DecompX: Explaining Transformers Decisions by Propagating Token Decomposition

Ali Modarressi[1,2⋆] **Mohsen Fayyaz**[3⋆] **Ehsan Aghazadeh**[3]
Yadollah Yaghoobzadeh[3,4] **Mohammad Taher Pilehvar**[4]
[1] Center for Information and Language Processing, LMU Munich, Germany
[2] Munich Center for Machine Learning (MCML), Germany [3] University of Tehran, Iran
[4] Tehran Institute for Advanced Studies, Khatam University, Iran
amodaresi@cis.lmu.de mohsen.fayyaz77@ut.ac.ir eaghazade1998@ut.ac.ir
y.yaghoobzadeh@ut.ac.ir mp792@cam.ac.uk

Abstract

An emerging solution for explaining Transformer-based models is to use vector-based analysis on how the representations are formed. However, providing a faithful vector-based explanation for a multi-layer model could be challenging in three aspects: (1) Incorporating all components into the analysis, (2) Aggregating the layer dynamics to determine the information flow and mixture throughout the entire model, and (3) Identifying the connection between the vector-based analysis and the model's predictions. In this paper, we present *DecompX* to tackle these challenges. DecompX is based on the construction of decomposed token representations and their successive propagation throughout the model without mixing them in between layers. Additionally, our proposal provides multiple advantages over existing solutions for its inclusion of all encoder components (especially nonlinear feed-forward networks) and the classification head. The former allows acquiring precise vectors while the latter transforms the decomposition into meaningful prediction-based values, eliminating the need for norm- or summation-based vector aggregation. According to the standard faithfulness evaluations, DecompX consistently outperforms existing gradient-based and vector-based approaches on various datasets. Our code is available at github.com/mohsenfayyaz/DecompX.

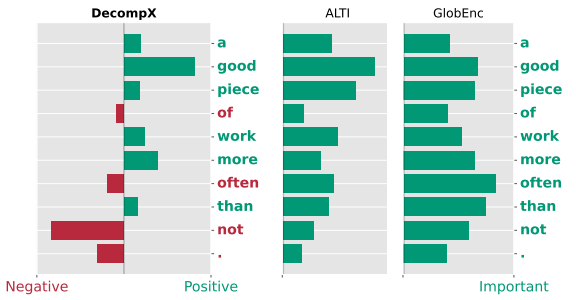

Figure 1: The explanation of our method (DecompX) compared with GlobEnc and ALTI for fine-tuned BERT on SST2 dataset (sentiment analysis). Our method is able to quantify positive or negative attribution of each token as well as being more accurate.

1 Introduction

While Transformer-based models have demonstrated significant performance, their black-box nature necessitates the development of explanation methods for understanding these models' decisions (Serrano and Smith, 2019; Bastings and Filippova, 2020; Lyu et al., 2022). On the one hand, researchers have adapted *gradient-based* methods from computer vision to NLP (Li et al., 2016; Wu and Ong, 2021). On the other hand, many have attempted to explain the decisions based on the components inside the Transformers architecture (*vector-based* methods). Recently, the latter has shown to be more promising than the former in terms of faithfulness (Ferrando et al., 2022).

Therefore, we focus on the vector-based methods which require an accurate estimation of (i) the mixture of tokens in each layer (*local-level* analysis), and (ii) the flow of attention throughout multiple layers (*global-level* analysis) (Pascual et al., 2021). Some of the existing local analysis methods include raw attention weights (Clark et al., 2019), effective attentions (Brunner et al., 2020), and vector norms (Kobayashi et al., 2020, 2021), which all attempt to explain how a single layer combines its input representations. Besides, to compute the global impact of the inputs on the outputs, the local behavior of all layers must be aggregated. *Attention rollout* and *attention flow* were the initial approaches for recursively aggregating the raw attention maps in each layer (Abnar and Zuidema, 2020). By employing rollout, GlobEnc (Modarressi et al., 2022) and ALTI (Ferrando et al., 2022) significantly improved

⋆ Equal contribution.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2649–2664

on previous work by substituting norm-based local methods (Kobayashi et al., 2021) for raw attentions. Despite their advancements, these vector-based methods still have three major limitations: (1) they ignore the encoder layer's Feed-Forward Network (FFN) because of its non-linearities, (2) they use rollout, which produces inaccurate results because it requires scalar local attributions rather than decomposed vectors which causes information loss, and (3) they do not take the classification head into account.

In an attempt to address all three limitations, in this paper, we introduce *DecompX*. Instead of employing rollout to aggregate local attributions, DecompX propagates the locally decomposed vectors throughout the layers to build a global decomposition. Since decomposition vectors propagate along the same path as the original representations, they accurately represent the inner workings of the entire model. Furthermore, we incorporate the FFNs into the analysis by proposing a solution for the non-linearities. The FFN workaround, as well as the decomposition, enable us to also propagate through the classification head, yielding per predicted label explanations. Unlike existing techniques that provide absolute importance, this per-label explanation indicates the extent to which each individual token has contributed towards or against a specific label prediction (Figure 1).

We conduct a comprehensive faithfulness evaluation over various datasets and models, that verifies how the novel aspects of our methodology contribute to more accurate explanations. Ultimately, our results demonstrate that DecompX consistently outperforms existing well-known gradient- and vector-based methods by a significant margin.

2 Related Work

Vector-based analysis has been sparked by the motivation that attention weights alone are insufficient and misleading to explain the model's decisions (Serrano and Smith, 2019; Jain and Wallace, 2019). One limitation was that it neglects the self-attention value vectors multiplied by the attention weights. Kobayashi et al. (2020) addressed it by using the norm of the weighted value vectors as a measure of inter-token attribution. Their work could be regarded as one of the first attempts at Transformer decomposition. They expanded their analysis from the self-attention layer to the entire attention block and found that residual connections are crucial to the information flow in the encoder layer (Kobayashi et al., 2021).

However, to be able to explain the multilayer dynamics, one needs to aggregate the local analysis into global by considering the attribution mixture across layers. Abnar and Zuidema (2020) introduce the attention rollout and flow methods, which aggregate multilayer attention weights to create an overall attribution map. Nevertheless, the method did not result in accurate maps as it was based on an aggregation of attention weights only. *GlobEnc* (Modarressi et al., 2022) and *ALTI* (Ferrando et al., 2022) improved this by incorporating decomposition at the local level and then aggregating the resulting vectors-norms with rollout to build global level explanations. At the local level, GlobEnc extended Kobayashi et al. (2021) by incorporating the second Residual connection and LayerNormalization layer after the attention block. GlobEnc utilizes the L2-norm of the decomposed vectors as an attribution measure; however, Ferrando et al. (2022) demonstrate that the reduced anisotropy of the local decomposition makes L2-norms an unreliable metric. Accordingly, they develop a scoring metric based on the L1-distances between the decomposed vectors and the output of the attention block. The final outcome after applying rollout, referred to as ALTI, showed improvements in both the attention-based and norm-based scores.

