

CafGa: Customizing Feature Attributions to Explain Language Models

Alan Boyle

Furui Cheng*

Vilém Zouhar

Mennatallah El-Assady

ETH Zurich

Abstract

Feature attribution methods, such as SHAP and LIME, explain machine learning model predictions by quantifying the influence of each input component. When applying feature attributions to explain language models, a basic question is defining the interpretable components. Traditional feature attribution methods, commonly treat individual words as atomic units. This is highly computationally inefficient for long-form text and fails to capture semantic information that spans multiple words. To address this, we present CafGa, an interactive tool for generating and evaluating feature attribution explanations at customizable granularities. CafGa supports customized segmentation with user interaction and visualizes the deletion and insertion curves for explanation assessments. Through a user study involving participants of various expertise, we confirm CafGa’s usefulness, particularly among LLM practitioners. Explanations created using CafGa were also perceived as more useful compared to those generated by two fully automatic baseline methods: PartitionSHAP and MExGen, suggesting the effectiveness of the system.

1 Introduction

Large Language Models (LLMs) continuously evolve in scale and capabilities across a wide range of tasks, such as question answering, reasoning, and text summarization (Kamalloo et al., 2023, *inter alia*). Meanwhile, their increasing complexity poses significant challenges in interpretability and human trust (Sun et al., 2024). Feature attribution methods, e.g. SHAP (Lundberg and Lee, 2017) and LIME (Ribeiro et al., 2016), offer a promising approach to explain LLM behaviors by quantifying the influences of each component in the

*Corresponding author

†CafGa is available as a pip-installable package `cafga`. The source code is hosted on <https://github.com/explain-llm/CafGa>, and a live demo is accessible at cafga.ivia.ch.

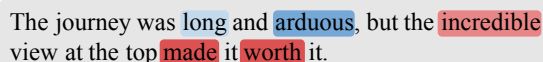


Figure 1: Feature attribution explanations quantify the contributions of each component (e.g., words), allowing users to validate the models’ reasoning.

input. These explanations calibrate users’ trust in model predictions (Bansal et al., 2021) and offer useful insights into the prediction shortcuts and model biases (Du et al., 2021; Ren and Xiong, 2023). Defining interpretable components is a fundamental question in generating meaningful feature attributions (Ribeiro et al., 2016). Traditional removal-based methods typically treat individual words as the basic units of interpretation (Figure 1), but this approach struggles to scale effectively for long-form text due to computational inefficiency and the fact that meaningful semantic cues often span multiple tokens or phrases, such as “*made it worth it*” in Figure 1. To address this, we propose CafGa, an interactive tool that enables users to create feature attribution explanations at customized levels of granularity. The tool offers default options to segment text at the word, sentence, or paragraph level, and supports users to further refine these segments by interactively isolating text spans. CafGa also provides functionality to evaluate and compare different segmentation strategies. The system visualizes deletion and insertion curves, showing how the model’s prediction changes as components are removed or added based on their attribution rankings, and calculates the area under the perturbation curve as the fidelity scores.

We conducted a two-stage user study to evaluate the usability and effectiveness of CafGa. In the first phase, ten participants used CafGa to create explanations that helped them understand the model’s reasoning. Based on self-reported usability scores, users with machine learning expertise generally found the system easy to use and learn,

while novice users experienced some difficulty due to the need to grasp new concepts before effectively interacting with the tool. Despite this, participants across all expertise levels reported that CafGa was helpful in improving their understanding of the LLM. In the second phase, we invited four expert users to compare the usefulness of the CafGa-generated explanations from the first phase with those produced by two automated baselines: PartitionSHAP (Lundberg, 2024) and MExGen (Paes et al., 2024). In 64% of the comparisons, participants rated the human-generated explanations as the most helpful, demonstrating the effectiveness of the proposed system in supporting meaningful model interpretation.

Input ⓘ

This restaurant is known as a premier location for meat-lovers. The chef originally worked at a BBQ restaurant where she became intimately familiar with all things meat. It was on her first trip to France that she was inspired to specialize in steak tartare, the restaurant's signature and only dish. The restaurant offers a few variations, but the customers appreciate the simplicity.

