Self-Attention Attribution: Interpreting Information Interactions Inside Transformer

Yaru Hao,[1,2*] **Li Dong,**[2] **Furu Wei,**[2] **Ke Xu** [1]
[1] Beihang University
[2] Microsoft Research
{haoyaru@,kexu@nlsde.}buaa.edu.cn
{lidong1,fuwei}@microsoft.com

Abstract

The great success of Transformer-based models benefits from the powerful multi-head self-attention mechanism, which learns token dependencies and encodes contextual information from the input. Prior work strives to attribute model decisions to individual input features with different saliency measures, but they fail to explain how these input features interact with each other to reach predictions. In this paper, we propose a self-attention attribution method to interpret the information interactions inside Transformer. We take BERT as an example to conduct extensive studies. Firstly, we apply self-attention attribution to identify the important attention heads, while others can be pruned with marginal performance degradation. Furthermore, we extract the most salient dependencies in each layer to construct an attribution tree, which reveals the hierarchical interactions inside Transformer. Finally, we show that the attribution results can be used as adversarial patterns to implement non-targeted attacks towards BERT.

Introduction

Transformer (Vaswani et al. 2017) is one of state-of-the-art NLP architectures. For example, most pre-trained language models (Devlin et al. 2019; Liu et al. 2019; Dong et al. 2019; Bao et al. 2020; Clark et al. 2020; Conneau et al. 2020; Chi et al. 2020a,b) choose stacked Transformer as the backbone network. Their great success stimulates broad research on interpreting the internal black-box behaviors. Some prior efforts aim at analyzing the self-attention weights generated by Transformer (Clark et al. 2019; Kovaleva et al. 2019). In contrast, some other work argues that self-attention distributions are not directly interpretable (Serrano and Smith 2019; Jain and Wallace 2019; Brunner et al. 2020). Another line of work strives to attribute model decisions back to input tokens (Sundararajan, Taly, and Yan 2017; Shrikumar, Greenside, and Kundaje 2017; Binder et al. 2016). However, most previous attribution methods fail on revealing the information interactions between the input words and the compositional structures learnt by the network.

To address the above issues, we propose a self-attention attribution method (ATTATTR) based on integrated gradient (Sundararajan, Taly, and Yan 2017). We conduct experiments for BERT (Devlin et al. 2019) because it is one of the most representative Transformer-based models. Notice that our method is general enough, and can be applied to other Transformer networks without significant modifications. Results show that our method well indicates the information flow inside Transformer, which makes the self-attention mechanism more interpretable.

Firstly, we identify the most important attention connections in each layer using ATTATTR. We find that attention weights do not always correlate well with their contributions to the model prediction. We then introduce a heuristic algorithm to construct self-attention attribution trees, which discovers the information flow inside Transformer. In addition, a quantitative analysis is applied to justify how much the edges of an attribution tree contribute to the final prediction.

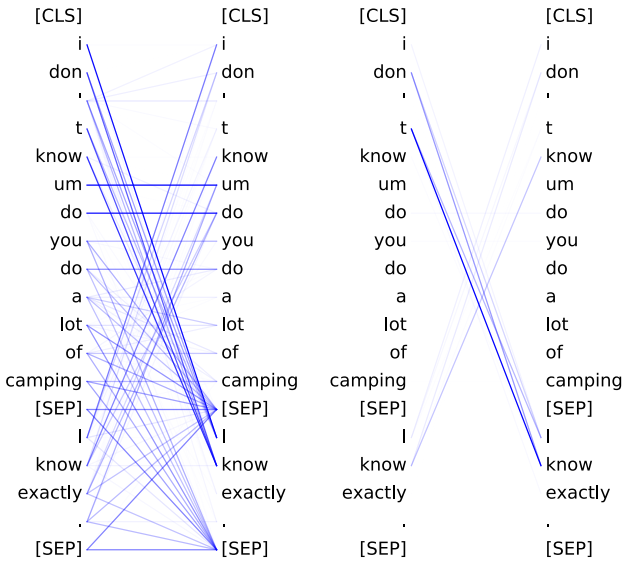
Next, we use ATTATTR to identify the most important attention heads and perform head pruning. The derived algorithm achieves competitive performance compared with the Taylor expansion method (Michel, Levy, and Neubig 2019). Moreover, we find that the important heads of BERT are roughly consistent across different datasets as long as the tasks are homogeneous.

Finally, we extract the interaction patterns that contribute most to the model decision, and use them as adversarial triggers to attack BERT-based models. We find that the fine-tuned models tend to over-emphasize some word patterns to make the prediction, which renders the prediction process less robust. For example, on the MNLI dataset, adding one adversarial pattern into the premise can drop the accuracy of `entailment` from 82.87% to 0.8%. The results show that ATTATTR not only can interpret the model decisions, but also can be used to find anomalous patterns from data.

The contributions of our work are as follows:

- We propose to use self-attention attribution to interpret the information interactions inside Transformer.
- We conduct extensive studies for BERT. We present how to derive interaction trees based on attribution scores, which visualizes the compositional structures learnt by Transformer.
- We show that the proposed attribution method can be used to prune self-attention heads, and construct adversarial triggers.

[*] Contribution during internship at Microsoft Research.



