Local Interpretation of Transformer Based on Linear Decomposition

Sen Yang, Shujian Huang[*], Wei Zou, Jianbing Zhang, Xinyu Dai, Jiajun Chen
National Key Laboratory for Novel Software Technology, Nanjing University
{yangsen,zouw}@smail.nju.edu.cn
{huangsj,zjb,daixinyu,chenjj}@nju.edu.cn

Abstract

In recent years, deep neural networks (DNNs) have achieved state-of-the-art performance on a wide range of tasks. However, limitations in interpretability have hindered their applications in the real world. This work proposes to interpret neural networks by linear decomposition and finds that the ReLU-activated Transformer can be considered as a linear model on a single input. We further leverage the linearity of the model and propose a linear decomposition of the model output to generate local explanations. Our evaluation of sentiment classification and machine translation shows that our method achieves competitive performance in efficiency and fidelity of explanation. In addition, we demonstrate the potential of our approach in applications with examples of error analysis on multiple tasks.[1]

1 Introduction

Deep neural networks (DNNs) such as Transformers (Vaswani et al., 2017) have achieved state-of-the-art results on various natural language tasks (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020; Dai et al., 2019) via learning complex nonlinear relationships of inputs.

However, the lack of interpretability of the predictions given by black-box models limits their application in real-world (Guidotti et al., 2018; Lipton, 2018; Ribeiro et al., 2016b).

A typical way to understand model prediction dynamics is to generate prediction explanations for each input, called local explanation generation (Chen et al., 2020). Most existing works on local explanation algorithms in NLP strive to understand such dynamics on word-level or phrase-level by assigning importance scores on input features (Ribeiro et al., 2016a; Lei et al., 2016; Lundberg et al., 2018; Plumb et al., 2018). However, nonlinearity in models makes it difficult to assign the contribution of individual words or phrases to predictions, while the linear counterparts are more interpretable as the weight of each component could be naturally interpreted as its contribution.

In this work, we present a linear decomposition theory to interpret linear models, which can be generalized to nonlinear DNNs. That is, we formalize the decomposition of the linear outputs into components corresponding to the input features, then mathematically propose the properties of linear decomposition and the uniqueness of the decomposition under these properties.

Furthermore, we prove that the ReLU-activated Transformer can be regarded as a linear function by given input features if the causal relationship between the input and certain intermediate variables is disregarded. Therefore, generalize the proposed linear decomposition to Transformer under such an assumption.

However, this decomposition yields a component corresponding to the parameters of additive bias (usually used in the linear layer), which contains a partial contribution of the inputs. Thus we separate and reallocate this part of the contribution to the input features while preserving the mathematical properties of the decomposition.

Quantitative experiments were conducted on sentiment classification and machine translation to identify important input features. We show that our local explanation algorithms efficiently outperform several competitive baselines. Additionally, we propose further implementation of our algorithm to explain the model errors on natural language tasks. The fidelity of our algorithm exceeds that of other baselines.

Our key contributions are summarized as follows:

- We prove the linearity of the ReLU-activated Transformer for a given input under reasonable assumptions.

[*] Corresponding author.

[1] We release our algorithm toolkit at `https://github.com/DoubleVII/pydec`.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 10270–10287
July 9-14, 2023 ©2023 Association for Computational Linguistics

- We design algorithms for the linear decomposition of Transformer hidden states and propose methods for reallocating the contribution of additive bias while maintaining the mathematical properties.
- Experimental results and case studies on sentiment classification and machine translation validate the fidelity and interpretability of the proposed methods.

2 Method

In this section, we propose the decomposition theory of linear functions. Then, we generalize it to nonlinear cases (i.e., Transformer) and present several decomposition methods accordingly. Finally, we analyze the mathematical properties of the different methods.

2.1 Linear Decomposition Theory

Decomposing the output of a linear system according to its input is relatively simple. The results of the decomposition are intuitively interpreted as the contributions of the inputs to the outputs. We present a theory of linear decomposition, including the definition of decomposition, linear decomposability, and the properties of interpretable decomposition.

Given a set $X = \{x_1, \cdots, x_m\}$ and a function f, the output is denoted as $h = f(X)$.

Definition 1. (*linearly decomposable*). The output h of the function f is linearly decomposable for input X if and only if h can be represented as a linear combination of X:

$$h = f(X) = \sum_{i}^{m} W_i^X x_i, \tag{1}$$

where $x_i \in R^{n(x_i)}$ denotes the i-th input vector, $W_i^X \in R^{n(h)\times n(x)}$ is the linear transformation matrix with respect to x_i, and the input X is defined as the *basis* of the decomposition. Here we use $n(\cdot)$ to denote the dimension of $\cdot$.

For linearly decomposable h, it is intuitive to regard $W_i^X x_i$ in Eq. (1) as the contribution of x_i. Sometimes input features are divided into different groups, and we are more interested in the overall impact of each group (e.g., tokens split from the same word can be divided into a group to produce word-level explanations). Specifically, a *group* is an element of a set P, where P is an arbitrary **partition** of the basis X.

Definition 2. (*decomposition*). A decomposition of h under partition P is the splitting of h into components corresponding to all groups in P, i.e.,

$$h = \sum_{g\in P} \left.\frac{\mathscr{D}h}{\mathscr{D}g}\right|_P,$$

where $\left.\frac{\mathscr{D}h}{\mathscr{D}g}\right|_P$ denotes the component corresponding to group g under partition P in the decomposition of h. In this paper, the partition P is omitted if there is no ambiguity.

Considering the given partition $P_1 = \{\{x_1, x_2\}, \{x_3, \cdots, x_m\}\}$ as an example, the decomposition of h under P_1 is denoted as

$$h = \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_1, x_2\}}\right|_{P_1} + \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_3, \cdots, x_m\}}\right|_{P_1}.$$

Since there are exponential decompositions for a function, each with unclear interpretability, we examine the following properties:

Property 1. *Orthogonality.*

$$\frac{\mathscr{D}x_i}{\mathscr{D}g} = \begin{cases} x_i, & \textit{if } x_i \in g \\ \mathbf{0}, & \textit{otherwise} \end{cases}.$$

Property 2. *Linearity.*

$$\frac{\mathscr{D}h_1}{\mathscr{D}g} + \frac{\mathscr{D}h_2}{\mathscr{D}g} = \frac{\mathscr{D}(h_1 + h_2)}{\mathscr{D}g},$$

$$W\frac{\mathscr{D}h}{\mathscr{D}g} = \frac{\mathscr{D}(Wh)}{\mathscr{D}g}.$$

Property 3. *Group Additivity.*

$$\frac{\mathscr{D}h}{\mathscr{D}g_1} + \frac{\mathscr{D}h}{\mathscr{D}g_2} = \frac{\mathscr{D}h}{\mathscr{D}g_1 \cup g_2}.$$

Definition 3. (*interpretable decomposition*). A decomposition $\mathscr{D}$ is interpretable if it satisfies *Orthogonality* and *Linearity*.

The interpretable decomposition specifies the necessary conditions that guarantee interpretability under linear operations. The *Group Additivity* is related to the consistency of a decomposition.

Definition 4. (*consistency*). A decomposition $\mathscr{D}$ is consistent if $\frac{\mathscr{D}h}{\mathscr{D}g}$ are equal for the same group g in any partition of the basis.

For example, given the partition $P_1 = \{\{x_1\}, \cdots, \{x_m\}\}$, the decomposition of h can be formulated as

$$h = \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_1\}}\right|_{P_1} + \cdots + \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_m\}}\right|_{P_1}, \tag{2}$$

and given another partition $P_2 = \{\{x_1\}, \{x_2, \cdots, x_m\}\}$, we have

$$h = \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_1\}}\right|_{P_2} + \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_2, \cdots, x_m\}}\right|_{P_2}. \quad (3)$$

If $\mathscr{D}$ is consistent, then $\left.\frac{\mathscr{D}h}{\mathscr{D}\{x_1\}}\right|_{P_1} = \left.\frac{\mathscr{D}h}{\mathscr{D}\{x_1\}}\right|_{P_2}$ holds.

