NormXLogit: The *Head-on-Top* Never Lies

Sina Abbasi[1] **Mohammad Reza Modarres**[1] **Mohammad Taher Pilehvar**[2]
[1] Tehran Institute for Advanced Studies, Khatam University, Iran
[2] Cardiff University, United Kingdom
s.abbasi@teias.institute mr.modarres@teias.institute
pilehvarmt@cardiff.ac.uk

Abstract

With new large language models (LLMs) emerging frequently, it is important to consider the potential value of model-agnostic approaches that can provide interpretability across a variety of architectures. While recent advances in LLM interpretability show promise, many rely on complex, model-specific methods with high computational costs. To address these limitations, we propose NormXLogit, a novel technique for assessing the significance of individual input tokens. This method operates based on the input and output representations associated with each token. First, we demonstrate that the norm of word embeddings can be utilized as a measure of token importance. Second, we reveal a significant relationship between a token's importance and how predictive its representation is of the model's final output. Extensive analyses indicate that our approach outperforms existing gradient-based methods in terms of faithfulness and offers competitive performance compared to leading architecture-specific techniques.

1 Introduction

Transformer-based models have gained widespread adoption across various natural language processing (NLP) tasks, demonstrating their versatility. However, the underlying mechanisms of these models are not quite understood. This means when the model fails and generates inaccurate or harmful content, we are unable to diagnose the source and improve the model's behavior. Consequently, a multitude of endeavors in recent years aimed at enhancing the interpretability of these models.

Architecture-agnostic methods, such as perturbation-based and gradient-based techniques, are widely used to identify influential input tokens that impact a model's predictions. However, these approaches face significant challenges, including the generation of out-of-distribution inputs and vulnerability to adversarial exploitation (Wang et al., 2020). Additionally, they often come with high computational costs, limiting their scalability. For instance, advanced gradient-based methods like Integrated Gradients (Sundararajan et al., 2017) involve repeated forward and backward passes, leading to substantial overhead.

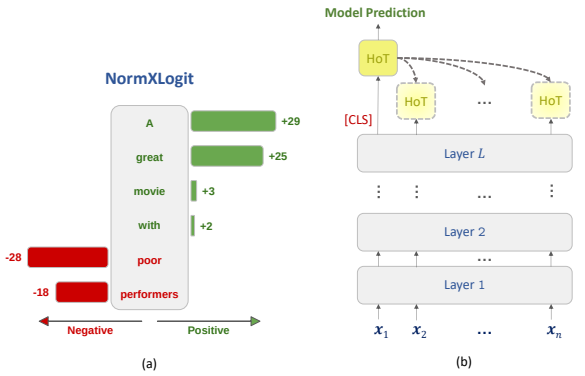

Figure 1: (a) Importance scores of NormXLogit for the sentiment analysis task. NormXLogit generates attributions per-label with signed scores denoting positive/negative impact. (b) Applying the head-on-top (HoT) on each of the final representations to obtain a prediction based on each token.

By leveraging the internal components of the target model, a new class of architecture-specific approaches, termed vector-based methods, has been developed (Kobayashi et al., 2021; Ferrando et al., 2022). Most of these approaches offer per-layer explanations, which are subsequently aggregated using the *rollout* technique (Abnar and Zuidema, 2020) to achieve a global interpretation that integrates all layers of the model. However, rollout can result in inaccurate outcomes due to the vanishing attribution problem (Mehri et al., 2024). Despite recent advances in faithfulness, a key drawback of vector-based methods is their architecture-specific design, which limits their adaptability to the rapidly evolving landscape of LLMs. Furthermore, almost

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 34926–34947
November 4-9, 2025

all of these methods ignore the *head-on-top*[1] which is crucial to produce task-dependent explanations.

In response to these limitations, this paper presents NormXLogit, a simple yet powerful, computationally efficient, and architecture-agnostic[2] approach that outperforms many sophisticated techniques and can be easily applied to any NLP task. NormXLogit eliminates the need for any backward passes, utilizing the pre-trained model only once. It leverages the informativeness of the norm of input embeddings, in conjunction with the rich semantic and syntactic information encoded in the model's final-layer representations. The latter enables the incorporation of all internal components, thereby removing the need for any aggregation method, such as rollout. We incorporate the head-on-top of pre-trained models into our analysis, which serves as a task-specific interpreter of token attributions[3], tailoring its attention to the nuances of the task. This helps NormXLogit to generate per-label attributions with positive and negative impacts of each input token (cf. Figure 1(a)).

Based on comprehensive experiments across multiple tasks and models, we show that in numerous instances, the faithfulness of NormXLogit surpasses that of widely recognized gradient-based approaches. In the regression setup, it demonstrates superior performance compared to a state-of-the-art architecture-specific baseline. Furthermore, through a targeted experiment on the BLiMP corpus (Warstadt et al., 2020), we evaluate the alignment between NormXLogit's attributions and known linguistic evidence across diverse grammatical phenomena. Our results demonstrate that NormXLogit consistently yields more faithful and plausible explanations than existing approaches, highlighting its effectiveness as a general-purpose token attribution method.

2 Related Work

In recent years, vector-based analyses have emerged as an architecture-specific approach to Transformer interpretability. These methods build on findings that attention weights can be misleading (Jain and Wallace, 2019; Serrano and Smith, 2019). Kobayashi et al. (2021) extended analysis beyond attention weights by decomposing the entire attention block. To aggregate per-layer attributions into global ones, Abnar and Zuidema (2020) proposed *attention rollout* to quantify the flow of information through self-attention.

GlobEnc (Modarressi et al., 2022) and ALTI (Ferrando et al., 2022) advanced previous work by further incorporating other Transformer components. ALTI challenged the use of ℓ^2 norms for measuring contributions, proposing the Manhattan distance for improved results. GlobEnc added the second residual connection and layer normalization into the analysis but overlooked the impact of the feed-forward network. To address this, DecompX (Modarressi et al., 2023) captured the influence of the feed-forward network by approximately decomposing the activation function and propagating decomposed vectors across layers, eliminating reliance on aggregation methods like rollout.

Perturbation-based methods, such as SHAP (Lundberg and Lee, 2017) and LIME (Ribeiro et al., 2016), investigate the causal link between input features and the model's final prediction by perturbing or erasing parts of the input. Recently, Mohebbi et al. (2023) proposed *Value Zeroing* which is based on the Explaining-by-Removing intuition (Covert et al., 2022), in order to quantify the context mixing. Their approach, in contrast to other perturbation-based methods, does not remove the input token representations. Value Zeroing instead suggests zeroing the value vector of each token to measure its contribution.

Gradient-based methods involve analyzing the gradients of the model's output with respect to the input features to understand their impact on the model's decision-making process. Methods such as Gradient Norm (Simonyan et al., 2014), Gradient$\times$Input (Kindermans et al., 2016), and Integrated Gradients (Sundararajan et al., 2017) are the most prominent ones in this category.

3 Proposed Approach

3.1 Background: Transformer Architecture

The Transformer architecture consists of multiple identical layers, each with a multi-head self-attention (MHA) block and a position-wise fully connected feed-forward network (FFN), where both are followed by a residual connection (RES) and layer normalization (LN).

The MHA component is responsible for creating contextualized representations for the input el-

[1]By 'head-on-top', we are referring to the classification or regression head used on top of the pre-trained models.

[2]By using the term "architecture-agnostic," we aim to differentiate our approach from vector-based methods, ensuring generalization across different Transformer-based models.

[3]We use 'attribution' and 'importance' interchangeably.

ements. The output representation of MHA for token x_i and head h in l-th layer is then calculated as the weighted sum of transformed input representations:

$$\tilde{x}_i^{l\,\textcircled{h}} = \sum_{j=1}^{n} \alpha_{i,j}^{l\,\textcircled{h}} v_j^{l\,\textcircled{h}}, \tag{1}$$

where $\alpha_{i,j}^{l\,\textcircled{h}}$ represents the attention weight of token i with respect to token j in the h-th head of the MHA of the l-th layer.

Kobayashi et al. (2021) demonstrated that the output representation of each token produced by the attention block can be explained via two effects: (i) "preserving" its original input using the RES and the contribution of the token itself through context mixing of MHA, and (ii) "mixing" the representations in the context (except the target token). They showed that the preserving effect is predominant, primarily due to the higher contribution of RES to the output representation.

3.2 Norm of Word Embedding

Oyama et al. (2023) demonstrated that the norm of input embeddings encodes information gain. They showed that tokens with higher ℓ^2 norm carry more information, effectively capturing the least frequent words in the text. Additionally, based on the Eq. 1, the MHA could be interpreted as the weighted sum of transformed vectors. In other words, the final representation of each token is built by mixing the representations of all tokens in the input sequence. Consequently, tokens with higher norms are expected to contribute more to the final representation of the target token. Higher contribution suggests greater importance, allowing us to utilize the ℓ^2 norm of word embeddings to identify crucial tokens influencing the model's decision.

3.3 LogAt: Logit Attribution

The tasks in the domain of NLP can be broadly divided into two main categories: classification tasks and regression tasks. For both of these setups, we utilize a special token (often known as `[CLS]`[4]), which is embedded in almost all pre-trained models. This token serves as a single vector representing the entire input sequence, which is then fed into head-on-top, an FFN placed on top of the pre-trained model to produce the output prediction.

[4]The name of this special token may vary depending on the model. Also, in auto-regressive models, the last token in the input is typically used for classification.

The intuition behind the attention mechanism implies that more important tokens have a greater contribution to building the final representation of the `[CLS]` token. In other words, an identical `[CLS]` embedding is fed into the model for all input sequences, and based on the fine-tuning objective, the attention block attempts to utilize the most relevant (i.e., important) tokens to construct the new representation of `[CLS]`. This suggests that the `[CLS]` token has a higher degree of similarity to the most important input tokens in the model's decision-making process. To identify the tokens that are most similar to `[CLS]`, we use the head-on-top to evaluate how each individual token in the input contributes to predicting the target task. In the following, we describe the approach for each setup.

Classification. In a classification setup, the output of the head-on-top for each sample is a vector of length equal to the number of labels (i.e., classes). The values in this output vector are referred to as logits, which are further processed using the softmax function to obtain probabilities over the output labels. The model's final prediction is the label associated with the highest logit value. To determine the most important input tokens for a model with L layers, we apply the head-on-top to each one of the output representations at layer L, as illustrated in Figure 1(b). Next, we extract the logits corresponding to the predicted class, which is already determined by applying the head-on-top to the `[CLS]` token. The logit value associated with each token represents its attribution, and tokens with the highest logits are regarded as the most important for the classification task. We call this method Logit Attribution (LogAt). To calculate the attribution ($\text{Att}_{\text{LogAt}}$) of the token i for a task with C classes and a classification head $\text{HoT}_{\text{clas}}(\cdot) \in \mathbb{R}^C$, we have:

$$\text{Att}_{\text{LogAt}}(x_i) = \text{HoT}_{\text{clas}}(x_i^L)[\hat{p}], \tag{2}$$

where x_i^L is the final representation of the i-th token in a model with L layers, and $\hat{p} \in \{0, 1, ..., C-1\}$ denotes the index of the predicted class. By changing the index of $\hat{p}$ to other class labels in the task, we can identify the important tokens relative to those classes as well, leading to a per-label attribution technique.