(a) Attention Score (b) Attribution Score

Figure 1: Attention score (left) and attribution score (right) of a single head in BERT. The color is darker for larger values. The prediction of the sentence from MNLI dataset is `contradiction`. ATTATTR tends to identify more sparse word interactions that contribute to the final model decision.

Background

Transformer (Vaswani et al. 2017) Given input tokens $\{x_i\}_{i=1}^{|x|}$, we pack their word embeddings to a matrix $X^0 = [x_1, \cdots, x_{|x|}]$. The stacked L-layer Transformer computes the final output via $X^l = \text{Transformer}_l(X^{l-1}), l \in [1, L]$.

The core component of a Transformer block is multi-head self-attention. The h-th self-attention head is described as:

$$Q_h = XW_h^Q,\ K = XW_h^K,\ V = XW_h^V \tag{1}$$

$$A_h = \text{softmax}(\frac{Q_h K_h^\intercal}{\sqrt{d_k}}) \tag{2}$$

$$H_h = \text{AttentionHead}(X) = A_h V_h \tag{3}$$

where $Q, K \in \mathbb{R}^{n \times d_k}$, $V \in \mathbb{R}^{n \times d_v}$, and the score $A_{i,j}$ indicates how much attention token x_i puts on x_j. There are usually multiple attention heads in a Transformer block. Let $|h|$ denote the number of attention heads in each layer, the output of multi-head attention is given by $\text{MultiH}(X) = [H_1, \cdots, H_{|h|}]W^o$, where $W^o \in \mathbb{R}^{|h|d_v \times d_x}$, $[\cdot]$ means concatenation, and H_i is computed as in Equation (3).

BERT (Devlin et al. 2019) We conduct all experiments on BERT, which is one of the most successful applications of Transformer. The pretrained language model is based on bidirectional Transformer, which can be fine-tuned towards downstream tasks. Notice that our method can also be applied to other multi-layer Transformer models with few modifications. For single input, a special token `[CLS]` is added to the beginning of the sentence, and another token `[SEP]` is added to the end. For pairwise input, `[SEP]` is also added as a separator between the two sentences. When BERT is fine-tuned on classification tasks, a softmax classifier is added on top of the `[CLS]` token in the last layer to make predictions.

Methods: Self-Attention Attribution

Figure 1a shows attention scores of one head in fine-tuned BERT. We observe that the attention score matrix is quite dense, although only one of twelve heads is plotted. It poses a huge burden on us to understand how words interact with each other within Transformer. Moreover, even if an attention score is large, it does not mean the pair of words is important to model decisions. In contrast, we aim at attributing model decisions to self-attention relations, which tends to assign higher scores if the interaction contributes more to the final prediction.

Given input sentence x, let $\text{F}_x(\cdot)$ represent the Transformer model, which takes the attention weight matrix A (Equation (2)) as the model input. Inspired by Sundararajan, Taly, and Yan (2017), we manipulate the internal attention scores $\bar{A}$, and observe the corresponding model dynamics $\text{F}_x(\bar{A})$ to inspect the contribution of word interactions. As the attribution is always targeted for a given input x, we omit it for the simplicity of notations.

Let us take one Transformer layer as an example to describe self-attention attribution. Our goal is to calculate an attribution score for each attention connection. For the h-th attention head, we compute its attribution score matrix as:

$$A = [A_1, \cdots, A_{|h|}]$$

$$\text{Attr}_h(A) = A_h \odot \int_{\alpha=0}^{1} \frac{\partial \text{F}(\alpha A)}{\partial A_h} d\alpha \in \mathbb{R}^{n \times n}$$

where $\odot$ is element-wise multiplication, $A_h \in \mathbb{R}^{n \times n}$ denotes the h-th head's attention weight matrix (Equation (2)), and $\frac{\partial \text{F}(\alpha A)}{\partial A_h}$ computes the gradient of model $\text{F}(\cdot)$ along A_h. The (i, j)-th element of $\text{Attr}_h(A)$ is computed for the interaction between input token x_i and x_j in terms of the h-th attention head.

The starting point ($\alpha = 0$) of the integration represents that all tokens do not attend to each other in a layer. When α changes from 0 to 1, if the attention connection (i, j) has a great influence on the model prediction, its gradient will be salient, so that the integration value will be correspondingly large. Intuitively, $\text{Attr}_h(A)$ not only takes attention scores into account, but also considers how sensitive model predictions are to an attention relation.

The attribution score can be efficiently computed via Riemman approximation of the integration (Sundararajan, Taly, and Yan 2017). Specifically, we sum the gradients at points occurring at sufficiently small intervals along the straightline path from the zero attention matrix to the original attention weight A:

$$\tilde{\text{Attr}}_h(A) = \frac{A_h}{m} \odot \sum_{k=1}^{m} \frac{\partial \text{F}(\frac{k}{m} A)}{\partial A_h} \tag{4}$$

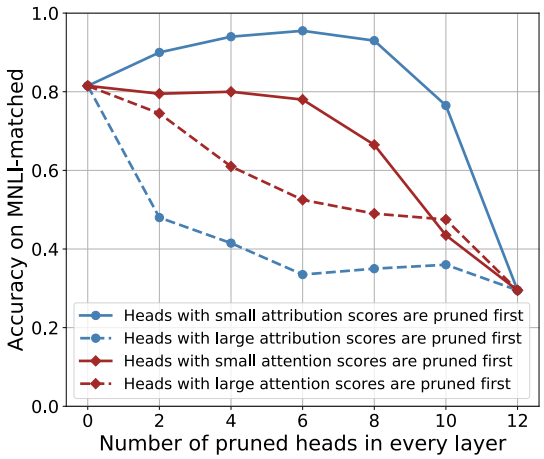


Figure 2: Effectiveness analysis of ATTATTR. The blue and red lines represent pruning attention heads according to attribution scores, and attention scores, respectively. The solid lines mean the attention heads with the smallest values are pruned first, while the dash lines mean the largest values are pruned first. The results show that ATTATTR better indicates the importance of attention heads.

where m is the number of approximation steps. In our experiments, we set m to 20, which performs well in practice.

