Multi-Level Explanations for Generative Language Models

Lucas Monteiro Paes[*1], **Dennis Wei**[*2], **Hyo Jin Do**[2], **Hendrik Strobelt**[2], **Ronny Luss**[2], **Amit Dhurandhar**[2], **Manish Nagireddy**[2], **Karthikeyan Natesan Ramamurthy**[2], **Prasanna Sattigeri**[2], **Werner Geyer**[2], **Soumya Ghosh**[†3]

[1]Harvard University [2]IBM Research [3]Merck Research Labs

lucaspaes@g.harvard.edu, dwei@us.ibm.com

Abstract

Despite the increasing use of large language models (LLMs) for context-grounded tasks like summarization and question-answering, understanding what makes an LLM produce a certain response is challenging. We propose Multi-Level Explanations for Generative Language Models (MExGen), a technique to provide explanations for context-grounded text generation. MExGen assigns scores to parts of the context to quantify their influence on the model's output. It extends attribution methods like LIME and SHAP to LLMs used in context-grounded tasks where (1) inference cost is high, (2) input text is long, and (3) the output is text. We conduct a systematic evaluation, both automated and human, of perturbation-based attribution methods for summarization and question answering. The results show that our framework can provide more faithful explanations of generated output than available alternatives, including LLM self-explanations. We open-source code for MExGen as part of the ICX360 toolkit: https://github.com/IBM/ICX360.

1 Introduction

Large language models (LLMs) are being deployed to generate text used for decision-making, e.g., summarizing meetings (Laskar et al., 2023), extracting key points from legal documents (Kanapala et al., 2017), and answering doctors' questions (Xiong et al., 2024). In these applications, the LLM is grounded in *context* (e.g., legal documents) provided as part of the input. Given an LLM response, users may wish to know which parts of the context were responsible for the response, or whether the response is grounded at all (Huang et al., 2023a). We propose **M**ulti-level **Ex**planations for **Gen**erative Language Models (MExGen) to fulfill this user necessity by providing explanations for context-grounded text generation.

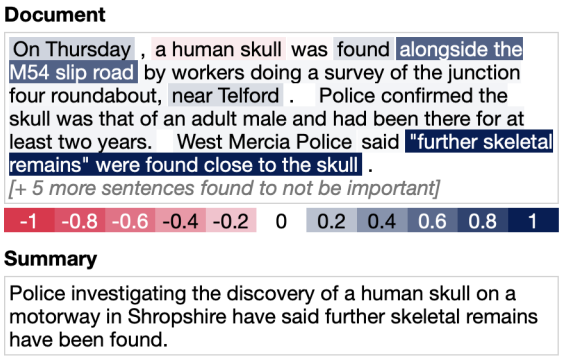


Figure 1: Explanation generated using MExGen C-LIME (with *BERT* scalarizer) for a summarization example. The most important phrases found by MExGen (darker blue) suggest that the LLM closely paraphrases text (*further skeletal remains*) and also abstracts concepts (*M54* → *motorway*).

MExGen computes explanations by comparing multiple model predictions given perturbed versions of the input context. The explanations take the form of attribution scores assigned to parts of the input, quantifying the effect of each part on the model output. Accordingly, we also refer to this type of explanation as *input attribution*. MExGen generalizes popular perturbation-based explanation methods such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) to generative LLMs. Such methods are widely used for text classification (Chen et al., 2020; Kim et al., 2020; Mosca et al., 2022; Ju et al., 2023), but their application to generative LLMs is still limited (Section 2).

Perturbation-based input attribution for generative LLMs presents challenges related to having (1) text outputs, (2) high inference cost, and (3) long text inputs. MExGen provides a framework to address these challenges. The framework can be instantiated with different attribution algorithms, and we do so using LIME- and SHAP-like algorithms.

Challenge of output text The first challenge is that LLMs output text rather than a real number

[*]Denotes equal contribution in alphabetical order.

[†]Work done while at IBM Research.

(e.g., the predicted log probability of a class). Attribution algorithms require a real-valued function to quantify that function's sensitivity to different inputs. We address this through the concept of *scalarizers*, functions that map output text to real numbers. We investigate and compare multiple scalarizers to provide guidance on this choice.

Importantly, most scalarizers that we consider address the truly "black-box" setting in which we receive only text as output from the model. This is common with LLMs that only provide API access or are proprietary.

Challenge of input length and LLM inference The input text can be long for context-grounded generative tasks, e.g., whole papers or news articles. Generating explanations for long inputs requires more model inferences, imposing higher computational and financial costs. Long inputs also pose interpretation issues, since attributions that are too fine-grained may be less interpretable to a user.

We address this challenge in three ways: *(1) Linguistic segmentation:* We segment the input text into linguistic units at multiple levels, for example sentences, phrases, and words. *(2) Multi-level attribution:* We use a refinement strategy that proceeds from attributing at a coarser level like paragraphs to a finer level like words, only refining the most important parts of the context. This controls the number of model inferences and resembles a binary search. *(3) Linear-complexity algorithms:* We instantiate our framework with attribution algorithms whose numbers of model inferences scale linearly with the number of units (e.g., number of sentences). We also propose a linear-complexity variant of LIME called C-LIME that limits the number of units perturbed at one time.

Evaluation We evaluate instantiations of `MExGen` on summarization and context-grounded QA tasks. We show that `MExGen` provides more faithful explanations compared to baselines (including LLM self-explanation), assigning higher importance to input parts that have the greatest effect on the model output. We also find that certain scalarizers yield similar explanations to each other. Some scalarizers that only use text generate explanations close to those that depend on log probabilities. Human evaluation corroborates the automated evaluation findings and also indicates that certain scalarizers and attribution methods, previously considered similar to others in the automated evaluation, were perceived as more faithful by users.

Our **main contributions** are:

- We propose the `MExGen` framework to extend perturbation-based input attribution to generative language models, with a multi-level strategy to combat the challenges of long inputs.
- We compare several scalarizers for mapping output text to real numbers, notably handling the case of text-only output from the model.
- We conduct a systematic evaluation, both automated and human, of input attribution methods for summarization and QA, showing that `MExGen` can provide more faithful explanations of generated output than the available alternatives. This advantage extends to self-explanations of LLMs (Huang et al., 2023b; Madsen et al., 2024), even from powerful LLMs such as DeepSeek-V3.

2 Related Work

We discuss works on perturbation-based explanations and self-explanations for *generative* LMs[1]. The literature on the former for generative LMs is limited, as corroborated by Mosca et al. (2022).

Perturbation-based methods `PartitionSHAP` in the SHAP library handles long inputs by dividing them into token spans and assigning the same score to each token in a span. `PartitionSHAP` produces separate attributions for each output token. This approach is less interpretable because it requires the selection of an output token and assigns multiple attribution scores to each input span. Although `PartitionSHAP` supports API-only access (SHAP, 2024b), understanding its operation required substantial effort (see Appendix B.1).

`CaptumLIME` is a modification of LIME (Ribeiro et al., 2016) tailored for text generation tasks in the Captum library (Miglani et al., 2023).[2] `CaptumLIME` allows the user to manually define units for attribution within the input. It handles text outputs by computing the log probability of the output. This does, however, require access to output probabilities, so `CaptumLIME` is not suitable for the text-only setting.

`TextGenSHAP` (Enouen et al., 2023) offers a more efficient sampling strategy for Shapley value estimation based on speculative decoding (Leviathan et al., 2023). As with `CaptumLIME`, such computations also require access to output probabilities. Also, `TextGenSHAP` is tailored to SHAP.

[1] Please see Appendix A for other forms of explanation.

[2] Captum also has variants of SHAP but we found them slow to run and obviated by `PartitionSHAP`.

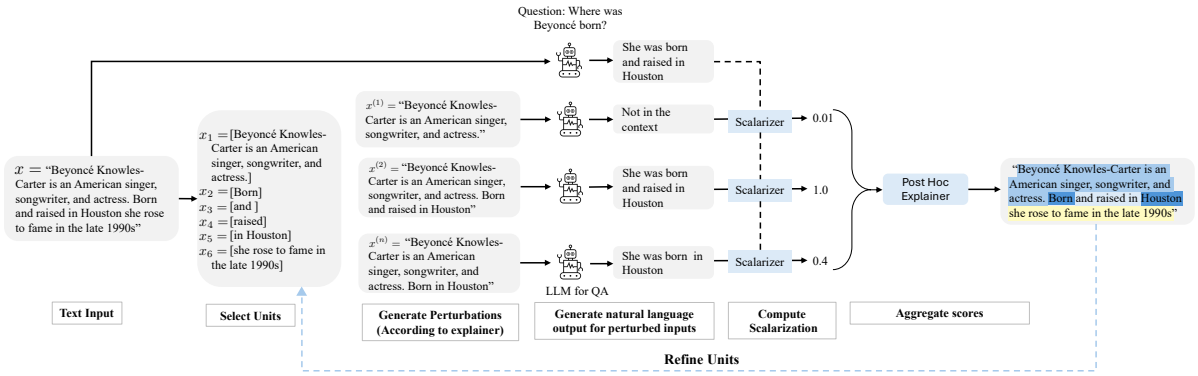


Figure 2: Diagram showing the workflow of MExGen.

Table 3 shows that the above methods each lack capabilities that are offered by MExGen.

Self-explanation methods These methods prompt the model to explain its predictions (e.g., by ranking the most important parts of its context) (Camburu et al., 2018; Huang et al., 2023b; Kroeger et al., 2024; Madsen et al., 2024). However, prior work has found that these self-explanations may be less faithful when used for in-context classification tasks (Madsen et al., 2024; Huang et al., 2023b). Fragkathoulas and Chlapanis (2024) demonstrated that self-explanations are not as faithful as numerical attributions (like MExGen) in identifying keywords for a QA task. We generalize their work to additional generative tasks and to self-explanations in the form of rankings.

3 Multi-Level Explanations for Generative Language Models

This section describes the proposed MExGen framework. Figure 2 provides an overview of MExGen.

In the setting of perturbation-based input attribution, we are given a generative LM f, an input text sequence of interest x^o (left side of Figure 2, superscript o for "original"), and a generated output $y^o = f(x^o)$ that is also a text sequence (top center of Figure 2). Our goal is to explain y^o (the *target output* for explanation) by attributing to parts of the input x^o. Each part of the input, denoted x_s, $s = 1, \ldots, d$, is to be assigned an attribution score ξ_s (represented by color on the right of Figure 2) quantifying the importance of x_s in generating the output, in the sense that if important parts are perturbed, then the output will change significantly. As the second through fourth paths in Figure 2 indicate, model f can be queried on perturbations x of x^o, with no further access to f.

Generative language tasks pose two main challenges: having text as output, and potentially long text as input. The following two subsections discuss our solutions to these challenges.

3.1 Handling Text Outputs

Input attribution algorithms such as SHAP and LIME require a real-valued function as the object to explain (see Section 3.2, "Linear-complexity algorithms" for examples of how this function is used). Since the LM f may only output text, we introduce *scalarizers*, which are functions S (shown as blue boxes in Figure 2) that map output text back to real numbers. We consider two types of access to f: (a) *full logit access*, where f can provide predicted logits for all tokens in its vocabulary, at each position in the output sequence; (b) *text-only* access, where we are limited to text outputs. See Appendix B.1 for why we assume vocabulary-wide logits for (a).

Full logit access When all logits are available, we use the probability of generating the target output y^o as the function to explain. We refer to this as the *Log Prob* scalarizer. Given output sequence y^o of length ℓ and an arbitrary input sequence x, we compute the model's log probability of generating each target output token y^o_t conditioned on x and previous output tokens $y^o_{<t}$. We then average over output tokens to obtain the scalarized output for x,

$$S(x; y^o, f) = \frac{1}{\ell} \sum_{t=1}^{\ell} \log p\left(y^o_t \mid y^o_{<t}, x; f\right). \quad (1)$$

Here the scalarizer S is parameterized by y^o and f since the latter is providing predicted probabilities.

The *Log Prob* scalarizer generalizes the log probability used in explaining text classification. This is seen from (1) by setting $\ell = 1$ (single prediction) and identifying y^o_1 with the predicted class.

Text-only access Given only the output text $y = f(x)$ generated from the perturbed input x, we consider similarity measures $S(y; y^o)$ between y and the target output y^o as scalarizers. These now depend on f only via composition, i.e., $S(f(x); y^o)$, and are not parameterized by f as in (1). Appendix B.1 has further details on these scalarizers.

- *Sim*: Cosine similarity between embeddings of y and y^o (e.g., SentenceTransformers embeddings).
- *BERT*: BERTScore (Zhang et al., 2020) between y and y^o.
- *BART*: BARTScore ("faithfulness" version) between y and y^o (Yuan et al., 2021). Measures the probability of an auxiliary LM f_{BART} generating y^o given y as input.
- *SUMM*: Similar to the *BART* scalarizer with a summarization model as f_{BART} (SHAP, 2024b).
- *Log NLI*: Log-odds of entailment given y^o as premise and y as hypothesis, computed using a natural language inference (NLI) model.