Despite continuous improvement, all these methods suffer from three main shortcomings. They all omitted the classification head, which plays a significant role in the output of the model. In addition, they only evaluate linear components for their decomposition, despite the fact that the FFN plays a significant role in the operation of the model (Geva et al., 2021, 2022). Nonetheless, the most important weakness in their analysis is the use of rollout for multi-layer aggregation.

Rollout assumes that the only required information for computing the global flow is a set of scalar cross-token attributions. Nevertheless, this simplifying assumption ignores that each decomposed vector represents the multi-dimensional impact of its inputs. Therefore, losing information is inevitable when reducing these complex vectors into one cross-token weight. On the contrary, by keeping and propagating the decomposed vectors in DecompX, any transformation applied to the representations can be traced back to the input tokens without information loss.

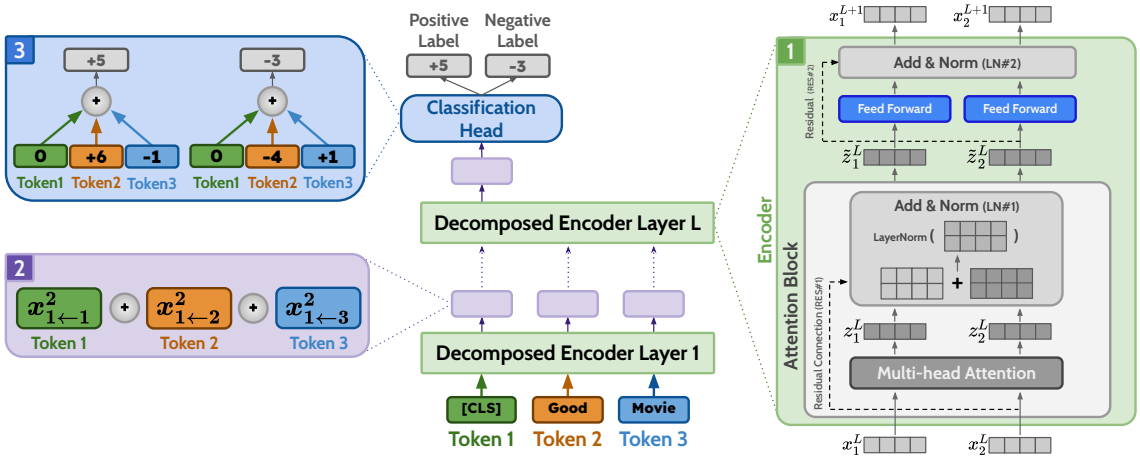

Figure 2: The overall workflow of DecompX. The contributions include: (1) incorporating all components in the encoder layer, especially the non-linear feed-forward networks; (2) propagating the decomposed token representations through layers which prevents them from being mixed; and (3) passing the decomposed vectors through the classification head, acquiring the exact positive/negative effect of each input token on individual output classes.

Gradient-based methods. One might consider gradient-based explanation methods as a workaround to the three issues stated above. Methods such as vanilla gradients (Simonyan et al., 2014), GradientXInput (Kindermans et al., 2016), and Integrated gradients (Sundararajan et al., 2017) all rely on the gradients of the prediction score of the model w.r.t. the input embeddings. To convert the gradient vectors into scalar per-token importance, various reduction methods such as L1-norm (Li et al., 2016), L2-norm (Poerner et al., 2018), and mean (Atanasova et al., 2020; Pezeshkpour et al., 2022) have been employed. Nonetheless, Bastings et al. (2022) evaluations showed that none of them is consistently better than the other. Furthermore, adversarial analysis and sanity checks both have raised doubts about gradient-based methods' trustworthiness (Wang et al., 2020; Adebayo et al., 2018; Kindermans et al., 2019).

Perturbation-based methods. Another set of interpretability methods, broadly classified as perturbation-based methods, encompasses widely recognized approaches such as LIME (Ribeiro et al., 2016) and SHAP (Shapley, 1953). However, these were excluded from our choice of comparison techniques, primarily due to their documented inefficiencies and reliability issues as highlighted by Atanasova et al. (2020). We follow recent work (Ferrando et al., 2022; Mohebbi et al., 2023) and mainly compare against gradient-based methods which have consistently proven to be more faithful than perturbation-based methods.

Mohebbi et al. (2023) recently presented a method called *Value zeroing* to measure the extent of context mixing in encoder layers. Their approach involves setting the value representation of each token to zero in each layer and then calculating attribution scores by comparing the cosine distances with the original representations. Although they focused on local-level faithfulness, their global experiment has clear drawbacks due to its reliance on rollout aggregation and naive evaluation metric (cf. A.3).

3 Methodology

Based on the vector-based approaches of Kobayashi et al. (2021) and Modarressi et al. (2022), we propose *decomposing* token representations into their constituent vectors. Consider decomposing the i^{th} token representation in layer $\ell \in \{0, 1, 2, ..., L, L+1\}$[1], i.e., $\boldsymbol{x}_i^\ell \in \{\boldsymbol{x}_1^\ell, \boldsymbol{x}_2^\ell, ..., \boldsymbol{x}_N^\ell\}$, into elemental vectors attributable to each of the N input tokens:

$$\boldsymbol{x}_i^\ell = \sum_{k=1}^{N} \boldsymbol{x}_{i \Leftarrow k}^\ell \tag{1}$$

According to this decomposition, we can compute the norm of the attribution vector of the k^{th} input ($\boldsymbol{x}_{i \Leftarrow k}^\ell$) to quantify its total attribution to $\boldsymbol{x}_i^\ell$. The main challenge of this decomposition, however, is how we could obtain the attribution vectors in accordance with the internal dynamics of the model.

[1] $\ell = 0$ is the input embedding layer and $\ell = L + 1$ is the classification head over the last encoder layer.

As shown in Figure 2, in the first encoder layer, the first set of decomposed attribution vectors can be computed as $\boldsymbol{x}^2_{i \Leftarrow k}$.[2] These vectors are passed through each layer in order to return the decomposition up to that layer: $\boldsymbol{x}^{\ell}_{i \Leftarrow k} \rightarrow \text{Encoder}^{\ell} \rightarrow \boldsymbol{x}^{\ell+1}_{i \Leftarrow k}$. Ultimately, the decomposed vectors of the [CLS] token are passed through the classification head, which returns a decomposed set of logits. These values reveal the extent to which each token has influenced the corresponding output logit.

In this section, we explain how vectors are decomposed and propagated through each component, altogether describing a complete propagation through an encoder layer. After this operation is repeated across all layers, we describe how the classification head transforms the decomposition vectors from the last encoder layer into prediction explanation scores.