Figure 1 is an example about the attention score map and the attribution score map of a single head in fine-tuned BERT. We demonstrate that larger attention scores do not mean more contribution to the final prediction. The attention scores between the `[SEP]` token and other tokens are relatively large, but they obtain little attribution scores. The prediction of the `contradiction` class attributes most to the connections between "*don't*" in the first segment and "*I know*" in the second segment, which is more explainable.

Experiments

We employ BERT-base-cased (Devlin et al. 2019) in our experiments. The number of BERT layers $|l| = 12$, the number of attention heads in each layer $|h| = 12$, and the size of hidden embeddings $|h|d_v = 768$. For a sequence of 128 tokens, the attribution time of the BERT-base model takes about one second on an Nvidia-v100 GPU card. Moreover, the computation can be parallelized by batching multiple input examples to increase throughput.

We perform BERT fine-tuning and conduct experiments on four classification datasets. **MNLI** (Williams, Nangia, and Bowman 2018) Multi-genre Natural Language Inference is to predict whether a premise entails the hypothesis (entailment), contradicts the given hypothesis (contradiction), or neither (neutral). **RTE** (Dagan, Glickman, and Magnini 2006; Bar-Haim et al. 2006; Giampiccolo et al. 2007; Bentivogli et al. 2009) Recognizing Textual Entailment comes from a series of annual textual entailment challenges. **SST-2** (Socher et al. 2013) Stanford Sentiment Treebank is to predict the polarity of a given sentence. **MRPC** (Dolan and Brockett 2005) Microsoft Research Paraphrase Corpus is to predict whether the pairwise sentences are semantically equivalent. We use the same data split as in (Wang et al. 2019). The accuracy metric is used for evaluation. When fine-tuning BERT, we follow the settings and the hyper-parameters suggested in (Devlin et al. 2019).

Effectiveness Analysis

We conduct a quantitative analysis to justify the self-attention edges with larger attribution scores contribute more to the model decision. We prune the attention heads incrementally in each layer according to their attribution scores with respect to the golden label and record the performance change. We also establish a baseline that prunes attention heads with their average attention scores for comparison.

Experimental results are presented in Figure 2, we observe that pruning heads with attributions scores conduces more salient changes on the performance. Pruning only two heads within every layer with the top-2 attribution scores can cause an extreme decrease in the model accuracy. In contrast, retaining them helps the model to achieve nearly 97% accuracy. Even if only two heads are retained in each layer, the model can still have a strong performance. Compared with attribution scores, pruning heads with average attention scores are less remarkable on the performance change, which proves the effectiveness of our method.

Use Case 1: Attention Head Pruning

According to the previous section, only a small part of attention heads contribute to the final prediction, while others are less helpful. This leads us to the research about identifying and pruning the unimportant attention heads.

Head Importance The attribution scores indicate how much a self-attention edge attributes to the final model decision. We define the importance of an attention head as:

$$I_h = E_x[\max(\mathrm{Attr}_h(A))] \tag{5}$$

where x represents the examples sampled from the held-out set, and $\max(\mathrm{Attr}_h(A))$ is the maximum attribution value of the h-th attention head. Notice that the attribution value of a head is computed with respect to the probability of the golden label on a held-out set.

We compare our method with other importance metrics based on the accuracy difference and the Taylor expansion, which are both proposed in (Michel, Levy, and Neubig 2019). The accuracy difference of an attention head is the accuracy margin before and after pruning the head. The method based on the Taylor expansion defines the importance of an attention head as:

$$I_h = E_x \left| A_h^\intercal \frac{\partial \mathcal{L}(x)}{\partial A_h} \right| \tag{6}$$

where $\mathcal{L}(x)$ is the loss function of example x, and A_h is the attention score of the h-th head as in Equation (2).

For all three methods, we calculate I_h on 200 examples sampled from the held-out dataset. Then we sort all the heads according to the importance metrics. The less important heads are first pruned.