3.2 Handling Long Text Inputs

Some generative language tasks require long input texts, such as in summarization and context-grounded QA. We address this challenge through a combination of three techniques: segmenting the input into linguistic units at multiple levels, using attribution algorithms with linear complexity in the number of input units, and obtaining attributions in a coarse-to-fine manner.

Linguistic segmentation We segment the input into linguistic units at multiple granularities: paragraphs, sentences, phrases, and words ("Select Units" box in Figure 2). This approach takes advantage of linguistic and other structure present in the input. For example, the input may already be broken into paragraphs or contain multiple distinct retrieved documents, in which case these paragraphs or documents can form the units at the highest level. In contrast, existing methods rely on the model's tokenizer, which can yield units (tokens) that are too fine, or treat the text as a flat sequence of tokens and let the algorithm decide how to segment it (Chen et al., 2020; SHAP, 2024a).

We use tokenization and dependency parsing from `spaCy` v3.6 (Honnibal et al., 2020) to segment paragraphs into sentences and words. To segment sentences into phrases, we implemented an algorithm that uses the dependency parse tree from `spaCy`. In the first pass, the algorithm recursively segments the tree and its subtrees into phrases that are no longer than a maximum phrase length. In the second pass, some short phrases are re-merged. More details are in Appendix B.2.

Our framework also allows for units at any level to be marked as not of interest for attribution. Textual elements like punctuation, prompt templates, and system prompts are usually not relevant for attribution and, therefore, marked as not of interest. Ultimately, the user may choose which parts of the input context are not of interest. Appendix C.3 gives more details on the choice of ignored units.

Linear-complexity algorithms Given a segmented input with possibly mixed units as in Figure 2, we have the task of attributing to units $x_1, \ldots, x_d$ (the "Post Hoc Explainer" block in Figure 2). Here, we consider only perturbation-based attribution algorithms that scale linearly with the number of units d in terms of model queries, to control this cost. We instantiate `MExGen` with three such methods: leave-one-out (`LOO`), a LIME-like algorithm with further constraints (`C-LIME`), and Local Shapley (`L-SHAP`). In this work, perturbing a unit means simply dropping it; see Appendix F for further discussion.

`LOO`: Units $x_1, \ldots, x_d$ are perturbed one at a time, yielding corresponding perturbed inputs $x^{(1)}, \ldots, x^{(d)}$. The attribution score for x_s is the decrease in scalarizer score due to leaving x_s out: $\xi_s = S(x^o; y^o, f) - S(x^{(s)}; y^o, f)$.

`C-LIME`: We use a linear model that operates on interpretable features z and approximates the model f in the vicinity of original input x^o. In our case, the interpretable features $z \in \{0, 1\}^d$ correspond to units $x_1, \ldots, x_d$, with $z_s = 0$ if unit s is perturbed and $z_s = 1$ otherwise. The linear model is fit using n perturbations $x^{(1)}, \ldots, x^{(n)}$ of x^o, with corresponding interpretable representations $z^{(1)}, \ldots, z^{(n)}$, and scalarized model outputs $S(x^{(1)}; y^o, f), \ldots, S(x^{(n)}; y^o, f)$:

$$\xi = \arg\min_w \sum_{i=1}^{n} \pi\left(z^{(i)}\right)\left(w^T z^{(i)} - S(x^{(i)}; y^o, f)\right)^2 + \lambda R(w), \quad (2)$$

where $\pi(z)$ is a sample weighting function and R is a regularizer. The best-fit linear model coefficients yield the attribution scores ξ.

We make two main departures from LIME, in addition to the use of a general scalarizer S as seen in (2). Firstly, we limit the number of perturbations n to a multiple of the number of units d. Since n

is the number of samples in the linear regression problem (2) while d is the number of parameters to fit, a ratio n/d of 5 or 10 can yield good results. LIME by default sets n to be in the thousands independently of d, which can be prohibitive for LLMs. Secondly, we limit the number of units K that can be perturbed simultaneously to a small integer. This concentrates the smaller number of perturbations on inputs that are closer to x^o, which aligns with work showing that doing so improves the fidelity of attributions (Tan et al., 2023). LIME by contrast samples the number of units to perturb uniformly from $\{1, \ldots, d\}$ (Mardaoui and Garreau, 2021).

`L-SHAP`: This variation of SHAP by Chen et al. (2019) limits perturbations to a local neighborhood around a unit of interest, overcoming SHAP's exponential complexity. We further extend `L-SHAP` by also limiting the number of simultaneously perturbed units (as in `C-LIME`). The equation for the SHAP score computation is given in (3). Please see Appendix B.3 for further details.

Multi-level explanations The multi-level approach is essential for obtaining fine-grained attributions to the most important units in the context without a drastic increase in the computational cost. Once we have computed attribution scores at a given level, we refine the input units and repeat the process (feedback path at the bottom of Figure 2), generating *multi-level* explanations. For example, given sentence-level attributions, we obtain word-level attributions for a few sentences and keep attributions at the sentence level for the remainder to avoid introducing too many new units.

The few units to be refined are selected by Algorithm 1,[3] which refines units with scores larger than a predefined threshold ϕ and that are among the k units with the highest scores. Our multi-level approach decreases explanation cost by avoiding attribution of scores to less relevant finer units. See Appendix C.3 for the used hyperparameters.

4 Automated Evaluation

We evaluate `MExGen` on two text generation tasks, summarization and context-grounded QA. Section 4.2 compares the scalarizers that we have considered and provides guidance on the choice of scalarizer. Sections 4.3 and 4.4 compare `MExGen` to other attribution algorithms and to LLM self-explanations respectively, showing that `MExGen` has higher fidelity in terms of identifying input units that are most important to the explained model.

[3]Users can also manually select the units to be refined.

Algorithm 1 Unit Refinement

Require: $x_1, \ldots, x_d$ ▷ Current units
$k \leq d$ ▷ Maximum # units to refine
$\phi \in [-1, 1]$ ▷ Significance threshold

Compute attribution scores ξ using (2) or (3).
$\psi \leftarrow 2\frac{\xi - \min_s \xi_s}{\max_s \xi_s - \min_s \xi_s} - 1$ ▷ Normalize scores
Compute Top-$k(\psi)$ ▷ k largest ψ_s
`units_to_be_refined` $\leftarrow \{\}$
for $s \in [d]$ **do**
  if $\psi_s \geq \phi$ **and** $\psi_s \in$ Top-$k(\psi)$ **then**
    `units_to_be_refined.add(`x_s`)`
return `units_to_be_refined`

4.1 Setup

Datasets and LMs For summarization, we evaluate on the Extreme Summarization (XSUM) (Narayan et al., 2018) and CNN/Daily Mail (CNN/DM) (See et al., 2017; Hermann et al., 2015) datasets. In Sections 4.2 and 4.3, we use three LMs: the 306M-parameter DistilBART [4], the 20B-parameter Flan-UL2 (Tay et al., 2023), and the Llama-3-8B-Instruct ("Llama-3") (Dubey et al., 2024) models. We treat the DistilBART model as one with full logit access, enabling use of the *Log Prob* scalarizer. We call Flan-UL2 and Llama-3 through an IBM API (IBM, 2024) that only returns text, thus representing the text-only setting precluding the use of *Log Prob*. We evaluate on the first 1000 test set examples of each dataset for DistilBART and the first 500 for Flan-UL2 and Llama-3.

For QA, we use the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) (1000 validation set examples, see Appendix C.1). We consider two LMs, Llama-3 through the above API, and the 770M-parameter Flan-T5-Large model[5] (Chung et al., 2022), which we treat as providing output logits.

Attribution algorithms We instantiate `MExGen` with the attribution algorithms discussed in Section 3.2: `LOO`, `C-LIME`, and `L-SHAP`. For summarization, we obtain mixed sentence- and phrase-level attributions while for QA, we obtain mixed sentence and word attributions, which are appropriate for SQuAD's shorter, paragraph-long contexts.

[4]https://huggingface.co/sshleifer/distilbart-xsum-12-6

[5]https://huggingface.co/google/flan-t5-large

Choices for scalarizers, segmentation, and algorithm parameters are described in Appendix C.3.

We compare against `PartitionSHAP` (P-SHAP) (SHAP, 2024a) and `CaptumLIME` (Miglani et al., 2023). For `PartitionSHAP`, recall from Section 2 that it requires the selection of an output token to explain, whereas we are interested in the output sequence as a whole. For this reason, we modify `PartitionSHAP` by summing across all attribution scores, corresponding to different output tokens, that it gives to each input span. This is equivalent to explaining the sum of the log probabilities of output tokens because of the linearity of Shapley values, and it corresponds to our *Log Prob* scalarizer. For `CaptumLIME`, since it accepts user-defined units for attribution, we provide it with the same units used by `MExGen`. Thus we can directly compare Captum's attribution algorithm (i.e., LIME) with ours, controlling for input segmentation. Captum's default for the target function to explain also corresponds to the *Log Prob* scalarizer. Additionally, `PartitionSHAP` and `CaptumLIME` take the number of model queries as an input; for a fair comparison, we allow them the greater of the numbers of model calls used by `MExGen L-SHAP` and `C-LIME`.

Metrics To measure local fidelity of explanations, we use perturbation curves as in Chen et al. (2020); Ju et al. (2023). Given a set of attribution scores, perturbation curves rank input units in decreasing importance according to their scores, perturb the top k units with k increasing, and plot the resulting change in some output scalarization measuring how much the perturbed output deviates from the target output. For the output scalarization, we select an *evaluation scalarizer* from those in Section 3.1. To accommodate `PartitionSHAP` and `CaptumLIME` (and slightly favor them), we choose an evaluation scalarizer corresponding to the target function that they use for explanation. For DistilBART and Flan-T5-Large, this means the *Log Prob* scalarizer, while for Flan-UL2 and Llama-3, we choose *SUMM* in keeping with `PartitionSHAP`. Please see Appendix C.5 for further details.

4.2 Scalarizer Evaluation

Ranking Similarity Across Scalarizers We first compare scalarizers by measuring the Spearman rank correlation between attribution score vectors ξ_S and $\xi_{S'}$ at the same granularity for all scalarizer pairs S, S'. Results with cosine similarity instead of Spearman are in Appendix D.1, Figure 7.

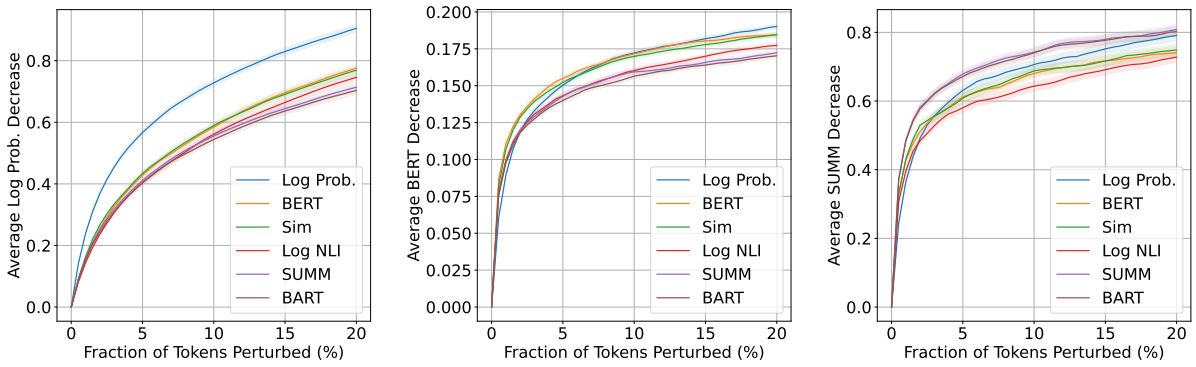
(a) `MExGen C-LIME`

(b) `MExGen L-SHAP`

Figure 3: Spearman's rank correlation between attribution scores using different scalarizers. Attributions were computed using multi-level (a) `MExGen C-LIME` for the DistilBART model on the XSUM dataset and (b) `MExGen L-SHAP` for Flan-T5-Large on SQuAD.

Figure 3 shows Spearman correlation matrices, averaged across examples from the respective dataset. Certain scalarizer pairs are highly similar, for example *BART* and *SUMM* as mentioned in Section 3.1, as well as *Sim* and *BERT*. Thus, it may suffice to consider only one from each pair. *Log Prob*, the only one that uses logits from the model being explained, clearly differs from the others. Additional results in Figure 7 show the same patterns. There are also noticeable differences between Figures 3a and 3b, which feature different tasks, datasets, and models. For example, Spearman correlation between *BERT* and *SUMM* is 0.75 in (a) and 0.83 in (b). This suggests the necessity of exploring different scalarizers for different tasks.

