsupervised metric that uses models like BERT [10] to evaluate the output of text generation models. They show their BERT-based scoring mechanism better reflects human judgement than ROUGE-L or BLEU. Recent work by Manakul et al. [11] finds that BERT-based methods may have difficulty capturing subtle differences in reasons, leading to similar scores for right and wrong reasons. Hence Manakul et al. [11] propose using LLMs as verifier to evaluate language model reasons. The LLMs can be prompted multiple times to obtain consistency or confidence scores [12]. However, computing LLM-based metrics can be computationally expensive as they involve potentially costly LLM calls.

Recent work by Golovneva et al. [13] proposes evaluating language models using smaller embedding based models to compute metrics across different dimensions like semantic alignment, semantic similarity, logical inference, and grammatical correctness. This approach, named ROSCOE, is useful for identifying specific areas in which a reason is deficient. Its main limitation is that it produces 14 metrics rather than one, which makes it difficult to rank or filter reasons. There is also no guarantee that these metrics are all aligned with human preferences.

In this paper, we first assess two state-of-the-art evaluation methods for the language model output: LLM-based evaluation and ROSCOE. Our experiments use different task descriptions from a diverse set of programs within Amazon, with the goal of classifying products based on their adherence to the given task description. While LLM-based evaluation yields metrics that align more closely with human judgment, it is hindered by a significant computational cost, to the tune of $2000 per million reason scores for our tasks [14]. On the other hand, ROSCOE metrics are inexpensive to compute at $225 per million reasons evaluated on 1 GPU of p3.2xlarge. However, as we show in this paper, the plurality of ROSCOE metrics and their misalignment with human preferences renders them ill-suited for reason ranking and filtering. Motivated by these limitations, we propose Reasoning Scorer (ReScorer), a modification to ROSCOE

*Equal contribution

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Genaiecom '24, October 25, 2024, Boise, ID
© 2024 Copyright held by the owner/author(s).

that maps its 14 metrics to a single output. We accomplish the mapping with a lightweight logistic regression model trained on a small set of human judgments.

Our findings reveal that ReScorer outperforms both LLM-based and ROSCOE metrics in detecting low quality reasons (7% better than LLM-based scoring and 59% better than ROSCOE), underscoring the robustness of the ReScorer algorithm, while preserving a lower computational cost—89% reduction in computational cost compared to LLM-based scoring.

The remainder of this paper is structured as follows - Section 2 introduces various approaches to evaluate LLM reasons, including ReScorer. Section 3 outlines our experimental setup. We present key findings of the study in Section 4, followed by conclusions in Section 5.

2 REASON SCORING ALGORITHMS

In this section, we highlight different approaches we used to evaluate LLM reasons for product classification use cases. Section 2.1 introduces the LLM-based approach we used for evaluating reasons from LLMs; Section 2.2 introduces ROSCOE suite of metrics used for evaluating LLM output; and we propose ReScorer in Section 2.3.

2.1 LLM-based scoring

LLMs have recently shown to be effective in assessing information consistency between a document and its summary in zero-shot settings [11, 15]. We adopt a prompting strategy similar to Manakul et al. [11] for evaluating reasons produced by LLMs. In this work we use Claude-v2 [1] as a verifier LLM to evaluate reason output from Claude-v1. Our use case is product classification based on a task description (also called a *policy document*). We used the following prompt to score a previously generated LLM reason:

Given the following policy document:
`{task_description}`

Given this product to classify:
`{product_features}`

Given this response from the model classifying the product according to the policy:
`{claude_response}`

Please provide a score as either 0 or 1 about how good the response from the LLM is. Assign a score of 0 for a bad response and 1 for a good response.

To assess the likelihood that a provided reason is "good", we employ a multi-prompt strategy. This involves prompting the model multiple times with different hyperparameters for the p in the top-p decoding strategy [16], resulting in various outputs. We then utilize these outputs to compute the probability of the reason being considered "good." The computation of this probability is conducted using the following equation:

$$P(\text{Good Reason}) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(O_i = 1)$$

where N represents the number of times LLM is prompted, O_i represents ith output from LLM and $\mathbb{I}$ represents the indicator function defined as follow:

$$\mathbb{I}(O_i = 1) = \begin{cases} 1 & \text{if } O_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

This approach is very computationally expensive as we have to prompt the verifier LLM multiple times.