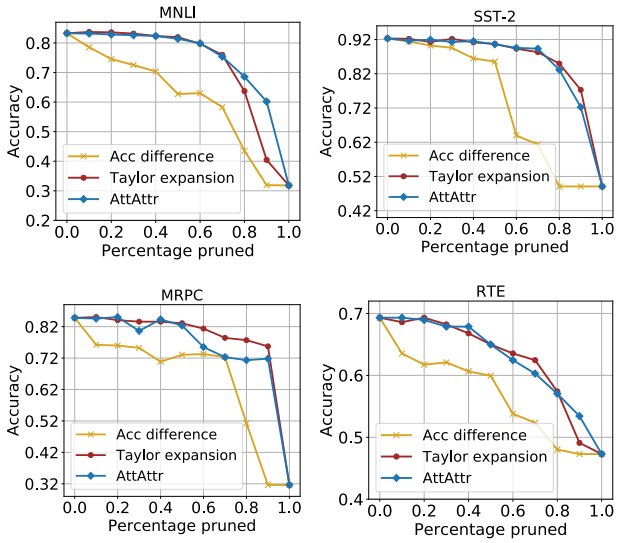


Figure 3: Evaluation accuracy as a function of head pruning proportion. The attention heads are pruned according to the accuracy difference (baseline; dash yellow), the Taylor expansion method (Michel, Levy, and Neubig 2019; solid red), and ATTATTR (this work; solid blue).

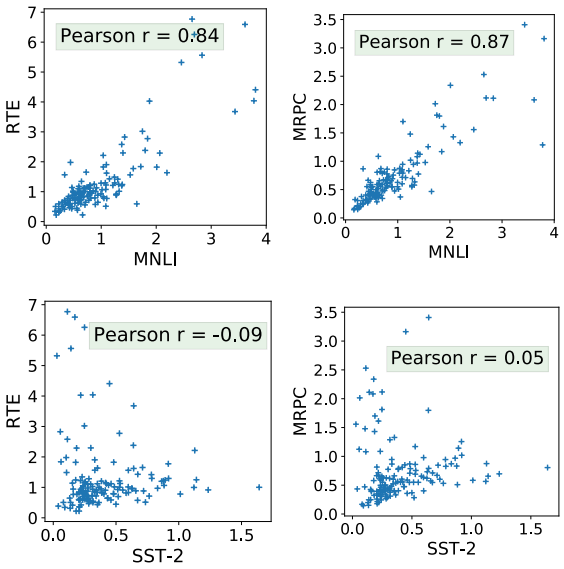


Figure 4: Correlation of attribution scores of different attention heads between datasets. Each point represents the attribution scores of a single attention head on two datasets.

Evaluation Results of Head Pruning Figure 3 describes the evaluation results of head pruning. The solid red lines represent pruning heads based on our method ATTATTR. We observe that pruning head with attribution score is much better than the baseline of accuracy difference.

Moreover, the pruning performance of ATTATTR is competitive with the Taylor expansion method, although ATTATTR is not specifically designed for attention head pruning. The result show that attention attribution successfully indicates the importance of interactions inside Transformer. On the MNLI dataset, when only 10% attention heads are retained, our method can still achieve approximately 60% accuracy, while the accuracy of the Taylor expansion method is about 40%.

Universality of Important Heads Previous results are performed on specific datasets respectively. Besides identifying the most important heads of Transformer, we investigate whether the important heads are consistent across different datasets and tasks. The correlation of attribution scores of attention heads between two different datasets is measured by the Pearson coefficient. As described in Figure 4, as long as the tasks are homogeneous (i.e., solving similar problems), the important attention heads are highly correlated. The datasets RTE, MPRC, and MNLI are about entailment detection, where the important self-attention heads (i.e., with large attribution scores) of BERT are roughly consistent across the datasets. In contrast, the dataset SST-2 is sentiment classification. We find that the important heads on SST-2 are different from the ones on RTE, and MRPC. In conclusion, the same subset of attention heads is fine-tuned for similar tasks.

Use Case 2: Visualizing Information Flow Inside Transformer

We propose a heuristic algorithm to construct attribution trees, the results discover the information flow inside Transformer, so that we can know the interactions between the input words and how they attribute to the final prediction. Such visualization can provide insights to understand what dependencies Transformer tends to capture. The post-interpretation helps us to debug models and training data.

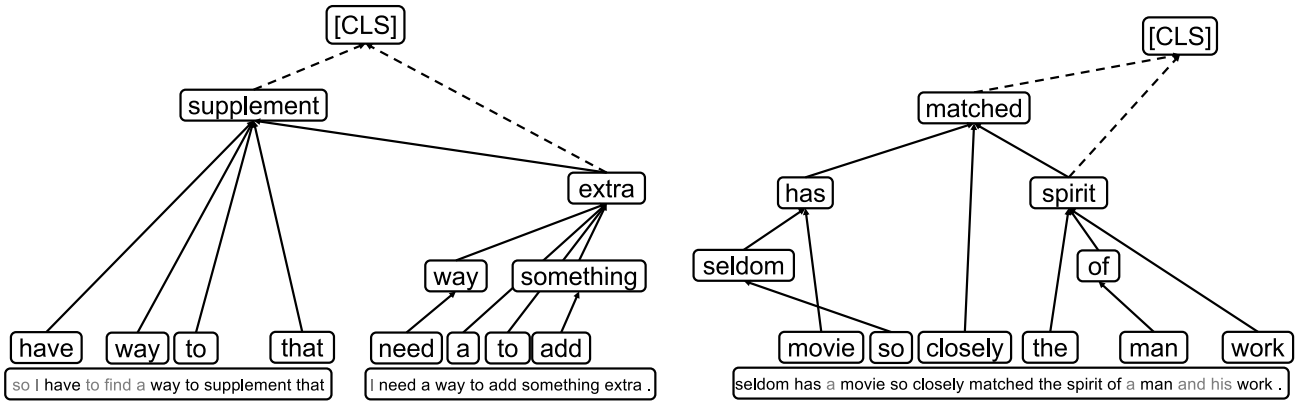
The problem is a trade-off between maximizing the summation of attribution scores and minimizing the number of edges in the tree. We present a greedy top-down algorithm to efficiently construct attribution trees. Moreover, we conduct a quantitative analysis to verify the effectiveness.