3.1 The Multi-head Self-Attention

The first component in each encoder layer is the multi-head self-attention mechanism. Each head, $h \in \{1, 2, ..., H\}$, computes a set of attention weights where each weight $\alpha^h_{i,j}$ specifies the raw attention from the i^{th} to the j^{th} token. According to Kobayashi et al. (2021)'s reformulation, the output of multi-head self-attention, $\boldsymbol{z}^{\ell}_i$, can be viewed as the sum of the projected value transformation ($\boldsymbol{v}^h(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{W}^h_v + \boldsymbol{b}^h_v$) of the input over all heads:

$$\boldsymbol{z}^{\ell}_i = \sum_{h=1}^{H}\sum_{j=1}^{N} \alpha^h_{i,j} \boldsymbol{v}^h(\boldsymbol{x}^{\ell}_j)\boldsymbol{W}^h_{\boldsymbol{O}} + \boldsymbol{b}_{\boldsymbol{O}} \tag{2}$$

The multi-head mixing weight $\boldsymbol{W}^h_{\boldsymbol{O}}$ and bias $\boldsymbol{b}_{\boldsymbol{O}}$ could be combined with the value transformation to form an equivalent weight $\boldsymbol{W}^h_{\boldsymbol{Att}}$ and bias $\boldsymbol{b}_{\boldsymbol{Att}}$ in a simplified format[3]:

$$\boldsymbol{z}^{\ell}_i = \sum_{h=1}^{H}\sum_{j=1}^{N} \underbrace{\alpha^h_{i,j}\boldsymbol{x}^{\ell}_j \boldsymbol{W}^h_{\boldsymbol{Att}}}_{\boldsymbol{z}^{\ell}_{i \leftarrow j}} + \boldsymbol{b}_{\boldsymbol{Att}} \tag{3}$$

Since Kobayashi et al. (2021) and Modarressi et al. (2022) both use local-level decomposition, they regard $\boldsymbol{z}^{\ell}_{i \leftarrow j}$ as the attribution vector of token i from input token j in layer ℓ's multi-head attention.[4] We also utilize this attribution vector, but only in the first encoder layer since its inputs are also the same inputs of the whole model ($\boldsymbol{z}^1_{i \leftarrow j} = \boldsymbol{z}^1_{i \Leftarrow j}$). For other layers, however, each layer's decomposition should be based on the decomposition of the previous encoder layer. Therefore, we plug Eq. 1 into the formula above:

$$\begin{aligned} \boldsymbol{z}^{\ell}_i &= \sum_{h=1}^{H}\sum_{j=1}^{N} \alpha^h_{i,j} \sum_{k=1}^{N} \boldsymbol{x}^{\ell}_{j \Leftarrow k} \boldsymbol{W}^h_{\boldsymbol{Att}} + \boldsymbol{b}_{\boldsymbol{Att}} \\ &= \sum_{k=1}^{N} \sum_{h=1}^{H}\sum_{j=1}^{N} \alpha^h_{i,j} \boldsymbol{x}^{\ell}_{j \Leftarrow k} \boldsymbol{W}^h_{\boldsymbol{Att}} + \boldsymbol{b}_{\boldsymbol{Att}} \end{aligned} \tag{4}$$

To finalize the decomposition we need to handle the bias which is outside the model inputs summation ($\sum_{k=1}^{N}$). One possible workaround would be to simply omit the model's internal biases inside the self-attention layers and other components such as feed-forward networks. We refer to this solution as *NoBias*. However, without the biases, the input summation would be incomplete and cannot recompose the inner representations of the model. Also, if the decomposition is carried out all the way to the classifier's output without considering the biases, the resulting values will not tally up to the logits predicted by the model. To this end, we also introduce a decomposition method for the bias vectors with *AbsDot*, which is based on the absolute value of the dot product of the summation term (highlighted in Eq. 4) and the bias:

$$\omega_k = \frac{|\boldsymbol{b}_{\boldsymbol{Att}} \cdot \boldsymbol{z}^{\ell}_{i \Leftarrow k,[\text{NoBias}]}|}{\sum_{k=1}^{N} |\boldsymbol{b}_{\boldsymbol{Att}} \cdot \boldsymbol{z}^{\ell}_{i \Leftarrow k,[\text{NoBias}]}|} \tag{5}$$

where ω_k is the weight that decomposes the bias and enables it to be inside the input summation:

$$\boldsymbol{z}^{\ell}_i = \sum_{k=1}^{N} \underbrace{\left(\sum_{h=1}^{H}\sum_{j=1}^{N} \alpha^h_{i,j} \boldsymbol{x}^{\ell}_{j \Leftarrow k} \boldsymbol{W}^h_{\boldsymbol{Att}} + \omega_k \boldsymbol{b}_{\boldsymbol{Att}} \right)}_{\boldsymbol{z}^{\ell}_{i \Leftarrow k}} \tag{6}$$

The rationale behind *AbsDot* is that the bias is ultimately added into all vectors at each level; consequently, the most affected decomposed vectors are the ones that have the greatest degree of alignment (in terms of cosine similarity) and also have larger norms. The sole usage of cosine similarity could be one solution but in that case, a decomposed vector lacking a norm (such as padding tokens) could also be affected by the bias vector. Although alternative techniques may be employed, our preliminary

[2] As $\boldsymbol{x}$ denotes the inputs, the output decomposition of the first layer is the input of the second layer.

[3] cf. A.1 for further detail on the simplification process.

[4] Note that even though they discard the bias within the head-mixing module, $\boldsymbol{b}_{\boldsymbol{O}}$, the value bias $\boldsymbol{b}^h_v$ is included.

quantitative findings suggested that *AbsDot* represents a justifiable and suitable selection.

Our main goal from now on is to try to make the model inputs summation $\sum_{k=1}^{N}$ the most outer sum, so that the summation term ($\boldsymbol{z}^{\ell}_{i \Leftarrow k}$ for the formula above) ends up as the desired decomposition.[5]

3.2 Finalizing the Attention Module

After the multi-head attention, a residual connection adds the layer's inputs ($\boldsymbol{x}^{\ell}_{i}$) to $\boldsymbol{z}^{\ell}_{i}$, producing the inputs of the first LayerNormalization (LN#1):

$$\begin{aligned}\tilde{\boldsymbol{z}}^{\ell}_{i} &= \mathrm{LN}(\boldsymbol{z}^{+\ell}_{i}) \\ &= \mathrm{LN}(\boldsymbol{x}^{\ell}_{i} + \sum_{k=1}^{N} \boldsymbol{z}^{\ell}_{i \Leftarrow k}) \\ &= \mathrm{LN}(\sum_{k=1}^{N} [\boldsymbol{x}^{\ell}_{i \Leftarrow k} + \boldsymbol{z}^{\ell}_{i \Leftarrow k}])\end{aligned} \tag{7}$$

Again, to expand the decomposition over the LN function, we employ a technique introduced by Kobayashi et al. (2021) in which the LN function is broken down into a summation of a new function $g(.)$:

$$\begin{aligned}\mathrm{LN}(\boldsymbol{z}^{+\ell}_{i}) &= \sum_{k=1}^{N} \underbrace{g_{\boldsymbol{z}^{+\ell}_{i}}(\boldsymbol{z}^{+\ell}_{i \Leftarrow k}) + \boldsymbol{\beta}}_{\tilde{\boldsymbol{z}}^{\ell}_{i \Leftarrow k}} \\ g_{\boldsymbol{z}^{+\ell}_{i}}(\boldsymbol{z}^{+\ell}_{i \Leftarrow k}) &:= \frac{\boldsymbol{z}^{+\ell}_{i \Leftarrow k} - m(\boldsymbol{z}^{+\ell}_{i \Leftarrow k})}{s(\boldsymbol{z}^{+\ell}_{i})} \odot \boldsymbol{\gamma}\end{aligned} \tag{8}$$

where $m(.)$ and $s(.)$ represent the input vector's element-wise mean and standard deviation, respectively.[6] Unlike Kobayashi et al. (2021) and Modarressi et al. (2022), we also include the LN bias ($\boldsymbol{\beta}$) using our bias decomposition method.