Due to the dominance of the "preserving" effect in the attention block, the contextualized representations in the last layer still retain the identity of

the original input tokens. As a result, the logits can be seen as a direct reflection of each token's contribution. The use of the head-on-top provides task-specific explanations, allowing us to semantically identify the tokens that are most critical for the target task. Furthermore, the sign of the logits provides insight into the direction of the contributions, indicating whether each label is positively or negatively influenced, specifically with respect to the model's predicted label.

To interpret the choice of token in various **language modeling objectives**, we categorize them as classification tasks, where the number of labels corresponds to the vocabulary size. In language modeling, the goal is to predict the correct word given the context, which yields a probability distribution over the vocabulary for generating each individual word. In this setup, we utilize the masked language modeling head as the head-on-top to identify the tokens that contribute most to predicting the `[Mask]` token.

Regression. For the regression setup, the approach typically involves generating a single value in the output rather than a vector of probabilities. So, instead of taking the largest logit corresponding to the prediction label, we take the absolute distance of the output for each token from the model's prediction. For the attribution of i-th token in a regression task, we have:

$$\text{Att}_{\text{LogAt}}(x_i) = \left|\text{HoT}_{\text{reg}}(x_i^L) - \text{HoT}_{\text{reg}}(\texttt{[CLS]}^L)\right|, \quad (3)$$

where $\text{HoT}_{\text{reg}}(\cdot) \in \mathbb{R}$ denotes the regression head, x_i^L is the final representation of the i-th token in a model with L layers, and $\texttt{[CLS]}^L$ is the final representation of the `[CLS]` token.

3.4 NormXLogit

Although LogAt provides valuable explanations of the model's decision-making process, our experiments show that considering the informativeness of the norm of word embeddings can yield more faithful results. Therefore, we introduce NormXLogit, an architecture-agnostic interpretation method that can be applied to any task and domain. The attribution of token i using NormXLogit is obtained as:

$$\text{Att}_{\text{NormXLogit}}(x_i) = \left\|x_i^0\right\|_2 \times \text{Att}_{\text{LogAt}}(x_i), \quad (4)$$

where the $\left\|x_i^0\right\|_2$ is the ℓ^2 norm of the input word embedding for the i-th token, and $\text{Att}_{\text{LogAt}}(\cdot)$ is the LogAt attribution according to the task setup.

4 Experiment 1: Faithfulness Analysis

To analyze the *faithfulness* of NormXLogit, we begin with a set of experiments on classic classification and regression tasks. Our goal is to evaluate whether the tokens identified as important are truly aligned with the model's internal reasoning.

4.1 Experimental Setup

Data. In the classification setup, we will use SST-2 (Socher et al., 2013) for sentiment analysis, MultiNLI (Williams et al., 2018) for recognizing textual entailment, and QNLI for question answering (Wang et al., 2018). SST-2 contains movie review sentences labeled as positive or negative, while MultiNLI includes sentence pairs labeled as entailment, contradiction, or neutral. QNLI consists of question–sentence pairs labeled for entailment. STS-B (Cer et al., 2017) is used for semantic similarity as a regression task, with scores from 0 (no similarity) to 5 (semantic equivalence).

Models. Our target models in this section involve three prominent models: LLAMA 2 (Touvron et al., 2023), DeBERTa (He et al., 2023), and BERT (Devlin et al., 2019).[5] We use the fine-tuned version of each model for the corresponding task. To fine-tune LLAMA 2 and perform inference, we employ the LoRA (Hu et al., 2021) technique with a rank of 4 from the PEFT library[6].

Input Attribution Methods. To analyze the performance of our proposed method, we compare NormXLogit with three well-known gradient-based input attribution methods: Gradient Norm (Grad. Norm), Gradient×Input (G×I), and Integrated Gradients (IG), using the ℓ^1 norm for aggregation. Notably, we focus on gradient-based methods over perturbation-based approaches due to their more faithful results. To account for vector-based approaches, we adopt DecompX as the current state-of-the-art method among them. However, this family of methods is primarily developed for BERT-like architectures and may not be applicable to all models. In addition, we consider a random baseline where tokens are ranked randomly from most important to least important.

[5] We employ the 7 billion parameter variant of LLAMA 2, the uncased $\text{BERT}_{\text{base}}$ model, and $\text{DeBERTaV3}_{\text{base}}$, all from HuggingFace's Transformers library (Wolf et al., 2020).

[6] https://github.com/huggingface/peft

4.2 Evaluation Metrics

To assess the faithfulness of the aforementioned methods, we adopt two complementary criteria: **comprehensiveness** and **sufficiency** (DeYoung et al., 2020). Both measure how the model's behavior changes when input tokens identified as important are perturbed. Each criterion is evaluated through two metrics: (i) *Confidence Drop*, which quantifies how the model's probability for the predicted class changes, and (ii) *Accuracy*, which evaluates how overall task performance is affected. For classification tasks, we report both metrics, while regression tasks use only Accuracy due to the absence of discrete class probabilities.

4.2.1 Comprehensiveness

Comprehensiveness evaluates model performance when different proportions (e.g., 10%, 20%, ...) of the most important input tokens are perturbed, reflecting the necessity of these tokens for the model's prediction. For the masked language modeling objectives, masking is used for token perturbations, while for auto-regressive models, deletions are employed due to the absence of a `[MASK]` token. We quantify comprehensiveness using two metrics: Confidence Drop and Accuracy.

Confidence Drop: This metric measures how much the model's confidence in its predicted label decreases when the most important tokens are removed. For a given input sequence X_i, the perturbed sequence $X_i \backslash K$ is generated by applying the perturbation on $K\%$ of the most important tokens. Then, for all of the instances in the dataset, the average Confidence Drop is defined as:

$$\text{Comp}_{\text{CD}}(K\%) = \frac{1}{m}\sum_{i=1}^{m}\left[f_{\hat{y}}(X_i) - f_{\hat{y}}(X_i \backslash K)\right], \tag{5}$$

where m is the number of instances, and $f_{\hat{y}}(X)$ is the model's output probability for label $\hat{y}$. For comprehensiveness, a higher confidence drop indicates better attribution performance, as the model's prediction is more strongly affected by the removal of important tokens.

Accuracy: This metric works by calculating the model's accuracy on the target dataset, where for each instance the top $K\%$ most important tokens are perturbed. For regression tasks, we utilize the Pearson correlation coefficient as the Accuracy metric. For classification tasks, we report standard accuracy (i.e., the percentage of correctly predicted labels) on the perturbed inputs. Formally,

$$\text{Comp}_{\text{Acc}}(K\%) = \text{Acc}(\mathcal{D}^{\backslash K}), \tag{6}$$

where $\mathcal{D}^{\backslash K} = \{(X_i \backslash K, y_i)\}_{i=1}^{m}$ is the perturbed dataset. Here, lower values indicate higher faithfulness, as perturbing key tokens is expected to decrease task accuracy.

4.2.2 Sufficiency

Sufficiency evaluates whether the highlighted tokens alone are adequate to preserve the model's prediction. Unlike comprehensiveness, which perturbs important tokens, sufficiency masks or removes all *unimportant* tokens, retaining only the top $K\%$ identified as most important. We again evaluate sufficiency through both Confidence Drop and Accuracy.

Confidence Drop: This variant measures the decrease in confidence when the input is reduced to only the important tokens:

$$\text{Suff}_{\text{CD}}(K\%) = \frac{1}{m}\sum_{i=1}^{m}\left[f_{\hat{y}}(X_i) - f_{\hat{y}}(X_i^K)\right], \tag{7}$$

where X_i^K denotes the sequence containing only the top $K\%$ tokens. Here, lower values are better, since keeping only the important tokens should ideally preserve the model's confidence.

Accuracy: Similarly, task performance can be measured on inputs restricted to the important tokens:

$$\text{Suff}_{\text{Acc}}(K\%) = \text{Acc}(\mathcal{D}^{K}). \tag{8}$$

For sufficiency, a higher accuracy indicates better attribution performance.

4.3 Results

4.3.1 Comprehensiveness

Figure 2 illustrates the superior performance of NormXLogit in LLAMA 2 fine-tuned on SST-2. NormXLogit achieves a higher Confidence Drop for the comprehensiveness criterion across all thresholds, indicating its effectiveness in identifying crucial tokens for the model's decision-making process. It should be noted that DecompX, due to its architecture-specific nature, may not be applicable to LLAMA 2. Additionally, even if it were, the computational cost of DecompX may not be easily manageable given the size of LLAMA 2 on many accessible hardware configurations.

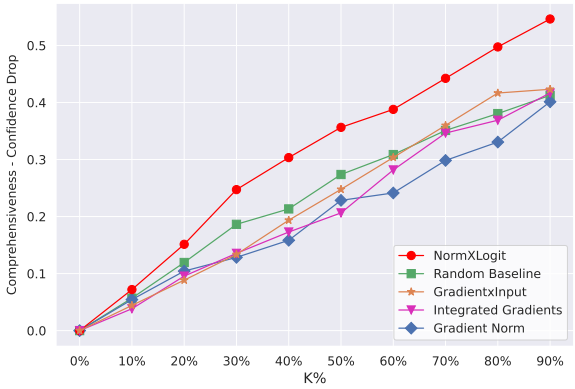

Figure 2: Comprehensiveness Confidence Drop of different attribution methods for LLAMA 2 fine-tuned on SST-2 (higher values are better).

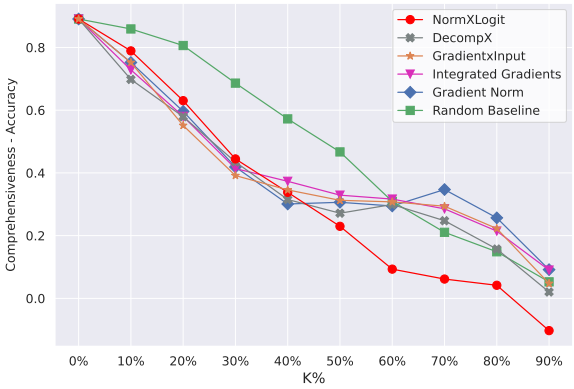

Figure 3: Comprehensiveness Accuracy of different attribution methods for BERT fine-tuned on STS-B (lower values are better).

	SST-2 ($\text{Comp}_{\text{CD}}\uparrow$)			**MNLI** ($\text{Comp}_{\text{CD}}\uparrow$)			**QNLI** ($\text{Comp}_{\text{CD}}\uparrow$)			**STS-B** ($\text{Comp}_{\text{Acc}}\downarrow$)		
	LLAMA 2	**DeBERTa**	**BERT**	**LLAMA 2**	**DeBERTa**	**BERT**	**LLAMA 2**	**DeBERTa**	**BERT**	**LLAMA 2**	**DeBERTa**	**BERT**
Random	0.256	0.266	0.245	0.421	0.445	0.361	0.284	0.306	0.273	0.283	0.430	0.457
Grad. Norm	0.216	0.320	0.331	0.364	0.535	0.460	0.334	0.365	0.360	0.351	0.338	0.374
G×I	0.236	0.345	0.339	0.442	0.565	0.456	0.353	0.382	0.364	0.255	0.214	0.358
IG	0.220	0.346	0.367	0.448	**0.571**	0.466	0.336	0.381	0.364	0.237	0.227	0.370
DecompX	N/A	N/A	**0.574**	N/A	N/A	**0.585**	N/A	N/A	**0.460**	N/A	N/A	0.336
ℓ^2 norm	0.299	0.360	0.311	0.420	0.473	0.393	0.272	0.339	0.304	0.251	**0.199**	0.321
LogAt	0.341	0.377	0.364	0.518	0.548	0.566	**0.378**	0.435	0.394	**0.167**	0.423	0.313
NormXLogit	**0.341**	**0.386**	0.423	**0.519**	0.566	0.556	0.363	**0.474**	0.402	0.233	0.320	**0.281**

Table 1: Comprehensiveness of NormXLogit against other methods across various model and dataset configurations. Each value is computed by averaging across all perturbation ratios (higher Confidence Drop and lower Accuracy are better). Best values are in **bold**, and second-best values are underlined.