Attribution Tree Construction After computing self-attention attribution scores, we can know the interactions between the input words in each layer and how they attribute to the final prediction. We then propose an attribution tree construction algorithm to aggregate the interactions. In other words, we build a tree to indicate how information flows from input tokens to the final predictions. We argue that such visualization can provide insights to understand what dependencies Transformer tends to capture.

For each layer l, we first calculate self-attention attribution scores of different heads. Then we sum them up over the heads, and use the results as the l-th layer's attribution:

$$\text{Attr}(A^l) = \sum_{h=1}^{|h|} \text{Attr}_h(A^l) = [a^l_{i,j}]_{n\times n}$$

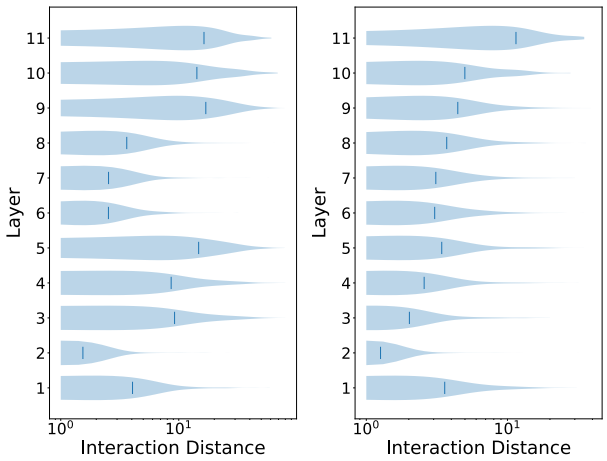
where larger $a^l_{i,j}$ indicates more interaction between x_i and x_j in the l-th layer in terms of the final model predictions.



(a) Example from MNLI

(b) Example from SST-2

Figure 5: Examples of attribution trees. (a) is from MNLI, which is predicted as `entailment` by BERT. (b) is from SST-2, which is predicted as `positive` by BERT. The grey words from the inputs do not appear in the attribution trees.



(a) MNLI

(b) SST-2

Figure 6: Distance distribution of interactions extracted by the attribution tree in each layers.

The construction of attribution trees is a trade-off between maximizing the summation of attribution scores and minimizing the number of edges in the tree. The objective is defined as:

$$\text{Tree} = \arg\max_{\{E^l\}_{l=1}^{|l|}} \sum_{l=1}^{|l|} \sum_{(i,j)\in E^l} a_{i,j}^l - \lambda \sum_{l=1}^{|l|} |E^l|,$$

$$E^l \subset \{(i,j) | \frac{a_{i,j}^l}{\max(\text{Attr}(A^l))} > \tau\}$$

where $|E^l|$ represents the number of edges in the l-th layer, λ is a trade-off weight, and the threshold τ is used to filter the interactions with relatively large attribution scores.

Rather than solving a combinatorial optimization problem, we use a heuristic top-down method to add these edges to the attribution tree. The process is detailed in Algorithm 1.

Settings We set $\tau = 0.4$ for layers $l < 12$. The larger τ tends to generate more simplified trees, which contains the more important part of the information flow. Because the special token `[CLS]` is the terminal of the information flow for classification tasks, we set τ to 0 for the last layer. We observe that almost all connections between `[CLS]` and other tokens in the last layer have positive attribution scores with respect to model predictions.

Case Studies As shown in Figure 5, the two attribution trees are from MNLI and SST-2, respectively. The attribution tree Figure 5a is generated from MNLI, whose golden label is `entailment`. At the bottom of Figure 5a, we find that the interactions are more local, and most information flows are concentrated within a single sentence. The information is hierarchically aggregated to "*supplement*" and "*extra*" in each sentence. Then the "*supplement*" token aggregates the information in the first sentence and "*add something extra*" in the second sentence, these two parts "*supplement*" and "*add something extra*" have strong semantic relevance. Finally, all the information flows to the terminal token `[CLS]` to make the prediction `entailment`. The attribution tree interprets how the input words interacts with each other, and reach the final prediction, which makes model decisions more interpretable.

Figure 5b is an example from SST-2, whose golden label is `positive`, correctly predicted by the model. From Figure 5b, we observe that information in the first part of the sentence "*seldom has a movie so closely*" is aggregated to the "*has*" token. Similarly, information in the other part of the sentence "*the spirit of a man and his work*" flows to the "*spirit*" token, which has strong positive emotional tendencies. Finally, with the feature interactions, all information aggregates to the verb "*matched*", which gives us a better understanding of why the model makes the specific decision.