3.3 Feed-Forward Networks Decomposition

Following the attention module, the outputs enter a 2-layer Feed-Forward Network (FFN) with a non-linear activation function (f_{act}):

$$\begin{aligned}\boldsymbol{z}^{\ell}_{\mathrm{FFN}} &= \mathrm{FFN}(\tilde{\boldsymbol{z}}^{\ell}_{i}) \\ &= f_{\mathrm{act}}(\underbrace{\tilde{\boldsymbol{z}}^{\ell}_{i} \boldsymbol{W}^{1}_{\mathrm{FFN}} + \boldsymbol{b}^{1}_{\mathrm{FFN}}}_{\boldsymbol{\zeta}^{\ell}_{i}}) \boldsymbol{W}^{2}_{\mathrm{FFN}} + \boldsymbol{b}^{2}_{\mathrm{FFN}}\end{aligned} \tag{9}$$

$\boldsymbol{W}^{\lambda}_{\mathrm{FFN}}$ and $\boldsymbol{b}^{\lambda}_{\mathrm{FFN}}$ represent the weights and biases, respectively, with λ indicating the corresponding layer within the FFN. In this formulation, the activation function is the primary inhibiting factor to continuing the decomposition. As a workaround, we approximate and decompose the activation function based on two assumptions: the activation function (1) passes through the origin ($f_{\mathrm{act}}(0) = 0$) and (2) is monotonic.[7] The approximate function is simply a zero intercept line with a slope equal to the activation function's output divided by its input in an elementwise manner:

$$\begin{aligned}f^{(\boldsymbol{x})}_{\mathrm{act}}(\boldsymbol{x}) &= \boldsymbol{\theta}^{(\boldsymbol{x})} \odot \boldsymbol{x} \\ \boldsymbol{\theta}^{(\boldsymbol{x})} &:= (\theta_1, \theta_2, ...\theta_d) \text{ s.t. } \theta_t = \frac{f_{\mathrm{act}}(x^{(t)})}{x^{(t)}}\end{aligned} \tag{10}$$

where (t) denotes the dimension of the corresponding vector. One important benefit of this alternative function is that when $\boldsymbol{x}$ is used as an input, the output is identical to that of the original activation function. Hence, the sum of the decomposition vectors would still produce an accurate result. Using the described technique we continue our progress from Eq. 9 by decomposing the activation function:

$$\begin{aligned}\boldsymbol{z}^{\ell}_{\mathrm{FFN},i} &= f^{(\boldsymbol{\zeta}^{\ell}_{i})}_{\mathrm{act}}(\sum_{k=1}^{N} \boldsymbol{\zeta}^{\ell}_{i \Leftarrow k}) \boldsymbol{W}^{2}_{\mathrm{FFN}} + \boldsymbol{b}^{2}_{\mathrm{FFN}} \\ &= \sum_{k=1}^{N} \underbrace{\boldsymbol{\theta}^{(\boldsymbol{\zeta}^{\ell}_{i})} \odot \boldsymbol{\zeta}^{\ell}_{i \Leftarrow k} + \boldsymbol{b}^{2}_{\mathrm{FFN}}}_{\boldsymbol{z}^{\ell}_{\mathrm{FFN},i \Leftarrow k}}\end{aligned} \tag{11}$$

In designing this activation function approximation, we prioritized completeness and efficiency. For the former, we ensure that the sum of decomposed vectors should be equal to the token's representation, which has been fulfilled by applying the same θ to all decomposed values ζ based on the line passing the activation point. While more complex methods (such as applying different θ to each ζ) which require more thorough justification may be able to capture the nuances of different activation functions more accurately, we believe that our approach strikes a good balance between simplicity and effectiveness, as supported by our empirical results.

The final steps to complete the encoder layer progress are to include the other residual connection and LayerNormalization (LN#2), which could be handled similarly to Eqs. 7 and 8:

[5] For a bias-included analysis, note that the bias weighting in all subsequent decomposition equations is always determined by the bias itself and its prior term (highlighted in the above formula).

[6] $\boldsymbol{\gamma} \in \mathbb{R}^d$ and $\boldsymbol{\beta} \in \mathbb{R}^d$ are respectively the trainable scaling and bias weights of LN. For extra details, please refer to Appendix A in Kobayashi et al. (2021) for the derivation.

[7] Even though the *GeLU* activation function, which is commonly used in BERT-based models, is not a monotonic function in its $x < 0$ region, we ignore it since the values are small.

$$\boldsymbol{x}_i^{\ell+1} = \mathrm{LN}(\sum_{k=1}^{N}[\underbrace{\tilde{\boldsymbol{z}}^{\ell}_{i\Leftarrow k} + \boldsymbol{z}^{\ell}_{\mathrm{FFN},i\Leftarrow k}}_{\boldsymbol{z}^{\ell}_{\mathrm{FFN}^+,i\Leftarrow k}}])$$
$$= \sum_{k=1}^{N} \underbrace{g_{\boldsymbol{z}^{\ell}_{\mathrm{FFN}^+,i}}(\boldsymbol{z}^{\ell}_{\mathrm{FFN}^+,i\Leftarrow k}) + \boldsymbol{\beta}}_{\boldsymbol{x}^{\ell+1}_{i\Leftarrow k}} \quad (12)$$

Using the formulations described in this section, we can now obtain $\boldsymbol{x}^{\ell+1}_{i\Leftarrow k}$ from $\boldsymbol{x}^{\ell}_{i\Leftarrow k}$, and by continuing this process across all layers, $\boldsymbol{x}^{L+1}_{i\Leftarrow k}$ is ultimately determined.