My favourite food is beef. I particularly love steak, which I prefer well-done, because I don't want to eat raw meat. I usually only season the steak with salt, because I appreciate the simplicity of just letting the beef speak for itself.

Template ⓘ

Below is a description of a restaurant and a customer.
{input}

Do you recommend this restaurant for the customer?
Answer with "yes" or "no".

Target Answer

Operator ⓘ

Figure 2: The user creates a task where a model is asked to decide whether to recommend a restaurant to a customer.

2 System Design

CafGa is an interactive system that includes a *task creation page* for defining prediction tasks and customizing explanation granularities, and an *explanation page* that displays explanations and their evaluation results. The following sections introduce the key features and design.

Define a Prediction Task. CafGa allows users to define a prediction task by constructing a prompt and defining an evaluator (Figure 2). To construct

the prompt, users need to complete the template field with the structure of the prompt and the input field with the content they wish to explain. We use the design in ChainForge (Arawjo et al., 2024) where an evaluator is used to structure the text generated by the model by transforming the text into a boolean value based on user-defined rules. To define the evaluator, users can choose from predefined operators such as Contains, which returns a boolean value indicating whether the text includes the specified Target Answer, and Entails, which assesses whether the text logically entails the Target Answer. See the full list of operators in Appendix B.

Customize Text Segmentation. CafGa supports default options for creating text segments, such as dividing text by word, sentence, or paragraph. Users can customize the segmentation through interactions. For example, they could brush to select a phrase and isolate the selected phrase from the sentence (Figure 4). The segments may not overlap to avoid ambiguous attributions.

Calculate and View Feature Attributions. CafGa uses the KernelSHAP (Lundberg and Lee, 2017) algorithm to sample and compute the feature attributions. The system generates random samples by removing text segments in the user-specified granularities. For each perturbed sample, the system requests ten responses, which are converted into Boolean values using the user-defined evaluator. The estimated probability, e.g., $P(\text{the answer contains "no"})$ is computed as the proportion of responses evaluated as true. Finally, the system uses a weighted linear regression model to estimate the Shapley values of each group. This step runs in the back-end and is hidden from users. After the calculation, we visualize the explanations using a heatmap with a color-blind friendly color schema (Figure 3B) accompanied by the task description (Figure 3A).

Evaluate Explanations. To ensure that the created explanations accurately reflect the model’s decision making and mitigate users’ confirmation bias, we adapt a commonly used and general fidelity metric, Area Over the Perturbation Curve (Samek et al., 2017). Following Petsiuk et al. (2018), we employ two variants of the perturbation curve: *Deletion* and *Insertion*. Deletion begins with all the features present and then iteratively removes the highest valued features. Insertion begins