Perturbation Curves Across Scalarizers Figures 4a–4c show perturbation curves with different evaluation scalarizers, using `MExGen C-LIME` with all proposed scalarizers (i.e., a cross-evaluation of scalarizers). The curves are averaged over 1000 samples from XSUM (see Appendix C.5 for how) and the shading shows one standard error above and below. The case where the attribution scalarizer is matched with the evaluation scalarizer is generally the best. The *Log Prob* scalarizer performs well across evaluation scalarizers, implying that it gives



(a) *Log Prob* as Evaluation (b) *BERT* Score as Evaluation (c) *SUMM* Score as Evaluation

Figure 4: Perturbation curves (higher is better) for MExGen C-LIME with different scalarizers, used to explain the DistilBART model on the XSUM dataset. The curves show the decrease in (a) log probability, (b) BERTScore, (c) *SUMM* score when removing the most important $p\%$ of tokens according to each explanation scalarizer. Shading shows standard error in the mean.

a more universal explanation (least dependent on the evaluation scalarizer). Hence, we suggest using the log probabilities when they are available.

When model logits are not available, the choice of scalarizer is not as clear. However, if (hypothetically) one were to use *Log Prob* as the evaluation scalarizer, Figures 3 and 4a indicate that the *BERT* scalarizer best approximates the *Log Prob* scalarizer (*Sim* is similar while *SUMM*, the scalarizer used by P-SHAP, performs worse in this regard). Results for MExGen L-SHAP and a second model-dataset pair in Appendix D.1, Figures 8–10 show similar trends as above.

The computational cost of different scalarizers is discussed in Appendix C.3.

4.3 Comparison Between Explainers

Perturbation Curves Figure 5 compares the perturbation curves of MExGen instantiations with P-SHAP and CaptumLIME. Mean curves and standard errors are again computed over the number of examples taken from each dataset (see Section 4.1).

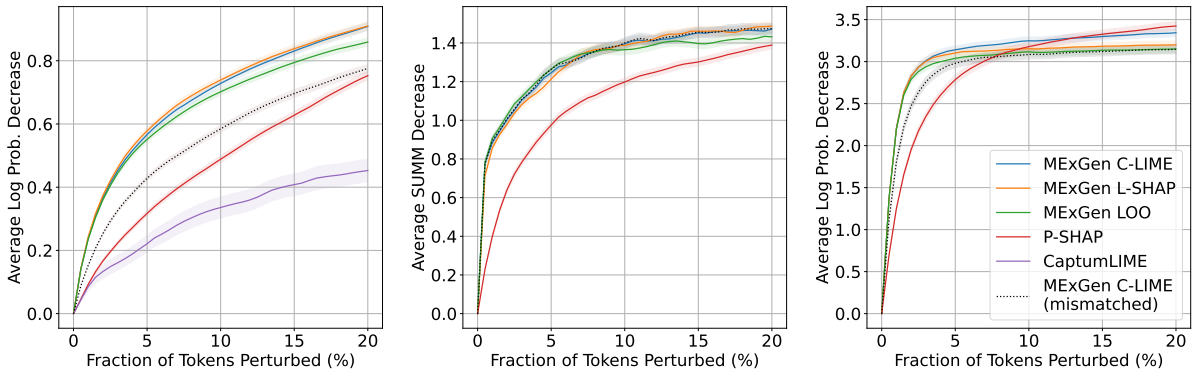
Regarding CaptumLIME, we were only able to obtain results for it in Figure 5a because (i) we had API access to Llama-3 and CaptumLIME needs output logits, and (ii) CaptumLIME does not support Flan-T5-Large. Figure 5a is a direct comparison between LIME (represented by CaptumLIME) and our modification C-LIME, using the same number of model queries and input segmentation as mentioned earlier. C-LIME is clearly more effective.

In comparison with P-SHAP, Figure 5 shows that the perturbation curves for MExGen rise more quickly from zero and are higher for the top $x\%$ of tokens, where $x > 20\%$ in Figures 5a and 5b, and x varies between 8% and 13% in Figure 5c. This pattern indicates that MExGen is better able to identify units that are most important to the model, as measured by the change in the evaluation scalarizer in the leftmost region of each plot. Figure 11 has results on additional model-dataset pairs. Across the tested models and datasets, MExGen C-LIME and L-SHAP are the top performers, while MExGen LOO, the simplest attribution algorithm, is close behind.

We also show in Figure 5 perturbation curves for MExGen C-LIME using different scalarizers than used for evaluation. Surprisingly, we find that even when using a mismatched scalarizer (notably *BERT* in Figures 5a, 5c, which does not even use logits), MExGen C-LIME can outperform P-SHAP in fidelity.

Area Under the Perturbation Curve Table 1 shows the area under the perturbation curve (AUPC) to summarize performance over all dataset-model pairs that we tested, including the ones in Figure 11. We evaluate AUPC up to 20% of tokens as done in (Chen et al., 2020). Across all dataset-model pairs, MExGen instantiations (including LOO) performed better (i.e., higher AUPC) than P-SHAP. The one exception is the (CNN/DM, Flan-UL2) pair, but even in this case, Figure 11b in Appendix D.2 shows that the MExGen curves are higher for the top 5% of tokens. The second highest AUPC is always from MExGen C-LIME or L-SHAP.

Computational Cost The computational cost of the compared explanation methods — measured by time and memory use — is primarily determined by the number of LLM inferences performed. As



(a) DistilBART on XSUM (b) Llama-3 on CNN/DM (c) Flan-T5-Large on SQuAD

Figure 5: Perturbation curves (higher is better) for different explanation methods, using the same scalarizer as the evaluation scalarizer in the y-axis label. Models and datasets: (a) DistilBART on XSUM, (b) Llama-3 on CNN/DM, (c) Flan-T5-Large on SQuAD. The legend in (c) applies to all panels, but we could run `CaptumLIME` only for (a). "`MExGen C-LIME` (mismatched)" refers to the use of mismatched scalarizers: BERTScore (a)(c) and BARTScore (b).

Datasets	Models	MExGen C-LIME	MExGen L-SHAP	MExGen LOO	P-SHAP
XSUM	DistilBART	13.6	**13.8**	13.1	9.4
	Flan-UL2	17.2	**17.4**	16.7	13.7
	Llama-3-8B-Instruct	**22.4**	22.2	22.1	20.2
CNN/DM	DistilBART	13.5	**14.7**	13.2	9.7
	Flan-UL2	32.1	32.0	32.1	**33.2**
	Llama-3-8B-Instruct	**26.4**	26.3	26.1	22.1
SQuAD	Flan-T5-Large	**62.7**	61.1	60.2	58.8
	Llama-3-8B-Instruct	56.4	**57.0**	54.9	38.5

Table 1: Areas under the perturbation curve (AUPC) up to 20% of tokens. For DistilBART and Flan-T5-Large, log probability is used as both the explanation and evaluation scalarizer. For Flan-UL2 and Llama-3-8B-Instruct, which do not provide access to log probabilites, *SUMM* is used as the explanation and evaluation scalarizer.

noted in Section 4.1, `P-SHAP` and `CaptumLIME` were allocated the same or slightly more inferences than `MExGen` in our experiments. Even so, `MExGen` consistently delivered more faithful explanations.

4.4 Comparison with LLM Self-Explanation

LLMs are capable of explaining their own outputs, including by providing numerical attributions to their inputs (Huang et al., 2023b; Madsen et al., 2024). In this experiment, we compare the fidelity of these self-explanations to that of `MExGen`. Specifically, since the perturbation curves that we use to evaluate fidelity depend only on the ranking of input units, we prompt the LLM to rank units in order of importance to the output that it generated.

LLMs and Datasets As a test of current LLMs' ability to self-explain, we chose a large and powerful open-weights LM, DeepSeek-V3. At the time of writing, we could not reliably obtain log probabilities from DeepSeek-V3, so we were unable to apply `MExGen` with the *Log Prob* scalarizer and could only use a text-only scalarizer. For this reason, we tested a second LLM, Granite-3.3-8B-Instruct, for which both *Log Prob* and text-only scalarizers were possible. We used the summarization datasets XSUM and CNN/DM.

Explanation Methods For self-explanation, we describe in Appendix C.4 the prompt that we used and other details. The most important point is that we made the ranking task easier by following Zhang et al. (2024) in numbering input units with tags and only asking for a list of tags. `MExGen` was run in the same way as before (see Appendix C.3).

Results Table 2 shows the AUPC values for this self-explanation experiment. The corresponding perturbation curves are in Figure 12. In all cases, `MExGen` is more faithful to the LLM's behavior (as measured by higher AUPC) than the LLM's self-explanations. Considering Granite-3.3, the advantage of `MExGen` is especially large (AUPC is double or more) when it uses the *Log Prob* scalarizer, and less so with the text-only *BART* scalarizer. The larger DeepSeek-V3 model narrows the gap further on the CNN/DM dataset. Overall, our results indi-

Dataset	Model	Scalarizer	MExGen C-LIME	MExGen L-SHAP	MExGen LOO	Self
XSUM	Granite-3.3	Prob	18.9	**19.0**	18.9	9.5
		BART	**15.4**	14.4	14.2	12.4
	DeepSeek-V3	BART	**12.7**	12.3	12.3	10.5
CNN/DM	Granite-3.3	Prob	17.3	**17.4**	16.9	7.1
		BART	**11.9**	11.0	11.0	8.9
	DeepSeek-V3	BART	**14.1**	14.0	13.5	13.5

Table 2: Areas under the perturbation curve (AUPC) up to 20% of tokens for the comparison with LLM self-explanation. The scalarizer in the "Scalarizer" column is used for both explanation and evaluation.

cate that while LLM self-explanations can be good, they are outperformed by algorithms that systematically quantify importance. It is also important to note that this experiment only assesses *ranking* ability, whereas `MExGen` also provides *real-valued* scores, a task that would be harder for an LLM.

5 User Study

We conducted a user study to understand how humans perceive explanations provided by different scalarizers and attribution methods, and whether they can discern performance differences akin to the quantitative evaluations in Section 4. To ease interpretation, we developed a visualization tool that highlights input text spans based on a color-coded scale for the attribution scores. In this section, we focus on the following research questions:

1. *Fidelity*: Which method is perceived to be better at explaining how the language model generated the summary?
2. *Preference*: Which method do people prefer?

We discuss additional research questions on concentration and granularity of attribution scores in Appendix E.

We selected ten examples from the XSUM dataset for the user study with diversity in topics, while ensuring that they do not contain sensitive issues or obvious errors. We designed an online survey consisting of three parts. Each part showed an input text, randomly drawn from the ten examples, and its summary, generated by the DistilBART model. This user study focused on algorithms and scalarizers that are of most interest based on the automated evaluation in Section 4. Specifically, we compared two scalarizers (*Log Prob*, *BERT*) and three attribution algorithms (`C-LIME`, `L-SHAP`, `PartitionSHAP`). The presentation order of the attribution algorithms was randomized to mitigate order effects. The survey consisted of seven pairwise comparisons in total followed by questions for the participants.

We recruited participants from a large technology company who self-identify as machine learning practitioners using language models and collected data from 88 of them after filtering. Here, we report a summary of key results only. See Appendix E for details including survey questions, analysis, and statistical results.

Scalarizers. Significantly more participants perceived *BERT* to be higher in fidelity than *Log Prob* (57% to 35%). They also preferred *BERT* over *Log Prob* (64% to 31%). This result is notable because *BERT* uses only text output from the LM while *Log Prob* depends on output logits.

Attribution methods. Significantly more participants perceived `C-LIME` to be higher in fidelity than `L-SHAP` (p-value $= 0.011$). They also preferred `C-LIME` over `L-SHAP` ($p = 0.007$). This result is notable because `C-LIME` and `L-SHAP` performed very similarly in the automated evaluation in Section 4.3.

6 Concluding Remarks

We proposed `MExGen`, a framework to provide explanations for LLMs used in context-grounded tasks like summarization and question answering. `MExGen` uses a multi-level strategy to efficiently explain model predictions in the case of long inputs. `MExGen` can produce explanations even when only text outputs are available (API access), thanks to *scalarizers* that map text to numerical values. Our experiments offer guidance on the choice of scalarizer and show that instances of `MExGen` provide more faithful explanations, outperforming the baselines `PartitionSHAP` and `CaptumLIME` as well as self-explanations by powerful LLMs. The user study results align with the automated evaluation, and reveal that people perceive the *BERT* scalarizer as more locally faithful than the *Log Prob* scalarizer. This result implies that in some cases, there may be no loss in having text-only access compared to full logit access.

Limitations

We see the following limitations and risks:

(1) MExGen is a framework for post hoc explanations. Although such explanations can help practitioners understand model behavior, they do not fully characterize how models generate output and only provide local explanations.

(2) Although the findings of our automated evaluation are consistent across the tested models and datasets, the results reported in Section 4 could still change in other experimental settings.

(3) Our user study analyzes the *perception* of participants of how well a method explains the predictions of a model, and not necessarily the fidelity of the explanation itself — fidelity is measured more directly in the automated evaluation. However, we believe the fact that the participant pool was composed of people with experience in ML and LLMs improves the quality of their evaluation.