Consistency guarantees the consistent contribution of a given group by arbitrary partitions from the perspective of interpretability. To determine a consistent decomposition, we propose the following lemma (proved in Appendix A):

Lemma 1. *A decomposition $\mathscr{D}$ is consistent if and only if it satisfies the Group Additivity.*

To the best of our knowledge, most of the current local explanation algorithms (Singh et al., 2019; Chen et al., 2020; Li et al., 2016; Sundararajan et al., 2017) are interpretable. Furthermore, for linearly decomposable h, these algorithms are essentially equivalent to the following decomposition:

Definition 5. (*decomposition $\bar{\mathscr{D}}$*). $\bar{\mathscr{D}}$ is defined on linearly decomposable h, where each component $\left.\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}g}\right|_P$ is the sum of terms corresponding to the given group $g \in P$, and each term comes from the linear combination of X about h, i.e.,

$$\left.\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}g}\right|_P := \sum_{x_i \in g} W_i^X x_i.$$

$\bar{\mathscr{D}}$ is intuitive, and more importantly, the unique interpretable decomposition for any linearly decomposable h (proved in Appendix B).

Obviously, $\bar{\mathscr{D}}$ satisfies *Group Additivity* thus is consistent. As forementioned, most existing methods are equivalent and consistent under linear conditions. However, they may lose consistency with nonlinear functions. This aspires us to transform nonlinear functions into locally linear ones for consistency guarantees of the interpretable decomposition, and extend the interpretable decomposition onto nonlinear activation functions while maintaining consistency.

2.2 ReLU-activated Transformer Is a Linear Function of The Input

A typical Transformer (Vaswani et al., 2017) is composed of a stack of identical layers. Each layer of the encoder consists of two major components: a multi-head self-attention mechanism and a feed-forward neural network. Besides, a residual connection (He et al., 2016) is employed around each of the two components, followed by layer normalization (Ba et al., 2016). The decoder is in a similar fashion to the encoder, but with additional attention to draw relevant information from the hidden states generated by the encoders.

Complicated as it may be, a Transformer can be seen as a combination of the above modules. Thus, if each module is linearly decomposable, the final result will be linearly decomposable. To achieve this, we disregard the input's contribution to the intermediate variables of attention scores and standard deviation of layer normalization. Consequently, these intermediate variables can be considered as coefficients of the linear transformation in the formula, analogous to the parameters of linear layers in the model. Though we partially ignore some of the influence propagations, the remainings retain the major causalities of the model, which are sufficient to provide adequate explanations. We verified this assumption by comparing the performance before and after cutting off the gradient of the attention scores and standard deviations (Section 3.2). To make life easier, this paper assumes the model uses ReLU as the activation function. We discuss the extensibility of our approach to other activation functions in Section 7.

Based on the above elaboration, we give the following lemma, which provides the condition to apply linear decomposition on Transformer.

Lemma 2. *For a given input $X = \{x_1, x_2, \cdots, x_m\}$, any hidden state h in Transformer can be represented as:*

$$h = \sum_i^m W_i^X x_i + \sum_l^L W_l^B b_l, \quad (4)$$

where x_i denotes the i-th input vector, b_l denotes the parameter of additive bias in the model[2]*.*

Proof. Proof by mathematical induction.

Base Case. For any input x_i, we have $x_i = x_i$, which is consistent with Eq. (4), i.e.

$$W_j^X = \begin{cases} I, & \text{if } j = i \\ \mathbf{0}, & \text{otherwise} \end{cases}, W_l^B = \mathbf{0}.$$

Induction step. Assume Eq. (4) holds for all input hidden states of a Transformer sub-layer, it holds for the output of the sub-layer, too. We prove each of the sub-layer types below respectively.

[2] For ease of expression, all the parameters of additive bias in the model are numbered from 1 to L.

For *Linear Layer*, we have

$$
\begin{aligned}
h' &= W'h + b_k \\
&= \sum_i^m W'W_i^X x_i + \sum_{l \neq k}^L W'W_l^B b_l + (I + W'W_k^B) b_k.
\end{aligned}
$$

For *Attention Layer*, since each attention score a_i is considered as a coefficient of the linear transformation, then we have

$$
\begin{aligned}
h' &= a_1 h_1 + \cdots + a_m h_m \\
&= \sum_j^m a_j \left[\sum_i^m W_{ij}^X x_i + \sum_l^L W_{lj}^B b_l \right] \\
&= \sum_i^m \left[\sum_j^m a_j W_{ij}^X \right] x_i + \sum_l^L \left[\sum_j^m a_j W_{lj}^B \right] b_l.
\end{aligned} \tag{5}
$$

For *Residual Connection*, we have

$$
\begin{aligned}
h' &= h_1 + h_2 \\
&= \sum_i^m \left[W_{i1}^X + W_{i2}^X \right] x_i + \sum_l^L \left[W_{l1}^B + W_{l2}^B \right] b_l.
\end{aligned}
$$

As for *Layer Normalization*, we rewrite a linear transformation

$$
h' = \mathrm{LN}(h) = s(h - W'h), \tag{6}
$$

where the scalar $s = 1/\sqrt{Var(h)}$ is the coefficient and W' is the averaging operator, i.e.

$$
W' = [1/n(h)]_{n(h) \times n(h)}.
$$

The *Activation Function* ReLU can be rewritten as a linear transformation $h' = \mathrm{relu}(h) = W'h$, where

$$
\begin{aligned}
W' &= \mathrm{diag}(d_1, \cdots, d_{n(h)}) \\
d_i &= \begin{cases} 1, & \text{if } h[i] \geq 0 \\ 0, & \text{otherwise} \end{cases}.
\end{aligned} \tag{7}
$$

□

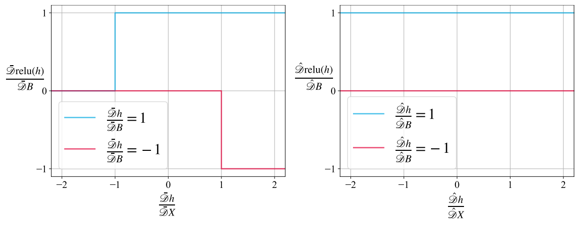
With Lemma 2, we raise the core theorem of this paper.

Theorem 1. *For a given input, any hidden state h in Transformer is linearly decomposable on the basis $X' = \{x_1, \cdots, x_m, b_1, \cdots, b_L\}$.*

In other words, we can obtain the decomposition $\bar{\mathscr{D}}$ of h as

$$
h = \sum_{g \in P} \frac{\bar{\mathscr{D}} h}{\bar{\mathscr{D}} g} + \frac{\bar{\mathscr{D}} h}{\bar{\mathscr{D}} B}, \tag{8}
$$

where $B = \{b^1, \cdots, b^L\}$ and we still use P to denote the partition of the input X instead of the basis X'. The partition of the basis X' can be recovered as $P' = P \cup \{B\}$ if $b_1, \cdots, b_L$ are considered as a single group.



(a) (b)

Figure 1: The curves of the bias component of the ReLU output given each component of the ReLU input h, where $h = \frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}X} + \frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}B}$. For $\bar{\mathscr{D}}$ (a), $\frac{\bar{\mathscr{D}}\mathrm{relu}(h)}{\bar{\mathscr{D}}B}$ is governed by both $\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}B}$ and $\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}X}$, while for $\hat{\mathscr{D}}$ (b), $\frac{\hat{\mathscr{D}}\mathrm{relu}(h)}{\hat{\mathscr{D}}B}$ is only governed by $\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}$ and is independent of $\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}X}$.