3.4 Classification Head

Norm- or summation-based vector aggregation could be utilized to convert the decomposition vectors into interpretable attribution scores. However, in this case, the resulting values would only become the attribution of the output token to the input token, without taking into account the task-specific classification head. This is not a suitable representation of the model's decision-making, as any changes to the classification head would have no effect on the vector aggregated attribution scores. Unlike previous vector-based methods, we can include the classification head in our analysis thanks to the decomposition propagation described above.[8] As the classification head is also an FFN whose final output representation is the prediction scores $\boldsymbol{y} = (y_1, y_2, ..., y_C)$ for each class $c \in \{1, 2, ..., C\}$, we can continue decomposing through this head as well. In general, the [CLS] token representation of the last encoder layer serves as the input for the two-layer (pooler layer + classification layer) classification head:

$$\boldsymbol{y} = u_{\mathrm{act}}(\boldsymbol{x}^{L+1}_{[\mathrm{CLS}]}\boldsymbol{W}_{\mathrm{pool}} + \boldsymbol{b}_{\mathrm{pool}})\boldsymbol{W}_{\mathrm{cls}} + \boldsymbol{b}_{\mathrm{cls}} \quad (13)$$

Following the same procedure as in Section 3.3, we can now compute the input-based decomposed vectors of the classification head's output $\boldsymbol{y}_k$ using the decomposition of the [CLS] token, $\boldsymbol{x}_{i\Leftarrow k}$. By applying this, in each class we would have an array of attribution scores for each input token, the sum of which would be equal to the prediction score of the model for that class:

$$y_c = \sum_{k=1}^{N} y_{c\Leftarrow k} \quad (14)$$

To explain a predicted output, $y_{c\Leftarrow k}$ would be the attribution of the k^{th} token to the total prediction score.

[8] We also discuss about alternative use cases in section A.2

4 Experiments

Our faithfulness evaluations are conducted on four datasets covering different tasks, SST-2 (Socher et al., 2013) for sentiment analysis, MNLI (Williams et al., 2018) for NLI, QNLI (Rajpurkar et al., 2016) for question answering, and HateXplain (Mathew et al., 2021) for hate speech detection. Our code is implemented based on HuggingFace's Transformers library (Wolf et al., 2020). For our experiments, we used fine-tuned BERT-base-uncased (Devlin et al., 2019) and RoBERTa-base (Liu et al., 2019), obtained from the same library.[9] As for gradient-based methods, we choose 0.1 as a step size in integrated gradient experiments and consider the L2-Norm of the token's gradient vector as its final attribution score.[10]

4.1 Evaluation Metrics

We aim to evaluate our method's *Faithfulness* by perturbing the input tokens based on our explanations. A widely-used perturbation method removes $K\%$ of tokens with the highest / lowest estimated importance to see its impact on the output of the model (Chen et al., 2020; Nguyen, 2018). To mitigate the consequences of perturbed input becoming out-of-distribution (OOD) for the model, we replace the tokens with [MASK] instead of removing them altogether (DeYoung et al., 2020). This approach makes the sentences similar to the pre-training data in masked language modeling. We opted for three metrics: AOPC (Samek et al., 2016), Accuracy (Atanasova et al., 2020), and Prediction Performance (Jain et al., 2020).

AOPC: Given the input sentence x_i, the perturbed input $\tilde{x}_i^{(K)}$ is constructed by masking $K\%$ of the most/least important tokens from x_i. Afterward, AOPC computes the average change in the predicted class probability over all test data as follows:

$$\mathrm{AOPC}(K) = \frac{1}{N}\sum_{i=1}^{N} p(\hat{y} \mid x_i) - p(\hat{y} \mid \tilde{x}_i^{(K)}) \quad (15)$$

where N is the number of examples, and $p(\hat{y} \mid .)$ is the probability of the predicted class. When masking the most important tokens, a higher AOPC is better, and vice versa.

[9] RoBERTa results can be found in section A.3.

[10] All were conducted on an RTX A6000 24GB machine.

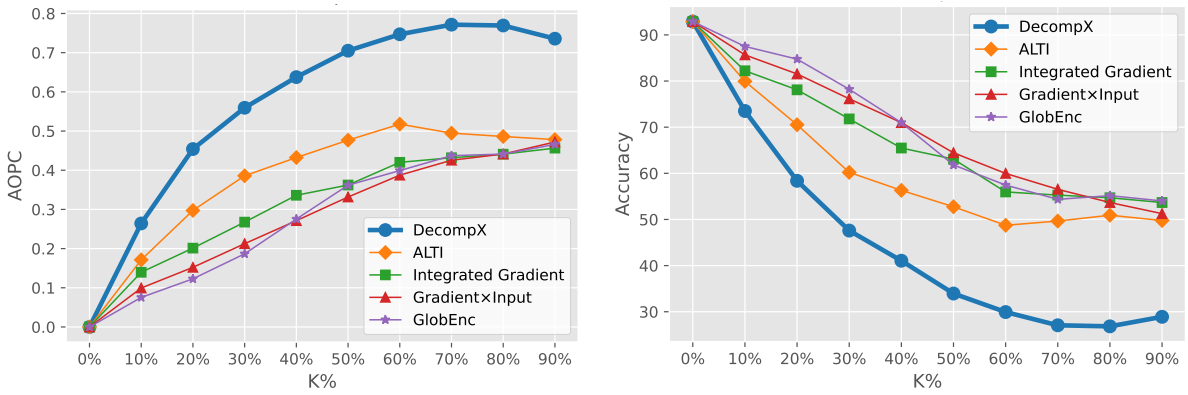

Figure 3: AOPC and Accuracy of different explanation methods on SST2 upon masking $K\%$ of the most important tokens (higher AOPC and lower Accuracy are better). DecompX outperforms existing methods by a large margin.

	SST2			MNLI			QNLI			HATEXPLAIN		
	ACC↓	AOPC↑	PRED↑	ACC↓	AOPC↑	PRED↑	ACC↓	AOPC↑	PRED↑	ACC↓	AOPC↑	PRED↑
GlobEnc (Modarressi et al., 2022)	67.14	0.307	72.36	48.07	0.498	70.43	64.93	0.342	84.00	47.65	0.401	56.50
+ FFN	64.90	0.326	79.01	45.05	0.533	75.15	63.74	0.354	84.97	46.89	0.406	59.52
ALTI (Ferrando et al., 2022)	57.65	0.416	88.30	45.89	0.515	74.24	63.85	0.355	85.69	43.30	0.469	64.67
Gradient×Input	66.69	0.310	67.20	44.21	0.544	76.05	62.93	0.366	86.27	46.28	0.433	60.67
Integrated Gradients	64.48	0.340	64.56	40.80	0.579	73.94	61.12	0.381	86.27	45.19	0.445	64.46
DecompX	**40.80**	**0.627**	**92.20**	**32.64**	**0.703**	**80.95**	**57.50**	**0.453**	**89.84**	**38.71**	**0.612**	**66.34**

Table 1: Accuracy, AOPC, and Prediction Performance of DecompX compared with the existing methods on different datasets. Each figure is the average across all perturbation ratios. As for Accuracy and AOPC, we mask the most important tokens while for Prediction Performance the least important tokens are removed (lower Accuracy, higher AOPC, and higher Prediction Performance scores are better).

Accuracy: Accuracy is calculated by averaging the performance of the model over different masking ratios. In cases where tokens are masked in decreasing importance order, lower Accuracy is better, and vice versa.

Predictive Performance: Jain et al. (2020) employ predictive performance to assess faithfulness by evaluating the sufficiency of their extracted rationales. The concept of sufficiency evaluates a rationale—a discretized version of soft explanation scores—to see if it adequately indicates the predicted label (Jacovi et al., 2018; Yu et al., 2019). Based on this, a BERT-based model is trained and evaluated based on inputs from rationales only to see how it performs compared with the original model. As mentioned by Jain et al. (2020), for each example, we select the top-$K\%$ tokens based on the explanation methods' scores to extract a rationale[11].