(4) Post hoc explanations in general come with the risk of being steered to obfuscate undesirable behavior from the model. One potential mitigation is to couple them with additional types of explanation, where possible. Another mitigation is to have a different party compute explanations, rather than the model developer. The black-box, perturbation-based explanations considered in this work lend themselves to such a two-party scenario, where the party computing explanations only needs to query the model to do so. This party would select which perturbed inputs to query the model on, avoiding one path to obfuscation where a model developer uses specially chosen perturbations to conceal undesirable behavior. The explaining party should also be given a large budget of model queries to better probe model behavior.

Ethics Statement

MExGen is a framework to explain generative language models. Hence, its objective is to elucidate how a model made a specific prediction. Methods that aim to understand how black-box models generate their output are essential for guaranteeing transparency during decision-making. For example, a generative language model can be used to summarize dialog and create minutes of meetings that can later be used to perform high-stakes decisions. Then, it is necessary to understand how the model generated the summary and ensure that the output content is based on the input dialog. Therefore, for such high-stakes applications, methods that can provide explanations for text generated by language models are necessary, highlighting the importance of MExGen.

Acknowledgements

The authors thank the following people: Inkit Padhi for suggestions on parsing sentences into phrases, using language models for perturbing words, and participation in multiple discussions; Ella Rabinovich and Samuel Ackerman for discussions on and recommendations of similarity measures as scalarizers, and Ella in particular for also providing suggestions on parsing sentences into phrases and using language models for perturbation; Pratap Kishore Varma Vemulamanda for helping to investigate the API-only case of SHAP's summarization example (SHAP, 2024b); Subhajit Chaudhury for suggestions of similarity measures as scalarizers; Keshav Ramji for discussion on related work and quality evaluation for the explanations; Kush Varshney and Eitan Farchi for general encouragement and support. The authors also thank the anonymous ARR reviewers and action editor for their high-quality reviews and positive, constructive comments, especially Reviewer GEAE for suggesting the comparison with LLM self-explanations, and Yannis Katsis for his advice on the setup for this experiment. The work of Lucas Monteiro Paes was supported by the Apple Scholars in AI/ML Fellowship.

References

Pepa Atanasova, Oana-Maria Camburu, Christina Lioma, Thomas Lukasiewicz, Jakob Grue Simonsen, and Isabelle Augenstein. 2023. Faithfulness tests for natural language explanations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 283–294, Toronto, Canada. Association for Computational Linguistics.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: natural language inference with natural language explanations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9560–9572, Red Hook, NY, USA. Curran Associates Inc.

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.

Jianbo Chen and Michael Jordan. 2020. LS-Tree: Model interpretation when the data are linguistic. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3454–3461.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. 2019. L-Shapley and C-Shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Benjamin Cohen-Wang, Harshay Shah, Kristian Georgiev, and Aleksander Madry. 2024. Contextcite: Attributing model generation to context. *arXiv preprint arXiv:2409.00729*.

Abhimanyu Dubey et al. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

James Enouen, Hootan Nakhost, Sayna Ebrahimi, Sercan O. Arik, Yan Liu, and Tomas Pfister. 2023. TextGenSHAP: Scalable post-hoc explanations in text generation with long documents. *Preprint*, arXiv:2312.01279.

David Firth. 2005. Bradley-Terry models in R. *Journal of Statistical software*, 12:1–12.

Christos Fragkathoulas and Odysseas Spyridon Chlapanis. 2024. Local explanations and self-explanations for assessing faithfulness in black-box llms. In *Proceedings of the 13th Hellenic Conference on Artificial Intelligence*, SETN '24, New York, NY, USA. Association for Computing Machinery.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023a. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ArXiv*, abs/2311.05232.

Shiyuan Huang, Siddarth Mamidanna, Shreedhar Jangam, Yilun Zhou, and Leilani H. Gilpin. 2023b. Can large language models explain themselves? a study of llm-generated self-explanations. *Preprint*, arXiv:2310.11207.

IBM. 2024. IBM Generative AI Python SDK (Tech Preview). `https://github.com/IBM/ibm-generative-ai`. Accessed: 2024-10-01.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. 2020. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *International Conference on Learning Representations*.

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.

Ambedkar Kanapala, Sukomal Pal, and Rajendra Pamula. 2017. Text summarization from legal documents: a survey. *Artificial Intelligence Review*, 51:371 – 402.

Siwon Kim, Jihun Yi, Eunji Kim, and Sungroh Yoon. 2020. Interpretation of NLP models through input marginalization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3154–3167.

Nicholas Kroeger, Dan Ley, Satyapriya Krishna, Chirag Agarwal, and Himabindu Lakkaraju. 2024. In-context explainers: Harnessing llms for explaining black box models. *Preprint*, arXiv:2310.05797.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Md Tahmid Rahman Laskar, Xue-Yong Fu, Cheng Chen, and TN ShashiBhushan. 2023. Building real-world

meeting summarization systems using large language models: A practical perspective. In *Conference on Empirical Methods in Natural Language Processing*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationaling neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Learning Representations*.

Xuhong Li, Jiamin Chen, Yekun Chai, and Haoyi Xiong. 2024. Gilot: interpreting generative language models via optimal transport. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Wei Liu, , Haozhao Wang, Jun Wang, Ruixuan Li, Zinyang Li, Yuankai Zhang, and Yang Qiu. 2023a. Mgr: Multi-generator based rationalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Wei Liu, , Jun Wang, Haozhao Wang, Ruixuan Li, Zhiying Deng, Yuankai Zhang, and Yang Qiu. 2023b. D-separation for causal self-explanation. In *NeurIPS*.

Wei Liu, , Jun Wang, Haozhao Wang, Ruixuan Li, Yang Qiu, Yuankai Zhang, Jie Han, and Yixiong Zou. 2023c. Decoupled rationalization with asymmetric learning rates: A flexible lipschitz restraint. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Wei Liu, Zhiying Deng, Zhongyu Niui, Jun Wang, Haozhao Wang, Yuankai Zhang, and Ruixuan Li. 2024a. Is the mmi criterion necessary for interpretability? degenerating non-causal features to plain noise for self-rationalization. In *NeurIPS*.

Wei Liu, Haozhao Wang, Jun Wang, Zhiying Deng, YuanKai Zhang, Cheng Wang, and Ruixuan Li. 2024b. Enhancing the rationale-input alignment for self-explaining rationalization. In *Proceedings of the IEEE 40th International Conference on Data Engineering*.

Wei Liu, Haozhao Wang, Jun Wang, Ruixuan Li, Chao Yue, and Yuankai Zhang. 2022. Fr: Folded rationalization with a unified encoder. In *NeurIPS*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. 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, Bangkok, Thailand. Association for Computational Linguistics.

Dina Mardaoui and Damien Garreau. 2021. An analysis of LIME for text data. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3493–3501. PMLR.

Vivek Miglani, Aobo Yang, Aram Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. 2023. Using Captum to explain generative language models. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 165–173, Singapore.

Edoardo Mosca, Ferenc Szigeti, 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.

Manish Nagireddy, Lamogha Chiazor, Moninder Singh, and Ioana Baldini. 2024. SocialStigmaQA: A benchmark to uncover stigma amplification in generative language models. In *Proceedings of the 2024 AAAI Conference on Artificial Intelligence*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Letitia Parcalabescu and Anette Frank. 2024. On measuring faithfulness or self-consistency of natural language explanations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6048–6089, Bangkok, Thailand. Association for Computational Linguistics.

Thang Pham, Trung Bui, Long Mai, and Anh Nguyen. 2022. Double trouble: How to not explain a text classifier's decisions using counterfactuals synthesized by masked language models? In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 12–31, Online only. Association for Computational Linguistics.

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 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*, pages 1135–1144.

Gabriele Sarti, Nils Feldhus, Ludwig Sickert, Oskar van der Wal, Malvina Nissim, and Arianna Bisazza. 2023. Inseq: An interpretability toolkit for sequence generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 421–435, Toronto, Canada. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

SHAP. 2024a. Text examples. https://shap.readthedocs.io/en/latest/text_examples.html. Accessed February 2024.

SHAP. 2024b. Text to text explanation: Abstractive summarization example. https://shap.readthedocs.io/en/latest/example_notebooks/text_examples/summarization/Abstractive%20Summarization%20Explanation%20Demo.html. Accessed February 2024.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Chandan Singh, W. James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Zeren Tan, Yang Tian, and Jian Li. 2023. GLIME: General, stable and local LIME explanation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023. UL2: unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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 *Thirty-seventh Conference on Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6233–6251, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. BARTScore: Evaluating generated text as text generation. In *Advances in Neural Information Processing Systems*.

Jiajie Zhang, Yushi Bai, Xin Lv, Wanjun Gu, Danqing Liu, Minhao Zou, Shulin Cao, Lei Hou, Yuxiao Dong, Ling Feng, and Juanzi Li. 2024. LongCite: Enabling LLMs to generate fine-grained citations in long-context QA. *arXiv preprint arXiv:2409.02897*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*.

A More on Related Work

Table 3 compares the features of the proposed MExGen framework to alternative perturbation-based explanation methods. The remainder of this appendix discusses other categories of explanation methods.

Hierarchical Explanations A line of work (Singh et al., 2019; Jin et al., 2020; Chen et al., 2020; Ju et al., 2023) has developed hierarchical explanations for sequence models including LLMs, which can reveal compositional interactions between words and phrases. In particular, the HEDGE algorithm of Chen et al. (2020) was identified by Mosca et al. (2022) as "arguably the most suitable choice" for NLP input attribution, in part because it builds its hierarchy in a top-down, divisive fashion (as opposed to bottom-up agglomeration (Singh et al., 2019; Ju et al., 2023)), which is more practical for long texts. However, HEDGE is specific to classification because it measures feature importance based on classification margin.

Gradient-Based Explanations Gradient-based methods provide input attribution explanations (Simonyan et al., 2013; Sundararajan et al., 2017; Shrikumar et al., 2017; Bach et al., 2015), but they require access to model gradients with respect to the input.

Attribution for Generative Language Models. Concurrently with our work, other studies have explored attribution techniques for generative LMs (Li et al., 2024; Cohen-Wang et al., 2024; Sarti et al., 2023). GiLOT (Li et al., 2024) employs optimal transport to compute explanations. However, it focuses on the overall distributional changes across all possible outputs rather than attributing specific generated outputs to particular inputs, as MExGen does.

ContextCite (Cohen-Wang et al., 2024) extends LIME (Ribeiro et al., 2016) to generative models by considering parts of the context as features. While effective, ContextCite operates at a single predefined granularity level (e.g., sentence level), lacking the multi-level nature of MExGen, which improves explanation performance.

Inseq (Sarti et al., 2023) is a toolkit for attribution of generative LMs, offering methods and visualization tools for context attributions. Its methods focus on computing the importance of each input token for every token in the generated output. However, this approach may not fully capture the influence of larger units in the input, and it also does not explain the entire output.

A key distinction of MExGen is its ability to function without access to the model's logits by utilizing text-only scalarizers, whereas GiLOT, ContextCite, and Inseq all assume access to these logits.

Self-Explanations & Numerical Attributions Other works have used the generative LM itself to provide explanations in line with subsequent outputs.

An example of self-explanation is the chain-of-thought (CoT) (Wei et al., 2022), where the model explicitly generates the reasoning in natural language used to produce its output. However, CoT has issues with stability (Turpin et al., 2023) and faithfulness (Atanasova et al., 2023; Lanham et al., 2023) in addition to significant variation in quality (Nagireddy et al., 2024). Moreover, evaluating the faithfulness of CoT is challenging, and Parcalabescu and Frank (2024) argued that existing faithfulness evaluations actually measure self-consistency. Parcalabescu and Frank (2024) proposed their own measure of self-consistency called CC-SHAP, which adapts SHAP in a way similar to how we adapt P-SHAP. CoT is less relevant to our work since it provides explanations in natural language rather than numerical attributions.

Another approach to self-explanation is prompting the model to explain its predictions (e.g., by ranking the most important parts of its context) (Camburu et al., 2018; Huang et al., 2023b; Kroeger et al., 2024; Madsen et al., 2024). However, prior work has found that these self-explanations may be less faithful when used for in-context classification tasks (Madsen et al., 2024; Huang et al., 2023b).

Recently, Fragkathoulas and Chlapanis (2024) employed a modified version of MExGen to identify words in the input context that were essential for the model's prediction in a question-answering task. Then, Fragkathoulas and Chlapanis (2024) compared the keywords they identified with the keywords extracted using self-explanation, showing that self-explanations are not as faithful as their method. This is a first step in demonstrating that self-explanations are not as faithful as numerical attributions, as provided by MExGen. Our work generalizes theirs to additional generative tasks and to self-explanations in the form of rankings.

Rationalization Yet another line of self-explanation methods falls under the class of

Method	Long Text Output	Long Text Input	API Access
LIME	✗	✗	✗
SHAP	✗	✗	✗
HEDGE	✗	✓	✗
P-SHAP	●	✓	✓
TextGenSHAP	✗	✓	✗
Captum	✓	✗	✗
MExGen	✓	✓	✓

Table 3: Comparing the features of our MExGen framework with existing perturbation-based explanation methods.

rationalization methods. Lei et al. (2016) propose to simultaneously train a rationale extractor with a predictor. The rationale extractor essentially selects parts of the text to use for prediction that fulfill two criteria: they are interpretable and maintain nearly the same prediction as using the full text. As these parts of the text are interpretable, they offer an explanation for the final prediction. Lei et al. (2016) employed regularizations to keep the rationales short while encouraging contiguous text, and a sequence of recent literature has focused on improving these rationales.