In the regression setup of STS-B depicted in Figure 3, dropping important tokens results in a decrease in Accuracy. To leverage DecompX for this setup, in the absence of a classification head, we applied the ℓ^2 norm to the decomposed vectors obtained from the final layer. The results for the initial $K\%$ ratios are very close, with DecompX and Gradient×Input showing a slight lead at the outset. However, after dropping 40% of the most important tokens, the performance of all these methods deteriorates, while NormXLogit continues to experience a drop in Accuracy.

Table 1 presents the average Confidence Drop and Accuracy across different ratios of perturbation, evaluated on various models and datasets[7]. For classification tasks, the Accuracy metric results are consistent with Confidence Drop measurements and are provided in the appendix to avoid repetition (cf. Table 4). In SST-2 and QNLI, NormXLogit consistently outperforms architecture-agnostic methods. However, DecompX, which is specific to the BERT architecture, results in a larger confidence drop. In the MultiNLI dataset, NormXLogit performs better than gradient-based approaches in LLAMA 2 and BERT, though Integrated Gradients show a slight edge in the DeBERTa model. In the BERT model, similar to SST-2, DecompX performs better, but the difference is notably smaller compared to the SST-2 dataset.

In the regression setup, the ℓ^2 norm performs surprisingly well and also boosts the performance of Gradient×Input. This strong performance on STS-B can be attributed to the structure of STS-B samples: sentence pairs consist of relatively short sentences with similar openings, where key information that determines the label often resides in less frequent words toward the end. Such words tend to have higher embedding norms, making the ℓ^2 norm approach more effective.

Furthermore, in the BERT model, the absence of a classification head diminishes DecompX's effectiveness, resulting in performance worse than

[7]The corresponding diagrams are provided in Section A.2.

Input Length	Attribution Method				
	Gradient Norm	Gradient×Input	Integrated Gradients	DecompX	NormXLogit
40	$0.77_{\pm 0.03}$	$0.80_{\pm 0.02}$	$0.82_{\pm 0.02}$	$0.39_{\pm 0.01}$	$\mathbf{0.36}_{\pm 0.00}$
120	$0.77_{\pm 0.02}$	$0.81_{\pm 0.02}$	$0.96_{\pm 0.03}$	$0.38_{\pm 0.00}$	$\mathbf{0.36}_{\pm 0.00}$
360	$0.76_{\pm 0.03}$	$0.80_{\pm 0.02}$	$1.47_{\pm 0.05}$	$0.99_{\pm 0.02}$	$\mathbf{0.38}_{\pm 0.01}$
512	$0.78_{\pm 0.02}$	$0.80_{\pm 0.02}$	$1.79_{\pm 0.06}$	$2.36_{\pm 0.02}$	$\mathbf{0.36}_{\pm 0.00}$

Table 2: Average time (in seconds) per instance for different attribution methods across various input lengths. NormXLogit demonstrates significantly lower computational costs compared to other methods and remains unaffected by input sequence length. The values in subscript represent the standard deviation.

that of the input embeddings' norms. In LLAMA 2, NormXLogit slightly surpasses all gradient-based methods, largely due to the strong performance of LogAt.

4.3.2 Sufficiency

While NormXLogit repeatedly outperforms gradient-based methods in comprehensiveness, it lags behind in sufficiency (cf. Table 4). This difference arises from how transformers encode meaning: tokens often matter through their interactions with surrounding context rather than in isolation. NormXLogit highlights such contextually important tokens, as their representations strongly contribute to the full-sequence logit. Removing these tokens therefore causes a substantial confidence drop, which explains the method's strength under comprehensiveness. However, when only the top-ranked tokens are retained, the contextual cues needed to sustain the original prediction are lost, leading to weaker sufficiency scores. In other words, NormXLogit excels at identifying *necessary* tokens within context but is less effective when tokens must stand alone as *sufficient* evidence.

4.4 Computational Efficiency

NormXLogit is not only faithful and generalizable, but also computationally lightweight. In perturbation-based methods, it is costly to search for appropriate combinations of tokens to intervene, as this requires multiple forward passes. Similarly, advanced gradient-based methods like Integrated Gradients incur substantial overhead due to repeated forward and backward passes. In contrast, NormXLogit performs attribution in a single forward pass, eliminating the need for any backpropagation.

Our runtime analysis shows that for longer inputs (e.g., 360 tokens or more), NormXLogit can be up to $6\times$ faster and up to $750\times$ more memory-efficient than its counterparts (cf. Tables 2 and 7).

5 Experiment 2: Evidence Alignment

In our second experiment, we extend our evaluation beyond classification and regression by focusing on the language modeling task, using the BLiMP dataset. This experiment serves four purposes: (1) to assess the *faithfulness* of NormXLogit in the masked language modeling setting, expanding the scope of our evaluation, (2) to examine its *plausibility*, i.e., the extent to which its attributions align with human-understood linguistic rationales provided by BLiMP, (3) to analyze the target model's sensitivity to specific linguistic phenomena, and (4) to investigate how *contextualization* evolves throughout the model by analyzing the output representations at different layers of the Transformer (i.e., per-layer attributions).

5.1 Experimental Setup

Data. The BLiMP dataset contains sentence pairs with minimal contrasts in syntax, morphology, or semantics. It is constructed to provide samples where the true label is uniquely determined by a single word in each sentence, providing an ideal benchmark for assessing attribution methods. This word, which serves as the decisive factor in determining grammatical acceptability, is termed the *evidence*. BLiMP offers a strong prior on which token is expected to drive the model's prediction, thereby supporting a targeted assessment of attribution faithfulness. Additionally, it enables evaluation of plausibility, as the evidence in this dataset inherently aligns with human reasoning. In summary, using BLiMP we are evaluating both the faithfulness and plausibility of these methods.

Following Mohebbi et al. (2023), we utilize a subset of the BLiMP dataset comprising five paradigms that represent three distinct linguistic phenomena. Examples of each phenomenon are

Phenomenon	UID	Example (Target ✔/Foil ✘)
Anaphor Number Agreement	ana	This government alarms itself ✔/themselves ✘.
Determiner-Noun Agreement	dna	Russell explored this ✔/these ✘ mall.
	dnaa	Patients scan this ✔/these ✘ orange brochure.
Subject-Verb Agreement	darn	The sister of doctors writes ✔/write ✘.
	rpsv	The pedestrian has ✔/have ✘ forgotten Grace.

Table 3: Examples of various linguistic phenomena that have been investigated in our experiments. Each paradigm is represented by a unique identifier (UID) from the BLiMP dataset. The target and foil words are denoted using check and cross marks. In each instance, the relevant evidence is underlined.

provided in Table 3. Using spaCy (Honnibal and Montani, 2017), we are able to identify the evidence of each linguistic phenomenon. For *anaphor number agreement*, we employ NeuralCoref[8] to detect the coreferent of the target word. To address *determiner-noun agreement*, we generate the dependency tree for each sample and extract the determiners corresponding to the target noun. Lastly, for *subject-verb agreement*, the same dependency tree can be used to identify the subjects associated with the verb.

Model. In this section, we employ the RoBERTa (Liu et al., 2019) model for our evaluations. We use both pre-trained[9] and fine-tuned versions of the model. For fine-tuning, the target token is replaced with `[MASK]`, and the model is optimized to select the correct target token (the grammatically appropriate word) over the foil token (a similar but grammatically incorrect alternative). Next, for inference, we use the head-on-top, also known as the unembedding matrix, to generate probabilities over the vocabulary.

Attribution Methods. GlobEnc, ALTI, and Value Zeroing are the attribution methods we consider for comparison in this experiment. Unlike Value Zeroing, which produces layer-wise attributions, the other two methods generate global importance scores. To acquire per-layer attributions for GlobEnc and ALTI, we bypass the rollout aggregation method to directly derive per-layer scores and we denote them as GlobEnc¬ and ALTI¬. We also consider a random baseline in which tokens are attributed equal scores.

Layer-wise Attribution with NormXLogit. Following the procedure described in Subsection 3.3, we treat the language modeling setup as a classification task, where the number of labels corresponds to the size of the vocabulary. In this setup, applying the head-on-top to the token representations yields a probability distribution over the vocabulary for each input token. For NormXLogit, to obtain the attributions of layer l, we apply LogAt to the output representations of the l-th layer, combined with the ℓ^2 norm of the input embeddings, where $l \in \{1, 2, \ldots, L\}$. Then, for each input token, $LogAt(target)$ is considered its attribution score with respect to the target token. Thanks to the per-label attributions obtained via LogAt, we can also identify the importance of evidence words in predicting both foil and target tokens. These attributions can be generated for any token in the vocabulary (cf. A.3).

5.2 Alignment Metrics

Following Yin and Neubig (2022), we define the known evidence (i.e., ground truth token-level rationale) as a binary vector $\boldsymbol{\mathcal{E}}$ with a length equal to the input sequence X. In this vector, a value of 1 at a given index indicates the presence of known evidence, while a value of 0 indicates its absence. Similarly, the explanation vector $\boldsymbol{\mathcal{S}}$ has the same length, with each element $\mathcal{S}_i$ representing the attribution score assigned to the i-th token for predicting the target token. To evaluate the alignment between evidence and explanation vectors, we take advantage of the Dot Product and Average Precision metrics (see Section A.4 for a worked example).

Dot Product: The dot product $\boldsymbol{\mathcal{E}} \cdot \boldsymbol{\mathcal{S}}$ measures the total score that the target attribution method assigns to the evidence tokens.

Average Precision: To evaluate whether an attribution method successfully identifies the most important tokens, we use Average Precision (AP). This metric concentrates on the ranking obtained via the attribution method rather than its raw scores. For a given sample, AP is computed as:

$$\text{AP} = \sum_{k=1}^{n} (R_k - R_{k-1}) P_k \tag{9}$$

where R_k and P_k indicate the recall and precision at threshold k, and n is the length of the input sequence.

5.3 Results

Figures 4 and 5 present the results of the alignment for different attribution methods and the known ev-

[8] https://github.com/huggingface/neuralcoref

[9] The results of the pre-trained model are covered in Section A.3.

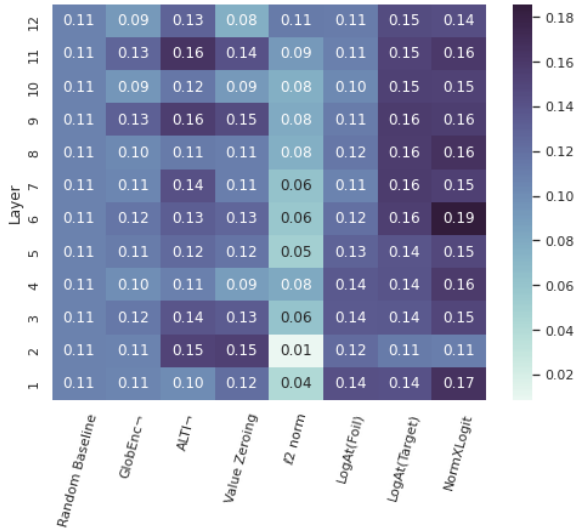

Figure 4: Per-layer alignment between evidence and explanation vectors in fine-tuned RoBERTa, calculated using Dot Product (higher values are better). Alignment for ℓ^2 norm of word embeddings (layer 0): 0.14.