2.3 Decomposing The Contribution of Additive Bias

Eq. (8) shows that the parameters of additive bias in the model contribute partially to h, by $\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}B}$. This is reasonable because the term $\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}B}$ represents a prior guess made by the model in the absence of inputs (e.g., even in the absence of inputs, a language model may predict ‘The’ as the beginning of a sentence with a certain probability). However, the term $\frac{\bar{\mathscr{D}}h}{\bar{\mathscr{D}}B}$ is also mixed with the contribution from inputs, since the bias component of the ReLU output may change due to the components of the input (Figure 1 (a)). To address this issue, we define a new decomposition $\hat{\mathscr{D}}$ and require the bias component of the ReLU output to be independent of the input components (Figure 1 (b)), i.e.,

$$
\frac{\hat{\mathscr{D}}\mathrm{relu}(h)}{\hat{\mathscr{D}}B} := \mathrm{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}). \tag{9}
$$

The remaining parts are to be assigned to each group of the input, which is

$$
\begin{aligned}
&\mathrm{relu}(h) - \mathrm{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}) \\
=&W'h - \mathrm{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}) \\
=&W' \left[\sum_{g \in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} + \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B} \right] - \mathrm{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}) \\
=&\sum_{g \in P} W' \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} + \left[W' \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B} - \mathrm{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}) \right],
\end{aligned} \tag{10}
$$

where W' comes from Eq. (7).

The first term of Eq. (10) is easily assigned to each group, and the second term implies the contribution separated from the original bias term, which is split in the assignment:

$$\frac{\hat{\mathscr{D}}\text{relu}(h)}{\hat{\mathscr{D}}g} := W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} + \alpha_g \left[W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B} - \text{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}) \right], \quad (11)$$

where $\sum_{g \in P} \alpha_g = 1$.

We designed two methods to calculate α.

Absolute-value-based:

$$\alpha_g = \frac{|\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}|}{\sum_{g \in P} |\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}|}. \quad (12)$$

Signed-value-based:

$$\alpha_g = \frac{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}}{\sum_{g \in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}}. \quad (13)$$

For linear functions, we introduce *Orthogonality* and *Linearity* into $\hat{\mathscr{D}}$ to make it interpretable:

$$\frac{\hat{\mathscr{D}}x_i}{\hat{\mathscr{D}}g} := \begin{cases} x_i, & \text{if } x_i \in g \\ \mathbf{0}, & \text{otherwise} \end{cases}, \quad (14)$$

$$\frac{\hat{\mathscr{D}}(h_1 + h_2)}{\hat{\mathscr{D}}g} := \frac{\hat{\mathscr{D}}h_1}{\hat{\mathscr{D}}g} + \frac{\hat{\mathscr{D}}h_2}{\hat{\mathscr{D}}g}, \quad (15)$$

$$\frac{\hat{\mathscr{D}}(Wh)}{\hat{\mathscr{D}}g} := W\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}. \quad (16)$$

Finally, we notice that the α in Eq. (13) explodes as the denominator gets close to 0, degrading the algorithm's performance. As a comparison, α in Eq. (12) is more stable when constrained by the probability simplex. To alleviate the stability issue, we switch to the absolute-value-based method in the unstable region of Eq. (13). The instability is measured by

$$r = \frac{\left|\sum_{g \in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}\right|}{\sum_{g \in P} \left|\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}\right|}, \quad (17)$$

where r indicates more stability when ascending from 0 to 1. In our experiments, we adopt a hyperparameter λ to interpolate different α schemes: absolute-value-based when $r < \lambda$ and signed-value-based when $r \geq \lambda$. When λ goes from 0 to 1, the decomposition $\hat{\mathscr{D}}$ will change from signed-value-based algorithm to absolute-value-based algorithm with more inconsistency.

2.4 Comparison

Our algorithms exhibit different properties under the two α schemes in Eq. (11), which lead to different final results. The signed-value-based $\hat{\mathscr{D}}$ satisfies *Group Additivity*, while the absolute-value-based approach does not satisfy it (Appendix C). More importantly, it can be proved that the signed-value-based α calculation is the only solution that satisfies *Group Additivity* (Appendix D), and the absolute-value-based approach aims at the numerical stability issue. By Lemma 1, we conclude that $\hat{\mathscr{D}}$ based on Signed-value is consistent, while the one based on Absolute-value is inconsistent.

3 Experiments

We evaluate our algorithms with SOTA Transformer implementations on text classification (RoBERTa, Liu et al., 2019) and machine translation (Vaswani et al., 2017). It is notable that the classification follows the encoder-only architecture, while the translation follows the encode-decode architecture.

3.1 Experiment Settings

Datasets. We use the SST-2 (Socher et al., 2013) and the IMDB (Maas et al., 2011) datasets for sentiment analysis, which is modeled as a binary classification. The SST-2 includes 6920/872/1821 instances in the train/dev/test sets. The IMDB includes 25000/25000 instances in the train/test sets. We adopt WMT14 English-to-German (En⇒De) for machine translation, with 4.5M parallel sentences consisting of 118M English and 111M German words for training. We use newstest 2013 for validation and newstest 2014 as the test set.

We evaluate the explanation on test sets of all datasets, except for the IMDB, where we test on a subset with 2000 randomly selected samples from test data due to computation expenses.

Models. We adopt the Transformer (Vaswani et al., 2017) base model with baseline settings for machine translation. We adopt the fine-tuned RoBERTa base model (Liu et al., 2019) for text classification. RoBERTa utilizes GELU (Hendrycks and Gimpel, 2016) as its activation function. To apply our decomposition, we replaced it with ReLU during fine-tuning. The impact on performance and other implementation details are explained in Appendix E.

Appendix F shows the best performance of the models on all datasets in our experiments.

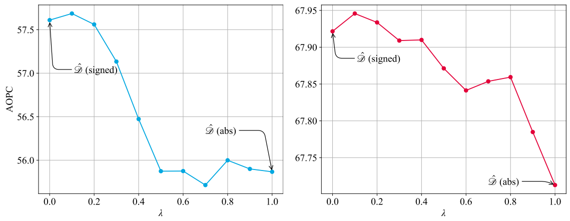
Methods	SST-2		IMDB		WMT14 En⇒De	
	AOPC↑	LAT./s ↓	AOPC↑	LAT./s ↓	AOPC↑	LAT./s ↓
Random	5.69	0.03	3.33	0.02	30.39	0.61
ACD (Singh et al., 2019)	8.87	2.30	failed	-	35.85	126.80
HEDGE (Chen et al., 2020)	44.25	0.30	65.14	2.88	43.62	21.79
LRP (Voita et al., 2021)	22.75	3.28	failed	-	59.92	122.29
GlobEnc (Modarressi et al., 2022)†	20.09	0.29	19.75	1.60	N/A	-
LIME (Ribeiro et al., 2016b)	37.39	0.53	19.09	3.57	**68.66**	9.90
LOO (Li et al., 2016)	53.29	0.38	59.67	3.09	**68.83**	21.23
IG (Sundararajan et al., 2017)	43.60	1.04	30.56	58.11	68.23	108.46
+ linearizing Attn & LN	45.58	1.00	46.08	46.95	67.92	74.72
Decomposition $\bar{\mathscr{D}}$	48.94	**0.06**	81.63	**0.82**	66.98	**1.31**
Decomposition $\hat{\mathscr{D}}$	**57.69**	**0.06**	**87.11**	1.96	67.95	**1.34**

† Not applicable to the encoder-decoder architecture.

Table 1: AOPCs and average latency of different methods on the SST-2, IMDB and WMT En-De datasets.