2.2 ROSCOE

Golovneva et al. [13] propose a suite of metrics to evaluate language models output on different dimensions such as *semantic alignment*, *semantic similarity*, *logical inference*, and *grammatical correctness*.

The *semantic alignment* and *semantic similarity* metrics utilize an embedding model to generate embeddings for task descriptions, product details, and LLM reasons. These embeddings facilitate the computation of various scores, including faithfulness of the LLM output (reason) to the input text (task description + product details), and informativeness, which gauges the model's ability to refer to the input text during output generation. In product classification tasks, it is crucial for the model to accurately refer to the task description. Higher informativeness scores indicate that the LLM output effectively refers to the task description, contributing to improved classification accuracy and good reasoning capability of the model.

Logical inference metrics gauge probability of contradiction between the source text and LLM output. Lower values of probability of contradiction are desirable, indicating minimal contradiction between the human-provided source text and the LLM output. Additionally, grammatical correctness assesses the coherence of the model output, with perplexity from GPT-2 utilized for this evaluation. Refer to Appendix B for comprehensive list of ROSCOE metrics.

Though ROSCOE is a powerful framework for unsupervised reasoning evaluation, there are two issues with using it to detect bad reasons. The first has to do with the fact that ROSCOE involves multiple scores as they evaluate LLM model capabilities across various dimensions. However, deriving decisions from multiple metrics poses a challenge, as it remains unclear how to prioritize different dimensions effectively. Additionally, ROSCOE metrics may not be fit for specific problem domains, being overly general in nature. In some scenarios, good reasons may score lower in certain metrics than bad ones. For example, reasons may exhibit poor grammar metrics because they quote product details or task descriptions, which are themselves grammatically incorrect. These limitations emphasize the need for an additional layer to *aggregate* and *align* metrics tailored to specific applications.

2.3 ReScorer

To address the aforementioned issues, we introduce a lightweight Reasoning Scorer—ReScorer (Figure 1). ReScorer uses a simple logistic regression model [17] to map ROSCOE metrics to a single score between 0 and 1. To ensure that ReScorer is aligned with human preferences, we fit the logistic regression model to a small number of human-assigned evaluations of LLM generated reasons.

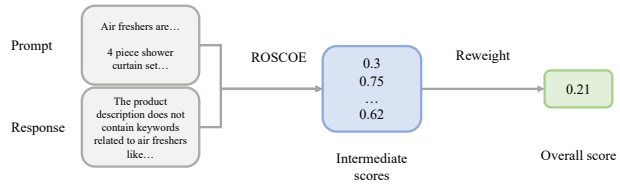

Figure 1: An overview of ReScorer. Given a prompt and corresponding LLM response, ReScorer first computes a set of unsupervised metrics via ROSCOE, then recombines them into a single score with a logistic regression model trained on human annotated data.

The benefits of ReScorer include its ability to align ROSCOE metrics with human judgements, its low computational cost, and the fact that it outputs a single summary score. If we want to use a score to rate LLM outputs, it is important for the score to adequately represent human preferences. Furthermore, when we need to assess a large number of LLM generated reasons, the low computational cost is critical. ReScorer achieves this by adding only a lightweight logistic regression model on top of ROSCOE. Consolidating performance into a single score, as opposed to utilizing a variety of metric, facilitates the decision making in the downstream tasks.

ReScorer's main drawback is that it requires labeled data to learn the mapping component, even though the metric itself is unsupervised, that is it does not require reference reasons during scoring of reasons. Fortunately, the labeled data needed to train ReScorer is generally easier to obtain compared to ground truth generative output. Annotators need only assign LLM output with scores based on a target task, rather than composing text outputs (e.g. summaries or answers to questions). Another potential limitation is that, ROSCOE metrics that are the foundation for ReScorer may sometimes fail to capture relevant characteristics of LLM answers. However, it is easy to supplement the current ROSCOE metrics with additional metrics tailored for specific applications within the ReScorer framework.