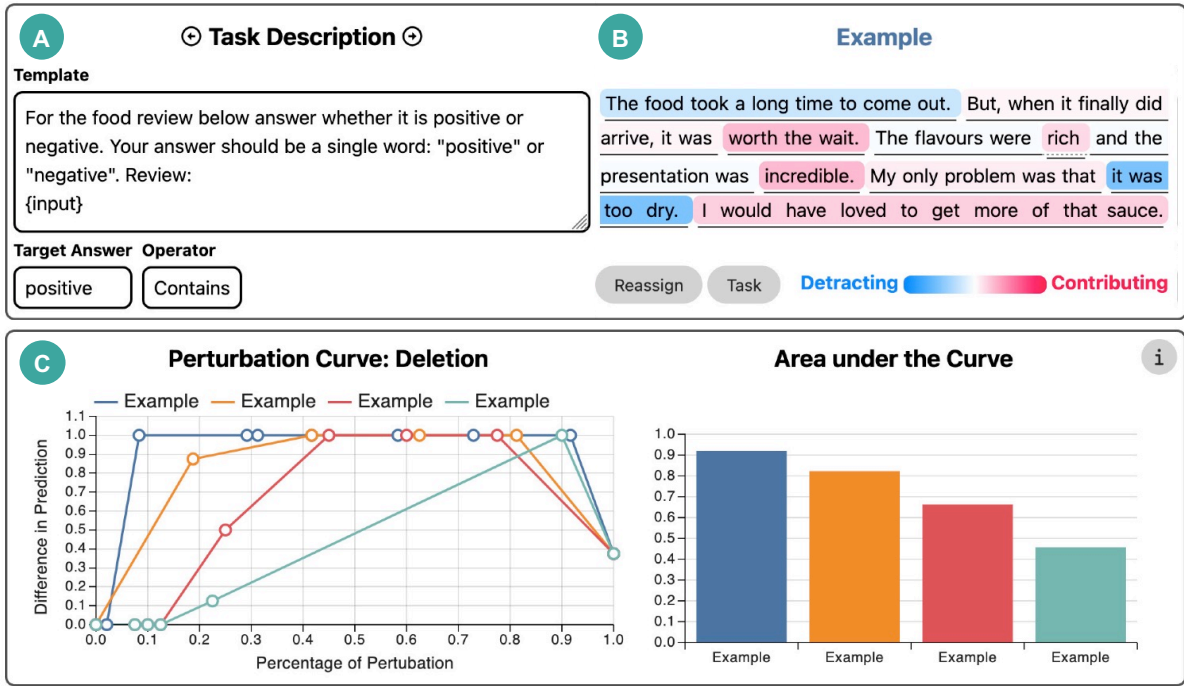


Figure 3: An overview of CafGa: The user first creates a predictive task (A) and then decides the granularities for explanations by assigning words into groups and gets an explanation based on the assignments (B). Using the perturbation curve the user can validate the fidelity of the generated explanation (C).

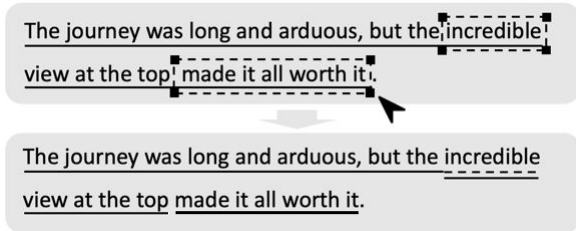


Figure 4: CafGa allows users to interactively segment text to customize the explanation granularity.

with no features present and iteratively inserts the highest valued features. In the resulting graph the x-axis represents the percentage of words added or removed and the y-axis the difference between the original prediction $f(x)$ and the perturbed prediction $f(x^k)$ when perturbing the top k features. Importantly, we plot the percentage of *words* perturbed on the x-axis rather than the percentage of *groups* to avoid a bias towards large groupings. For deletion the area under the curve should be large as the perturbed prediction should quickly diverge from the original one. For insertion it should be small as the perturbed prediction should quickly converge back to the original one. These metrics are visualized in an intuitive way to users such that they can follow and compare the computation exactly (Figure 3C).

Implementation Details. CafGa is implemented as a web application composed of a front-end and a back-end communicating via HTTP requests. The front-end is written in TypeScript and React and served via Vite. The back-end server uses FastAPI and Uvicorn. The model used in the back-end can be chosen by the user from a selection of OpenAI models. In the implementation of the KernelSHAP algorithm, we use the sampling strategy and functions from the SHAP library. This implementation uses $2 * n_{features} + 2048$ as the maximum number of samples used to approximate the shapley value. In the interactive setting we prioritize user experience and thus set the number of samples so that the explanation can be provided in a reasonable time t_{max} . Given a rate of API requests r_{API} , the maximum number of samples is defined as $n_{samples} = t_{max} * r_{API}$. We provide a PyPI package that contains all of the algorithms used in the back-end. The package also comes with Jupyter widgets created using AnyWidget (Manz et al., 2024) that recreate parts of the front-end. A Jupyter Notebook example is shown in Appendix A. This in particular allows users to also run CafGa on their local models.