ID	Variables with retained gradients	Variables with cut off gradients	SST-2	IMDB	WMT14
1	a_i and s	h_i and $h - W'h$	5.73×10^{-4}	1.46×10^{-4}	2.26
2	h_i and $h - W'h$	a_i and s	10.35	1.55	15.05

Table 2: Averaged gradient norms passed to input via different intermediate variables. The intermediate variables are from Eq. (5) (attention layer) and Eq. (6) (layer normalization), which denote the attention scores and values, mean-subtracted hidden states and their standard deviation, respectively. The gradients for other layers are intact.



(a)

(b)

Figure 2: AOPC curves given different λ on the SST (a) and WMT (b) datasets.

Evaluations. We adopt *the area over the perturbation curve* (AOPC, Chen et al., 2020; Nguyen, 2018; Samek et al., 2016) to evaluate token-level explanations, which measures local fidelity by comparing the probability change on the predicted label after deleting $k\%$ top-scored tokens assigned by explanation algorithms. We set $k = 20$ for sentiment analysis. For machine translation, the number of deleted tokens is fixed at 4. This is because a complete generation consists of multiple token predictions, while each generated target-side token depends on only a few input tokens rather than the entire input sequence. In addition, we average the AOPC scores for the decoding process of the machine translation model.

In this paper, we generate contribution scores by decomposing the logits of the model. Specifically, for a classification of n classes, the model generates a n-dimensional vector of logits $h^o \in R^n$ for a prediction $\hat{y} = \arg\max_i h^o[i]$. Thus, the importance score of feature x_i can be expressed as $\frac{\mathscr{D}h^o}{\mathscr{D}\{x_i\}}[\hat{y}]$.

3.2 Main Results

We compare our algorithms with the following baselines: Leave-One-Out (LOO, Li et al., 2016), LIME (Ribeiro et al., 2016b), GlobEnc (Modarressi et al., 2022), Integrated Gradient (IG, Sundararajan et al., 2017), Agglomerative Contextual Decomposition (ACD, Singh et al., 2019), Layer-wise Relevance Propagation (LRP, Voita et al., 2021), and HEDGE (Chen et al., 2020). We also report the AOPC as a reference when *random* scores are assigned to tokens. For our algorithms, we adopt $\bar{\mathscr{D}}$ and $\hat{\mathscr{D}}$ in the evaluation, and fix the hyperparameter λ of $\hat{\mathscr{D}}$ at 0.1.

As shown in Table 1, the improved decomposition $\hat{\mathscr{D}}$ outperforms our base decomposition $\bar{\mathscr{D}}$ and other baselines in the quality of explanations over the SST-2 dataset, especially the IMDB dataset. Our decomposition $\hat{\mathscr{D}}$ achieves comparable performance to IG on the WMT En-De dataset. IG performs well on the translation but poorly on the sentiment classification with excessive computational complexity. We suspect that this is because the loss scale of the sentiment classification is significantly smaller than that of the translation, weakening the salience of the gradient. Occlusion-based methods, such as LOO and LIME, achieve relatively good performance on the WMT dataset because they are very similar to the evaluation metrics when k is small. Furthermore, on the IMDB dataset, LOO and LIME become weaker as the sequence becomes longer due to the diminished impact of a single token deletion in a sentence. The ACD fails the IMDB due to accumulated precision error, while the LRP suffers from exponential overhead.

Nevertheless, IG comprehensively considers the influence of each variable, including attention weights and standard deviations of layer normalization. We additionally consider linearizing the attention layer and layer normalization by cutting off the gradients of attention weights and standard deviations for comparison, where we justify our hypothesis of ignoring their influence propagations by looking into its impact on performance. Surprisingly, this hypothesis even gains improvements on the SST-2 and IMDB datasets. To further validate our hypothesis, we investigated the contribution of inputs to outputs through different intermediate variables by examining the norms of the gradients propagated to inputs from different variables. As shown in Table 2, when the gradients of attention scores and standard deviation are retained in the sentiment classification task, the gradient norms reflected on the input are negligible. In the translation tasks, the weight is larger but still much smaller than that of group 2, which the decoder may introduce. Finally, it's notable that the connection between our method and these experiments lies in the fact that the gradient produced by group 2 is equal to the transition matrix of each input in the decomposition $\bar{\mathscr{D}}$ (i.e., W^X in Eq. (4)). Therefore, our decomposition indeed captures the major causalities of the model.

Overall, the results show that our approach is applicable and efficient in classification and end-to-end generation. We provide additional results of AOPCs by different k in Appendix G, including an extra natural language understanding task from GLUE (Wang et al., 2018).

3.3 Ablation Study

We investigate the impact of different λ on the SST-2 and WMT14 datasets, which controls the interpolation of $\hat{\mathscr{D}}$ (signed) and $\hat{\mathscr{D}}$ (abs).

We achieve the best AOPC score with λ near 0.1 (Figure 2). Compared with the absolute-value-based decomposition ($\lambda = 1$). The AOPC scores of the pure signed-value-based decomposition ($\lambda = 0$) differ slightly from the best results. As the λ increases, the AOPC scores on both datasets decrease, demonstrating that improved consistency leads to better interpretability.

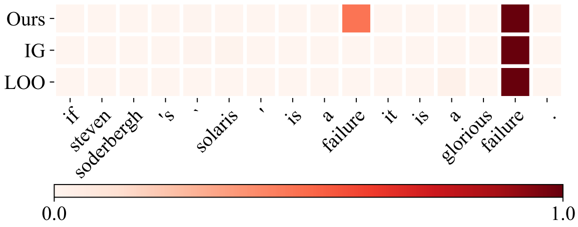


Figure 3: The contribution heatmaps generated by our algorithm, IG (Integrated Gradient) and LOO (Leave-One-Out). All contribution scores are normalized within $[0, 1]$.

4 Applications

Our method can be applied to various scenarios by designing different partitions. In this section, we analyze the causes of model errors in sentiment classification and translation at the instance level. We set up our algorithm with $\hat{\mathscr{D}}$ (signed) for strict consistency. We also compare the results of IG (Integrated Gradient) and LOO (Leave-One-Out).

4.1 Errors in Sentiment Classification

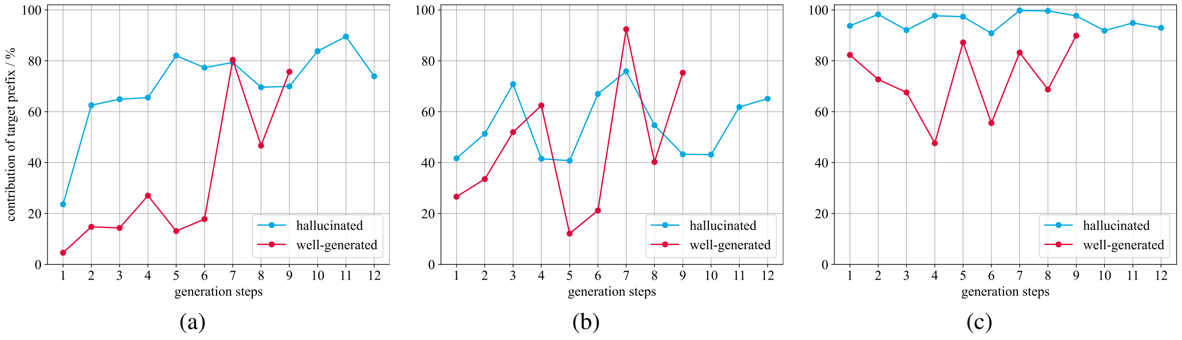
We find that over half of the errors of the SST-2 test occur when the sentiment expressed at the sentence level is opposite to the polarity of the sentiment words in the input. For example, the sentence "*if steven soderbergh's 'solaris' is a failure it is a glorious failure.*" is a positive comment, but the model's prediction is negative.