Receptive Field The self-attention mechanism is supposed to have the ability to capture long-range dependen-

Algorithm 1 Attribution Tree Construction

Input: $[a^l_{i,j}]_{n\times n}$: Attribution scores
$\{E^l\}_{l=1}^{|l|}$: Retained attribution edges
Output: $\mathcal{V}, \mathcal{E}$: Node set and edge set of Attr tree

▷ *Initialize the state of all tokens*
for $i \leftarrow n, \cdots, 1$ **do**
  $\text{State}_i \leftarrow$ `NotAppear`
▷ *Choose the top node of the attribution tree*
$[AttrAll_i]_n = \sum_{l=1}^{|l|} \sum_{j=1, j\neq i}^{n} a^l_{i,j}$
$TopNode = \arg\max([AttrAll_i]_n)$
$\mathcal{V} \leftarrow \{TopNode\}$; $\text{State}_{TopNode} \leftarrow$ `Appear`
▷ *Build the attribution tree downward*
for $l \leftarrow |l|-1, \cdots, 1$ **do**
  for $(i,j)^l_{i\neq j} \in E^l$ **do**
    if State_i is `Appear` **and**
      State_j is `NotAppear` **then**
      $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{(i,j)\}$; $\mathcal{V} \leftarrow \mathcal{V} \bigcup \{j\}$
      $\text{State}_i \leftarrow$ `Fixed`; $\text{State}_j \leftarrow$ `Appear`
    if State_i is `Fixed` **and**
      State_j is `NotAppear` **then**
      $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{(i,j)\}$; $\mathcal{V} \leftarrow \mathcal{V} \bigcup \{j\}$
      $\text{State}_j \leftarrow$ `Appear`
▷ *Add the terminal of the information flow*
$\mathcal{V} \leftarrow \{$`[CLS]`$\}$
for $j \leftarrow n, \cdots, 1$ **do**
  if $\text{State}_j \in \{$`Appear`, `Fixed`$\}$ **then**
    $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{($`[CLS]`$, j)\}$
return $\{\mathcal{V}, \mathcal{E}\}$

cies. In order to better understand the layer-wise effective receptive field in Transformer, we plot the distance distribution of interactions extracted by the attribution tree. As shown in Figure 6, we observe that for the paired input of MNLI, the effective receptive field is relatively local in the first two layers and the 6-8th layers, while are more broad in the top three layers. For the single input of SST-2, the effective receptive field is monotonically increasing along with the layer number. Generally, the effective receptive field in the second layer is more restricted, while the latter layers extract more broad dependencies. Moreover, for pairwise-input tasks (such as MNLI), the results indicate that Transformer models tend to first conduct local encoding and then learn to match between the pair of input sentences, which is different with training from scratch (Bao et al. 2019).

Use Case 3: Adversarial Attack

The model decision attributes more to the attention connections with larger attribution scores. We observe that the model tends to over-emphasize some individual patterns to make the prediction, while omitting most of the input. We then use the over-confident patterns as adversarial triggers (Wallace et al. 2019) to attack the BERT model.

Trigger Construction We extract the attention dependencies with the largest attribution scores across different layers (i.e., $\max_{l=1}^{L} \{a^l_{i,j}\}$) from the input, and employ these patterns as the adversarial triggers. During the attack, the adversarial triggers are inserted into the test input at the same relative position and segment as in the original sentence.

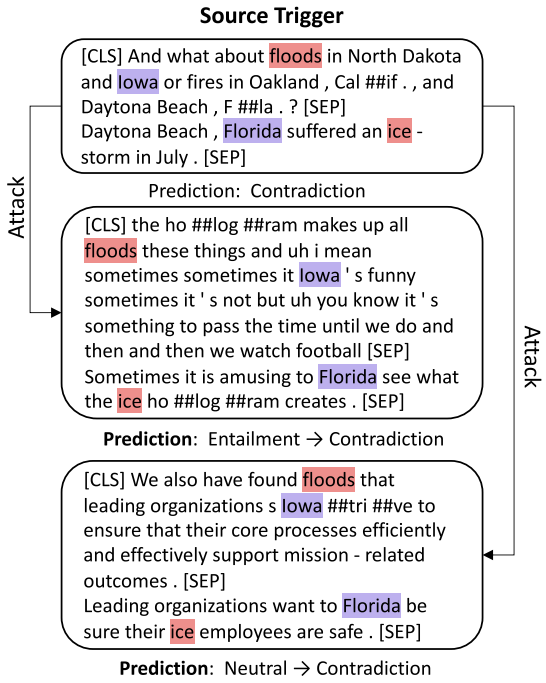


Figure 7: We use ATTATTR to extract the trigger (i.e., highlighted word patterns) from the MNLI instance that is labeled as `contradict`. After adding the adversarial trigger to the examples in other categories, the model predictions flip from `neutral` and `entailment` to `contradict`.

The specific attack process is shown in Figure 7. The two patterns "*floods-ice*" and "*Iowa-Florida*" contribute most to the prediction `contradict` in the source sentence. Next we employ them as the trigger to attack other examples, the model predictions flip from both `neutral` and `entailment` to `contradict`. Our attack method relies on attribution scores, which utilizes the gradient information, therefore it belongs to white-box non-targeted attacks.

We extract the dependencies with the largest attribution scores as the adversarial triggers from 3,000 input examples. Each trigger contains less than five tokens. The score of a trigger is defined as the maximum attribution value identified within it. When attacking the BERT model on SST-2, we use a lexicon[1] to blacklist the words with the obvious emotional tendencies (such as "*disgust*" for `negative` triggers). The adversarial triggers are inserted into the attack text at the same relative position as in the original sentence.