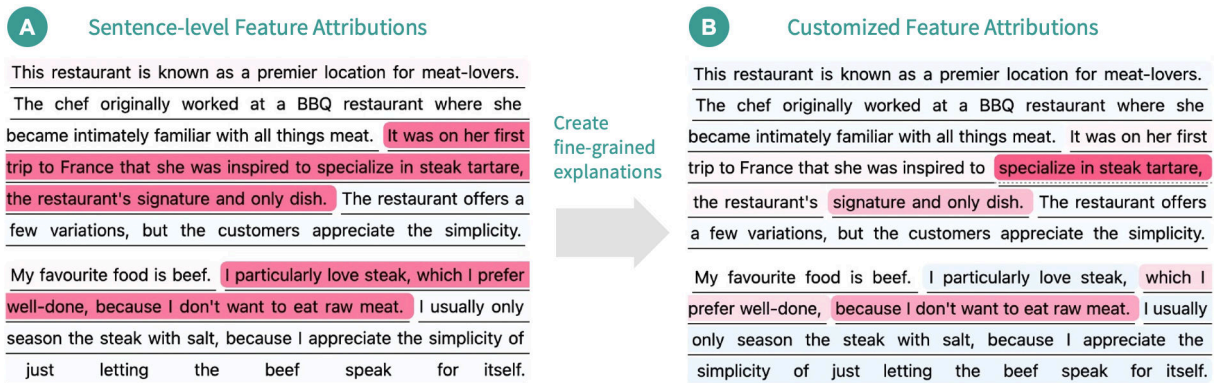


Figure 5: The user first used the sentence preset (A). The resulting explanation gives a rough overview of the model’s reasoning. The user specifically assigned all relevant parts into separate groups (B). In (B), one can now clearly see the reason: tartare is the restaurant’s sole signature dish, but the customer does not want to eat raw meat.

3 A Worked-out Example

We introduce a use case in understanding GPT4o-mini’s predictions in multi-hop reasoning. In this task, the model needs to decide whether to recommend the restaurant to a customer based on the restaurant’s review and the customer’s preferences. The model makes a correct prediction that does not recommend the restaurant. The user, an LLM developer, wanted to understand if the model followed the correct reasoning behind this decision.

The user first defined the task (see Figure 2) and used a preset that calculates sentence-level attributions. From the results (see Figure 5 A), the user noticed that the two sentences with the highest attribution include the necessary information for making this decision – *the restaurant only serves steak tartare, while the customer doesn’t like raw meat* (see highlighted parts in Figure 5 B). The user found this explanation unsatisfactory and wanted to know if the model specifically captured the key information that *steak tartare is the only dish served* in these sentences.

So the user then manually separated the sentence into segments, like *specialize in steak tartare, signature and only dish*, to get a more precise explanation. The new explanation, with a higher fidelity (from 0.77 to 0.96) suggests that all the necessary information is well-captured by the model and has high contributions to the prediction. From this two-stage exploration, the user confirmed that the model follows the correct reasoning in this decision.

4 User Study

To evaluate the usability of CafGa, we conducted a two-stage user study in which participants cre-

ated and assessed explanations. In the first stage, participants used CafGa to generate customized explanations and then provided feedback on the tool’s usability. In the second stage, experts compared these participant-generated explanations with those produced by existing fully automatic methods to assess the quality of the generated explanations.

4.1 Setup

We now describe the two parts of the user study.

Creating explanations. For the first stage we recruited ten participants: six users who had experience working with machine learning, noted as experts and four novice users. None of the participants had any prior experience working with attribution-based explanations.

At the beginning of the study, the participants were shown a video that explained the concepts behind CafGa and their role in this part of the study. The video also showed an example task that the participants could work through while following the video. After viewing the video participants were encouraged to ask questions. The participants then created explanations for five types of tasks of increasing complexity. The tasks were local question answering taken from SQuAD (Rajpurkar et al., 2016), sentiment analysis on reviews taken from the YELP academic dataset (YELP, 2015), a few-shot prompt engineering task inspired by Tenney et al. (2024), multi-hop reasoning taken from HotpotQA (Yang et al., 2018) and long-form text comprehension taken from BARQA (Hou, 2020) (see Appendix C for details). For each task type there were five tasks except for prompt-engineering and long-form text comprehension, which each had three tasks. The given task for each type was se-

lected at random. Once participants had created an explanation for each type of task, they filled out a survey rating the usability and usefulness of CafGa.

Comparing explanations. In the second stage, we recruited four expert participants who all studied machine learning and currently work with AI. The participants were again shown a video that explained the concepts behind CafGa and their role in this part of the study. The participants were asked to choose the most helpful explanations among 10 groups. Each group contained three feature attributions: a human-created explanation from the first stage of the study, one created by MExGen (Paes et al., 2024), and one created by PartitionSHAP (Lundberg, 2024). We choose MExGen because it provides a good baseline of what static granularities can achieve and PartitionSHAP because it is often cited as a comparison (Amara et al., 2024; Paes et al., 2024; Mosca et al., 2022) and is recommended as an efficient method by Mosca et al. (2022). Unlike most methods they also scale to long inputs like those in BARQA. Participants were asked to select the most helpful explanation – the one that best informed them of the model’s decision logic and correctness – among the three.