3 EXPERIMENTAL SETUP

Dataset. We curated a corpus with each sample of the form `<task description, product features, task label, reason>` , with 764 total tasks. Figure 2 shows a sample of data in the corpus. The `<task description, product features, task label>` part of the corpus are human annotated. The remaining `reason` part of the corpus is derived by passing `<task description, product features, task label>` to an off-the-shelf LLM, Claude-v1 [1]. Table 1 records the statistics of our corpus. Human annotators were asked to score these reasons as low, medium and high quality. Refer to Appendix C for annotation guidelines provided to human annotators.

After annotation, there were 2,906 reasons with high scores, 4 medium scores, and 52 low scores, leaving us with a severely imbalanced dataset. Therefore we simplified the scores and added two types of synthetic bad reasons to the dataset. For simplicity, we mapped the original scores to binary scores by assigning 0 to low scores and 1 to both medium and high scores. We also added two types of synthetic negative examples: easy and hard perturbations, P_e and P_h. Both mismatch `<task description, product features>` of a sample with `<reason>` from another sample in the original dataset. And we assign both scores of 0. We obtain easy perturbations by fixing a task and product then randomly selecting a reason from another example in the dataset. Such examples should be easy to detect since the perturbed reason is usually from a different task and hence easy to detect. We construct hard perturbations similarly, except we sample the new reason from another example with the *same task*. These negatives are more subtle because the reason references the correct task, but the wrong product. Further, we create a 80/20 train/test split, stratifying on the ground truth score. Only ReScorer uses the train set, and we evaluate all approaches on the test set.

Task description → Defines the classification task
I am interested in classifying products into the following category: Personal Flotation Device. Below I have described the category: A personal flotation device is a flotation device in the form of a vest or suite that is donned and fastened to a user to prevent the wearer from drowning in a body of water. Products that satisfies the above description often contain keywords: boatcushion, lifejacket, lifepreserver, lifevest. Products that should be excluded from this classification task contain keywords: limetal, surgical, newborns, suitable from birth

Product features →Describes the query product
item_name:o'neill reactor teen life vest; bullet_point:flexible multidensity design|||closed cell pvc marine foam|||zipper safety tab|||quick release dual safety buckles|||mesh for drainage

Task label → Classification labels provided by human annotator
Yes

Reason → Claude generated classification reason based on <Task description, Product features, Task label>
This product contains keywords like life vest and is intended for safety on water, so it fits the category of Personal Flotation Devices.

Figure 2: Sample data from the dataset.

Perturbations	# Samples	% Positives	Avg. Samples per Task
H	2962	98.2%	3.9
P_e	2962	0%	3.9
P_h	2962	0%	3.9
Overall	8886	32.7%	11.6

Table 1: Summary of dataset statistics. H is the set of human annotations, P_e is the set of easy perturbations, and P_h is the set of hard perturbations. "Overall" represents the full dataset. All perturbations include all 764 tasks.

ROSCOE. As baselines, we test two straightforward techniques for mapping ROSCOE metrics to a single score. **ROSCOE avg.** computes the mean of the 14 ROSCOE metrics. And **ROSCOE one-hot** uses the metric which has the best correlation with the target score. We specifically use Somers' D correlation [18], elaborated at the end of this section. In this case, the selected metric was `repetition_step` (refer Appendix B for details on ROSCOE metrics). Both methods generate unsupervised metrics. To calculate semantic alignment and semantic similarity metrics, we employ the off-the-shelf embedding model from Hugging Face, specifically using `all-MiniLM-L6-v2` [19]. For logical inference metrics, we adopt the model recommended by Golovneva et al. 13: `DeBERTa-v3-large-mnli-fever-anli-ling-wanli`. This model, based on the DeBERTa-large architecture [20], is fine-tuned on the MNLI, ANLI, and WNLI datasets [21]. To assess grammatical correctness, we use perplexity output from `GPT-2` [22].

ReScorer training. We trained ReScorer's supervised module (logistic regression) on the train set with hyperparameter optimization and 3-fold cross-validation using scikit-learn [17] (see Appendix A for details). We obtained the best overall performance by training on human annotations plus hard perturbations (see Section 4.2). Unless otherwise noted, the results in the following section all use this model.