Liu et al. (2022) propose to share the encoder parameters of the extractor and predictor, with the intuition that the extractor and predictor are both seeking to find the most informative part of the input text. Liu et al. (2023b) takes a different approach and considers removing what is unimportant rather than selecting what is important; they minimize a criterion for dependence of the prediction on the text which essentially detects what parts of the text is not required for prediction. Liu et al. (2023c) use a theoretical analysis of how the Lipschitz constant of the predictor affects the extractor to propose how to control the learning rate for obtaining better rationales. Liu et al. (2023a) introduce an ensemble of extractors (multiple extractors with different initializations) into the rationalization training framework and find improved prediction performance. At inference time, only the first extractor of the ensemble is used for interpretability. Liu et al. (2024b) further include a predictor on the full text into a regularization term in order to better align predictions on extracted text with those of the full text. Most recently, Liu et al. (2024a) build on Liu et al. (2023b) and propose a loss criterion that attempts to treat spurious correlations in the text as if it were noise.

B Further Details on MExGen

This appendix provides more details on the MExGen framework that are of a more general nature. For parameter settings and other details specific to our experiments, please refer to Appendix C.

B.1 Scalarizers

Vocabulary-wide logits The inputs x in our case are perturbed versions of the original input x^o. Some of these perturbations however may be significantly different semantically and cause the probability of generating the target output y^o to decrease dramatically. For this reason, the *Log Prob* scalarizer may require access to logits for improbable tokens conditioned on x, which can be ensured if logits are available for the entire vocabulary, but not if only the top k logits are provided.

Aggregation for *Log Prob* scalarizer As alternatives to the average over the output sequence in (1), other ways of aggregating include using the sum or taking the product or geometric mean of the probabilities. We choose the average to normalize for sequence length and because log probabilities tend to be a more "linear" function of inputs than probabilities.

BERTScore BERTScore uses an LM to obtain contextual embeddings for the tokens in y and y^o, matches the two sets of embeddings based on cosine similarity, and computes a score to quantify the degree of match.

BARTScore BARTScore is the same as (1) except with y in place of x and an auxiliary LM f_{BART} (not necessarily a BART model) in place of the LM f being explained. It thus measures the probability of f_{BART} generating y^o given y as input.

***SUMM* scalarizer** We included this scalarizer to represent our understanding of how the abstractive summarization example (SHAP, 2024b) in the

SHAP library handles the text-only API case. We investigated the code behind this SHAP example, and specifically the `TeacherForcing` class that it uses. Our understanding of the code is that it obtains proxy log-odds for tokens in the target output y^o by taking log-odds from an auxiliary summarization model f_{SUMM}, with input y and output set to y^o. If we then average the log-odds over the tokens in y^o (similar to (1)), the result is similar to the *BART* scalarizer with $f_{BART} = f_{SUMM}$ (with a possible discrepancy between log-odds versus log probabilities). Our experiments show that the *BART* and *SUMM* scalarizers are indeed very similar.

***Log NLI* scalarizer** This scalarizer is based on the intuition that *Log NLI* entailment is a kind of similarity. We use the NLI model to predict the log-odds of entailment given y^o as premise and y as hypothesis, and optionally in the other direction as well. If both directions are used, we take the geometric mean of the two entailment probabilities and then convert back to log-odds. We note that bi-directional entailment has been used to approximate semantic equivalence, for example in Kuhn et al. (2023), but here we use the log-odds scores and not just the predicted labels.

B.2 Phrase Segmentation

Here we describe the phrase segmentation algorithm mentioned in Section 3.2. The algorithm starts with the root token of the tree and checks whether each child subtree of the root is shorter (in terms of tokens) than a maximum phrase length parameter. If it is shorter, then the subtree constitutes a phrase, and if it is not, the algorithm is recursively applied to the subtree. The root token of each (sub)tree is also taken to be a phrase. Once the sentence has been recursively segmented into phrases in this manner, a second pass is performed to re-merge some phrases that have become too fragmented, thus controlling the number of phrases which is desirable for computation and interpretation. Specifically, phrases that constitute noun chunks (as identified by `spaCy`) are merged, and certain single-token phrases are merged with their neighbors. Further notes:

- Subtrees of the dependency parse tree are usually contiguous spans of text (in English), but sometimes they correspond to multiple spans. In this case, we treat each span as its own subtree since we wish to have contiguous phrases.
- In measuring the token length of a span, we do not count punctuation or spaces.
- In merging phrases that fall within a noun chunk, we check conditions that are consistent with noun chunks, for example that there is a root phrase that is a single token (the noun), and that the other phrases are children of the root phrase.
- For merging single-token phrases with their neighbors, we use the following criteria:
  - The single-token phrase (*singleton*) is a non-leaf phrase (is the parent of other phrases) or a coordinating conjunction.
  - If the singleton is a coordinating conjunction (e.g. "and"), the neighbor is a corresponding conjunct (e.g. "Bob" in "Alice and Bob").
  - If the singleton is a preposition (e.g. "to"), the neighbor is a child of the preposition (e.g. "the store" in "to the store").
  - If the singleton is of some other type, the neighbor is a leaf phrase and is either adjacent to the singleton or a singleton itself.
  - The merged phrases do not exceed the maximum phrase length parameter.

B.3 `L-SHAP`

This *local* variation of SHAP was proposed by Chen et al. (2019) to decrease the number of model inferences relative to SHAP, which requires exponentially many inferences. In our context, `L-SHAP` does so by only perturbing units that are within a constant-size neighborhood of the current unit being attributed to. This makes the number of inferences scale linearly with the number of units. More precisely, for unit of interest $s \in [d]$, we consider only the radius-M neighborhood $\mathcal{N}_s^M = \{s-M, \ldots, s-1, s+1, \ldots, s+M\}$ (truncated to $1, \ldots, d$ if necessary) and subsets of $\mathcal{N}_s^M$ with cardinality up to K. Then the attribution score ξ_s for unit s is given by

$$\xi_s = \frac{1}{K+1} \sum_{A \subseteq \mathcal{N}_s^M : |A| \leq K} \binom{|\mathcal{N}_s^M|}{|A|}^{-1} \times \Big(S(x^{(A)}; y^o, f) - S(x^{(A \cup \{s\})}; y^o, f) \Big), \quad (3)$$

where $x^{(A)}$ is a perturbation of x^o in which units $j \in A$ are perturbed (just dropped in our work).

C Automated Evaluation Details

This appendix documents choices and parameter settings used in the experiments in Section 4. We follow the order of presentation in Section 4.1: datasets and LM inference, choices for the `MExGen` framework (Appendix C.3), baseline attribution methods (Appendix C.4), perturbation curve details, and computing environment.

C.1 Datasets

The GitHub repository[6] that enables the XSUM dataset (Narayan et al., 2018) to be rebuilt is made available under the MIT license. The CNN/DM dataset (See et al., 2017) is made available by HuggingFace[7] under the Apache-2.0 license. Both datasets are intended for abstractive summarization, which is how we use them. Both datasets consist of news articles and thus contain the names of individuals. Since these names were reported by the media in the original articles, they are already in the public domain and there is no further issue.

SQuAD is distributed under the CC BY-SA 4.0 license. The dataset is intended for evaluating reading comprehension by answering questions on the contexts in the dataset, which is how we use it as well. Since the contexts are taken from Wikipedia articles, some of them name individuals, but again these articles are already in the public domain. We selected 1000 validation set examples at random from SQuAD because selecting the first 1000 examples yielded insufficient diversity.

C.2 Language model inference

DistilBART and Flan-T5-Large The two models that we treated as providing full logit access, DistilBART and Flan-T5-Large, were downloaded from HuggingFace under the Apache-2.0 license. The DistilBART model was trained on the summarization datasets XSUM and CNN/DM, and we use it for summarization. Flan-T5-Large is a general-purpose LM intended for research on LMs, consistent with our use.

The models were called through their `.generate` method. For generating the original (target) output (corresponding to the original input), `max_new_tokens` was set to `None` (i.e., the default). When using the text-only scalarizers, perturbed output texts (corresponding to perturbed inputs) are also generated, and for these, `max_new_tokens` was set to 1.5 times the number of tokens in the target output. The Log Prob scalarizer computes the log probability of generating the target output, so `max_new_tokens` is not needed in this case. All other hyperparameters were left at default settings (for example greedy decoding was used).

Flan-UL2 and Llama-3 Flan-UL2 and Llama-3-8B-Instruct are general-purpose LMs. Flan-UL2 is distributed by HuggingFace under the Apache-2.0 license while Llama-3 is distributed under its own Meta Llama 3 Community License.[8] We however accessed Flan-UL2 and Llama-3 using an LLM API service provided by IBM (IBM, 2024) (no longer in existence).

For these API calls to Flan-UL2 and Llama-3-8B-Instruct, we used greedy decoding and `max_new_tokens` $= 100$. In the case of Llama-3, we used its chat template and provided the following system prompts for summarization and QA respectively: (summarization) "Summarize the following article in one sentence. Do not preface the summary with anything." (QA) "Please answer the question based on the provided context. Answer with a short phrase or sentence." No chat template or system prompt was necessary to prompt Flan-UL2 to summarize.

DeepSeek-V3 and Granite-3.3 The DeepSeek-V3 and Granite-3.3-8B-Instruct models used in the self-explanation experiment of Section 4.4 were called using a different LLM API service provided by IBM. Similar to Llama-3, we used greedy decoding, `max_tokens` $= 100$, the model's chat template, and the same system prompt for summarization above. We also fixed the LLM's random seed so that it would generate the same output given the same input. For generating self-explanations in the form of a list of tags, `max_tokens` was increased to 500 to accommodate long lists. For Granite-3.3, we were able to obtain log probabilities by setting `max_tokens` $= 0$ and `logprobs` $= 0$, which returns log probabilities of tokens in the input given to the LLM.

C.3 `MExGen`

Scalarizer models The text-only scalarizers presented in Section 3.1 can be instantiated with different models. The ones used in our experiments

[6] https://github.com/EdinburghNLP/XSum

[7] https://huggingface.co/datasets/abisee/cnn_dailymail

[8] https://www.llama.com/llama3/license/

are as follows:

- "*Sim*": We use the all-MiniLM-L6-v2 embedding model from the SentenceTransformers package (Reimers and Gurevych, 2019).[9]
- "*BERT*": We use the model deberta-v2-xxlarge-mnli[10] (He et al., 2021) to compute BERTScore (Zhang et al., 2020) (MIT license). This is the same model that we use for the *Log NLI* scalarizer. Our initial reason for doing so was to see whether the two scalarizers would be very similar because of this choice (they are not as they operate on different principles). We take the "F1-score" output as the BERTScore.
- "*BART*" and "*SUMM*": For "*SUMM*", we follow SHAP (2024b) in using the same distilbart-xsum-12-6 model as both the scalarizing summarization model as well as the primary summarization model to explain. For "*BART*", i.e., BARTScore (Yuan et al., 2021), we use the code[11] from the authors and also instantiate it with the distilbart-xsum-12-6 model. The purpose was to determine whether the "*BART*" and "*SUMM*" scalarizers are very similar when instantiated with the same model, which is indeed the case.
- "*Log NLI*": As mentioned above, we use deberta-v2-xxlarge-mnli as the *Log NLI* model. We also choose to compute the *Log NLI* entailment probability in both directions and take the geometric mean of the two before taking the logit.

All other parameters of these scalarizers are kept at default values.

In the case where log probabilities are not available from the LLM being explained, the above text-only scalarizers contribute to the overall computational cost of `MExGen`, but only in a secondary way. This is because our choices of scalarizers are small LMs (at most 1.5B parameters for deberta-v2-xxlarge-mnli). The *Sim* scalarizer was the most computationally efficient because it uses a SentenceTransformer model. The other scalarizers were more comparable to each other, with the *Log NLI* scalarizer being the most costly because of our choice to compute bi-directional entailment (two inferences versus one).

Linguistic segmentation As mentioned in Section 3.2, we use `spaCy` v3.6 (Honnibal et al., 2020) (distributed under the MIT license) for sentence and word segmentation. Our custom phrase segmentation algorithm has one main parameter, the maximum phrase length, which we set to 10 `spaCy` tokens (not counting spaces and punctuation).

In terms of units that are not of interest for attribution, we exclude non-alphanumeric units (generally punctuation and newlines) and prompt template elements, for example start-of-message and end-of-message tokens like "<|start_header_id|>user<|end_header_id|>", prefixes such as "Context: ", and system prompts. For QA, we exclude the question and allow only the context to be attributed to.