4.2 Study Results

From the usability survey results (Table 1), we observed that the experts could use the system well. They generally found the system easy to learn and easy to use. In comparison, non-expert users faced more challenges in learning the system, primarily due to difficulties in quickly understanding the underlying concepts of CafGa. However, after getting familiar with the system, both groups of participants reported that the system is helpful in supporting them in understanding the LLM.

From the comparative study results (Table 2), we find that the explanations made by humans were generally preferred, with MExGen outperforming PartitionSHAP. We see that in total, the explanations created by humans were preferred 64% of the time over MExGen and PartitionSHAP. This demonstrates that CafGa is effective in supporting meaningful model interpretation.

Task-level preferences (Table 2) show that human-customized explanations performed especially well on the HotpotQA task. This is likely because the input text in these tasks typically contains multiple facts, often expressed through distinct clauses and phrases. The participants can make

Statement	Non-experts	Experts
The system is easy to use.	3.0	5.0
The system is easy to learn.	3.5	5.2
The system is enjoyable to use.	3.5	5.8
The various functions in the system are well integrated.	5.0	5.5
The system is helpful in understanding the LLM.	6.3	5.8
I would like to use this system frequently.	3.3	4.0

Table 1: After-task survey results. Participants were asked to rate the statements on a scale from 1 (strongly disagree) to 7 (strongly agree) with 4 (neutral). For each statement, we show the distribution and average.

	Human	MExGen	PartitionSHAP
SQuAD	63%	38%	0%
YELP	67%	33%	0%
Prompt	50%	0%	50%
HotpotQA	75%	25%	0%
BARQA	63%	38%	0%
Average	64%	28%	8%

Table 2: Preference win-rate for generated explanations across tasks.

semantic segmentations to the input text, while automatic methods create unreasonable n-grams that do not align with human intuitions.

5 Related Work

In this section we discuss related work on creating feature attributions at interpretable granularities and existing user interfaces.

5.1 Interpretable Feature Attributions

Recent work in attribution-based explanations has focused on creating attributions at granularities beyond the word level. This approach can not only improve computational efficiency, but also make for more interpretable explanations.

Early work in this direction focuses on finding specific decompositions of the neural network to get contextual attributions in addition to token-level attributions (Murdoch et al., 2018; Singh et al., 2019; Jin et al., 2019). However, given that there are now many settings where one does not have access to the model’s internal structure, such model-specific methods may no longer be applicable. Thus, there has been a shift towards model-agnostic methods that do not rely on the model’s

internal structure.

Chen et al. (2020) present a model-agnostic approach that iteratively splits the input sequence into smaller groups generating an attribution for each group. This hierarchical approach in particular allows users to validate whether the model can reason about the compositional nature of language.

Ju et al. (2023) further improve on this by addressing the limitation that the spans of grouped features must be contiguous. This is important because modern NLP architectures can model long-range non-contiguous dependencies in the input. However, both of these methods struggle to scale to large input sequences.

Paes et al. (2024) accordingly propose a method in which they first group the input at the coarsest granularity (e.g., paragraphs) and then iteratively refine the granularity only for the highest attributed group. This approach allows them to greatly reduce the computation time, but it can fail to precisely pinpoint the important parts of the input that are in the groups which do not get the highest attribution score. It also relies on the assumption that the fixed granularities used for the explanation are interpretable.

Cheng et al. (2025) emphasize the importance of the human understanding of language in creating explanations. They thus group the features in the input by use of a syntactic tree allowing them to make semantically meaningful groups.

Despite research into improving explanations by attributing features at differing granularities, little work has been done to directly involve humans in the creation of such explanations. We close this gap by empowering users to customize the attributions in a way that best suits their needs, while still informing them of the fidelity of the created explanations to stay aligned with the model.

5.2 LLM Explanation Interfaces

Recent research has focused on helping users gain a deeper understanding of LLMs through interactive visual interfaces.