Figure 3 shows the contribution heatmap generated by our algorithm and the baseline algorithms, where tokens belonging to the same word are divided into the same group for word-level explanations[3]. The results of the analysis show that the model focuses on both "*failure*" and fails classification, indicating the model's insufficient understanding of the overall sentence meaning. It is notable that our method not only considers the last "*failure*" as the main basis of the model decision but the first "*failure*" as well. This is more intuitive since the model's prediction only inverts as soon as both "*failure*" are masked. For comparison, the

[3] For other baseline algorithms, we sum the token-level scores within the group to obtain the group-level scores, despite of inconsistency.

Source	**Prediction**
This hotel is bad.	Das$_1$ Hotel$_2$ ist$_3$ $\underline{\text{sehr}_4\ \text{zentral}_5\ \text{gelegen}_6\ ,_7\ \text{aber}_8\ \text{trotzdem}_9\ \text{ruhig}_{10}\ ._{11}}$ $\langle$EOS$\rangle_{12}$ [The hotel is very centrally located , but still quiet.]
Many of my customers are very young.	Viele$_1$ meiner$_2$ Kunden$_3$ sind$_4$ sehr$_5$ j@@$_6$ ung$_7$.$_8$ $\langle$EOS$\rangle_9$ [Many of my customers are very young .]

Table 3: Examples of hallucinated and well-generated samples. The sequence is generated in the order according to the number marked at each token, with an English translation in brackets. The hallucination is underlined.



(a)

(b)

(c)

Figure 4: The contribution of target prefix (%) generated by our algorithm (a), IG (b), and LOO (c).

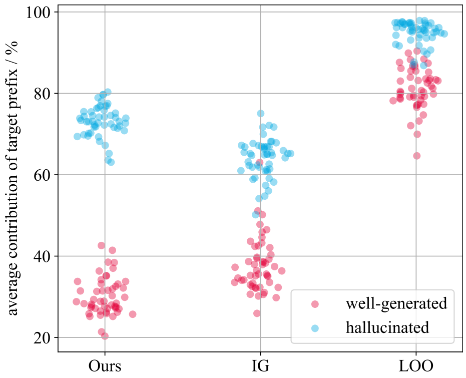


Figure 5: The contribution of target prefix (%) averaged over the generation steps. We sample 100 samples (50 for each class) from the training data and vertically distribute them for presentation.

other two baselines fail to indicate the impact of the first "*failure*".

4.2 Errors in Translation

We noticed that, despite fluency in the target language, machine translation produces hallucinated outputs (Müller et al., 2020) that are semantically decoupled from the source sequence (Table 3).

We divide inputs into two groups to inspect their contributions to outputs: the source and the target prefix. Figure 4 shows the percentage of the contribution by target prefix at each generation step for the case in Table 3. Our algorithm indicates that the model tries to generate a sentence without accessing source information during hallucination since the target prefix dominates the contribution. On the contrary, the contribution of the target prefix stays relatively low in a well-generated sequence. It only escalates at the generation of subword tails (step 7) or ⟨EOS⟩ tokens (step 9), where more language modeling takes over.

As a comparison, we did not find the above pattern in the results of IG. The results of LOO overestimate the contribution of the target prefix and lack interpretability of the trends on the well-generated sample. We further verify this pattern on more test samples, as shown in Figure 5. The contributions of target prefix to hallucinated samples are generally more than that to well-generated samples amongst all three methods, but only our algorithm distinguishes the two clusters.

5 Related Work

Interpreting DNNs involves various techniques, such as feature visualization (Olah et al., 2017; Yosinski et al., 2015), probing (Conneau et al., 2018; Shi et al., 2016), and analyzing learned weights (Tsang et al., 2018). Local interpretation belongs to another paradigm, which tries to interpret individual predictions of a DNN.

Existing works of local interpretation focus on assigning importance to individual features with respect to the prediction, such as pixels in an image or words in a sentence. The assignment employs methods like input occlusion (Li et al., 2016; Ribeiro et al., 2016b), gradient-based algorithms (Hechtlinger, 2016; Sundararajan et al., 2017), layer-wise relevance propagation (LRP, Voita et al., 2021; Bach et al., 2015), decomposition-based methods (Murdoch et al., 2018; Singh et al., 2019; Jin et al., 2020; Kobayashi et al., 2021; Modarressi et al., 2022; Ferrando et al., 2022), and others (Hao et al., 2021; Shrikumar et al., 2017).

Specifically in NLP, Voita et al. (2021) extend LRP to the Transformer to analyze NMT models. Murdoch et al. (2018) introduces a contextual decomposition to track the word-level importance in LSTM (Hochreiter and Schmidhuber, 1997). Singh et al. (2019) extend the aforementioned to produce hierarchical clustering of words along with the contribution of each cluster.

Backpropagation-based algorithms such as gradient-based algorithms (Sundararajan et al., 2017) and LRP (Voita et al., 2021) have exponential time or space complexity, making their application on long sequences infeasible. The occlusion algorithms (Li et al., 2016; Chen et al., 2020) also suffer from performance degradation on long sentences since occlusion has a limited impact on the semantics of long sentences. Our methods are similar to those based on additive decomposition (Kobayashi et al., 2021; Modarressi et al., 2022; Ferrando et al., 2022; Mickus et al., 2022). Despite not being explicitly noted, these methods all rely on the same assumption to linearize attention scores and layer normalization. However, they do not decompose the FFN layer and instead use heuristic algorithms to aggregate contributions across layers.

6 Conclusion

In this paper, we find that specific DNNs satisfy linearity under proper assumptions. We further leverage the linearity of the model to generate local explanations. We test proposed algorithms with the standard and pretrained Transformer architecture on two benchmark datasets. Experimental results show that our method achieves competitive performance in efficiency and fidelity of explanation. Additionally, we offer examples of different tasks to apply our algorithms for error analysis. We leave the analysis of other DNNs and the intermediate states of the models as future work.

7 Limitations

Although based on the Transformer model, our methods also apply to various DNN modules, including CNNs, Poolings, and their compositions. The applications of the proposed method in computer vision are left for future work.

An obvious limitation of this work is that we only verify our algorithm on models activated by ReLU. This issue can be alleviated because our algorithm is theoretically compatible with any piecewise linear activation function. For other functions in the ReLU family, such as the GELU (Hendrycks and Gimpel, 2016) used by BERT (Devlin et al., 2019; Liu et al., 2019), we replace the activations with ReLU, then fine-tune on downstream tasks and pretrain tasks (Appendix E). Our algorithms bog down on more complex nonlinear functions (e.g., sigmoid and tanh). It's intuitive to fit these nonlinear functions with ReLU-activated FNNs. However, this leads to additional computational and space complexity, which degrades performance after fitting.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments and suggestions that helped us to improve the quality of this manuscript. Their feedback was invaluable in helping us to refine our ideas and present them more effectively. Shujian Huang is the corresponding author. This work is supported by National Science Foundation of China (No. 62176115, 62176120), the Liaoning Provincial Research Foundation for Basic Research (No. 2022-KF-26-02).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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:e0130140.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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, Online. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

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.

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.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. Self-attention attribution: Interpreting information interactions inside transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12963–12971.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yotam Hechtlinger. 2016. Interpretation of prediction models using the input gradient. *arXiv preprint arXiv:1611.07634*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

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.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

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. *arXiv preprint arXiv:1907.11692*.

Scott M Lundberg, Gabriel G Erion, and Su-In Lee. 2018. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Timothee Mickus, Denis Paperno, and Mathieu Constant. 2022. How to dissect a Muppet: The structure of transformer embedding spaces. *Transactions of the Association for Computational Linguistics*, 10:981–996.

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.

Mathias Müller, Annette Rios, and Rico Sennrich. 2020. Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 151–164, Virtual. Association for Machine Translation in the Americas.