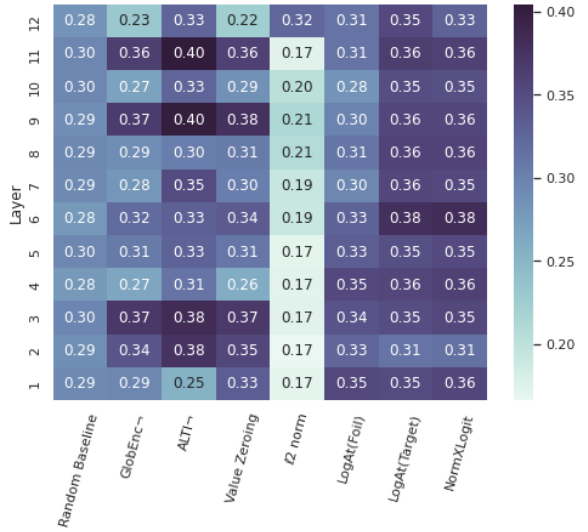

Figure 5: Per-layer alignment between evidence and explanation vectors in fine-tuned RoBERTa, calculated using Average Precision (higher values are better). Alignment for ℓ^2 norm of word embeddings (layer 0): 0.35.

idence enforcing a linguistic paradigm. In Figure 4, it can be seen that across almost all layers, NormXLogit consistently outperforms other methods in the experiment. The LogAt scores corresponding to the foil token in both alignment metrics are lower than those obtained from the target token. Specifically, as we progress to higher Transformer layers, there is a drop in alignment for the foil token and an increase for the target token. This pattern can be explained by the fact that token representations become more contextualized as they pass through layers. Increased context mixing from evidence words can lead to a correct prediction ($LogAt(Target)$), while reduced context mixing can result in incorrect predictions ($LogAt(Foil)$).

As noted earlier, the LogAt scores can be calculated for other tokens in the vocabulary as well. Our analysis shows that the LogAt score for the word 'plural' ($LogAt(\text{“}plural\text{”})$) outperforms all other methods in our experiments by a notable margin. This superior performance, unlike that of other random words, might be attributed to the *number agreement* phenomena underlying this experiment. At layer 7, the results of $LogAt(\text{“}plural\text{”})$ for Dot Product and Average Precision are 0.28 and 0.50, respectively (cf. A.3).

As mentioned in the caption of Figures 4 and 5, the high alignment between the norm of input word embeddings and the evidence confirms that indeed they are informative.

6 Conclusion and Future Work

In this paper, we introduced NormXLogit, an architecture-agnostic interpretation method that can be applied to any setup to reveal the opaque mechanism behind the decision-making process of LLMs. This method is fast and scalable, and it can be applied to models of any size. By utilizing the head-on-top, we gain the advantage of producing per-label explanations, which can be used to identify the most important tokens with respect to each label. Through extensive experiments, we showed that the attributions produced by NormXLogit are not only more faithful than those generated by gradient-based methods but also competitive with architecture-specific approaches.

Future work could explore the applicability of our proposed method to other domains and models, such as vision and non-Transformer architectures. Another promising direction is to investigate how aggregating attributions across all labels in a classification setup could lead to improved explanations.

Limitations

As with most attempts to interpret deep learning models, our evaluation, as well as those of existing methods, is inherently constrained by the lack of a definitive gold standard for understanding model internals. To mitigate this, we employed independent and complementary evaluation frameworks, including the AOPC metric, to measure the faithfulness

of our approach.

Our work primarily targets models with a transformer architecture, with evaluations focused on models applied to text. Nevertheless, we believe our method is broadly applicable to scenarios where inputs are represented numerically and the model generates token-wise representations.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4197, Online. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Ian Covert, Scott Lundberg, and Su-In Lee. 2022. Explaining by removing: A unified framework for model explanation. *Preprint*, arXiv:2011.14878.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *Preprint*, arXiv:2111.09543.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. 2016. Investigating the influence of noise and distractors on the interpretation of neural networks. *Preprint*, arXiv:1611.07270.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2021. Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4547–4568, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Preprint*, arXiv:1705.07874.

Faridoun Mehri, Mohsen Fayyaz, Mahdieh Soleymani Baghshah, and Mohammad Taher Pilehvar. 2024. Skipplus: Skip the first few layers to better explain vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 204–215.

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2023. DecompX: Explaining transformers decisions by propagating token decomposition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2649–2664, Toronto, Canada. Association for Computational Linguistics.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics.

Hosein Mohebbi, Willem Zuidema, Grzegorz Chrupała, and Afra Alishahi. 2023. Quantifying context mixing

in transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3378–3400, Dubrovnik, Croatia. Association for Computational Linguistics.

Momose Oyama, Sho Yokoi, and Hidetoshi Shimodaira. 2023. Norm of word embedding encodes information gain. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2108–2130, Singapore. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. *Preprint*, arXiv:1602.04938.

Sofia Serrano and Noah A. Smith. 2019. Is attention interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2931–2951, Florence, Italy. Association for Computational Linguistics.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. *Preprint*, arXiv:1312.6034.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Junlin Wang, Jens Tuyls, Eric Wallace, and Sameer Singh. 2020. Gradient-based analysis of NLP models is manipulable. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 247–258, Online. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: A benchmark of linguistic minimal pairs for English. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 409–410, New York, New York. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kayo Yin and Graham Neubig. 2022. Interpreting language models with contrastive explanations. *Preprint*, arXiv:2202.10419.

A Appendix

A.1 Computing Infrastructure

All experiments were conducted on a machine with an Nvidia Quadro RTX 8000 GPU with 48GB of memory. The operating system used is Ubuntu 22.04.3 LTS.

A.2 Experiment 1: Complete Results

Figures 6 to 46 illustrate the Comprehensiveness and Sufficiency across different models and datasets. In Figure 6, we demonstrate the global performance of Value Zeroing on the SST-2 dataset. The results show that this method is not faithful to the model's decision-making process. This issue may stem from the inherent limitations of the rollout aggregation method, as previously discussed. Additionally, since Value Zeroing is a perturbation-based method, it may also inherit some of the challenges associated with these approaches. For instance, this method zeros out each token's value vector one at a time, which can lead to problems like ignoring dependencies between features. Consider the following example:

> *"The movie is mediocre, maybe even bad."*

In this case, erasing "*mediocre*" or "*bad*" independently may not significantly impact the overall sentiment of the sentence.

For our Integrated Gradients experiments, we generally used 50 steps. However, for LLAMA2, we reduced the number of steps to 25 due to resource constraints.

Computational Efficiency. We conducted experiments to compare the computational efficiency of the attribution methods, using the BERT model with a maximum input length of 512 tokens. Timing results, averaged over five runs and reported in seconds, reveal that the runtime for Gradient Norm and Gradient×Input remains nearly constant across different input lengths, while Integrated Gradients and DecompX show noticeable increases as input length grows. NormXLogit consistently demonstrates the fastest processing times, with minimal sensitivity to input length. In terms of memory efficiency, we identified the maximum batch size each method could handle within 48GB of GPU memory. NormXLogit significantly outperforms others, processing up to 750 samples per batch, whereas Integrated Gradients and DecompX are limited to very small batch sizes (2 and 1, respectively). Consequently, NormXLogit offers both superior speed and memory efficiency, enabling more scalable attribution computations.

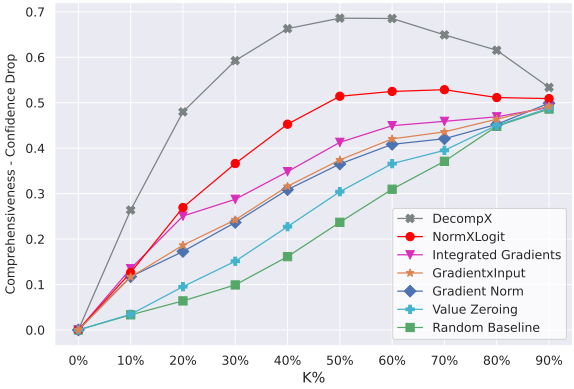

Figure 6: Comprehensiveness Confidence Drop of different attribution methods for BERT fine-tuned on SST-2 (higher values are better).

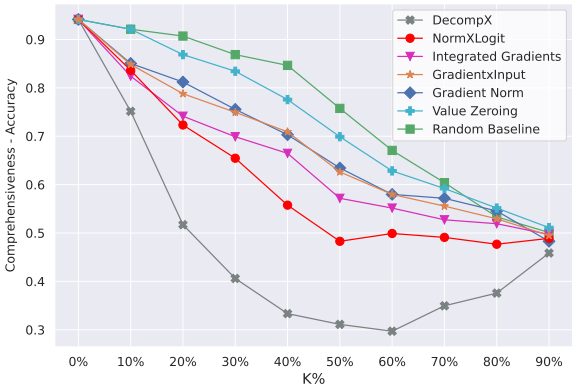

Figure 7: Comprehensiveness Accuracy of different attribution methods for BERT fine-tuned on SST-2 (lower values are better).

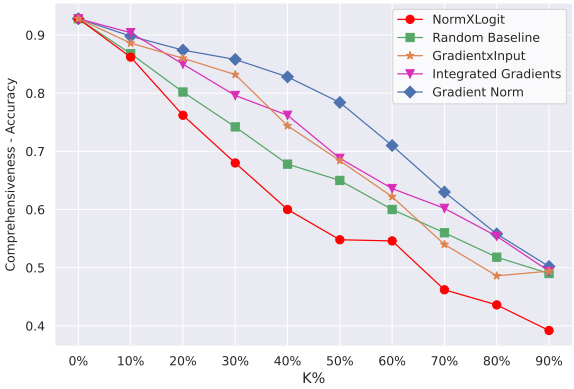

Figure 8: Comprehensiveness Accuracy of different attribution methods for LLAMA 2 fine-tuned on SST-2 (lower values are better).

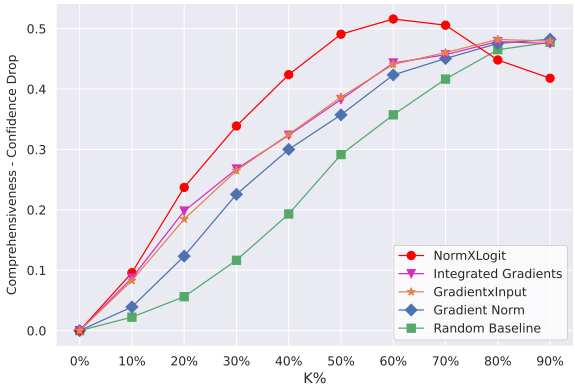

Figure 9: Comprehensiveness Confidence Drop of different attribution methods for DeBERTa fine-tuned on SST-2 (higher values are better).