[11]We select the top 20% for the single sentence and top 40% for the dual sentence tasks.

4.2 Results

Figure 3 demonstrates the AOPC and Accuracy of the fine-tuned model on the perturbed inputs at different corruption rates K. As we remove the most important tokens in this experiment, higher changes in the probability of the predicted class computed by AOPC and lower accuracies are better. Our method outperforms comparison explanation methods, both vector- and gradient-based, by a large margin at every corruption rate on the SST2 dataset. Table 1 shows the aggregated AOPC and Accuracy over corruption rates, as well as Predicted Performance on different datasets. DecompX consistently outperforms other methods, which confirms that a holistic vector-based approach can present higher-quality explanations. Additionally, we repeated this experiment by removing the *least* important tokens. Figure A.2 and Table A.2 in the Appendix demonstrate that even with 10%-20% of the tokens selected by DecompX the task still performs incredibly well. When keeping only 10% of the tokens based on DecompX, the accuracy only

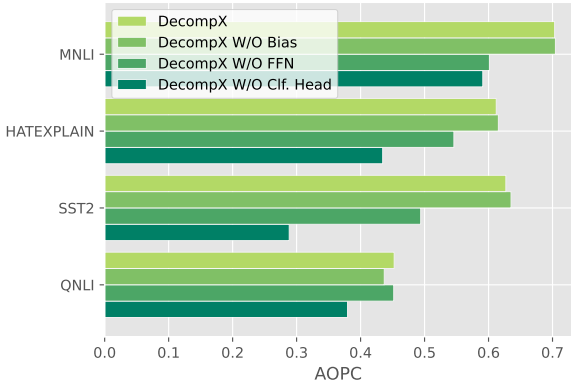

Figure 4: Leave-one-out ablation study of DecompX components. Higher AOPC scores are better.

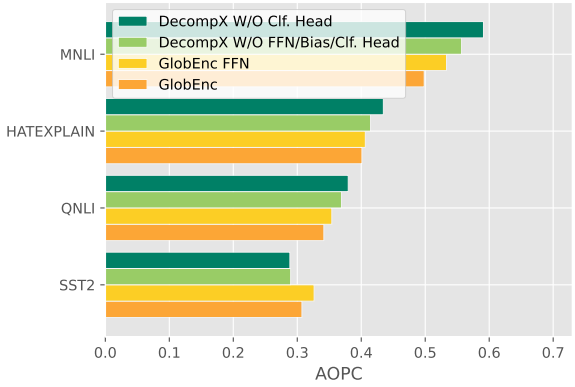

Figure 5: Ablation study for illustrating the effect of decomposition. Higher AOPC scores are better.

drops by 2.64% (from 92.89% of the full sentence), whereas the next best vector- and gradient-based methods suffer from the respective drops of 7.34% and 15.6%. In what follows we elaborate on the reasons behind this superior performance.

The role of feed-forward networks. Each Transformers encoder layer includes a feed-forward layer. Modarressi et al. (2022) omitted the influence of FFN when applying decomposition inside each layer due to FFN being a non-linear component. In contrast, we incorporated FFN's effect by a point-wise approximation (cf. §3.3). To examine its individual effect we implemented GlobEnc + FFN where we incorporated the FFN component in each layer. Table 1 shows that this change improves GlobEnc in terms of faithfulness, bringing it closer to gradient-based methods. Moreover, we conducted a leave-one-out ablation analysis[12] to ensure FFN's effect on DecompX. Figure 4 reveals that removing FFN significantly decreases the AOPC.

The role of biases. Even though Figure 4 demonstrates that considering bias in the analysis only has a slight effect, it is important to add biases for the human interpretability of DecompX. Figure 6 shows the explanations generated for an instance from MNLI by different methods. While the order of importance is the same in DecompX and DecompX W/O Bias, it is clear that adding the bias fixes the origin and describes which tokens had positive (green) or negative (red) effect on the predicted label probability. Another point is that without considering the biases, presumably less influential special tokens such as [SEP] are weighed disproportionately which is corrected in DecompX.[13]

The role of classification head. Figure 4 illustrates the effect of incorporating the classification head by removing it from DecompX. AOPC drastically drops when we do not consider the classification head, even more than neglecting bias and FFN, highlighting the important role played by the classification head. Moreover, incorporating the classification head allows us to acquire the exact effect of individual input tokens on each specific output class. An example of this was shown earlier in Figure 1, where the explanations are for the predicted class (Positive) in SST2. Figure 6 provides another example, for an instance from the MNLI dataset. Due to their omitting of the classification head, previous vector-based methods assign importance to some tokens (such as "or bolted") which are actually not important for the predicted label. This is due to the fact that the tokens were important for another label (contradiction; cf. Figure A.1). Importantly, previous methods fall short of capturing this per-label distinction. Consequently, we believe that no explanation method that omits the classification head can be deemed complete.

The role of decomposition. In order to demonstrate the role of propagating the decomposed vectors instead of aggregating them in each layer using rollout, we try to close the gap between DecompX and GlobEnc by simplifying DecompX and incorporating FFN in GlobEnc. With this simplification,

[12] In all our ablation studies, we use norm-based aggregation when not incorporating the classification head: $\|\tilde{\boldsymbol{x}}^{L+1}_{[\text{CLS}]\Leftarrow k}\|$

[13] The importance of special tokens does not change our results as it is not possible to remove the special tokens in the perturbed input.

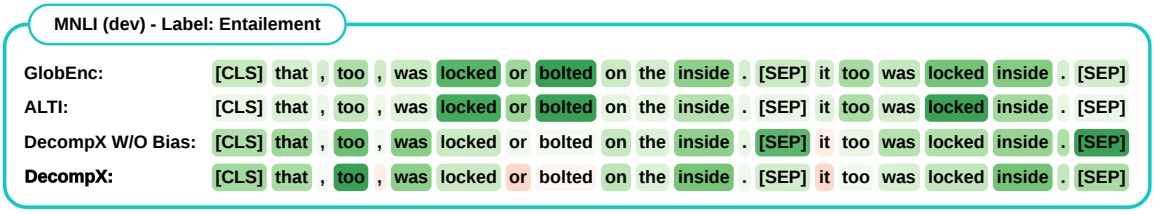

Figure 6: An example from MNLI dataset with Entailment label. In DecompX, green/red indicates the positive/negative impact of the token on the predicted label (Entailment, See Figure A.1 for Neutral and Contradiction). GlobEnc and ALTI only provide the general importance of tokens, not their positive or negative effect on each output class.

the difference between DecompX W/O classification head and GlobEnc with FFN setups is that the former propagates the decomposition of vectors while the latter uses norm-based aggregation and rollout between layers. Figure 5 illustrates the clear positive impact of our decomposition. We show that even without the FFN and bias, decomposition can outperform the rollout-based GlobEnc. These results demonstrate that aggregation in-between layers causes information loss and the final attributions are susceptible to this simplifying assumption.