W James Murdoch, Peter J Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *ICLR*.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. 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 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill*, 2(11):e7.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. 2018. Model agnostic supervised local explanations. *Advances in neural information processing systems*, 31.

Marco Ribeiro, Sameer Singh, and Carlos Guestrin. 2016a. “why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016b. " 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.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

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.

Chandan Singh, W James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *ICLR*.

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 *International conference on machine learning*, pages 3319–3328. PMLR.

Michael Tsang, Dehua Cheng, and Yan Liu. 2018. Detecting statistical interactions from neural network weights. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Elena Voita, Rico Sennrich, and Ivan Titov. 2021. Analyzing the source and target contributions to predictions in neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1126–1140, Online. Association for Computational Linguistics.

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.

Jason Yosinski, Jeff Clune, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. In *In ICML Workshop on Deep Learning*. Citeseer.

A Consistency Condition

Proof. Prove the sufficiency and necessity of Lemma 1, respectively.

Sufficiency. To prove the sufficiency, we introduce decomposition under the *elementary partition* as an intermediate, where the *elementary partition* P_e is a partition in which each element in X forms a set., i.e. $P_e = \{\{x_1\}, \cdots, \{x_m\}\}$.

For any two partitions P_a and P_b that $P_a \cap P_b \neq \emptyset$ and $g \in P_a \cap P_b$, if $\mathscr{D}$ satisfies *Group Additivity*, then there is

$$\begin{aligned} \left.\frac{\mathscr{D}h}{\mathscr{D}g}\right|_{P_a} &= \sum_{x\in g} \left.\frac{\mathscr{D}h}{\mathscr{D}\{x\}}\right|_{P_e} \\ &= \left.\frac{\mathscr{D}h}{\mathscr{D}g}\right|_{P_b}. \end{aligned} \tag{18}$$

Necessity. For any two groups $g_1, g_2 \in X$ that $g_1 \cap g_2 = \emptyset$ and $g_1, g_2 \neq \emptyset$, and any partitions P_a and P_b that $g_1, g_2 \in P_a, g_1 \cup g_2 \in P_b$. Without loss of generality, assume that $P_a = \{g_1, g_2, g_3^a, \cdots, g_m^a\}$ and $P_b = \{g_1 \cup g_2, g_3^b, \cdots, g_n^b\}$.

There are two different partitions $P'_a = \{g_1, g_2, X\backslash(g_1 \cup g_2)\}$ and $P'_b = \{g_1 \cup g_2, X\backslash(g_1 \cup g_2)\}$. And we have

$$\sum_{i=3}^{m} \left.\frac{\mathscr{D}h}{\mathscr{D}g_i^a}\right|_{P_a} = h - \left.\frac{\mathscr{D}h}{\mathscr{D}g_1}\right|_{P_a} - \left.\frac{\mathscr{D}h}{\mathscr{D}g_2}\right|_{P_a}, \tag{19}$$

$$\left.\frac{\mathscr{D}h}{\mathscr{D}(X\backslash(g_i \cup g_j))}\right|_{P'_a} = h - \left.\frac{\mathscr{D}h}{\mathscr{D}g_1}\right|_{P'_a} - \left.\frac{\mathscr{D}h}{\mathscr{D}g_2}\right|_{P'_a}. \tag{20}$$

By the consistency of $\mathscr{D}$, we have $\left.\frac{\mathscr{D}h}{\mathscr{D}g_1}\right|_{P_a} = \left.\frac{\mathscr{D}h}{\mathscr{D}g_1}\right|_{P'_a}$ and $\left.\frac{\mathscr{D}h}{\mathscr{D}g_2}\right|_{P_a} = \left.\frac{\mathscr{D}h}{\mathscr{D}g_2}\right|_{P'_a}$. Thus

$$\sum_{i=3}^{m} \left.\frac{\mathscr{D}h}{\mathscr{D}g_i^a}\right|_{P_a} = \left.\frac{\mathscr{D}h}{\mathscr{D}[X\backslash(g_i \cup g_j)]}\right|_{P'_a}. \tag{21}$$

Similarly, there is

$$\sum_{i=3}^{n} \left.\frac{\mathscr{D}h}{\mathscr{D}g_i^b}\right|_{P_b} = \left.\frac{\mathscr{D}h}{\mathscr{D}[X\backslash(g_i \cup g_j)]}\right|_{P'_b}. \tag{22}$$

Now we get

$$\begin{aligned} \left.\frac{\mathscr{D}h}{\mathscr{D}g_1}\right|_{P_a} + \left.\frac{\mathscr{D}h}{\mathscr{D}g_2}\right|_{P_a} &= h - \sum_{i=3}^{m} \left.\frac{\mathscr{D}h}{\mathscr{D}g_i^a}\right|_{P_a} \\ &= h - \left.\frac{\mathscr{D}h}{\mathscr{D}[X\backslash(g_i \cup g_j)]}\right|_{P'_a}. \end{aligned} \tag{23}$$

$$\begin{aligned} \left.\frac{\mathscr{D}h}{\mathscr{D}g_i \cup g_j}\right|_{P_b} &= h - \sum_{i=3}^{n} \left.\frac{\mathscr{D}h}{\mathscr{D}g_i^b}\right|_{P_b} \\ &= h - \left.\frac{\mathscr{D}h}{\mathscr{D}[X\backslash(g_i \cup g_j)]}\right|_{P'_b}. \end{aligned} \tag{24}$$

Again according to the consistency, we have

$$\left.\frac{\mathscr{D}h}{\mathscr{D}[X\backslash(g_i \cup g_j)]}\right|_{P'_a} = \left.\frac{\mathscr{D}h}{\mathscr{D}[X\backslash(g_i \cup g_j)]}\right|_{P'_b}. \tag{25}$$

So

$$\left.\frac{\mathscr{D}h}{\mathscr{D}g_1}\right|_{P_a} + \left.\frac{\mathscr{D}h}{\mathscr{D}g_2}\right|_{P_a} = \left.\frac{\mathscr{D}h}{\mathscr{D}g_i \cup g_j}\right|_{P_b}. \tag{26}$$

□

B The Uniqueness of Interpretable Decomposition

We claim that the interpretable decomposition of linearly decomposable h is unique.

Proof. Assuming $h = f(X) = \sum_i^m W_i^X x_i$. Based on *Orthogonality*, we have

$$\frac{\mathscr{D}x_i}{\mathscr{D}g} = x_i \text{ for } x_i \in g, \tag{27}$$

$$\frac{\mathscr{D}x_j}{\mathscr{D}g} = 0 \text{ for } x_j \notin g. \tag{28}$$

By the linear transformation of *Linearity*, we have

$$\frac{\mathscr{D}(W_i^X x_i)}{\mathscr{D}g} = W_i^X \frac{\mathscr{D}x_i}{\mathscr{D}g} = W_i^X x_i \text{ for } x_i \in g, \tag{29}$$

$$\frac{\mathscr{D}(W_j^X x_j)}{\mathscr{D}g} = W_i^X \frac{\mathscr{D}x_j}{\mathscr{D}g} = 0 \text{ for } x_j \notin g. \tag{30}$$

By the addition of *Linearity*, we have

$$\begin{aligned} \frac{\mathscr{D}h}{\mathscr{D}g} &= \frac{\mathscr{D}(\sum_i^m W_i^X x_i)}{\mathscr{D}g} \\ &= \sum_i^m \frac{\mathscr{D}(W_i^X x_i)}{\mathscr{D}g} \\ &= \sum_{x_i \in g} W_i^X x_i. \end{aligned} \tag{31}$$

□

C Mathematical Properties of $\hat{\mathscr{D}}$

By definition, it is clear that $\hat{\mathscr{D}}$ satisfies *Linearity*.

Proof. proof of *Group Additivity* by mathematical induction.