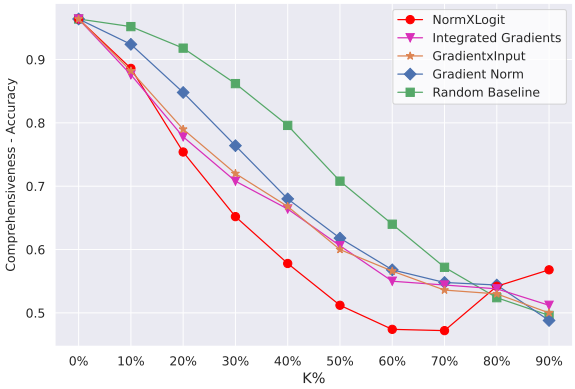

Figure 10: Comprehensiveness Accuracy of different attribution methods for DeBERTa fine-tuned on SST-2 (lower values are better).

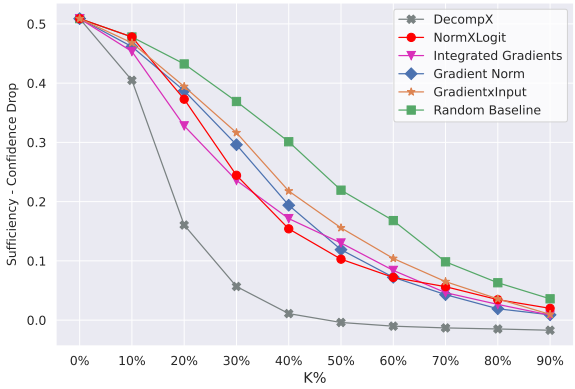

Figure 11: Sufficiency Confidence Drop of different attribution methods for BERT fine-tuned on SST-2 (lower values are better).

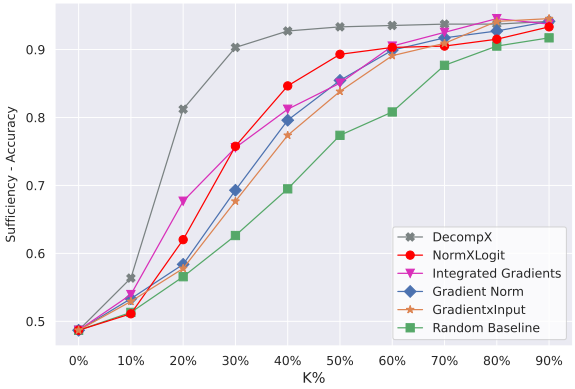

Figure 12: Sufficiency Accuracy of different attribution methods for BERT fine-tuned on SST-2 (higher values are better).

A.3 Experiment 2: RoBERTa Complete Results

The evidence alignment experiment is conducted on a masked language modeling task to understand why a particular target token is chosen. The LogAt method provides per-label attribution scores, enabling us to apply it to other labels (i.e., tokens in the vocabulary) to identify the most important tokens in the input sequence for predicting each specific label. Figures 48 and 49 display the results of the Dot Product and Average Precision alignment metrics for the pre-trained RoBERTa model. An important observation is the notable performance of $LogAt(\text{"}plural\text{"})$, which demonstrates its effectiveness in identifying evidence words. This level of performance is not seen with two other randomly chosen words. Specifically, the results are more pronounced in the top layers, indicating that increased context mixing enhances the connection between the evidence and the word “plural." In other words, as we progress through the layers, the contextualized representation of the evidence word becomes increasingly similar to the word “plural," resulting in a higher attribution for this word. We attribute the superior performance for the word “plural" primarily to the nature of the phenomena used from the BLiMP dataset, which focused on *number agreement*. Figures 50 and 51 demonstrate the results for the fine-tuned RoBERTa model.

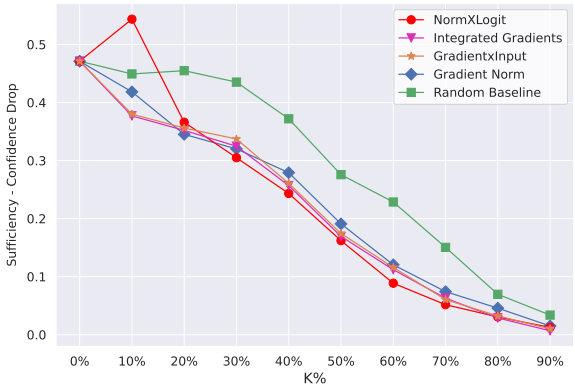

Figure 13: Sufficiency Confidence Drop of different attribution methods for DeBERTa fine-tuned on SST-2 (lower values are better).

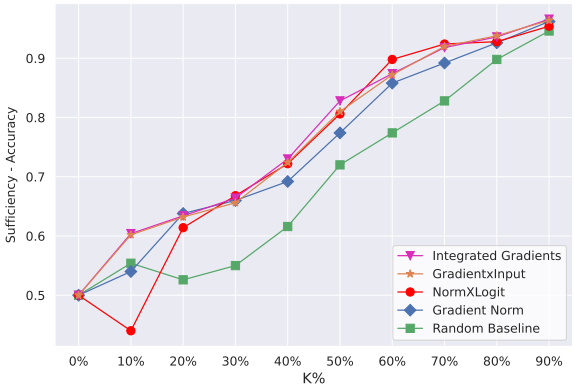

Figure 14: Sufficiency Accuracy of different attribution methods for DeBERTa fine-tuned on SST-2 (higher values are better).

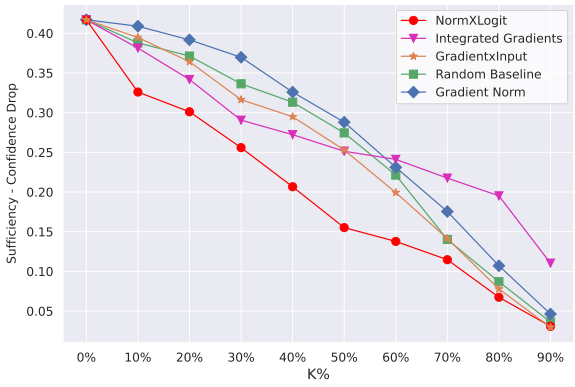

Figure 15: Sufficiency Confidence Drop of different attribution methods for LLAMA 2 fine-tuned on SST-2 (lower values are better).

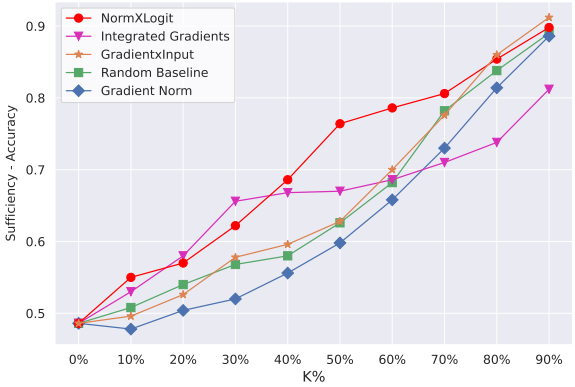

Figure 16: Sufficiency Accuracy of different attribution methods for LLAMA 2 fine-tuned on SST-2 (higher values are better).

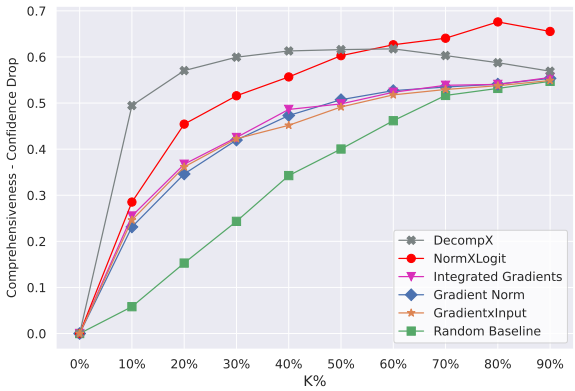

Figure 17: Comprehensiveness Confidence Drop of different attribution methods for BERT fine-tuned on MultiNLI (higher values are better).

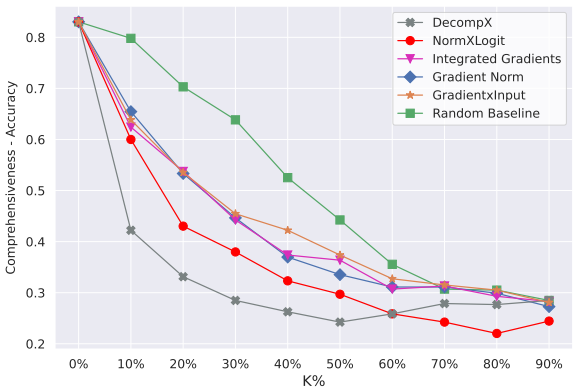

Figure 18: Comprehensiveness Accuracy of different attribution methods for BERT fine-tuned on MultiNLI (lower values are better).

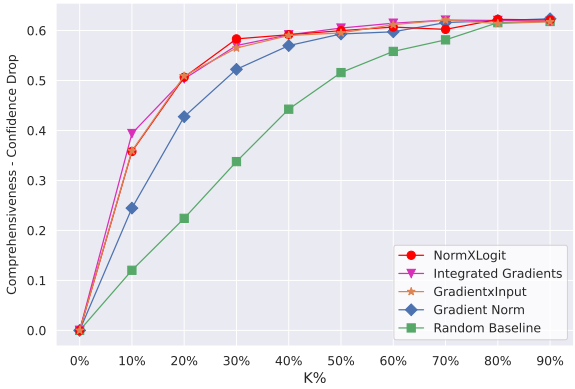

Figure 19: Comprehensiveness Confidence Drop of different attribution methods for DeBERTa fine-tuned on MultiNLI (higher values are better).

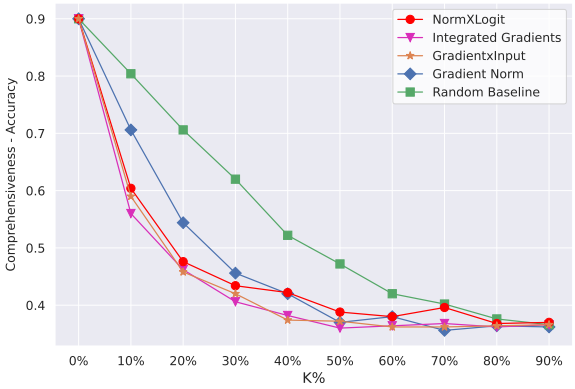

Figure 20: Comprehensiveness Accuracy of different attribution methods for DeBERTa fine-tuned on MultiNLI (lower values are better).

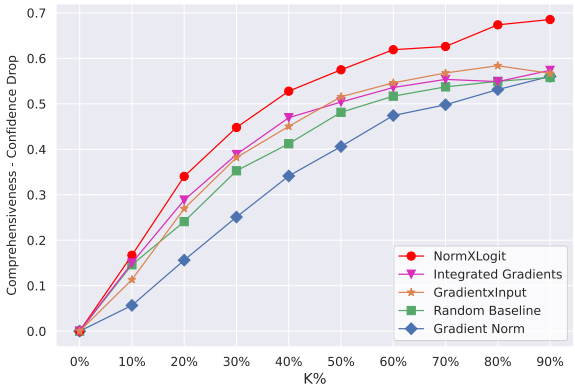

Figure 21: Comprehensiveness Confidence Drop of different attribution methods for LLAMA 2 fine-tuned on MultiNLI (higher values are better).