5 Conclusions

In this work, we introduced *DecompX*, an explanation method based on propagating decomposed token vectors up to the classification head, which addresses the major issues of the previous vector-based methods. To achieve this, we incorporated all the encoder layer components including non-linear functions, propagated the decomposed vectors throughout the whole model instead of aggregating them in-between layers, and for the first time, incorporated the classification head resulting in faithful explanations regarding the exact positive or negative impact of each input token on the output classes. Through extensive experiments, we demonstrated that our method is consistently better than existing vector- and gradient-based methods by a wide margin. Our work can open up a new avenue for explaining model behaviors in various situations. As future work, one can apply the technique to encoder-decoder Transformers, multilingual, and Vision Transformers architectures.

Limitations

DecompX is an explanation method for decomposing output tokens based on input tokens of a Transformer model. Although the theory is applicable to other use cases, since our work is focused on English text classification tasks, extra care and evaluation experiments may be required to be used safely in other languages and settings. Due to limited resources, evaluation of large language models such as GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2022) was not viable.

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.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online. Association for Computational Linguistics.

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. "will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 976–991, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. 2020. On identifiability in transformers. In *International Conference on Learning Representations*.

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.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, 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.

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.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium. Association for Computational Linguistics.

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.

Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C. Wallace. 2020. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, Online. Association for Computational Linguistics.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T. Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. 2019. *The (Un)reliability of Saliency Methods*, pages 267–280. Springer International Publishing, Cham.

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. *arXiv*, abs/1611.07270.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

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.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. 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. *arXiv*, abs/1907.11692.

Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. 2022. Towards faithful model explanation in nlp: A survey. *arXiv*, abs/2209.11326.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *AAAI*.

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.

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.

Damian Pascual, Gino Brunner, and Roger Wattenhofer. 2021. Telling BERT's full story: from local attention to global aggregation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 105–124, Online. Association for Computational Linguistics.

Pouya Pezeshkpour, Sarthak Jain, Sameer Singh, and Byron Wallace. 2022. Combining feature and instance attribution to detect artifacts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1934–1946, Dublin, Ireland. Association for Computational Linguistics.

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Melbourne, Australia. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, UW EDU, Sameer Singh, and Carlos Guestrin. 2016. Model-Agnostic Interpretability of Machine Learning. In *ICML Workshop on Human Interpretability in Machine Learning*.

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.

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.

Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/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.

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.

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.

Zhengxuan Wu and Desmond C. Ong. 2021. On explaining your explanations of bert: An empirical study with sequence classification. *arXiv*, abs/2101.00196.

Mo Yu, Shiyu Chang, Yang Zhang, and Tommi Jaakkola. 2019. Rethinking cooperative rationalization: Introspective extraction and complement control. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4094–4103, Hong Kong, China. Association for Computational Linguistics.

A Appendix

A.1 Equivalent Weight and Bias in the Attention Module

$$
\begin{aligned}
\boldsymbol{z}_i^{\ell} &= \sum_{h=1}^{H}\sum_{j=1}^{N} \alpha_{i,j}^{h}(\boldsymbol{x}_j^{\ell}\boldsymbol{W}_v^{h} + \boldsymbol{b}_v^{h})\boldsymbol{W}_{\boldsymbol{O}}^{h} + \boldsymbol{b}_{\boldsymbol{O}} \\
&= \sum_{h=1}^{H}\sum_{j=1}^{N} \alpha_{i,j}^{h}(\boldsymbol{x}_j^{\ell}\boldsymbol{W}_v^{h}\boldsymbol{W}_{\boldsymbol{O}}^{h} + \boldsymbol{b}_v^{h}\boldsymbol{W}_{\boldsymbol{O}}^{h}) + \boldsymbol{b}_{\boldsymbol{O}} \\
&= \sum_{h=1}^{H}\sum_{j=1}^{N} \alpha_{i,j}^{h}\boldsymbol{x}_j^{\ell}\underbrace{\boldsymbol{W}_v^{h}\boldsymbol{W}_{\boldsymbol{O}}^{h}}_{\boldsymbol{W}_{\boldsymbol{Att}}^{h}} \\
&\quad + \underbrace{\sum_{h=1}^{H}\boldsymbol{b}_v^{h}\boldsymbol{W}_{\boldsymbol{O}}^{h}\sum_{j=1}^{N}\alpha_{i,j}^{h} + \boldsymbol{b}_{\boldsymbol{O}}}_{\boldsymbol{b}_{\boldsymbol{Att}}} \quad (16)
\end{aligned}
$$

A.2 Alternative use cases

The versatility of DecompX allows for explaining various NLP tasks and use cases. Since each output representation is decomposed based on the inputs ($\boldsymbol{x}_{i\Leftarrow k}^{L+1}$), it can be propagated through the task-specific head. In Question Answering (QA), for instance, there are two heads to identify the beginning and end of the answer span (Devlin et al., 2019). Thanks to the fact that DecompX is applied post-hoc and the final predicted span is known ($\boldsymbol{x}_{i=\text{Start}}^{L+1}$ and $\boldsymbol{x}_{i=\text{End}}^{L+1}$), we can continue propagation through the heads as described in Section 3.4. In the end, DecompX can indicate the impact of each input token on the span selection: $\boldsymbol{y}_{\text{Start}\Leftarrow k} \in \mathbb{R}^N$ & $\boldsymbol{y}_{\text{End}\Leftarrow k} \in \mathbb{R}^N$.

A.3 RoBERTa Results

Figures A.3 and A.4 demonstrate the results of our evaluations over the RoBERTa-base model.

In a contemporaneous work, Mohebbi et al. (2023) introduced the concept of *ValueZeroing* to incorporate the entire encoder layer and compute context mixing scores in each layer. Our experiments, as shown in Figures A.3 and A.4, demonstrate the poor performance of this technique at global-level. While it's possible that mismatching configurations[14] contributed to this inconsistency, we believe that the main issue lies in their reliance on an oversimplified evaluation measure for their global-level assessments. Their global level evaluation is based on the Spearman's correlation between the blank-out scores and various attribution methods (see Section 7 in Mohebbi et al. (2023)). The issue with this evaluation is that the blank-out baseline scores were obtained by removing only one token from the input (leave-one-out) and measuring the change in prediction probability, which cannot capture feature interactions (Lyu et al., 2022). For instance, in the sentence "The movie was great and amusing", independently removing "great" or "amusing" may not change the sentiment, resulting in smaller scores for these words.

[14] The authors of the study evaluated the models using blimp probing tasks in a prompting format, whereas we fine-tuned our models on SST-2 and MNLI tasks.