Metric. Similar to Golovneva et al. [13], we use Somers' D[1] [18] to measure the correlations between the scorers and human judgements. The scorers output scores between 0 and 1, and human judgements are either 0 (for bad reasons) or 1 (for good reasons). Somers' D is a way to measure the correspondence between two rankings. Somers' D closer to 1 represent strong agreement, while closer to -1 indicates strong disagreement. To better understand how well the methods detect different types of negative examples, we evaluate each on different subsets of the test data: human annotations H, human annotations + easy perturbations (H, P_e), human annotations + hard perturbations (H, P_h), and the full test set —human annotations + easy perturbations + hard perturbations (H, P_e, P_h).

4 RESULTS

Our results are organized into four sections. Section 4.1 gives the main results: a comparison of the methods of Section 2 on our reasoning dataset. Section 4.2 explores the effects on ReScorer of adding perturbations to the training data. We consider specific examples of where ReScorer correctly identifies bad reasons in Section 4.3. And Section 4.4 further discusses why ROSCOE metrics alone are insufficient for scoring reasons for specific applications.

4.1 Main results

	Somers' D			
Method	**H**	**(H, P_e)**	**(H, P_h)**	**(H, P_e, P_h)**
Random	-0.3081	0.0159	0.0356	0.0278
LLM-based [1]	**0.2294**	<u>0.8485</u>	<u>0.2629</u>	<u>0.5470</u>
ROSCOE avg. [13]	<u>0.2138</u>	-0.0165	0.0651	0.0213
ROSCOE one-hot [13]	0.1242	-0.0032	0.0686	0.0310
ReScorer	-0.1111	**0.9616**	**0.2934**	**0.6195**

Table 2: Somers' D between ground truth scores and scores assigned by reason evaluation methods. We evaluated on subsets of the test set with various types of perturbations: human annotations H, easy perturbations P_e, and hard perturbations P_h. The best results on each subset are bold and the second best are <u>underlined</u>. We compare our method, ReScorer, against four baselines: Random–randomly assigned scores (selected uniformly from $\{0, 1\}$), LLM-based–scores assigned by Claude, ROSCOE avg.–the mean of ROSCOE metrics, and ROSCOE one-hot–the best-performing single ROSCOE metric.

[1]We use the SciPy implementation [23].

Table 2 shows the Somers' D values for all reason evaluation methods on different types of perturbations. ReScorer demonstrates strong performance in distinguishing good and bad reasons when we include any or all perturbation types, improving significantly over LLM-based scoring by 7% (Somers' D increase from 0.54 to 0.61) and ROSCOE by 59% (Somers' D increase from 0.03 to 0.62). However, it struggles on the severely imbalanced human annotated data, many of which it confuses for negatives. LLM-based scoring provides a reasonable level of performance across different strata. Whereas the ROSCOE approaches do poorly at detecting perturbations, suggesting that ReScorer is needed to re-weigh the unsupervised metrics. Interestingly, we find that a simple average of ROSCOE metrics performs reasonably well on human annotations H, but this may be because this set is over 98% good reasons.

The Somers' D values reported here are generally lower than the values in [13], which suggests that our task may be more challenging. Indeed, the bad reasons from the human annotations and the hard perturbations are often wrong in subtle ways. For example, it may be hard to classify as good or bad a reason that is trying to determine whether a vegetable-derived oil is meant for cooking or personal use. Both uses have similar keywords such as "organic", "vegan", and "cold-pressed". To determine whether the reason is good, one must essentially solve the underlying classification problem. It is not reasonable to expect methods based on unsupervised metrics, like ROSCOE and ReScorer, to consistently solve such tasks.

4.2 Training data and ReScorer

We explore the effects of supplementing the human-annotated training data by varying the types of perturbations added to the training data and evaluating ReScorer's performance on different subsets of the test set. Table 3 shows the results. Unsurprisingly, we find that adding perturbed examples improves ReScorer's ability to detect perturbations, i.e. performance on (H, P_e, P_h)—Somers' D 0.6195 in Table 3. Adding only hard negatives/perturbations improves performance the most, but the larger message is this: the biggest gains come from adding *any* extra negative examples. As with any supervised model, ReScorer performs best when its train and test distributions are aligned. Therefore, on human annotated test data H, ReScorer does best when no perturbations are added to the training data, in which case it is comparable to the LLM baseline (0.2294 from Table 2). To optimize performance on such data, it may be best to omit added perturbations. However, given that our annotated dataset is highly imbalanced, it is still important to add perturbations to ensure robustness against some reasoning errors. Since the perturbations are essentially free, the main cost incurred is a minor increase in training time.