Multi-level explanations The decision here is how many top sentences are refined into phrases in the summarization experiments and into words in the QA experiments. For this we follow Algorithm 1 with the following parameter settings: For summarization with the DistilBART model, the number of top sentences is set to $k = 3$ and the significance threshold is set to $\phi = 1/3$ for sentence-level scores from `C-LIME` and `LOO` and $\phi = 0.3$ for `L-SHAP`. For summarization with Flan-UL2, Llama-3, DeepSeek-V3, and Granite-3.3, we set k to correspond to the top 25% of sentences, rounding to the nearest integer and ensuring $k \geq 1$. A significance threshold is not used, i.e., $\phi = -1$. For QA, we simply take the top $k = 1$ sentence as the context paragraphs in SQuAD tend to have only a handful of sentences, and again do not use a significance threshold ($\phi = -1$).

`C-LIME` For `C-LIME`, the parameters controlling the perturbations are the constant of proportionality between the number of perturbations n and number of units d, and the maximum number of units K perturbed at one time. For the smaller DistilBART and Flan-T5-Large models, n/d is set to 10 and K to 3, except for sentence-level attributions on SQuAD where $K = 2$.

For Flan-UL2, Llama-3-8B-Instruct, DeepSeek-V3, and Granite-3.3-8B-Instruct, since inference is more costly for these larger models, we take the following approach. First, at the sentence level of attribution, we only use `LOO` to identify the top sentences and do not use `C-LIME`. At the phrase level

[9] This model can be found at `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2` and is distributed under the Apache-2.0 license.

[10] This model can be found at `https://huggingface.co/microsoft/deberta-v2-xxlarge-mnli` under the MIT license.

[11] `https://github.com/neulab/BARTScore`, Apache-2.0 license

You provided the summary below of an article, also below. The article is divided into units (sentences or phrases), which are numbered in the format: <u0> unit 0 <u1> unit 1 ... Please list the {top_k} units that were most important for you to produce this summary. List them in order from most important to least important. List only the unit numbers, for example "<u3>, <u1>, <u4>".

Summary:
{summary}

Article:
<u0> {units[0]} <u1> {units[1]} ...

Figure 6: Prompt for LLM self-explanation

for the summarization datasets, we use C-LIME with $K = 2$ and n equal to the total number of phrases (in the entire document, not just in the sentences selected for refinement). Similarly, at the word level for SQuAD, we set $K = 2$ and n equal to the total number of words. In both cases this corresponds to a ratio $n/d \approx 5$.

C-LIME has two additional aspects to consider:

Sample weighting: ($\pi(z)$ in (2)) Since we limit the number of units K that are simultaneously perturbed to a small integer, we also no longer use LIME's sample weighting scheme. Instead, we give each subset cardinality $k = 0, \ldots, K$ the same total weight (say 1 without loss of generality), and then distribute this weight uniformly over the subsets of that cardinality.

Regularization: Different regularizers $R(w)$ can be used in (2), e.g. ℓ_2 or ℓ_1. In our experiments however, we do not regularize (i.e., $R(w) \equiv 0$) and compute a fully dense solution. This allows all units to be ranked to facilitate evaluation of perturbation curves.

L-SHAP For L-SHAP, the two main parameters are the local neighborhood radius M and the maximum number of neighbors K perturbed at one time. (Note that K does not include the unit of interest, so altogether the maximum number of perturbed units is $K + 1$.) For the DistilBART and Flan-T5-Large models, we take $M = 2$ and $K = 2$. For Flan-UL2, Llama-3, DeepSeek-V3, and Granite-3.3, we again use only LOO at the sentence level and do not use L-SHAP. For phrase-level or word-level L-SHAP, we set $M = 1$ and $K = 2$.

C.4 Baseline attribution methods

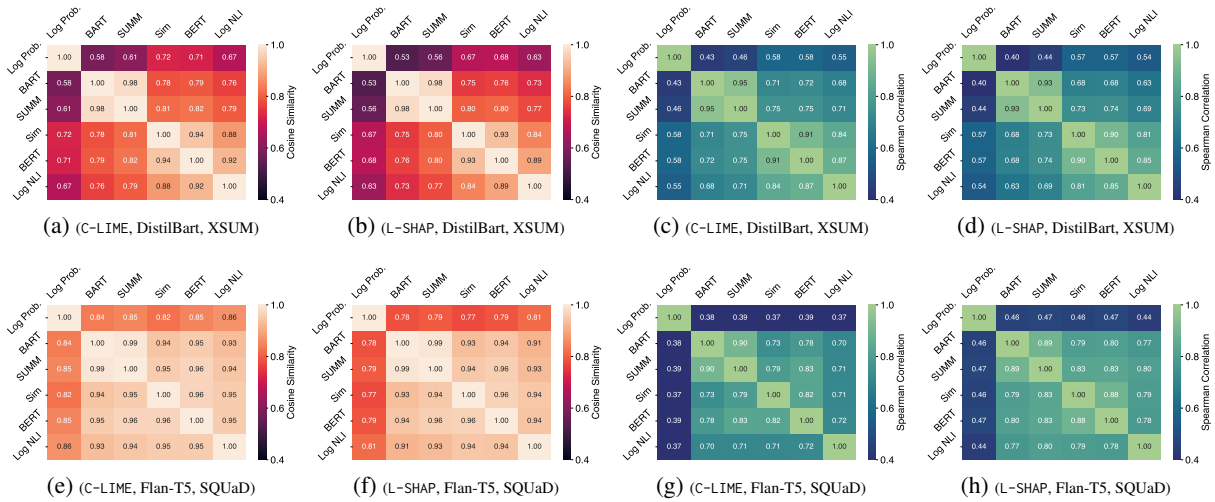
PartitionSHAP We set the number of model queries (parameter max_evals) to be approximately equal to the number used by our L-SHAP and C-LIME algorithms. More specifically, since obtaining mixed-level attributions with MExGen requires first performing sentence-level attribution, we add the numbers of queries used during sentence-level and mixed-level attribution. We then take the larger of these two sums for L-SHAP and C-LIME as the number of queries allowed for PartitionSHAP. All other parameters were kept at their default values in the SHAP library.[12]

CaptumLIME The number of model queries was also set as described above for PartitionSHAP. As discussed in Section 4.1, the units for attribution are the same as used by MExGen C-LIME. All other parameters were kept at their default values in the Captum library.[13]

LLM Self-Explanation Figure 6 shows the prompt used to produce LLM self-explanations for the experiment in Section 4.4, where the task is summarization. We first repeat the LLM's summary back to it. We then segment the input article into the same units units[0], units[1], ..., (sentence-level or mixed-level) used by MExGen. This controls for input segmentation as done in the comparison with CaptumLIME. To make the ranking task easier, we followed LongBench-Cite (Zhang et al., 2024) in prepending a numbered tag (<u0>, <u1>, ...) to each unit. The LLM is asked to list only the tags corresponding to units, in order of decreasing importance for generating the summary. The number top_k of units to be ranked is set to 30% of the total number of units. Since we evaluate perturbation curves up to 20% of the total tokens,

[12] https://github.com/shap/shap, made available under the MIT license

[13] https://github.com/pytorch/captum, distributed under the BSD 3-Clause license



(a) (C-LIME, DistilBart, XSUM) (b) (L-SHAP, DistilBart, XSUM) (c) (C-LIME, DistilBart, XSUM) (d) (L-SHAP, DistilBart, XSUM)

(e) (C-LIME, Flan-T5, SQUaD) (f) (L-SHAP, Flan-T5, SQUaD) (g) (C-LIME, Flan-T5, SQUaD) (h) (L-SHAP, Flan-T5, SQUaD)

Figure 7: Spearman's rank correlation and cosine similarity for different explanation methods in varying datasets.

the 30% setting provides some margin while not requiring the LLM to rank all units.

The comma-separated output from the LLM is parsed into a list of unit indices. Elements that do not yield an integer or yield an out-of-range integer are dropped (this happened rarely however).

C.5 Perturbation curves

MExGen can attribute to mixed units of different lengths in terms of the number of tokens, and these units also differ from those produced by P-SHAP. To account for these differences in computing perturbation curves, we consider both the attribution score and the number of tokens for each unit. We rank units in decreasing order of the attribution score divided by the number of tokens (since we plot perturbation curves as functions of tokens perturbed, this ratio can be seen as the slope in the score-tokens plane). Thus, a shorter unit is ranked higher than a longer unit if both have the same attribution score. We then perturb (more precisely remove) the top k units, compute the output score given by the evaluation scalarizer, and increase k until at least 20% of the tokens have been removed.

To average perturbation curves over examples, which have different numbers of tokens, we divide the numbers of tokens perturbed by the total number of tokens in each example to obtain percentages of tokens. We then linearly interpolate onto a common grid of percentages before averaging and computing standard errors.

C.6 Computing environment

Experiments were run on a computing cluster providing nodes with 32 GB of CPU memory, V100 GPUs with 32 GB of GPU memory, and occasionally A100 GPUs with 40 or 80 GB of GPU memory. One CPU and one GPU were used at a time. The total computation time is estimated to be on the order of 1000 hours.

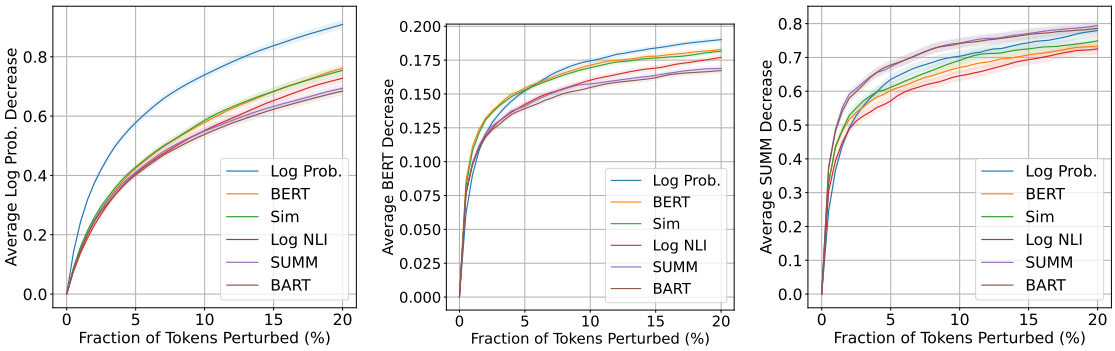
D More Automated Evaluation Results

D.1 Scalarizer Evaluation

On the Cosine Similarity of Explanations Figure 7 shows the cosine similarity between all the pairs of scalarizers we use. We show the cosine similarity between the explanations of each method to analyze how aligned the explanations are from different scalarizers. Each text input $x = x_1, ..., x_d$ receives a multi-level explanation given by $\xi_S(x) = (\xi_S(x)_1, ..., \xi_S(x)_d) \in \mathbb{R}^d$ where each $\xi_S(x)_i$ represents the contribution of unit i to the model prediction scalarization computed using the scalarizer S. We define the cosine similarity between scalarizers S and S' as the average of the cosine similarities between the explanations for all available input texts, i.e.,

$$\texttt{CosSim}(S, S') \triangleq \frac{1}{|X|} \sum_{x \in X} \frac{\langle \xi_S(x), \xi_{S'}(x) \rangle}{||\xi_S(x)|| ||\xi_{S'}(x)||}.$$

Figure 7 (a) shows the cosine similarities for the scalarizers used by MExGen C-Lime to explain the predictions of the DistilBert model in the XSUM dataset. Figure 7 (b) shows the cosine similarities



(a) *Log Prob* as Evaluation (b) *BERT* Score as Evaluation (c) *SUMM* Score as Evaluation

Figure 8: Perturbation curves for `MExGen L-SHAP` with different scalarizers, used to explain the DistilBART model on the XSUM dataset. The curves show the decrease in (a) log probability, (b) BERTScore, and (c) *SUMM* score when removing the most important $p\%$ of tokens according to each explanation scalarizer. Shading shows standard error.

for the scalarizers used by `MExGen L-SHAP` to explain the predictions of the Flan-T5-Large model on the SQUaD dataset.

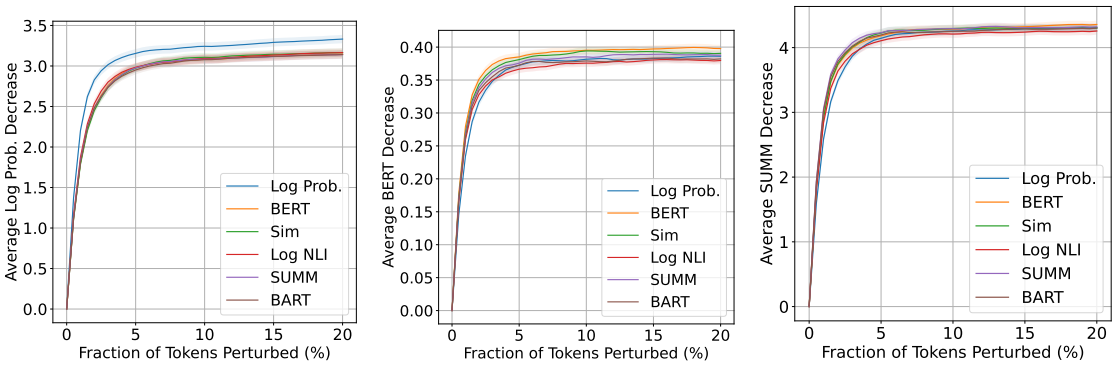
Similarity Across Scalarizers. Figure 7 indicates that, although some scalarizers lead to similar model explanations, there are occasions where scalarizers are more dissimilar. Moreover, the similarities between scalarizers not only depend on the scalarizer itself but also on the model and dataset being explained. For example, Figure 7 (b) shows that when using `MExGen L-SHAP` to provide explanations for the predictions in the XSUM dataset using DistilBrat, the scalarizers *BERT* and *SUMM* are fairly similar ($\texttt{CosSim}(S_{BERT}, S_{SUMM}) = 0.96$). On the other hand, looking at the same pair of scalarizers but for `MExGen` LIME to provide explanations to the predictions in the SQUaD dataset using Flan-T5-Large, BERT and *SUMM* are more dissimilar ($\texttt{CosSim}(S_{BERT}, S_{SUMM}) = 0.82$). This result highlights the necessity of exploring different scalarizers to explain natural language generation, taking into account the task being performed and the main objective of the explanation, i.e., target scalarization.

