Exploration of Visualizations for Interpreting Longformer Model's Predictions

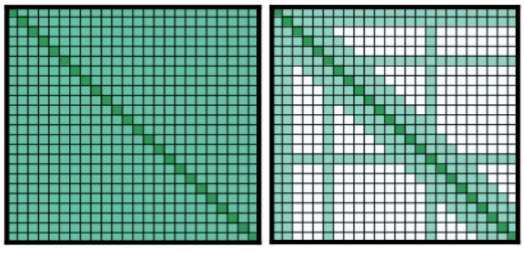
Daniel Hou
Cogs 402
danielhou13@gmail.com

Abstract

Transformers achieved state-of-the-art results on a variety of NLP tasks, but are unable to process long sequences due to their self-attention being quadratic in complexity. The Longformer is a Transformer model designed to deal with long sequences due to a modified self-attention mechanism and is linear in complexity. The Longformer's unique attention mechanism can process text with thousands of tokens. Unlike traditional Transformer Networks, visualizations for interpreting Longformers have not been explored to nearly the same robustness. In this paper, we explore methods of visualizing Longformer model predictions by analyzing its attention and salience. Our methods provide effective visualizations for identifying keywords that explain the model's prediction when masking out non-alpha tokens and stopwords. Furthermore, we find that our interpretations have high correlations with known faithful and plausible methods of explaining model predictions. The code is available at https://github.com/danielhou13/cogs402longformer

1 Introduction

Transformer Neural Networks are the leading architecture for Natural Language Processing (NLP) tasks [Vaswani et al., 2017]. Their success is attributed to the self-attention mechanism that allows each token to attend to all tokens, allowing the model to capture features from entire sequences. How much a token attends to another is represented by an attention weight or score. For each token, this weight divided across all tokens adds to 1, with some tokens getting more attention than others, similar to how human attention is divided. [1]



(a) Full n^2 attention (b) Sliding+Global attention

Figure 1: A comparison of the full n^2 self-attention matrix and the Longformer's global+sliding window attention matrix

[1] A quick detailed summary of self-attention mechanism can be found in Appendix A.1

While effective for many NLP tasks, the Transformer's self-attention is expensive in terms of memory and runtime ($O(n^2)$). Many Transformer models, such as BERT, have a limit of 512 tokens for this reason [Devlin et al., 2018]. The Longformer Model has a unique attention output designed to handle long sequences of text [Beltagy et al., 2020]. Unlike traditional Transformer models, the Longformer model does not have attention between all pairs of tokens, but rather, each token can only attend to the succeding $w/2$ tokens, the preceding $w/2$, and itself. They call this the sliding window attention. A small number of tokens are able to attend to all tokens, which they call global attention (Fig 1). Global attention is primarily for task-specific fine-tuning as it allows the model to better represent the task. For classification, only the opening [CLS] token uses global attention, keeping the memory and computational requirements low ($O(n)$). These augmentations allow the Longformer to process a maximum of 4096 tokens (8x of BERT) [Devlin et al., 2018]

There have been many different methods of interpreting Transformer model predictions. These methods usually look at the Transformer's attention weights to identify what features are learned [Mullenbach et al., 2018, Xie et al., 2017] or use saliency methods to attribute importance to input features [Sundararajan et al., 2017, Chrysostomou and Aletras, 2021]. Furthermore, many visualization frameworks have been created to support the understanding of Transformers [Vig, 2019, Li et al., 2021, 2022b], but lack of scalability to handle thousands of input tokens makes it difficult to visualize long-document tasks.

Our goal is to create and adapt visualizations able to support the understanding of the Longformer model using both attention and saliency methods. We fine-tune a Longformer model for a binary classification task on custom datasets and interpret the model's prediction. Furthermore, we evaluate the attention interpretations to determine their faithfulness (explains the inner workings of the model) and plausibility (agree with human judgment) as a method of explaining the model's prediction [Jacovi and Goldberg, 2020].

2 Related Works

Long-document classification Prior to the Longformer, the leading architecture for long-document classification was Recurrent Neural Networks (RNNs). For example, He et al. [2019] designed an RNN able to locate the most discriminative groups of words for their task. However, the biggest drawback was splitting the full text into blocks and individually examining each piece before moving on to the next block. When compared to Transformers which are able to process the entire text at once, RNNs cannot capture the same long-range dependencies that Transformers can. [Vaswani et al., 2017] With the advent of the Longformer and other long-document Transformers, fields like clinical and biomedical, where the document length is often longer than the limits of models like BERT, are slowly adapting to use these long-document Transformers [Li et al., 2022a].