Results of Attack We conduct the adversarial attacks on multiple datasets. The top-3 adversarial triggers for MNLI and SST-2 are listed in Table 1. We report the attack results with these triggers in Table 2. For MNLI, after inserting the

[1] www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

	MNLI			SST-2	
	contradict	entailment	neutral	positive	negative
Trigger1	{also, sometimes, S}	{with, math}	{floods, Iowa, ice, Florida}	{[CLS], nowhere}	{remove, ##fies}
Trigger2	{nobody, should, not}	{light, morning}	{never, but}	{but, has, nothing}	{not, alien, ##ate}
Trigger3	{do, well, Usually, but}	{floods, Iowa, ice, Florida}	{Massachusetts, Mexico}	{offers, little}	{##reshing, ##ly}

Table 1: Top-3 adversarial triggers for the MNLI and SST-2 datasets. The tokens are inserted into input sentences at the specific positions for non-targeted attacks. We omit the tokens' positions in the table for brevity.

	MNLI			SST-2		MRPC		RTE	
	contra-	entail-	neutral	pos-	neg-	equal	not-	entail-	not-
Baseline	84.94	82.87	82.00	92.79	91.82	90.32	72.87	72.60	65.65
Trigger1	34.17	0.80	34.77	54.95	72.20	29.39	51.94	9.59	59.54
Trigger2	39.81	1.83	47.36	59.68	74.53	32.62	55.04	11.64	62.50
Trigger3	41.83	2.99	51.49	70.50	77.80	36.56	58.91	13.70	62.60
Avg. Δ	-46.34	-80.00	-37.46	-31.08	-16.98	-57.46	-17.57	-60.96	-12.31

Table 2: Attack results of the top-3 triggers. We abbreviate `not equal` and `not entailment` to `not-` for MRPC and RTE, respectively. The baseline represents the original accuracy of model on each category.

words ("*with*", and "*math*") to the second segment of the input sentences, the model accuracy of the `entailment` class drops from 82.87% to 0.8%. For SST-2, adding the top-1 adversarial trigger to the input causes nearly 50% `positive` examples to be misclassified.

Analysis of Triggers For both MNLI and RTE, the `entailment` class is more vulnerable than others, because the current models and data seem to heavily rely on word matching, which would result in spurious patterns. Moreover, we also observe that the trigger is sensitive to the insertion order and the relative position in the sentence, which exhibits the anomalous behaviors of the model, i.e., overrelying on these adversarial triggers to make the prediction.

Related Work

Previous work has explored attributing predictions to the input features with various saliency measures, such as DeepLift (Shrikumar, Greenside, and Kundaje 2017), layerwise relevance propagation (Binder et al. 2016), and Integrated Gradients (IG; Sundararajan, Taly, and Yan 2017).

Specific to the NLP domain, Murdoch and Szlam (2017) introduce a decomposition method to track the word importance in LSTM (Hochreiter and Schmidhuber 1997). Murdoch, Liu, and Yu (2018) extend the above method to contextual decomposition in order to capture the contributions of word combinations. Another strand of previous work generates the hierarchical explanations, which aims at revealing how the features are composed together (Jin et al. 2020; Chen, Zheng, and Ji 2020). However, they both detect interaction within contiguous chunk of input tokens. The attention mechanism (Bahdanau, Cho, and Bengio 2015) rises another line of work. The attention weights generated from the model indicate the dependency between two words intuitively, but Jain and Wallace (2019) and Serrano and Smith (2019) draw the same conclusion that they largely do not provide meaningful explanations for model predictions. However, Wiegreffe and Pinter (2019) propose several alternative tests and conclude that prior work does not disprove the usefulness of attention mechanisms for interpretability. Furthermore, Ghaeini, Fern, and Tadepalli (2018) aim at interpreting the intermediate layers of NLI models by visualizing the saliency of attention and LSTM gating signals.

For Transformer, Clark et al. (2019) propose a attention-based visualization method and a probing classifier to explain the behaviors of BERT (Devlin et al. 2019). Brunner et al. (2020) study the identifiability of attention weights of BERT, which shows that self-attention distributions are not directly interpretable. Moreover, some work extracts the latent syntactic trees from hidden representations (Hewitt and Manning 2019; Rosa and Marecek 2019; Coenen et al. 2019) and attention weights (Marecek and Rosa 2019).

Conclusion

We propose self-attention attribution (ATTATTR), which interprets the information interactions inside Transformer and makes the self-attention mechanism more explainable. First, we conduct a quantitative analysis to justify the effectiveness of ATTATTR. Moreover, we use the proposed method to identify the most important attention heads, which leads to a new head pruning algorithm. We then use the attribution scores to derive the interaction trees, which visualizes the information flow of Transformer. We also understand the receptive field in Transformer. Finally, we show that ATTATTR can also be employed to construct adversarial triggers to implement non-targeted attacks.

Acknowledgements

The work was partially supported by National Natural Science Foundation of China (NSFC) [Grant No. 61421003].

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bao, H.; Dong, L.; Wei, F.; Wang, W.; Yang, N.; Cui, L.; Piao, S.; and Zhou, M. 2019. Inspecting Unification of Encoding and Matching with Transformer: A Case Study of Machine Reading Comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, 14–18. Association for Computational Linguistics.