Similarity to Logits. In the SHAP library, *SUMM* is proposed to provide explanations to LLMs that do not provide access to the logits — hence, the main objective of *SUMM* is to approximate the explanations for the logit when it is not available. However, Figure 7 shows that *SUMM* is not always the best scalarizer for approximating the explanations that would be given if logits were available. For example, Figure 7 (a) shows that the similarity between *SUMM* and logit is near 0.61. In contrast, Figure 7 (b) shows that the similarity between the explanations generated using *SUMM* as scalarizer has a similarity of 0.78 with the onex generated using logits.

We are also aware that only comparing the similarities between explanations might not be enough; once, many researchers use the scores to compute the ranking of the importance across all input features (input text units here). For this reason, next, we compare Spearman's rank correlation to measure the rank stability across different scalarizers.

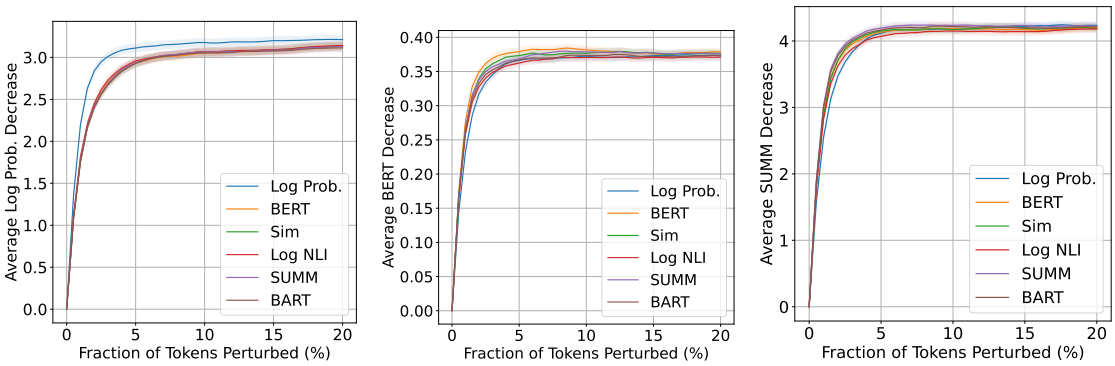
Similarity to Log Probability Scalarizer. In the SHAP library, *SUMM* is proposed to provide explanations when access to logits is not available. However, Figure 3 shows that *SUMM* is not always the best scalarizer for approximating the explanations that would be given if logits were available. For example, Figure 3 (a) shows the ranking correlation between the scalarizer and the Log Prob scalarizer is higher for *BERT* score. Figure 3 (b) shows that the rank generated by both the explanations generated using *SUMM* and *BERT* scalarizer are equally similar to the rank of explanations using the Log Prob scalarizer.

Perturbation curves for different combinations of scalarizers Figures 8, 9, 10 show the perturbation curve for different scalarizations.



(a) Log Prob as Evaluation (b) *BERT* Score as Evaluation (c) *SUMM* Score as Evaluation

Figure 9: Perturbation curves for `MExGen C-LIME` with different scalarizers, used to explain the Flan-T5-Large model on the SQuAD dataset. The curves show the decrease in (a) log probability, (b) BERTScore, and (c) *SUMM* score when removing the most important $p\%$ of tokens according to each explanation scalarizer. Shading shows standard error.



(a) Log Prob as Evaluation (b) *BERT* Score as Evaluation (c) *SUMM* Score as Evaluation

Figure 10: Perturbation curves for `MExGen L-SHAP` with different scalarizers, used to explain the Flan-T5-Large model on the SQuAD dataset. The curves show the decrease in (a) log probability, (b) BERTScore, and (c) *SUMM* score when removing the most important $p\%$ of tokens according to each explanation scalarizer. Shading shows standard error.

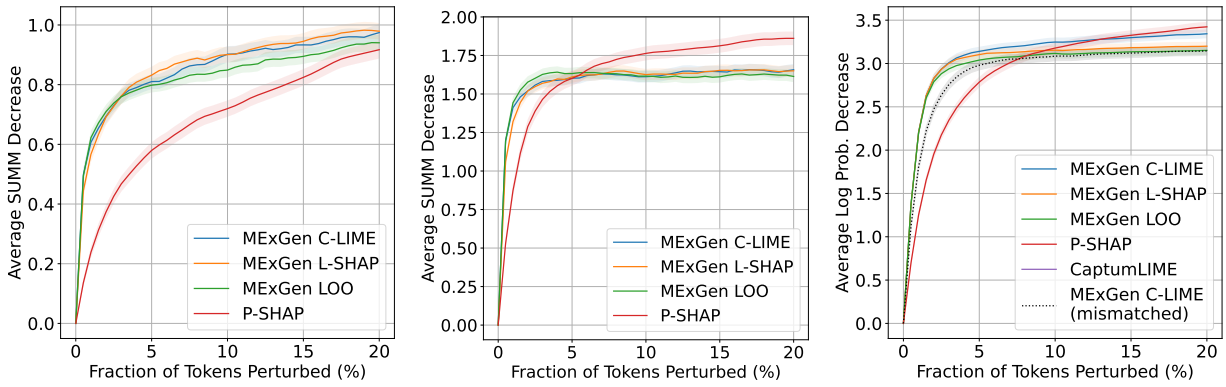
D.2 Comparison Between Explainers

Figure 11 compares the perturbation curves from `MExGen` instantiations and `P-SHAP` for additional model-dataset pairs. The patterns are similar to those in Figure 5. The one exception is in Figure 11b where `P-SHAP` attains a larger *SUMM* decrease as more tokens are perturbed, but the `MExGen` curves are still higher for the top 5% of tokens.

A possible reason for why `P-SHAP` performs better after a certain fraction of tokens in Figures 11b, 11c is as follows: `P-SHAP` perturbs larger subsets of the input than `MExGen`, for which we intentionally limit the size of perturbed subsets. These larger subsets may enable `P-SHAP` to find larger changes in output (higher perturbation curve) at larger fractions of tokens perturbed.

D.3 Comparison with LLM Self-Explanation

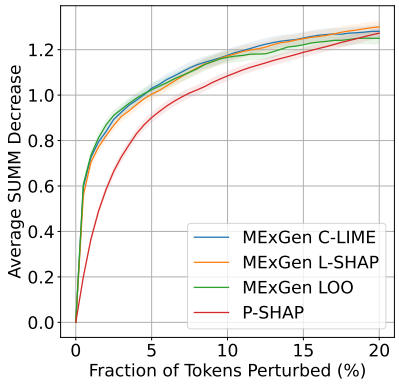
Figure 12 shows the perturbation curves corresponding to the AUPC values reported in Table 2. We also plot the perturbation curve for the intermediate sentence-level attributions (labelled "`MExGen LOO (sent)`") used to obtain the mixed-level `MExGen` attributions, as well as the curve for the self-explanation using the same sentence-level units ("self (sent)"). The curves closely reflect the AUPC comparison already seen in Table 2. For example, `MExGen` greatly outperforms self-explanation when using the *Log Prob* scalarizer in Figure 12a, 12d. On the other hand, for the text-only *BART* scalarizer and larger DeepSeek-V3 model in Figure 12f, the gap is closed at higher perturbed fractions, but a small gap remains at lower fractions.



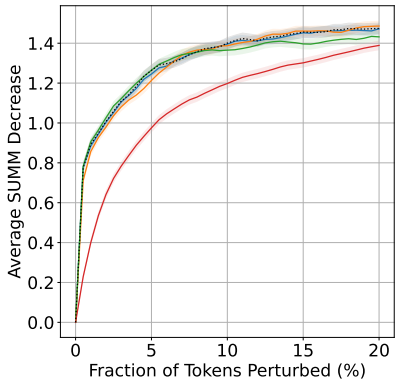
(a) Flan-UL2 on XSUM

(b) Flan-UL2 on CNN/DM

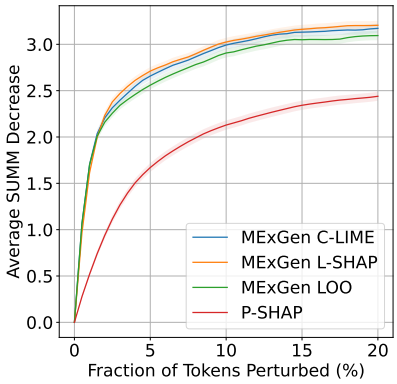
(c) Flan-T5-Large on SQuAD



(d) Llama-3 on XSUM



(e) Llama-3 on CNN/DM



(f) Llama-3 on SQuAD

Figure 11: Perturbation curves (higher is better) from different explanation methods for additional models and datasets: (a) Flan-UL2 on XSUM, (b) Flan-UL2 on CNN/DM, (d) Llama-3-8B-Instruct on XSUM, (f) Llama-3-8B-Instruct on SQuAD, plus (c) Flan-T5-Large on SQuAD and (e) Llama-3-8B-Instruct on CNN/DM repeated from Figure 5 to facilitate comparison. Shading shows standard error in the mean.

E User Study

In this section, we describe our user study.

E.1 Participants

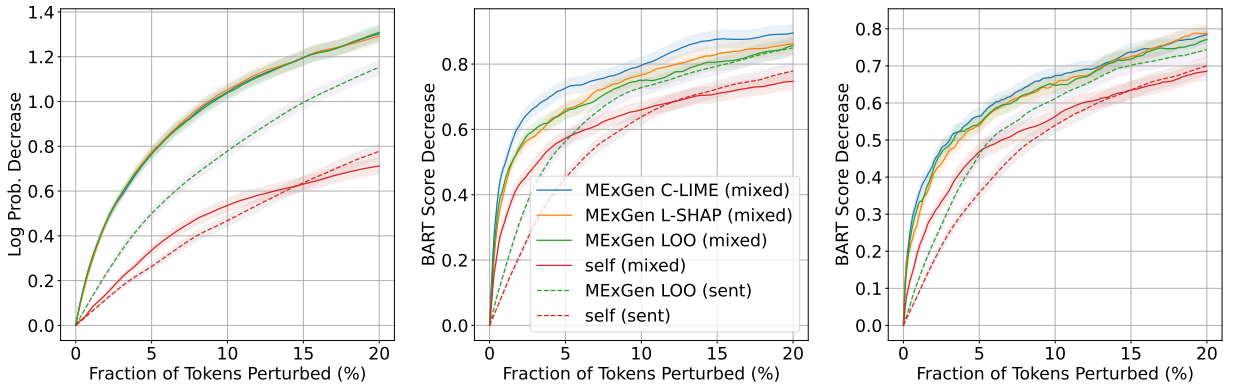
We recruited 96 participants from a large technology company. Those who self-identified as machine learning practitioners using language models were eligible for the study. We filtered out 8 participants who did not pass eligibility checks or did not provide valid responses, resulting in data from 88 participants for our analysis.

Participation in this study did not involve any significant risks beyond those present in daily life, which we explained in the consent form. At the beginning of the study, all participants read about the study procedure, risks, benefits, compensation, and costs, and provided informed consent. They voluntarily participated in the study and were free to withdraw their consent and discontinue participation at any time. Although a formal IRB process does not exist in our institution, we went through an equivalent informal process, including reviewing the study with our peers, and treated participants in accordance with ethical guidelines for human subjects.