Base Case. The same as Eq. (14), $\hat{\mathscr{D}}$ degenerates to $\bar{\mathscr{D}}$ and therefore inherits the *Group Additivity* property.

Induction step. For any hidden state h^l, it is obtained either by linear transformation and addition or by ReLU. Assume that the hidden states involved in the operation to get h^l all satisfy *Group Additivity*.

For addition and linear transformation, without loss of generality, suppose $h' = W_1h_1 + W_2h_2$, then there is

$$\begin{aligned}
&\frac{\hat{\mathscr{D}}h'}{\hat{\mathscr{D}}g_1} + \frac{\hat{\mathscr{D}}h'}{\hat{\mathscr{D}}g_2} = \frac{\hat{\mathscr{D}}(W_1h_1 + W_2h_2)}{\hat{\mathscr{D}}g_1} + \frac{\hat{\mathscr{D}}(W_1h_1 + W_2h_2)}{\hat{\mathscr{D}}g_2}\\
&= W_1\frac{\hat{\mathscr{D}}h_1}{\hat{\mathscr{D}}g_1} + W_2\frac{\hat{\mathscr{D}}h_2}{\hat{\mathscr{D}}g_1} + W_1\frac{\hat{\mathscr{D}}h_1}{\hat{\mathscr{D}}g_2} + W_2\frac{\hat{\mathscr{D}}h_2}{\hat{\mathscr{D}}g_2}\\
&= W_1\frac{\hat{\mathscr{D}}h_1}{\hat{\mathscr{D}}g_1 \cup g_2} + W_2\frac{\hat{\mathscr{D}}h_2}{\hat{\mathscr{D}}g_1 \cup g_2}\\
&= \frac{\hat{\mathscr{D}}h'}{\hat{\mathscr{D}}g_1 \cup g_2}.
\end{aligned} \tag{32}$$

For ReLU, suppose $h' = \text{relu}(h) = W'h$ and Θ is the separated contribution in Eq. (11), i.e.

$$\Theta := W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B} - \text{relu}(\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}B}). \tag{33}$$

Then we have

$$\begin{aligned}
&\frac{\hat{\mathscr{D}}h'}{\hat{\mathscr{D}}g_1} + \frac{\hat{\mathscr{D}}h'}{\hat{\mathscr{D}}g_2} = W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1} + \alpha_{g_1}\Theta + W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_2} + \alpha_{g_2}\Theta\\
&= W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + (\alpha_{g_1} + \alpha_{g_2})\Theta\\
&= W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + (\frac{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1}}{\sum_{g\in P}\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}} + \frac{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_2}}{\sum_{g\in P}\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}})\Theta\\
&= W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + \frac{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1} + \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_2}}{\sum_{g\in P}\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}}\Theta\\
&= W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + \frac{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1\cup g_2}}{\sum_{g\in P}\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}}\Theta\\
&= W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + \alpha_{g_1\cup g_2}\Theta\\
&= \frac{\hat{\mathscr{D}}h'}{\hat{\mathscr{D}}g_1 \cup g_2}.
\end{aligned} \tag{34}$$

□

Notice that we apply the signed-value-based decomposition (Eq. (13)) in line 3 of Eq. (34), while the absolute-value-based one does not make the derivation to hold.

D The Uniqueness of α

We claim that the signed-value-based α calculation is the only continuous solution that makes the decomposition $\hat{\mathscr{D}}$ satisfies consistency.

Proof. Since consistency and *Group Additivity* are equivalent, we will use both of their properties in the proof.

First prove that α itself satisfies *Group Additivity*, i.e, $\alpha_{g_1} + \alpha_{g_2} = \alpha_{g_1 \cup g_2}$.

According to the *Group Additivity* property of $\hat{\mathscr{D}}$, we have

$$\frac{\hat{\mathscr{D}}\text{relu}(h)}{\hat{\mathscr{D}}g_1} + \frac{\hat{\mathscr{D}}\text{relu}(h)}{\hat{\mathscr{D}}g_2} = \frac{\hat{\mathscr{D}}\text{relu}(h)}{\hat{\mathscr{D}}g_1 \cup g_2}, \tag{35}$$

$$\begin{aligned}
W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1} + \alpha_{g_1}\Theta + W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_2} + \alpha_{g_2}\Theta =&\\
W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + \alpha_{g_1\cup g_2}\Theta&,
\end{aligned} \tag{36}$$

$$\begin{aligned}
W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + \alpha_{g_1}\Theta + \alpha_{g_2}\Theta =&\\
W'\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2} + \alpha_{g_1\cup g_2}\Theta&,
\end{aligned} \tag{37}$$

$$\alpha_{g_1} + \alpha_{g_2} = \alpha_{g_1\cup g_2}, \tag{38}$$

where Θ is defined in Eq. (33).

Suppose that α is calculated by the function A, that is

$$\alpha_g = A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}), \tag{39}$$

where $H = \{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} | g \in P\}$.

Next, we prove that the value of $A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g})$ is only related to $\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}$ and $\sum_{g\in P}\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}$, instead of a specific values of other elements in H.

Since the sum of α is 1, we have

$$A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}) = 1 - \sum_{e\in H, e\neq g} A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}e}). \tag{40}$$

By the *Group Additivity* of α,

$$\sum_{e\in H, e\neq g} A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}e}) = A(H', \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}\bigcup_{e\in H, e\neq g} e}), \tag{41}$$

where $H' = \{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}\bigcup_{e\in H, e\neq g} e}\}$.

By the *Group Additivity* of $\hat{\mathscr{D}}$, there is

$$\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}\bigcup_{e\in H, e\neq g} e} = \sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} - \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}. \tag{42}$$

With Eq. (40) and Eq. (41), we have

$$A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}) = 1 - A(H', \sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} - \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}), \tag{43}$$

and $H' = \{\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}, \sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} - \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}\}$.

The proposition is proved. Let's replace the function $A(H, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g})$ with function $A'(\sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g})$.

Notice that we have

$$\begin{aligned} A'(s, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1}) + A'(s, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_2}) =& A'(s, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1 \cup g_2}) \\ =& A'(s, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_1} + \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g_2}). \end{aligned} \tag{44}$$

This means that $A'(s, x_1) + A'(s, x_2) = A'(s, x_1 + x_2)$ always holds. Thus $A'(s, ax) = aA'(s, x)$ holds for all $a \in \mathbb{Z}$ and all $x, s \in \mathbb{R}$. Further, $A'(s, \frac{a}{b}x) = \frac{a}{b}A'(s, x)$ holds for all $\frac{a}{b} \in \mathbb{Q}$ and all $x, s \in \mathbb{R}$.

Finally, we prove that $A'(s, rx) = rA'(s, x)$ holds for all $r \in \mathbb{R}$ and all $x, s \in \mathbb{R}$.

If $r \in \mathbb{R}$ and $r \notin \mathbb{Q}$, consider a sequence q_i in $\mathbb{Q}$ converging to r. Then the sequence $q_i x$ converges to rx and the sequence $q_i A'(s, x)$ converges to $rA'(s, x)$. If A' is continuous, then

$$\begin{aligned} A'(s, rx) =& A'(s, \lim_{i\to\infty} q_i x) \\ =& \lim_{i\to\infty} A'(s, q_i x) \\ =& \lim_{i\to\infty} q_i A'(s, x) \\ =& rA'(s, x). \end{aligned} \tag{45}$$

Therefore, $A'(s, x)$ is a linear function with respect to x. Suppose $A'(s, x) = cx$, we have

$$1 = \sum_{g\in P} A'(\sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}, \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}) = \sum_{g\in P} c\frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g} = c\sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}, \tag{46}$$

$$c = 1/\sum_{g\in P} \frac{\hat{\mathscr{D}}h}{\hat{\mathscr{D}}g}. \tag{47}$$

□

E Experiment Details

Data preprocessing All input text of GLEU and IMDB datasets are encoded by Byte-Pair Encoding (BPE, Sennrich et al., 2016) of RoBERTa, containing 50K subword units of byte-level vocabulary.