Bao, H.; Dong, L.; Wei, F.; Wang, W.; Yang, N.; Liu, X.; Wang, Y.; Piao, S.; Gao, J.; Zhou, M.; and Hon, H.-W. 2020. UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training. *arXiv preprint arXiv:2002.12804* .

Bar-Haim, R.; Dagan, I.; Dolan, B.; Ferro, L.; and Giampiccolo, D. 2006. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Bentivogli, L.; Dagan, I.; Dang, H. T.; Giampiccolo, D.; and Magnini, B. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *In Proc Text Analysis Conference*.

Binder, A.; Montavon, G.; Bach, S.; Müller, K.; and Samek, W. 2016. Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers. *CoRR* abs/1604.00825.

Brunner, G.; Liu, Y.; Pascual, D.; Richter, O.; Ciaramita, M.; and Wattenhofer, R. 2020. On Identifiability in Transformers. In *International Conference on Learning Representations*.

Chen, H.; Zheng, G.; and Ji, Y. 2020. Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection. In *ACL*.

Chi, Z.; Dong, L.; Wei, F.; Wang, W.; Mao, X.; and Huang, H. 2020a. Cross-Lingual Natural Language Generation via Pre-Training. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 7570–7577. AAAI Press.

Chi, Z.; Dong, L.; Wei, F.; Yang, N.; Singhal, S.; Wang, W.; Song, X.; Mao, X.; Huang, H.; and Zhou, M. 2020b. InfoXLM: An Information-Theoretic Framework for Cross-Lingual Language Model Pre-Training. *CoRR* abs/2007.07834.

Clark, K.; Khandelwal, U.; Levy, O.; and Manning, C. D. 2019. What Does BERT Look At? An Analysis of BERT's Attention. *CoRR* abs/1906.04341.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*.

Coenen, A.; Reif, E.; Yuan, A.; Kim, B.; Pearce, A.; Viégas, F. B.; and Wattenberg, M. 2019. Visualizing and Measuring the Geometry of BERT. *CoRR* abs/1906.02715.

Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; and Stoyanov, V. 2020. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–8451. Association for Computational Linguistics.

Dagan, I.; Glickman, O.; and Magnini, B. 2006. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, 177–190. Berlin, Heidelberg: Springer-Verlag. ISBN 3-540-33427-0, 978-3-540-33427-9.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Dolan, W. B.; and Brockett, C. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*.

Ghaeini, R.; Fern, X. Z.; and Tadepalli, P. 2018. Interpreting Recurrent and Attention-Based Neural Models: a Case Study on Natural Language Inference. *CoRR* abs/1808.03894.

Giampiccolo, D.; Magnini, B.; Dagan, I.; and Dolan, B. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 1–9. Prague: Association for Computational Linguistics.

Hewitt, J.; and Manning, C. D. 2019. A Structural Probe for Finding Syntax in Word Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 4129–4138. Minneapolis, Minnesota: Association for Computational Linguistics.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9: 1735–1780. ISSN 0899-7667.

Jain, S.; and Wallace, B. C. 2019. Attention is not Explanation. *CoRR* abs/1902.10186.

Jin, X.; Wei, Z.; Du, J.; Xue, X.; and Ren, X. 2020. Towards Hierarchical Importance Attribution: Explaining Compositional Semantics for Neural Sequence Models. In *ICLR*.

Kovaleva, O.; Romanov, A.; Rogers, A.; and Rumshisky, A. 2019. Revealing the Dark Secrets of BERT. 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)*, 4364–4373. Hong Kong, China: Association for Computational Linguistics.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* .

Marecek, D.; and Rosa, R. 2019. From Balustrades to Pierre Vinken: Looking for Syntax in Transformer Self-Attentions. *CoRR* abs/1906.01958.

Michel, P.; Levy, O.; and Neubig, G. 2019. Are Sixteen Heads Really Better than One? *CoRR* abs/1905.10650.

Murdoch, W. J.; Liu, P. J.; and Yu, B. 2018. Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs. *CoRR* abs/1801.05453.

Murdoch, W. J.; and Szlam, A. 2017. Automatic Rule Extraction from Long Short Term Memory Networks. *CoRR* abs/1702.02540.

Rosa, R.; and Marecek, D. 2019. Inducing Syntactic Trees from BERT Representations. *CoRR* abs/1906.11511.

Serrano, S.; and Smith, N. A. 2019. Is Attention Interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2931–2951. Florence, Italy: Association for Computational Linguistics.

Shrikumar, A.; Greenside, P.; and Kundaje, A. 2017. Learning Important Features Through Propagating Activation Differences. *CoRR* abs/1704.02685.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642. Seattle, Washington, USA: Association for Computational Linguistics.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic Attribution for Deep Networks. *CoRR* abs/1703.01365.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, 5998–6008. Curran Associates, Inc.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. 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)*, 2153–2162. Hong Kong, China: Association for Computational Linguistics.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*.

Wiegreffe, S.; and Pinter, Y. 2019. Attention is not not Explanation. In *EMNLP-IJCNLP*, 11–20. Hong Kong, China: Association for Computational Linguistics.

Williams, A.; Nangia, N.; and Bowman, S. 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)*, 1112–1122. New Orleans, Louisiana: Association for Computational Linguistics.