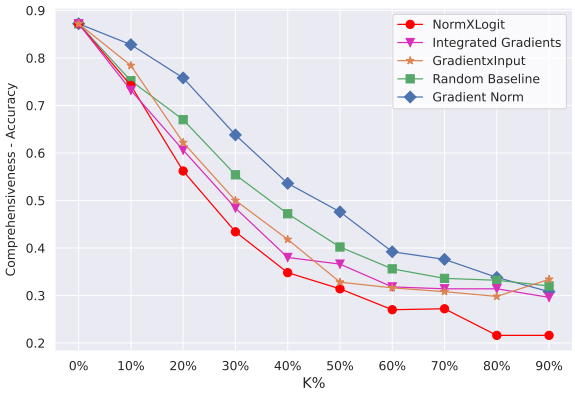

Figure 22: Comprehensiveness Accuracy of different attribution methods for LLAMA 2 fine-tuned on MultiNLI (lower values are better).

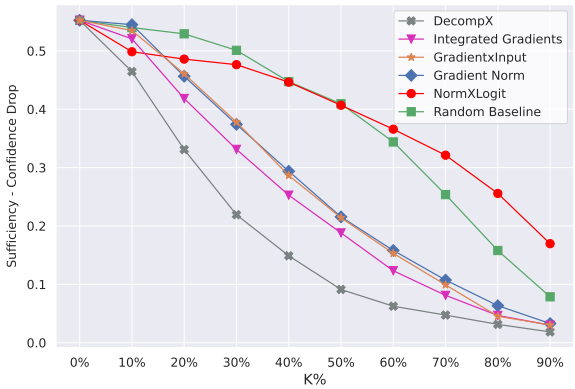

Figure 23: Sufficiency Confidence Drop of different attribution methods for BERT fine-tuned on MultiNLI (lower values are better).

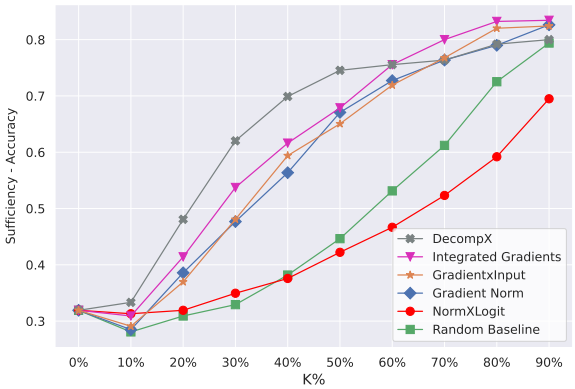

Figure 24: Sufficiency Accuracy of different attribution methods for BERT fine-tuned on MultiNLI (higher values are better).

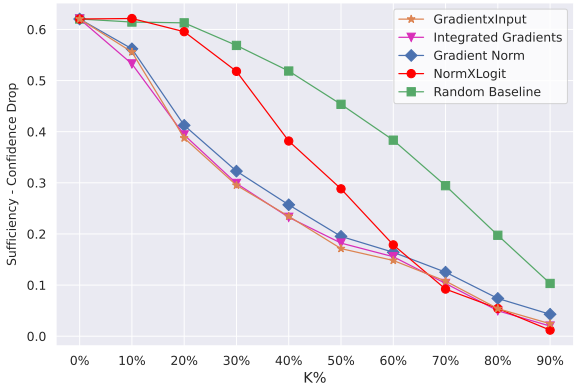

Figure 25: Sufficiency Confidence Drop of different attribution methods for DeBERTa fine-tuned on MultiNLI (lower values are better).

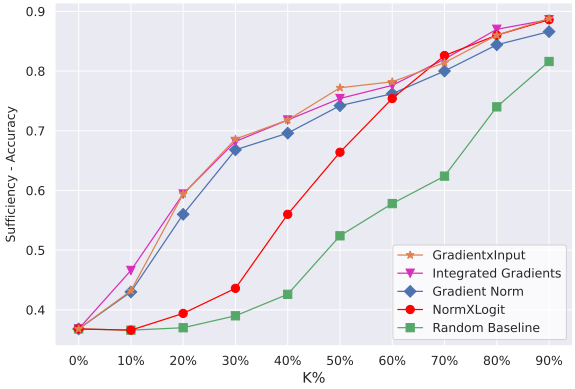

Figure 26: Sufficiency Accuracy of different attribution methods for DeBERTa fine-tuned on MultiNLI (higher values are better).

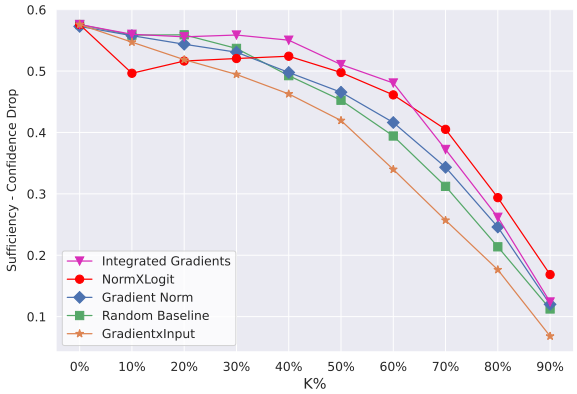

Figure 27: Sufficiency Confidence Drop of different attribution methods for LLAMA 2 fine-tuned on MultiNLI (lower values are better).

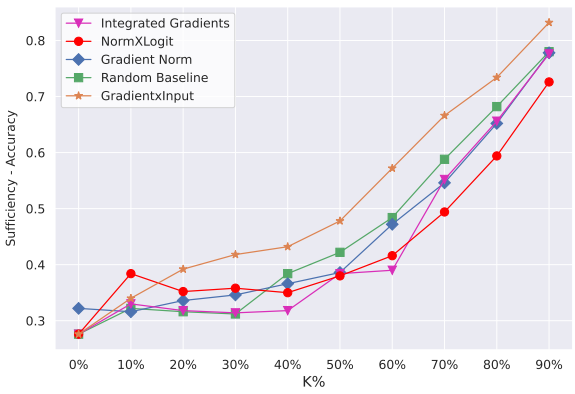

Figure 28: Sufficiency Accuracy of different attribution methods for LLAMA 2 fine-tuned on MultiNLI (higher values are better).

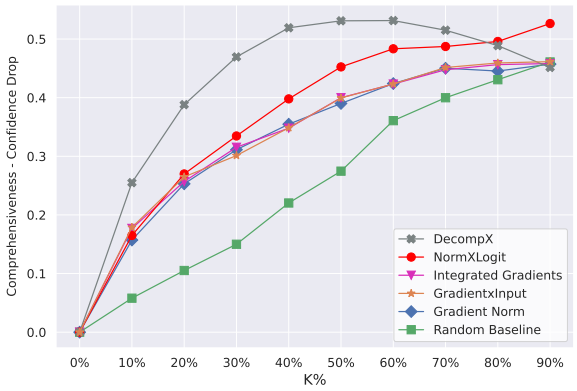

Figure 29: Comprehensiveness Confidence Drop of different attribution methods for BERT fine-tuned on QNLI (higher values are better).

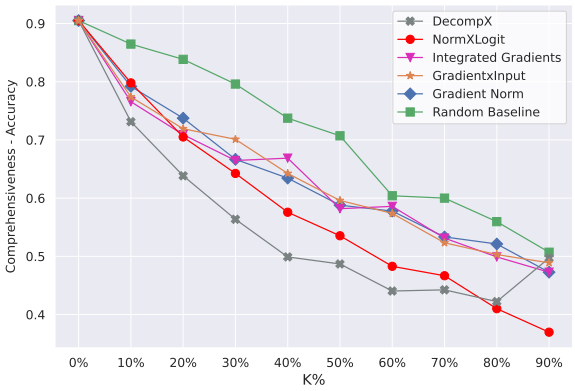

Figure 30: Comprehensiveness Accuracy of different attribution methods for BERT fine-tuned on QNLI (lower values are better).

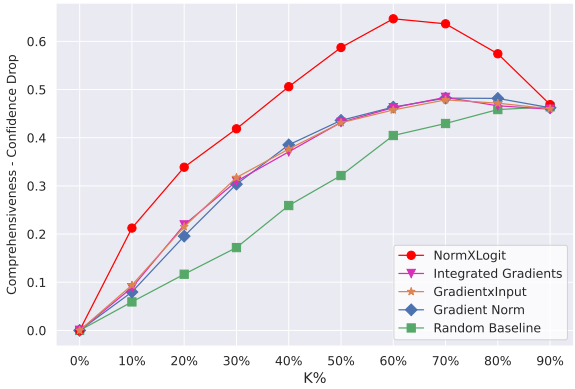

Figure 31: Comprehensiveness Confidence Drop of different attribution methods for DeBERTa fine-tuned on QNLI (higher values are better).

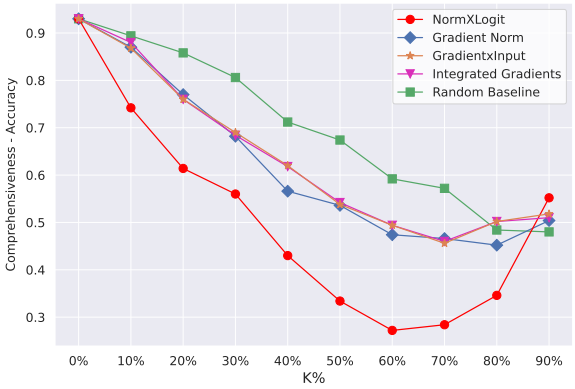

Figure 32: Comprehensiveness Accuracy of different attribution methods for DeBERTa fine-tuned on QNLI (lower values are better).

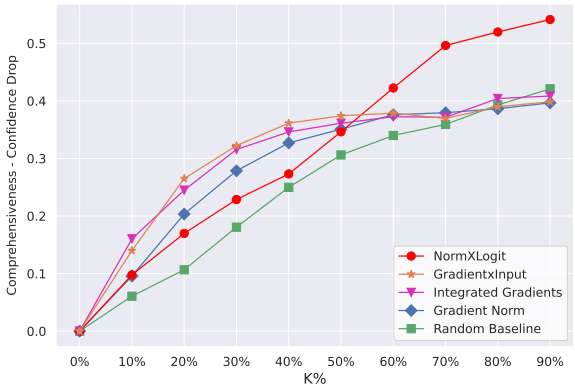

Figure 33: Comprehensiveness Confidence Drop of different attribution methods for LLAMA 2 fine-tuned on QNLI (higher values are better).

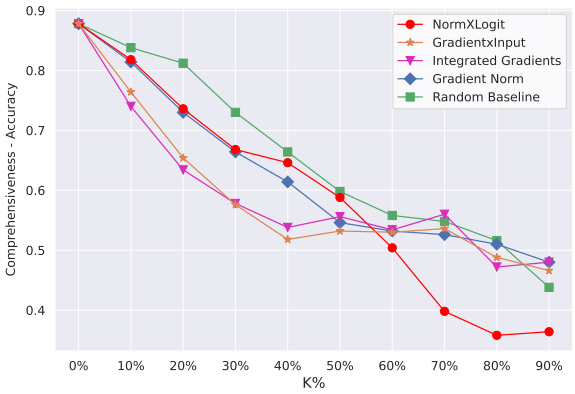

Figure 34: Comprehensiveness Accuracy of different attribution methods for LLAMA 2 fine-tuned on QNLI (lower values are better).