For WMT14 En-De dataset, sentences have been jointly tokenized and byte-pair encoded with 32k merge operations using a shared vocabulary.

Training details For GLUE (Wang et al., 2018), we follow the hyperparameter settings of RoBERTa (Liu et al., 2019), with batch sizes $\in \{16, 32\}$, and learning rates $\in \{1e-5, 2e-5, 3e-5\}$, with a linear warmup for the first 6% of steps followed by a linear decay to 0. We use the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e-6$. We fine-tune 10 epochs in each dataset. More details about hyperparameter configurations can be found in https://github.com/facebookresearch/fairseq/tree/main/examples/roberta/config/finetuning. For the IMDB dataset we set batch $= 16$, lr $= 1e-5$ and warmup $= 1256$, other settings are the same as GLEU benchmark.

Since the GELU activation (Hendrycks and Gimpel, 2016) in RoBERta is incompatible with our theory, we replace it with ReLU at fine-tuning, which leads to performance degradation, especially with small datasets. This issue can be solved by fine-tuning pre-training tasks prior to the downstream tasks: we re-train the pretraining tasks (i.e., masked language modeling) on a smaller dataset with ReLU activation function. We adopt the WikiText-103 dataset as the retraining corpus and use the same training configuration as fine-tuning, including batch $= 16$, lr $= 1e-5$ and warmup $= 1500$. The model with additional fine-tuning by pretraining tasks is comparable, and sometimes better than RoBERTa (Table 4).

For machine translation, we adopt $\beta = [0.9, 0.98]$ and $\epsilon = 1e-8$ for Adam optimizer. The learning rate linearly increases from $1e-7$ to $7e-4$ with 4000 warmup steps, then decay by the inverse square root schedule. We additionally adopt label smoothing at 0.1. Training instances are batched together by approximate sequence length. Input tokens in the batch are restricted to 8102 per GPU. The model is updated for 300k steps. We average the last 5 checkpoints, each of which is saved at the end of an epoch.

	MNLI	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	AVG.
RoBERTa$_{\text{BASE}}$	87.6	**92.8**	**91.9**	**78.7**	94.8	**90.2**	63.6	**91.2**	86.35
Our Impl.	86.9	89.7	91.1	56.3	92.1	75.5	75.5	87.1	81.8
+ FT. on MLM.	**87.7**	**92.8**	91.6	77.3	**95.0**	89.5	**83.5**	90.5	**88.49**

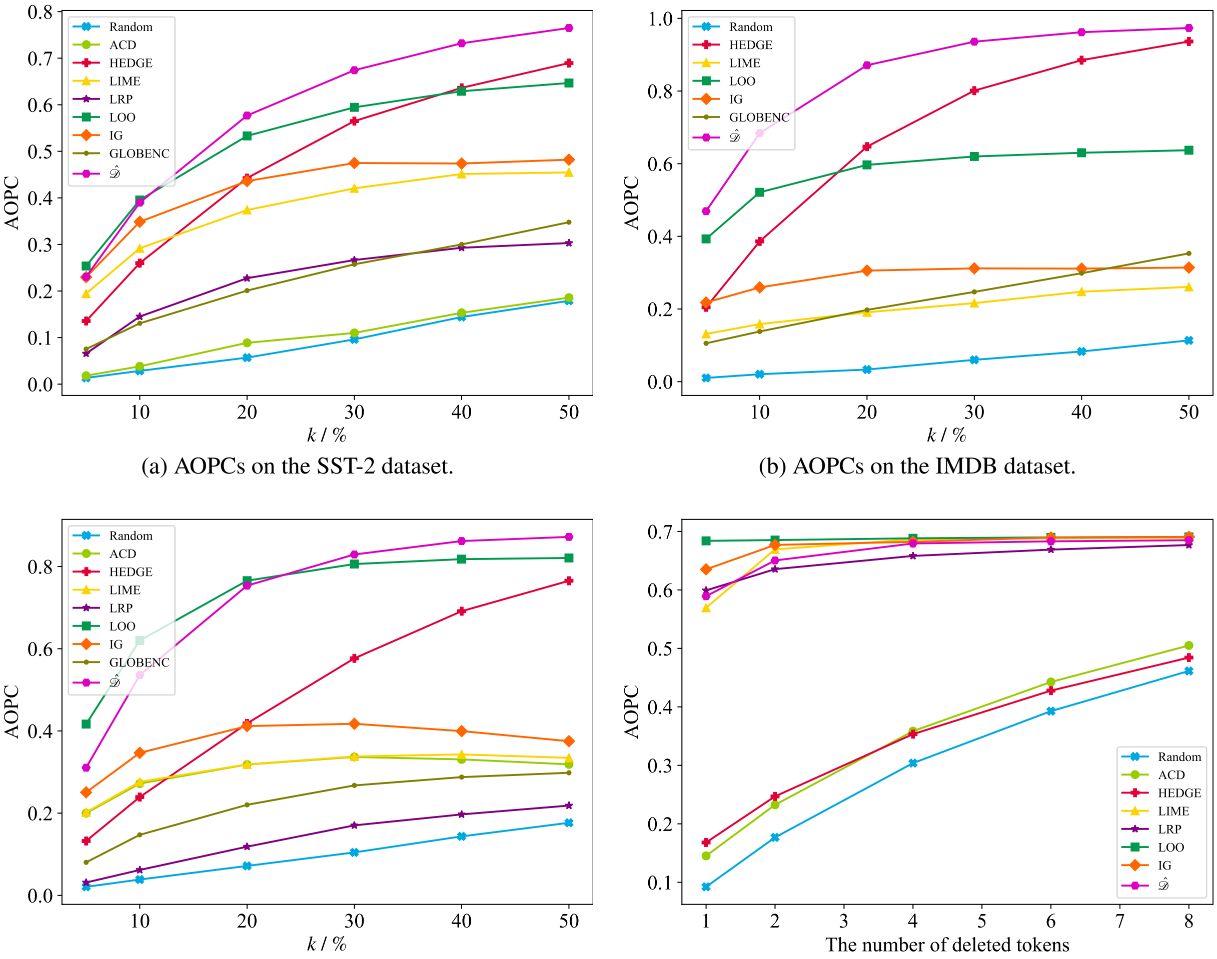
Table 4: Development set results on GLUE tasks for RoBERTa and our implementations.

All experiments were trained and evaluated using a single RTX 3090 Ti GPU, except for the translation model, which was trained on 2 RTX 3090 Ti GPUs.

F Performance Experiments

We present the full RoBERTa results of our implementation on development sets in Table 4. For IMDB, the fine-tuned RoBERTa model achieves 93.8% accuracy on the full test set. The model achieves a BLEU score (Papineni et al., 2002) of 27.19 on the WMT14 when trained from scratch.

G Results of AOPCs changing with different k



(a) AOPCs on the SST-2 dataset.

(b) AOPCs on the IMDB dataset.

(c) AOPCs on the RTE dataset.

(d) AOPCs on the WMT dataset.

Figure 6: AOPCs with different k on the SST-2, IMDB, RTE and WMT En-De datasets.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
7

☒ A2. Did you discuss any potential risks of your work?
Our work contains little potential risk.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

2, 3, 4

☑ B1. Did you cite the creators of artifacts you used?
3

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
All artifacts are publicly available and used in academic research.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use it for research purposes only.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We have used only publicly available datasets whose sensitive information has passed the provider's checks.

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Documentation of our algorithms will be provided in the future along with the code.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
3

C ☑ Did you run computational experiments?

3, 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix E

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
3, Appendix E

☒ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
The experimental results are not randomized.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix E

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.