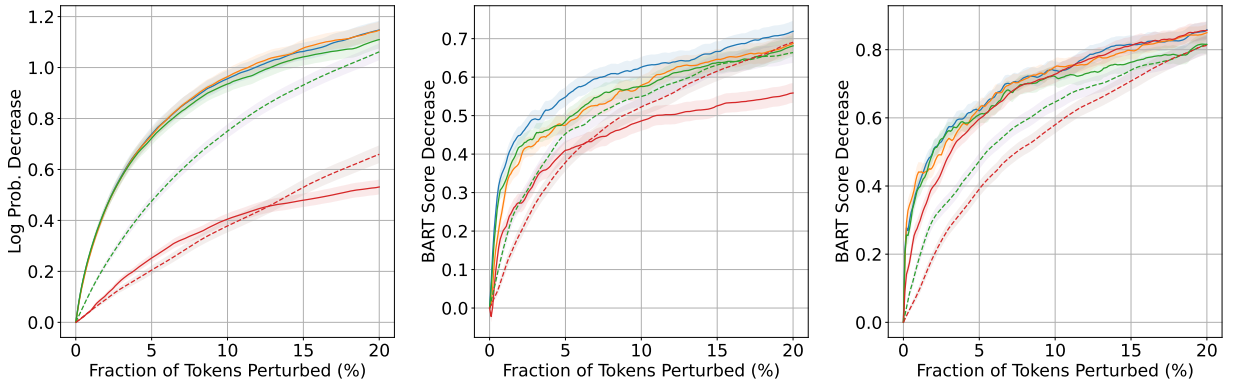
The study was expected to take about 30 minutes or less. For compensation, each participant received 50 points (a digital currency used within the institution), which was equivalent to $12.5 USD.

E.2 Scalarizers

Participants perceived *BERT* to be higher in fidelity than *Log Prob.* We ran a binomial test and found that the selection of *BERT* was significantly higher than the random chance ($p < .05$, 95% CI [.50, .65]). They also preferred *BERT* over *Log Prob* and the choice was statistically significant ($p < .05$, 95% CI [.56, .71]). The type of attribution methods (e.g., `C-LIME`, `L-SHAP`) did not affect the results. Participants perceived that the attribution concentration was adequate overall, as the av-



(a) Granite-3.3 on XSUM with *Prob* (b) Granite-3.3 on XSUM with *Prob* (c) DeepSeek-V3 on XSUM with *BART*



(d) Granite-3.3 on CNN/DM with *Prob* (e) Granite-3.3 on CNN/DM with *Prob* (f) DeepSeek-V3 on CNN/DM with *BART*

Figure 12: Perturbation curves comparing MExGen variants to LLM self-explanation. LLMs, datasets, and scalarizers (the latter used for both explanation and evaluation): (a) Granite-3.3-8B-Instruct on XSUM with *Prob* scalarizer, (b) Granite-3.3-8B-Instruct on XSUM with *BART* scalarizer, (c) DeepSeek-V3 on XSUM with *BART* scalarizer, (d) Granite-3.3-8B-Instruct on CNN/DM with *Prob* scalarizer, (e) Granite-3.3-8B-Instruct on CNN/DM with *BART* scalarizer, (f) DeepSeek-V3 on CNN/DM with *BART* scalarizer. The legend in (bow) applies to all panels. Shading shows standard error in the mean.

erage ratings (*Log Prob*: M=4.32, SD=1.65; *BERT*: M=4.66, SD=1.44) were close to the median of 4 on the 7-point Likert scale. A paired t-test revealed that the difference in the concentration perceptions between scalarizers was not statistically significant.

Selected Option	C-LIME	L-SHAP
Log Prob	35.2%	34.1%
BERT	54.5%	60.2%
Identical	10.2%	5.7%

Table 4: The proportions of participants who selected one of the three options – *Log Prob*, *BERT*, or 'They are identical'. Regardless of attribution methods, significantly more participants chose *BERT* over *Log Prob* when asked to select the one with higher perceived fidelity.

E.3 Attribution Methods

We fitted a Bradley-Terry model (Firth, 2005) for the outcome of pairwise comparisons between attribution methods. The model computes an 'ability' estimate of each method, yielding a complete ranking of methods. Regarding the perceived fidelity, we found that there is a significant difference between C-LIME and L-SHAP ($p < .05/3$ with Bonferroni adjustment), with C-LIME having the highest ability and L-SHAP having the lowest ability. The preference data showed the same pattern in which there is a significant difference between

Preferred Option	C-LIME	L-SHAP
Log Prob	29.5%	31.8%
BERT	62.5%	64.8%
Identical	8%	3.4%

Table 5: The proportions of participants who selected one of the three options – *Log Prob*, *BERT*, or 'They are identical' based on their preference. Regardless of attribution methods, significantly more participants preferred *BERT* over *Log Prob*.

C-LIME and L-SHAP ($p < .05/3$ with Bonferroni adjustment), with C-LIME having the highest ability and L-SHAP having the lowest ability. Other pairs of methods were not significantly different. Participants perceived that the attribution concentration was adequate overall, as the average rating was close to the median on the 7-point Likert scale (M=4.39, SD=1.47). A repeated ANOVA showed that the differences in perceived concentration among the attribution methods were not significant.

Selected vs. Rejected Options	p-value
C-LIME vs. L-SHAP	0.0107 **
PartitionSHAP vs. L-SHAP	0.0455
C-LIME vs. PartitionSHAP	0.5691

Table 6: There is a significant difference in perceived fidelity between C-LIME and L-SHAP. Significant p-values after Bonferroni adjustment are noted with ** (p<0.05/3).

Selected vs. Rejected Options	p-value
C-LIME vs. L-SHAP	0.0074 **
PartitionSHAP vs. L-SHAP	0.1758
C-LIME vs. PartitionSHAP	0.1758

Table 7: There is a significant difference in preference between C-LIME and L-SHAP. Significant p-values after Bonferroni adjustment are noted with ** (p<0.05/3).

E.4 Granularity Preference

Participants were asked to select their preferred granularity of attributions (sentence-level vs. multi-level). While the number of participants who preferred multi-level granularity (56.2%) was slightly higher than those who preferred sentence-level granularity (43.8%), binomial tests indicated that their granularity choice was not statistically significant. The preference for granularity did not significantly vary across attribution algorithms (C-LIME, L-SHAP).

E.5 Survey

The survey was structured as below with the following instructions and questions:

1. **Consent**
   - Select 'I agree' if you are eligible and agree to the terms above. By selecting 'I agree', you give consent to [*Institution Name*] to use your anonymized responses for research and development purposes. You also agree not to provide any information that is confidential or proprietary.
2. **Input document (1)**: Suppose you wanted to summarize an input document and used a language model to generate a summary. Please read the text below and answer the following questions. [*A randomly selected input document and a summary were inserted here*]
   - Rate the overall quality of the summary. A good summary should be coherent, consistent, fluent, relevant, and accurate. (1: Poor - 7: Excellent)
3. **Scale**: Introducing the scale that was used to annotate the attribution scores. See Fig. 13
4. **Scalarizer (2 pairwise comparisons)**: Using the selected 'input document (1)', same instructions and questions were used as shown in Fig. 14.
5. **Input document (2)**: Suppose you wanted to summarize an input document and used a language model to generate a summary. Please read the text below and answer the following questions. [*Another randomly selected input document and a summary were inserted here*]
   - Rate the overall quality of the summary. A good summary should be coherent, consistent, fluent, relevant, and accurate. (1: Poor - 7: Excellent)
6. **Attribution algorithms (3 pairwise comparisons)**: Using the selected 'input document (2)', same instructions and questions were used as shown in Fig. 14.
7. **Granularity**: The attribution scores can be presented in two levels of granularity, which are sentence- level and mixed-level granularities.
   - Sentence-level granularity: each sentence in the input document is color coded based on how much of it was used by the language model.
   - Mixed-level granularity: each phrase within a few high-scoring sentences is color coded based on how much of it was used by the language model. Low-scoring sentences are coded in the sentence-level granularity.

   In the next two pages, you will select your preferred granularity for each of the following examples.
   - Which granularity do you prefer in this example? [*A randomly selected summary and two highlighted input documents*] See Figure 15.
   - Which granularity do you prefer in this

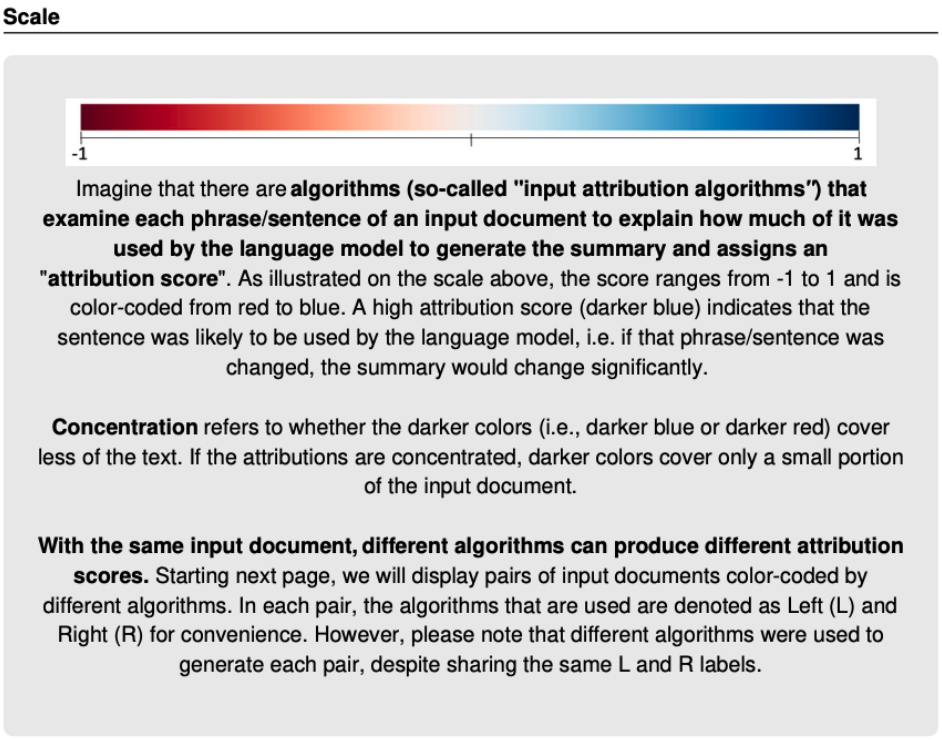


Figure 13: A scale used to color-code the attribution scores.

example? [*Another randomly selected summary and two highlighted input documents*]. See Figure 15.

8. **Background**
    - What is your job title?
    - Where is your work location?
    - What type of work do you do? (Please check all that apply.)
    - What is your proficiency level in English?
    - How often do you use language models either as part of your job or as a hobby?
    - What kind of tasks do you usually do with language models?
    - Besides the attribution scores/highlights that are shown in this study, what other information would you like to know to help you understand how a language model generated a summary?

Figure 14 and Figure 15 show primary questions we asked in the survey with screenshots.

F Future Directions

Hierarchical explanations It could be profitable in future work to incorporate the hierarchical explanations discussed in Appendix A into the multi-level `MExGen` framework. The method of Chen and Jordan (2020) may be especially relevant since it leverages a constituency parse tree to compute word-level importances, which may be related to our use of dependency parse trees.

Word infilling with BERT We have explored perturbing words by masking them and then calling a BERT model to fill the masks with different words that fit within the sentence. However, we have thus far not seen a quantifiable benefit to using BERT compared to replacing with a fixed baseline value (such as an empty string). Our experience is in line with the the mixed results reported by Pham et al. (2022) on using BERT in this manner.

Phrase segmentation Segmentation of sentences into phrases could of course be done in ways other than our dependency parsing algorithm, for example using constituency parsing instead. A possible advantage of using dependency parsing is that each phrase can be labeled with the dependency label of its root token and treated differently on this basis.

Figure 14: Survey questions. Participants answered a series of questions related to perceived fidelity and general preference comparing a pair of attribution methods, followed by questions related to perceived concentrations of each method.

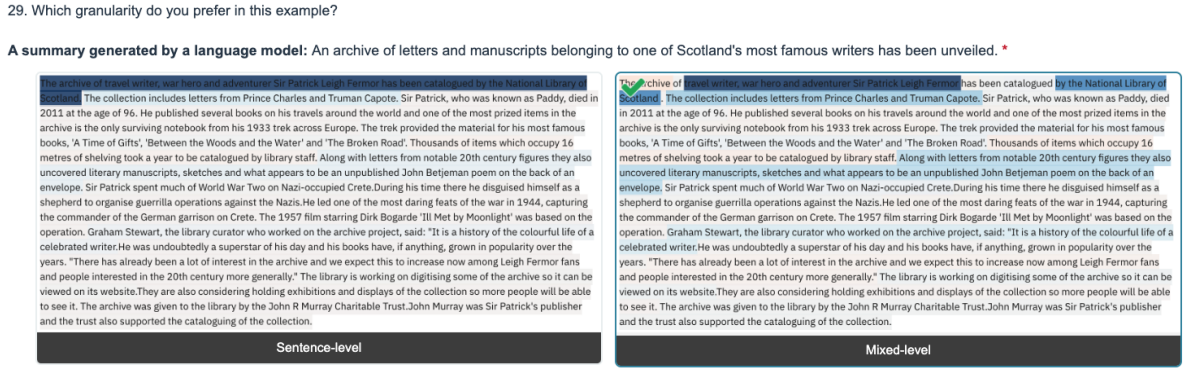


Figure 15: Granularity question. Participants were asked to select their preferred granularity of attributions (sentence-level vs. multi-level).