Many of these interfaces target specific neural networks and explain the model by visualizing the architecture and the model’s inner computations (Zeiler and Fergus, 2013; Ming et al., 2017; Strobel et al., 2016, 2018).

However, as it may not always be possible to access the model’s internal structure, there has also been a rise in model-agnostic LLM interfaces (Cos-

cia and Endert, 2024; Cheng et al., 2024; Arawjo et al., 2024; Kahng et al., 2024). Many of these tools allow users to investigate *what* a model will answer, but they do not provide users with precise tools to explain *why* a model answers that way.

LLM Checkup (Wang et al., 2024) allows users to explore the model’s reasoning by chatting with the model. This approach relies on model self-explanation, however, which can be misleading as the explanations often lack fidelity (Madsen et al., 2024; Turpin et al., 2023).

Closely related to our work, Cheng et al. (2025) analyze the LLM’s predictions by perturbing the input with meaningful counterfactuals. They use a syntactic tree to define semantically meaningful segments in the input which can be ablated in a perturbation. This allows users to perform an analysis similar to attribution-based explanation, as they can test which of the defined segments are relevant for the model’s prediction.

Tenney et al. (2024) also present an interface where users can choose the granularity of feature attribution. However, their tool relies on model-specific methods limiting its applicability and only allows users to specify the features at fixed granularities or by use of a regex, which means customizing explanations is largely not possible.

We further emphasize the need for users to be able to fully customize the explanation rather than just choosing from a limited set of options. Our model-agnostic approach also makes our system more broadly applicable and the use of the fidelity metric allows users to make certain that their custom explanations have high fidelity.

6 Conclusion

We introduce CafGa, a tool that enables users to generate explanations at arbitrary granularities by grouping words in a customized manner. We argue that using CafGa users can create more interpretable explanations. CafGa also provides users with a fidelity metric to ensure that the created explanations still align with the model’s decision making. Through a case study and a user study, we demonstrate CafGa’s usability and effectiveness, finding that users successfully created explanations that were preferred over automatic methods. This highlights the value of human involvement in creating interpretable explanations.

Limitations

Our user study was limited by a small sample size due to the constrained number of available participants. In future work, we would like to conduct a larger-scale evaluation. In particular, we believe that evaluating the practical use of CafGa in real-world, non-laboratory settings would provide valuable insights and offer a more accurate assessment of its usability and effectiveness.

Approximating the Shapley value involves a trade-off between speed and accuracy. Using fewer groups increases computational efficiency but risks obscuring important information within those groups. Conversely, creating many word groups improves fidelity but substantially reduces speed. Providing users with appropriate guidance in balancing this trade-off and thereby determining a suitable number of groups remains an important direction for future investigation.

We use deletion and insertion as fidelity metrics which are the most common fidelity metrics for attribution-based explanations and can be intuitively visualized for users. However, these metrics introduce a degree of circularity as both the explanation and the evaluation are based on perturbation. Their applicability also becomes questionable when features represent groups rather than atomic units (i.e., words). To mitigate this, we plot the results with respect to atomic units rather than grouped features. Nonetheless, further research is needed to develop more reliable fidelity metrics.

Ethics Statement

The participants were sourced from a pool of academic labs, which work on reciprocal basis instead of monetary rewards.