4.3 Anecdotes

It is informative to consider specific examples where ReScorer detects bad reasons (Table 4). In each example, the reason contains relevant information along with one or more hallucinations, and in each case ReScorer correctly assigns the reason a low score, i.e. below 0.5. It is promising that ReScorer is able to detect such nuances in LLM-based reasons.

Train perturbations	Test perturbations			
	H	(H, P_e)	(H, P_h)	(H, P_e, P_h)
H	**0.2188**	0.4929	-0.0919	0.1902
(H, P_e)	0.1243	0.9820	0.0643	0.5090
(H, P_h)	-0.1111	0.9616	**0.2934**	**0.6195**
(H, P_e, P_h)	-0.0183	**0.9834**	0.2593	0.6119

Table 3: Somers' D for ReScorer models with different types of perturbations included in the training data (Train perturbations). We report results on different segments of the test set (Test perturbations).

Class	Product features	LLM reason
Cooking thermometers	gomesme - dual channels wireless wireless waterproof bbq food thermometer thermometer digital cooking meat food oven grilling with timer...	This product mentions wireless and waterproof features but **does not explicitly mention cooking or food thermometers**.
Memory cards	new ssk usb 2.0 card reader for ms, xd, cf, mmc, sd, microsd, m2 card. guarantee the quality of products|||delivery time 20-30 days|||if you have any questions, please feel free to contact us	This product contains keywords like "**64gb**", "microsd", "**memory card**", and **mentions it is for a Samsung tablet**. Based on the SOP, this satisfies the description of a Memory Card.
Apple laptops	letter h laptop carrying bag sleeve,neoprene sleeve case/ancient jaded wooden h symbol from alphabet uppercase scratched worn design print decorative/for apple macbook air samsung google acer hp...	This product mentions Apple Macbook Air and **includes specifications like processor speed and graphics** that match the SOP criteria for Apple Laptops.

Table 4: Examples of bad reasons that ReScorer detects. For brevity we show the class names in place of the full class descriptions and truncate long product features at the item name. Hallucinations are shown in bold. Note that "SOP" is synonymous with "class description."

4.4 Why can't we just use ROSCOE?

There are two reasons behind our decision to put a supervised fine-tuning layer on top of ROSCOE metrics instead of using them directly. The first is simply the issue of deriving a decision from multiple metrics. For example, how should we decide which of two reasons is better when reason A scores $(0.9, 0.1)$ and reason B scores $(0.7, 0.7)$? Two obvious approaches are to pick one dimension/metric or to average across multiple dimensions. But we showed that these two strategies perform worse than ReScorer (Table 2).

The second issue is that the ROSCOE metrics are sometimes too general-purpose for specific problem domains. In some applications good reasons can have score lower in some metrics than bad reasons. For example, `repetition_word` and `repetition_step` measure the repetitiveness of a text. Usually it is bad for text to be too repetitive, but in our classification use-case, it is sometimes desirable. The following reason was assigned low repetition metrics despite being satisfactory:

> *This product does not mention a glue gun, only glue bars. The SOP defines Glue Guns as devices that dispense hot melt glue, while this item only includes glue bars.*

Similarly, we found that some good reasons had low `grammar_step` and `grammar_step_max` metrics because they quoted product details or class descriptions, which were themselves grammatically incorrect. These examples show the need for an extra step on top of ROSCOE to align and aggregate the metrics for the application at hand.

5 CONCLUSION

We explored the *evaluation* of LLM generated explanations alongside classification decisions. Using a e-commerce product classification dataset, we assessed two reasoning evaluation strategies: one using an LLM-based scorer and the other using ROSCOE metrics. Our analysis highlighted the computational intensity of LLM-based approaches and the difficulties in aligning ROSCOE metrics with human judgment. Consequently, we proposed ReScorer, an extension to the ROSCOE framework, achieving a 7% improvement over LLM-based scoring and a 59% improvement over ROSCOE, while reducing computational costs by 89% compared to LLM-based scoring. These findings emphasize the importance of efficient evaluation techniques and highlight ReScorer's potential to enhance the reliability of language model predictions in critical applications.