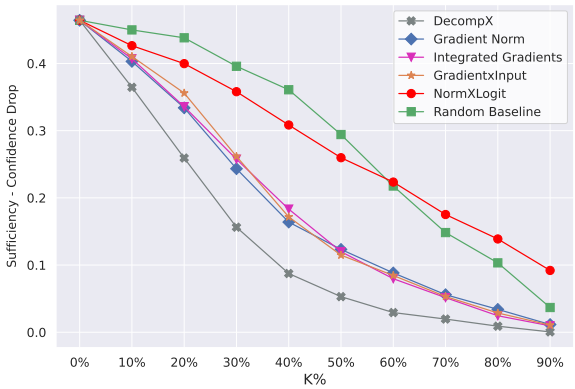

Figure 35: Sufficiency Confidence Drop of different attribution methods for BERT fine-tuned on QNLI (lower values are better).

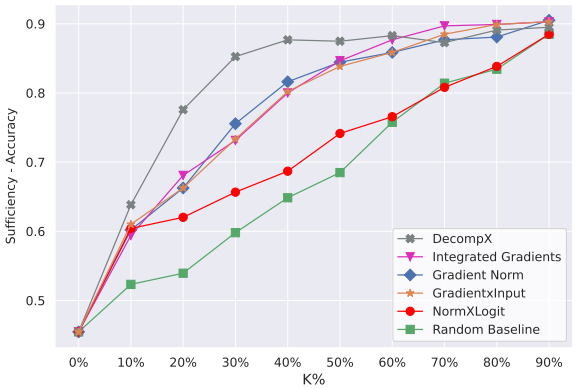

Figure 36: Sufficiency Accuracy of different attribution methods for BERT fine-tuned on QNLI (higher values are better).

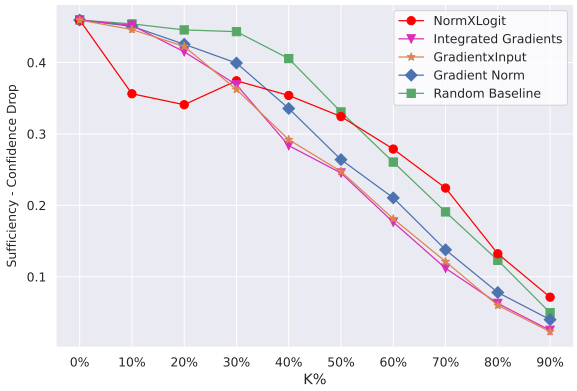

Figure 37: Sufficiency Confidence Drop of different attribution methods for DeBERTa fine-tuned on QNLI (lower values are better).

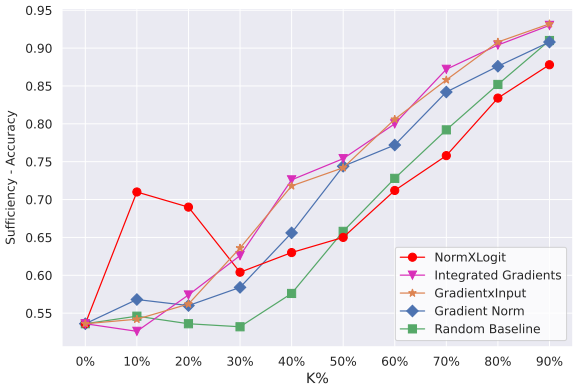

Figure 38: Sufficiency Accuracy of different attribution methods for DeBERTa fine-tuned on QNLI (higher values are better).

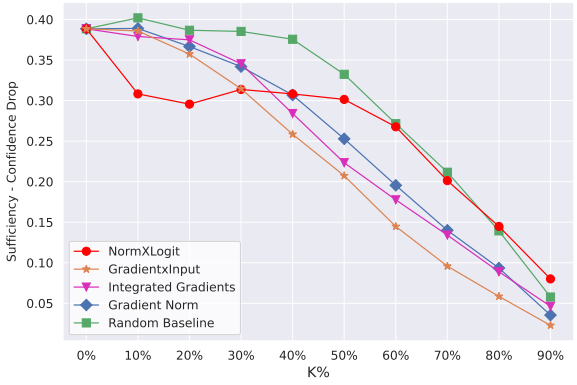

Figure 39: Sufficiency Confidence Drop of different attribution methods for LLAMA 2 fine-tuned on QNLI (lower values are better).

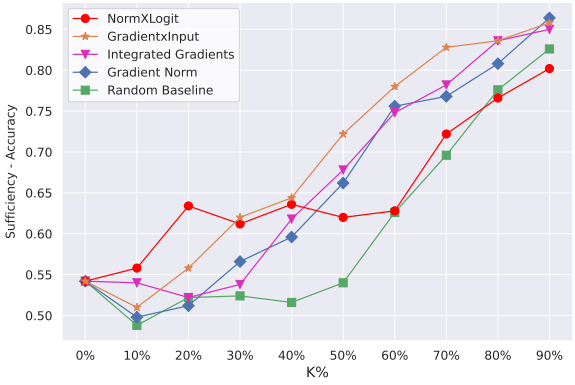

Figure 40: Sufficiency Accuracy of different attribution methods for LLAMA 2 fine-tuned on QNLI (higher values are better).

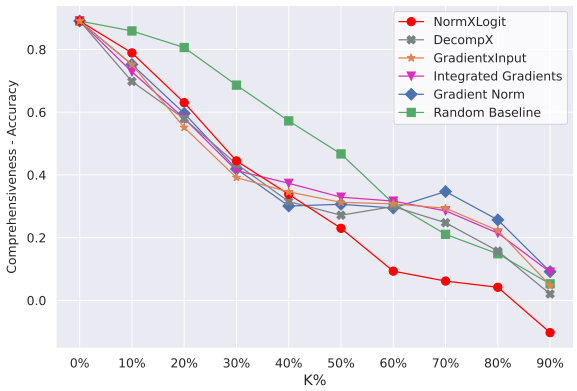

Figure 41: Comprehensiveness Accuracy of different attribution methods for BERT fine-tuned on STS-B (lower values are better).

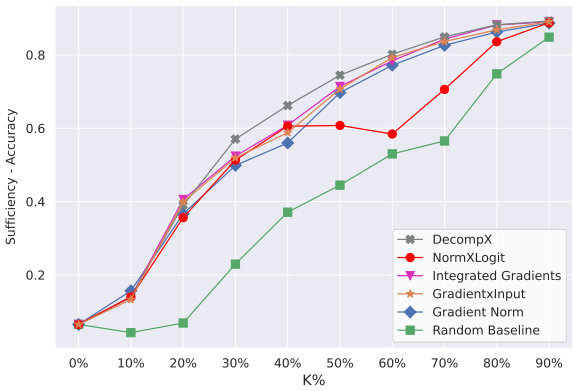

Figure 42: Sufficiency Accuracy of different attribution methods for BERT fine-tuned on STS-B (higher values are better).

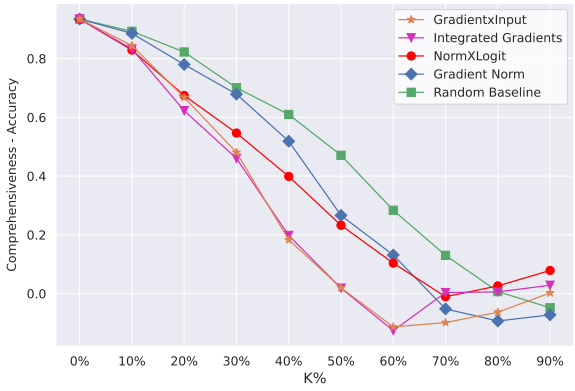

Figure 43: Comprehensiveness Accuracy of different attribution methods for DeBERTa fine-tuned on STS-B (lower values are better).

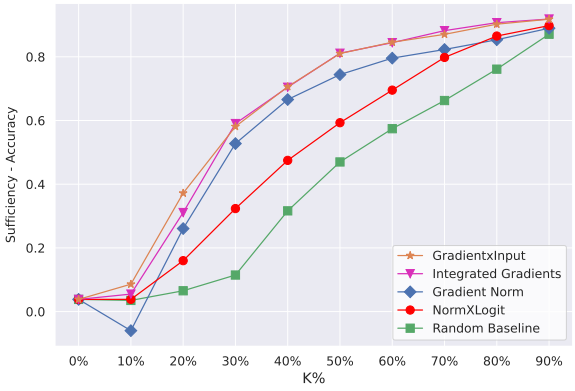

Figure 44: Sufficiency Accuracy of different attribution methods for DeBERTa fine-tuned on STS-B (higher values are better).

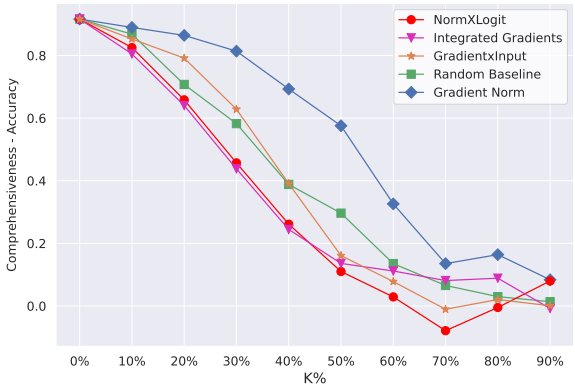

Figure 45: Comprehensiveness Accuracy of different attribution methods for LLAMA 2 fine-tuned on STS-B (lower values are better).

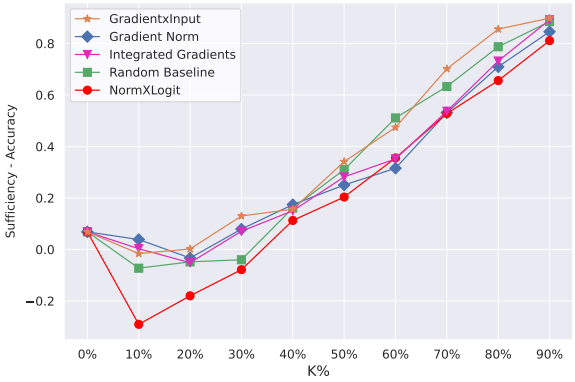

Figure 46: Sufficiency Accuracy of different attribution methods for LLAMA 2 fine-tuned on STS-B (higher values are better).

	SST-2 ($\text{Comp}_{\text{Acc}}\downarrow$)			**MNLI** ($\text{Comp}_{\text{Acc}}\downarrow$)			**QNLI** ($\text{Comp}_{\text{Acc}}\downarrow$)		
	LLAMA 2	**DeBERTa**	**BERT**	**LLAMA 2**	**DeBERTa**	**BERT**	**LLAMA 2**	**DeBERTa**	**BERT**
Random Baseline	0.652	0.719	0.734	0.466	0.521	0.484	0.634	0.675	0.690
Gradient Norm	0.738	0.665	0.659	0.517	0.440	0.393	0.602	0.591	0.614
Gradient×Input	0.683	0.644	0.653	0.434	0.408	0.406	0.563	0.605	0.613
Integrated Gradients	0.698	0.642	0.622	0.423	**0.403**	0.393	0.566	0.606	0.609
DecompX	N/A	N/A	**0.422**	N/A	N/A	**0.294**	N/A	N/A	**0.525**
ℓ^2 norm	0.595	0.624	0.670	0.497	0.502	0.477	0.651	0.645	0.665
LogAt	0.599	0.610	0.636	**0.367**	0.440	0.324	**0.556**	0.510	0.563
NormXLogit	**0.590**	**0.604**	0.579	0.375	0.426	0.333	0.564	**0.459**	0.554

Table 4: Comprehensiveness of NormXLogit against other methods across various model and dataset configurations. Each value is computed by averaging across all perturbation ratios (lower Accuracy is better). Best values are in **bold**, and second-best values are underlined.