References

- Kenza Amara, Rita Sevastjanova, and Mennatallah El-Assady. 2024. [SyntaxShap: Syntax-aware explainability method for text generation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4551–4566. Association for Computational Linguistics.
- Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. [Chainforge: A visual toolkit for prompt engineering and LLM hypothesis testing](#). In *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI '24*, page 1–18. ACM.
- Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. [Does the whole exceed its parts? the effect of ai explanations on complementary team performance](#). In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, CHI '21*, New York, NY, USA. Association for Computing Machinery.
- Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. [Generating hierarchical explanations on text classification via feature interaction detection](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593. Association for Computational Linguistics.
- Furui Cheng, Vilém Zouhar, Robin Shing Moon Chan, Daniel Furst, Hendrik Strobelt, and Mennatallah El-Assady. 2025. [Understanding large language model behaviors through interactive counterfactual generation and analysis](#). *IEEE Transactions on Visualization and Computer Graphics*.
- Furui Cheng, Vilém Zouhar, Simran Arora, Mrinmaya Sachan, Hendrik Strobelt, and Mennatallah El-Assady. 2024. [RELIC: Investigating large language model responses using self-consistency](#). In *Proceedings of the 2024 CHI conference on human factors in computing systems*.
- Adam Coscia and Alex Endert. 2024. [KnowledgeVIS: Interpreting language models by comparing fill-in-the-blank prompts](#). *IEEE Transactions on Visualization and Computer Graphics*, 30(9):6520–6532.
- Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. 2021. [Towards interpreting and mitigating shortcut learning behavior of NLU models](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929. Association for Computational Linguistics.
- Yufang Hou. 2020. [Bridging anaphora resolution as question answering](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1428–1438. Association for Computational Linguistics.
- Xisen Jin, Junyi Du, Zhongyu Wei, Xiangyang Xue, and Xiang Ren. 2019. [Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models](#). *CoRR*, abs/1911.06194.
- Yiming Ju, Yuanzhe Zhang, Kang Liu, and Jun Zhao. 2023. [A hierarchical explanation generation method based on feature interaction detection](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12600–12611. Association for Computational Linguistics.
- Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif,

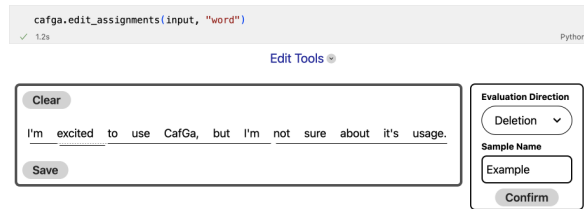
- Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2024. [LLM comparator: Visual analytics for side-by-side evaluation of large language models](#). In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA.
- Ehsan Kamaloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. [Evaluating open-domain question answering in the era of large language models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606. Association for Computational Linguistics.
- Scott M. Lundberg. 2024. [Exact explainer — SHAP latest documentation](#).
- Scott M. Lundberg and Su-In Lee. 2017. [A unified approach to interpreting model predictions](#). In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.
- Andreas Madsen, Sarath Chandar, and Siva Reddy. 2024. [Are self-explanations from large language models faithful?](#) In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 295–337. Association for Computational Linguistics.
- Trevor Manz, Nezar Abdennur, and Nils Gehlenborg. 2024. [anywidget: reusable widgets for interactive analysis and visualization in computational notebooks](#). *Journal of Open Source Software*, 9(102):6939. Publisher: The Open Journal.
- Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. 2017. [Understanding hidden memories of recurrent neural networks](#). In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–24.
- Edoardo Mosca, Ferenc Szegedi, Stella Tragianni, Daniel Gallagher, and Georg Groh. 2022. [SHAP-based explanation methods: A review for NLP interpretability](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4593–4603, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. [Beyond word importance: Contextual decomposition to extract interactions from LSTMs](#). In *International Conference on Learning Representations*.
- Lucas Monteiro Paes, Dennis Wei, Hyo Jin Do, Hendrik Strobelt, Ronny Luss, Amit Dhurandhar, Manish Nagireddy, Karthikeyan Natesan Ramamurthy, Prasanna Sattigeri, Werner Geyer, and Soumya Ghosh. 2024. [Multi-level explanations for generative language models](#). *ArXiv*, abs/2403.14459.
- Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. [RISE: Randomized input sampling for explanation of black-box models](#). *ArXiv*, abs/1806.07421.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Yuqi Ren and Deyi Xiong. 2023. [HuaSLIM: Human attention motivated shortcut learning identification and mitigation for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12350–12365. Association for Computational Linguistics.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. ["why should i trust you?": Explaining the predictions of any classifier](#). In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, page 1135–1144, New York, NY, USA.
- Wojciech Samek, Alexander Binder, Gregoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2017. [Evaluating the visualization of what a deep neural network has learned](#). *IEEE Transactions on Neural Networks and Learning Systems*, 28:2660–2673.
- Damien Sileo. 2024. [tasksource: A large collection of NLP tasks with a structured dataset preprocessing framework](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 15655–15684, Torino, Italia. ELRA and ICCL.
- Chandan Singh, W. James Murdoch, and Bin Yu. 2019. [Hierarchical interpretations for neural network predictions](#). In *International Conference on Learning Representations*.
- Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander M. Rush. 2018. [Seq2seq-vis: A visual debugging tool for sequence-to-sequence models](#). *CoRR*, abs/1804.09299.
- Hendrik Strobelt, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M. Rush. 2016. [Visual analysis of hidden state dynamics in recurrent neural networks](#). *CoRR*, abs/1606.07461.
- Lichao Sun, Yue Huang, Haoran Wang, Siyuan Wu, Qihui Zhang, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao, Chunyuan Li, Eric P. Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, John C. Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan

- Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Wang, Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, and Yue Zhao. 2024. [TrustLLM: Trustworthiness in large language models](#). *CoRR*, abs/2401.05561.
- Ian Tenney, Ryan Mullins, Bin Du, Shree Pandya, Min-suk Kahng, and Lucas Dixon. 2024. [Interactive prompt debugging with sequence salience](#).
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: unfaithful explanations in chain-of-thought prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Qianli Wang, Tatiana Anikina, Nils Feldhus, Josef Genabith, Leonhard Hennig, and Sebastian Möller. 2024. [LLMCheckup: Conversational examination of large language models via interpretability tools and self-explanations](#). In *Proceedings of the Third Workshop on Bridging Human-Computer Interaction and Natural Language Processing*, pages 89–104. Association for Computational Linguistics.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- YELP. 2015. [Yelp academic dataset](#).
- Matthew D. Zeiler and Rob Fergus. 2013. [Visualizing and understanding convolutional networks](#). *CoRR*, abs/1311.2901.

A Jupyter Notebook Usage

Editing the Assignments

To define the groups in the input over which you would like to get attributions you can use the edit widget to define the assignment of input segments to groups.



Visualizing the Explanation

Finally, we would like to visualize the explanation. Since we used a two-dimensional scalarizer we default to get two explanations. Set the scalarizer_index to the index of the scalarizer for which you would like to see the explanation.

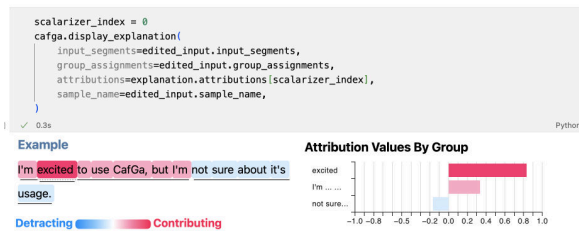


Figure 6: The assignment editor and the attributed text display as Jupyter notebook widgets.

B Operator Descriptions

The operators provided in CafGa can be seen in Table 3. For the boolean operators we sample 10 model responses and use the percentage of times the operator is true to evaluate the models response. For the logical operators we sample a single response and use a DeBERTa-based NLI model (Sileo, 2024) to evaluate the response.

Name	Description
Contains	Checks whether the response contains the target answer
Equals	Checks whether the response equals the target answer
Starts With	Checks whether the response starts with the target answer
Ends With	Checks whether the response ends with the target answer
Entails	Checks whether the response logically entails the target answer
Contradicts	Checks whether the response logically contradicts the target answer
Semantically Equals	Checks whether the response semantically equals the target answer by applying entailment in both directions.

Table 3: The operators available in CafGa.

C Task Descriptions

SQuAD: We take local question answering tasks from SQuAD (Rajpurkar et al., 2016) and place the question in the template and the context in the input. The model is asked to answer the question given the context.

YELP: We take reviews from the YELP academic dataset (YELP, 2015) and ask the model to predict whether the review is of positive or negative sentiment. We place the review in the input and the prediction instructions in the template.

Prompt: Inspired by the example presented in Tenney et al. (2024) we create a set of few-shot prompts that contain errors. The explanation can be used to detect the errors and also to note parts in the instructions that cause the model to recreate the errors in the examples. We put the instructions and the examples in the input. The template only contains the new sample for which the user wants the model to follow the prompt.

HotpotQA: To create complex questions we use Hotpot QA Yang et al. (2018), which contains questions that require chaining multiple supporting facts. We put supporting facts, potentially misleading facts and the question in the input. The template only contains the instructions to answer the question.

BARQA: For long-form text comprehension we use examples from the Bridging Anaphora dataset Hou (2020). We place the article in the input and the question in the template.