REFERENCES

[1] Introducing Claude — anthropic.com. https://www.anthropic.com/index/introducing-claude, . [Accessed 15-01-2024].

[2] 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. Language models are few-shot learners, 2020.

[3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways, 2022.

[4] Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. AnnoLLM: Making large language models to be better crowdsourced annotators, 2023.

[5] Jay Mohta, Kenan Emir Ak, Yan Xu, and Mingwei Shen. Are large language models good annotators? In *I Can't Believe It's Not Better Workshop: Failure Modes in the Age of Foundation Models*. Conference and Workshop on Neural

Information Processing Systems, December 2023. URL https://openreview.net/forum?id=0RwbmLUU2o.
[6] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
[7] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.
[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.
[9] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT, 2020.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
[11] Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models, 2023.
[12] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.
[13] Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=xYlJRpzZtsY.
[14] Claude amazon bedrock pricing, . URL https://aws.amazon.com/bedrock/pricing/.
[15] Zheheng Luo, Qianqian Xie, and Sophia Ananiadou. ChatGPT as a factual inconsistency evaluator for text summarization, 2023.
[16] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.
[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
[18] Robert H. Somers. A new asymmetric measure of association for ordinal variables. *American Sociological Review*, 27:799, 1962. URL https://api.semanticscholar.org/CorpusID:144232738.
[19] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.
[20] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.
[21] Adina Williams, Tristan Thrush, and Douwe Kiela. Anlizing the adversarial natural language inference dataset. 2022.
[22] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://api.semanticscholar.org/CorpusID:160025533.
[23] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

A RESCORER HYPERPARAMETER OPTIMIZATION

When training ReScorer, we used the scikit-learn logistic regression implementation and randomized 3-fold cross-validation search over the following hyperparameters with random state 42 and 128 iterations:

```
"penalty": [None, "l1", "l2"],
"C": [1e-3, 1e-1, 5e-1, 1, 5, 10, 100],
"solver": ["liblinear"],
"tol": [1e-5, 5e-5, 1e-4, 5e-4, 1e-3],
"class_weight": [None, "balanced"],
"max_iter": [100, 200, 500]
```

B LIST OF ROSCOE METRICS

ROSCOE [13] semantic alignment metrics is the reasoning alignment vector from the N-step hypothesis h (LLM response) to the source s (task description + product details) of length T:

$$r\text{-}align(h \rightarrow s) = \{\alpha_1, \alpha_2, \ldots, \alpha_N\},$$

where each alignment value

$$\alpha_i = r\text{-}align(h_i \rightarrow s) = \frac{1}{2}\left[1 + \max_{j=1}^{T}\left(\cos(h_i, s_j)\right)\right] \in [0, 1]$$

is the normalized cosine similarity between the hypothesis step and the most similar sentence in a context. It explicitly measures the grounding of the step-wise reasoning with respect to the source text. The ROSCOE metrics are summarized in Table 5

C ANNOTATION GUIDELINES

Following are the annotation guidelines provided to human reviewers for evaluating reasoning output from LLM:

- Provide score 1, 2 or 3 depending on quality of reason in `<task description, product features, task label, reason>`
- A low score of 1 should be given when there are outright hallucinations, such as task asks for bicycles or scooters, product is a scooter, but reason provided is that product is a bicycle.
- A medium score of 2 should be given when the reason is correct but contains general knowledge not appears in product attributes - this can happen since LLMs are pre-trained on billions of internet-scale data and hence could bring general knowledge during reasoning.
- A high score of 3 is given when none of the above are true.