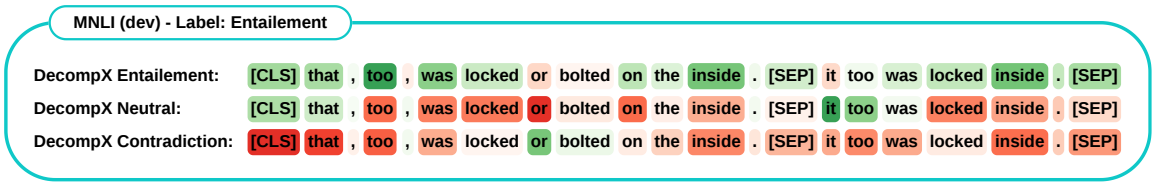

Figure A.1: An example from MNLI dataset with the *entailment* label. DecompX can provide explanations for each output class, and the sum of input explanations is equal to the final predicted logit for the corresponding class.

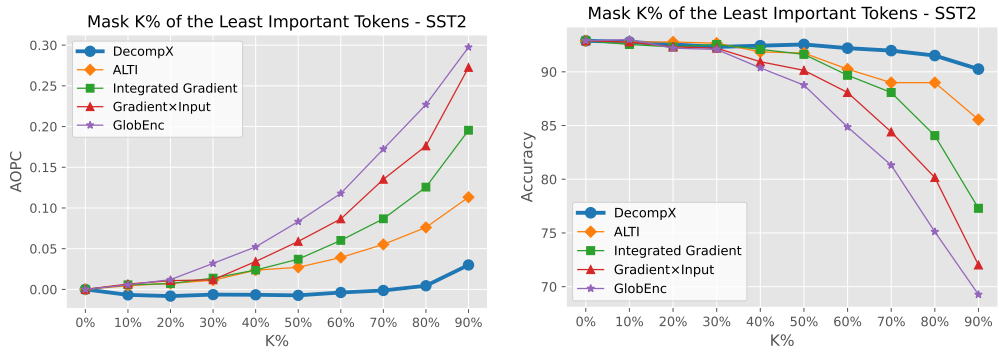

Figure A.2: AOPC and Accuracy of different explanation methods on the SST2 dataset after masking $K\%$ of the *least* important tokens (lower AOPC and higher Accuracy scores are better).

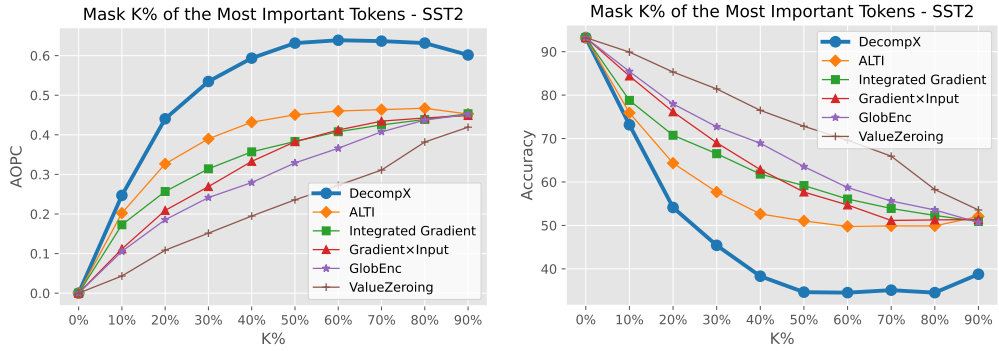

Figure A.3: RoBERTa-base AOPC and Accuracy of different explanation methods on the SST2 dataset after masking $K\%$ of the most important tokens (higher AOPC and lower Accuracy scores are better).

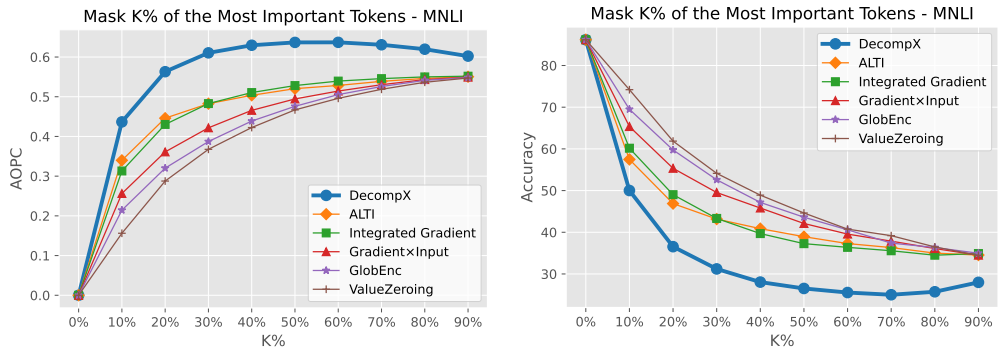

Figure A.4: RoBERTa-base AOPC and Accuracy of different explanation methods on the MNLI dataset after masking $K\%$ of the most important tokens (higher AOPC and lower Accuracy scores are better).

	SST2		MNLI		QNLI		HATEXPLAIN	
	AOPC↑	ACC↓	AOPC↑	ACC↓	AOPC↑	ACC↓	AOPC↑	ACC↓
DecompX	0.627	40.80	0.703	32.64	**0.453**	57.50	0.612	**38.71**
w/o Bias	**0.635**	**39.95**	**0.705**	**32.55**	0.437	58.66	**0.615**	38.73
w/o FFN	0.494	53.05	0.601	40.22	0.452	**55.97**	0.546	41.24
w/o Classification Head	0.288	69.93	0.591	39.80	0.380	61.83	0.435	45.31

Table A.1: Complete results of our ablation study when masking the *most* important tokens. We employ Leave-one-out ablation analysis to demonstrate the effects of bias, FFN, and classification head on the faithfulness of our method.

	SST2		MNLI		QNLI		HATEXPLAIN	
	AOPC↓	ACC↑	AOPC↓	ACC↑	AOPC↓	ACC↑	AOPC↓	ACC↑
GlobEnc (Modarressi et al., 2022)	0.111	0.852	0.205	0.715	0.151	0.817	0.204	0.600
+ FFN	0.087	0.872	0.171	0.744	0.134	0.832	0.185	0.613
ALTI (Ferrando et al., 2022)	0.040	0.906	0.191	0.731	0.121	0.844	0.135	0.644
Gradient×Input	0.088	0.870	0.164	0.746	0.125	0.839	0.175	0.620
Integrated Gradients	0.062	0.889	0.203	0.705	0.127	0.837	0.156	0.635
DecompX	**-0.001**	**0.921**	**0.104**	**0.767**	**0.085**	**0.853**	**0.035**	**0.657**

Table A.2: AOPC and Accuracy of DecompX compared with existing methods on different datasets. AOPC and Accuracy are the averages over perturbation ratios while masking the *least* important tokens (lower AOPC and higher Accuracy are better).

ACL 2023 Responsible NLP Checklist

A For every submission:

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

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

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

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

B ☑ Did you use or create scientific artifacts?

4. Experiments

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

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☑ 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)?
4. Experiments

☐ 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?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☒ 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.
The size of the datasets does not affect explanation extraction.

C ☑ Did you run computational experiments?

4. Experiments

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

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?
4. Experiments

☑ 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?
4. Experiments

☑ 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.)?
4. Experiments

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.