	SST-2 ($\text{Suff}_{\text{CD}}\downarrow$)			MNLI ($\text{Suff}_{\text{CD}}\downarrow$)			QNLI ($\text{Suff}_{\text{CD}}\downarrow$)			STS-B ($\text{Suff}_{\text{Acc}}\uparrow$)		
	LLAMA 2	DeBERTa	BERT	LLAMA 2	DeBERTa	BERT	LLAMA 2	DeBERTa	BERT	LLAMA 2	DeBERTa	BERT
Random	0.241	0.274	0.241	0.404	0.416	0.362	0.285	0.300	0.272	0.347	0.430	0.428
Grad. Norm	0.260	0.201	0.178	0.413	0.240	0.250	0.236	0.260	0.162	0.323	0.611	0.625
G×I	0.230	0.192	0.196	0.365	0.220	0.245	**0.205**	0.240	0.166	0.394	0.677	0.637
IG	0.256	**0.188**	0.165	0.442	**0.219**	0.221	0.228	**0.238**	0.163	0.330	0.669	0.643
DecompX	N/A	N/A	**0.064**	N/A	N/A	**0.157**	N/A	N/A	**0.109**	N/A	N/A	**0.659**
ℓ^2 norm	0.210	0.201	0.230	**0.315**	0.361	0.389	0.241	0.351	0.262	**0.658**	**0.714**	0.599
LogAt	**0.176**	0.193	0.189	0.423	0.351	0.373	0.237	0.316	0.268	0.123	0.490	0.495
NormXLogit	0.177	0.200	0.171	0.432	0.305	0.381	0.247	0.273	0.265	0.235	0.538	0.582

Table 5: Sufficiency of NormXLogit against other methods across various model and dataset configurations. Each value is computed by averaging across all perturbation ratios (lower Confidence Drop and higher Accuracy are better). Best values are in **bold**, and second-best values are underlined.

	SST-2 ($\text{Suff}_{\text{Acc}}\uparrow$)			MNLI ($\text{Suff}_{\text{Acc}}\uparrow$)			QNLI ($\text{Suff}_{\text{Acc}}\uparrow$)		
	LLAMA 2	DeBERTa	BERT	LLAMA 2	DeBERTa	BERT	LLAMA 2	DeBERTa	BERT
Random Baseline	0.668	0.712	0.742	0.477	0.537	0.490	0.613	0.681	0.698
Gradient Norm	0.638	0.771	0.794	0.466	0.708	0.610	0.670	0.723	0.800
Gradient×Input	0.675	0.791	0.787	0.540	0.727	0.613	**0.706**	0.745	0.799
Integrated Gradients	0.672	**0.795**	0.816	0.449	**0.730**	0.642	0.679	**0.746**	0.803
DecompX	N/A	N/A	**0.877**	N/A	N/A	**0.666**	N/A	N/A	**0.840**
ℓ^2 norm	0.702	0.787	0.763	**0.589**	0.597	0.451	0.658	0.560	0.727
LogAt	**0.730**	0.781	0.786	0.460	0.578	0.459	0.666	0.671	0.724
NormXLogit	0.726	0.773	0.809	0.450	0.638	0.451	0.664	0.718	0.734

Table 6: Sufficiency of NormXLogit against other methods across various model and dataset configurations. Each value is computed by averaging across all perturbation ratios (higher Accuracy is better). Best values are in **bold**, and second-best values are underlined.

	ATTRIBUTION METHOD				
Evaluation Criteria	**Gradient Norm**	**Gradient×Input**	**Integrated Gradients**	**DecompX**	**NormXLogit**
Maximum Batch Size	100	100	2	1	**750**
Average Time per Instance (s)	$0.0076_{\pm 0.00}$	$0.0081_{\pm 0.00}$	$1.4104_{\pm 0.01}$	$2.3625_{\pm 0.02}$	$\mathbf{0.0005}_{\pm 0.00}$

Table 7: Efficiency of attribution methods when maximizing batch size under a 48GB memory constraint (input length = 512). NormXLogit supports significantly larger batch sizes and achieves the lowest per-instance time, demonstrating superior scalability and memory efficiency. The values in subscript represent the standard deviation.

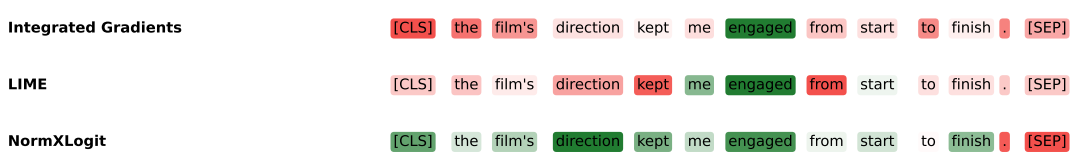

Figure 47: Qualitative comparison of token attribution methods for the sentiment analysis task. NormXLogit is compared with Integrated Gradients and LIME.

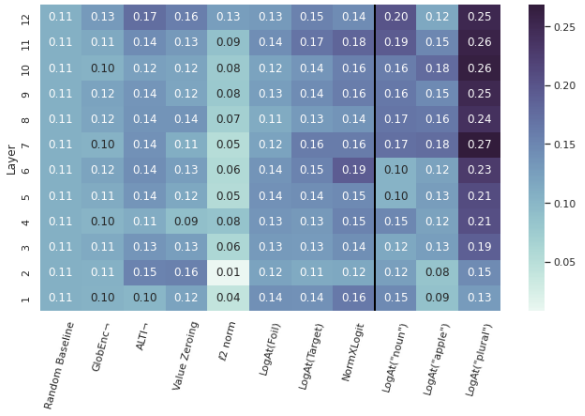

Figure 48: Per-layer alignment between evidence and explanation vectors for the pre-trained version of RoBERTa, calculated using Dot Product metric (higher values are better). The alignment for ℓ^2 norm of word embeddings (layer 0) is 0.14.

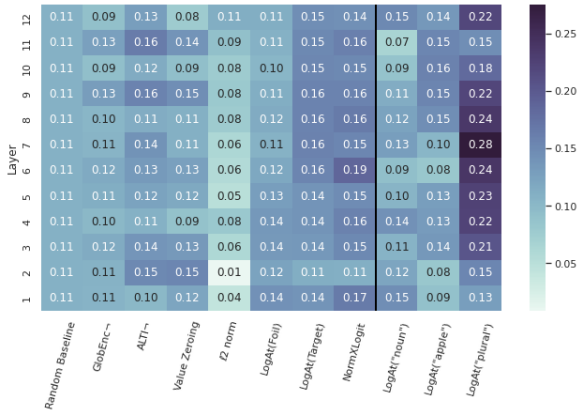

Figure 50: Per-layer alignment between evidence and explanation vectors for the fine-tuned version of RoBERTa, calculated using Dot Product metric (higher values are better). The alignment for ℓ^2 norm of word embeddings (layer 0) is 0.14.

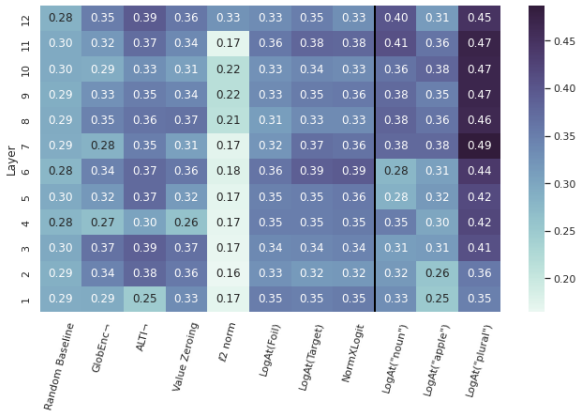

Figure 49: Per-layer alignment between evidence and explanation vectors for the pre-trained version of RoBERTa, calculated using Average Precision metric (higher values are better). The alignment for ℓ^2 norm of word embeddings (layer 0) is 0.35.

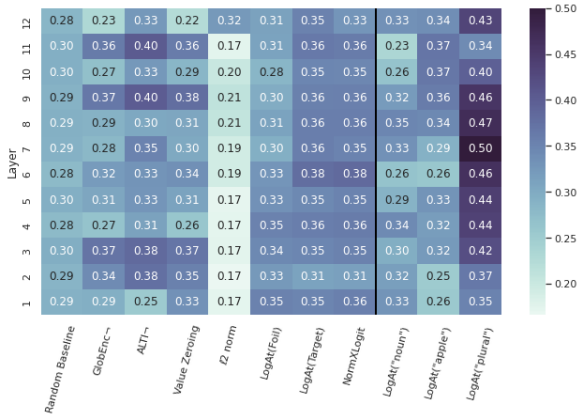

Figure 51: Per-layer alignment between evidence and explanation vectors for the fine-tuned version of RoBERTa, calculated using Average Precision metric (higher values are better). The alignment for ℓ^2 norm of word embeddings (layer 0) is 0.35.

A.4 Experiment 2: Worked Example for Alignment Metric Computation

To illustrate how the alignment metrics work, we provide an example using the following sentence:

"Karla thinks/think about it"

In this example, **"Karla"** serves as the evidence that determines the correct verb (*"thinks"*).

Vector Definitions. We first define a binary evidence vector $\boldsymbol{\mathcal{E}}$ to indicate which tokens are considered known evidence (1 for evidence, 0 otherwise). For this example, the input sequence is tokenized as:

[Karla, thinks, about, it]

Since only "Karla" is the evidence, we define:

$$\boldsymbol{\mathcal{E}} = [1, 0, 0, 0]$$

Next, we define the attribution vector $\boldsymbol{\mathcal{S}}$, where each element $\mathcal{S}_i$ corresponds to the score assigned by the attribution method to token i:

$$\boldsymbol{\mathcal{S}} = [0.3, 0.1, 0.5, 0.1]$$

Dot Product: We compute the dot product between the two vectors:

$$\begin{aligned}\boldsymbol{\mathcal{E}} \cdot \boldsymbol{\mathcal{S}} &= 1 \cdot 0.3 + 0 \cdot 0.1 + 0 \cdot 0.5 + 0 \cdot 0.1 \\ &= 0.3\end{aligned}$$

This score reflects the total attribution the method assigns to the known evidence ("Karla"). A higher dot product indicates better alignment with known evidence.

Average Precision (AP): This metric evaluates how well the attribution method ranks the evidence token(s). We start by sorting the attribution scores in descending order and recording their indices:

$$\text{Rank}(\boldsymbol{\mathcal{S}}) = [2, 0, 1, 3]$$

This means the token at index 2 ("about") has the highest score, followed by index 0 ("Karla"), and so on. The known evidence index (based on $\boldsymbol{\mathcal{E}}$) is:

$$\text{Evidence Index} = \{0\}$$

To compute AP, we iterate through the ranked list and monitor when the evidence index is included. Precision is calculated at each step where recall increases (i.e., when a new evidence token is found):

1. **[2]** — does not include index 0 $\rightarrow$ no change in recall
2. **[2, 0]** — includes index 0 $\rightarrow$ recall changes $\rightarrow$ precision = 1/2 = 0.5
3. **[2, 0, 1]** — no change in recall
4. **[2, 0, 1, 3]** — no change in recall

So, the final AP is:

$$\text{AP} = 0.5$$

Note that if the evidence index were 2 (evidence ranked first), the AP would be 1.0. If it were 1 (ranked third), the AP would be approximately 0.33. A higher AP indicates better alignment between the attribution method's ranking and the true evidence.