Metric	Description
Faithfulness-Step ($h \rightarrow s$)	This step-level score is based on the alignment from the hypothesis steps to the source sentences, and is calculated as the mean reasoning alignment score over the steps of reasoning (see illustration in Appendix D, Figure 3): $\frac{1}{N}\sum_{i=1}^{N} r\text{-}align(h_i \rightarrow s)$. Faithfulness measures if the model misinterpreted the problem statement, or the reasoning chain is too vague, irrelevant, or misuses information.
Faithfulness-Token ($h \rightarrow s$)	We extend step-level embeddings of the Faithfulness-Step by measuring similarities between the token embeddings: $\frac{1}{N+M}\sum_{i=1}^{N}[r\text{-}align(h_i \rightarrow s) + \sum_{j=1}^{M_i} r\text{-}align_{\text{token}}(h_{i,j} \rightarrow s)]$, as shown in App. D, Fig. 3. M_i is the number of tokens in step h_i, $M = \sum_{i=1}^{N} M_i$ is the total number of tokens in the reasoning chain, $h_{i,j}$ is the jth token in the ith step, and $r\text{-}align_{\text{token}}$ is the alignment vector from tokens in step h_i to all tokens in s.
Informativeness-Step (Info-Step) ($h \leftrightarrow s$)	Measures how well information present in the source is used in the reasoning steps: $\left[\frac{1}{T}\sum_{t=1}^{T} r\text{-}align(s_t \rightarrow h) + \frac{1}{N}\sum_{i=1}^{N} r\text{-}align(h_i \rightarrow s)\right]/2$. Info-step gives a higher score to reasoning steps that are well-grounded with respect to the source, and identifies the degree of information from source that is covered by the generated hypothesis. A lower Info-Step score corresponds to the reasoning steps that are not related to the source sentences or have missed information provided in the context.
Repetition-Token ($h_i \rightarrow h_j$)	To identify repeated, or paraphrased steps, we look at the token alignment scores between all steps in the hypothesis chain: $1 - \max_{i=2..N} \max_{j=1..i-1}\left[\frac{1}{M_i}\sum_{l=1}^{M_i} r\text{-}align_{\text{token}}(h_{i,l} \rightarrow h_j)\right]$. For each pair of sentences, we look at the mean token alignment, and find those sentences that maximize this alignment score. In other words, Repetition-Token will punish chains where there are at least two steps with high overlap in token embeddings.
Informativeness-Chain (Info-Chain) ($h \rightarrow s$)	Similar to Info-Step, this metric quantifies the degree of agreement between the hypothesis chain and the source and is calculated as $[1+\cos(h, s)]/2$. We embed reasoning chain and source context as a whole, as opposed to using step-wise embeddings.
Repetition-Step ($h_i \leftrightarrow h_j$)	Measures repetition-related errors on the step level by checking if it paraphrases information already mentioned in the previous steps: $(1 - \max_{i=2..N} \max_{j=1..i-1}[\cos(h_i, h_j)])/2$. Unlike Repetition-Token, which is orderless and compares individual tokens in pairs of steps, Repetition-Step considers step embeddings similarity and is more robust to changing contexts.
Semantic Coverage-Chain ($r \leftrightarrow h$)	Reflects the overall degree of similarity between the reference and hypothesis chains, comparing reference and hypothesis embeddings as a whole: $[1 + \cos(r, h)]/2$.
Self-Consistency ($h_i \leftrightarrow h_j$)	Measures logical entailment errors within the reasoning steps: $1 - \max_{i=2..N} \max_{j<i} p_{\text{contr}}(h_i, h_j)$. This metric will punish chains where there is a pair of steps that are likely to contradict each other.
Source-Consistency ($h \leftrightarrow s$)	Measures logical entailment errors between any generated reasoning h and the source context s: $1 - \max_{i=1..N} \max_{j=1..T} p_{\text{contr}}(h_i, s_j)$. Specifically, for each reasoning step we measure the probability that it contradicts any sentence in the context. We take the maximum probability of contradiction over all steps, following the logic that a contradiction anywhere in the reasoning chain signals a failure of the overall argument.
Perplexity-Chain (h)	Average perplexity of all tokens in the generated reasoning steps: $1/\text{PPL}(h)$. The context used to score each token is the previous tokens in the current and all previous steps. Steps are joined with a space character. To keep the range and orientation consistent with the other scores, we invert the perplexity.
Perplexity-Step (h_i)	Average perplexity of all tokens in the generated reasoning steps, where the context used to score each token is only the previous tokens within the current step: $1/[(1/N)\sum_{i=0} \text{PPL}(h_i)]$. To keep the range and orientation consistent with the other scores, we invert the perplexity.
Grammar (h_i)	Probability of grammatical acceptability of each step, averaged over all steps: $(1/N)\sum_{i=0} p_{\text{gram}}(h_i)$.

Table 5: Description of ROSCOE metrics