Transformer visualization There have been many visualization frameworks that help interpret the model attention such as Bertviz and Tensor2Tensor frameworks [Vig, 2019, Vaswani et al., 2018]. Bertviz takes an in-depth look at the attention between tokens allowing for the identification of syntactic relations such as which layers specialize in which parts-of-speech tags [Vig and Belinkov, 2019]. While BertViz does not help interpret the model's prediction, it works well to identify how the attention is structured in a Transformer model. Frameworks such as T3-vis and DeepNLPVis present a detailed framework for visualizing model behaviors through the use of saliency methods and assessing model attention but struggle with long documents due to limitations of token size on memory and runtime [Li et al., 2021, 2022b]. However, they are effective for visualizing models suited for low-text documents such as BERT [Devlin et al., 2018].

Attention as explanation Many papers have discussed what goes into making good interpretations for Neural Network model predictions, but the consensus is that the explanations must be faithful and plausible [Jacovi and Goldberg, 2020]. Attributions have been proven to be a faithful and plausible method of interpretation [Ding and Koehn, 2021, Chrysostomou and Aletras, 2021]; however, there is a huge debate whether attention meets the same requirements [Bibal et al., 2022]. A potential question is "why care about visualizing attention when saliency methods already provide an explanation?" [Bastings and Filippova, 2020, Thorne et al., 2019]. A few answers have already addressed this question, but the most relevant is that attention is something that is part of the model's architecture and is learned, so being able to use it as an explanation is more useful than using post-hoc tools such

as saliency methods. [Bibal et al., 2022]. As such, it is valuable to consider attention even if you use attributions.

3 Methods

We fine-tune and train a Longformer model to perform binary text classification and explore our model's predictions in two different ways. The first way is to analyze the model's sliding and global attention outputs. Specifically, we aim to gain a model-wide understanding of the Longformer's attention pattern as well as take an in-depth look at which tokens the model gives the most attention to. The difficulty of this task is that the Longformer Model, with 12 hidden layers and 12 attention heads, generates $12 \times 12 = 144$ unique sliding window attention and global attention matrices for each input example.

The second way is to analyze the model's saliency with respect to an input example. Saliency methods give an explanation for a model's predictions by assigning importance scores, also called attribution scores, to each feature in our input. In the case of text classification, each token in our input represents a feature. There are many different saliency methods, but for this paper, we use Integrated Gradients [Sundararajan et al., 2017]. Integrated Gradients is a faithful and plausible method of explaining the model's prediction [Ding and Koehn, 2021]. The formula for Integrated Gradients is as follows:

$$\text{IntegratedGrads}_i(x) ::= (x_i - x_i') \cdot \int_{\alpha=0}^{1} \frac{\partial F(x' + \alpha \cdot (x - x'))}{\partial x_i} d\alpha \tag{1}$$

To summarize, Integrated Gradients a baseline input sequence x' and our input sequence x, integrates the gradients between x' and x for feature i, and repeats for all features in x. For each feature, it returns a list of attribution scores of the same size as the model's embedding size. They sum up all the values in their respective list and normalize the result to get the final attribution score for a token. Repeating this process for every token, we get a list of attribution scores where the score at index i represents token i in our example. The attribution score is more precise the higher the number of steps taken to approximate the integral; however, the time taken will increase as well. Attribution scores can be positive or negative, and the higher the magnitude of the score is, the more strongly the token influences the model into predicting positive or negative respectively.

Lastly, we wish to confirm whether attention scores are comparable to attribution. The idea is that the tokens with the highest attention may also have the largest attributions (in terms of magnitude); therefore, may show high degrees of similarity.

3.1 Attention Visualization

Unlike other Transformer models, the Longformer has two attention outputs: the sliding window attention and the global attention. We concatenate both attention outputs and convert to a matrix of shape (sequence_len, sequence_len) for each head and layer. This allows us to easily gather attention weights between tokens as every token now has an attention weight/score for all tokens in the example rather than the succeeding and preceding $w/2$ tokens.

Scaling Attentions by Head Importance The multi-headed attention mechanism allows the Longformer Model to learn multiple different dependencies between tokens; however, the importance of each head in the Longformer is dependent on the task performed. As such, we adopt the method used by Molchanov et al. [2019] and Li et al. [2021] to scale each attention head using Taylor Expansion.

3.1.1 Attention Heatmap Visualization

We visualize the attention for each head and layer by converting each normalized attention matrix into a 2D heatmap where the saturation of the colour represents the magnitude of the attention between the associated tokens, replicating the implementation by [Li et al., 2021]. Given a large input sequence and 144 attention matrices, it is unfeasible to use this plot to identify which tokens seem to attend to which other tokens the most; however, we can identify patterns in each attention matrix and see how they change between attention heads or evolve in deeper layers. We can obtain these heatmaps

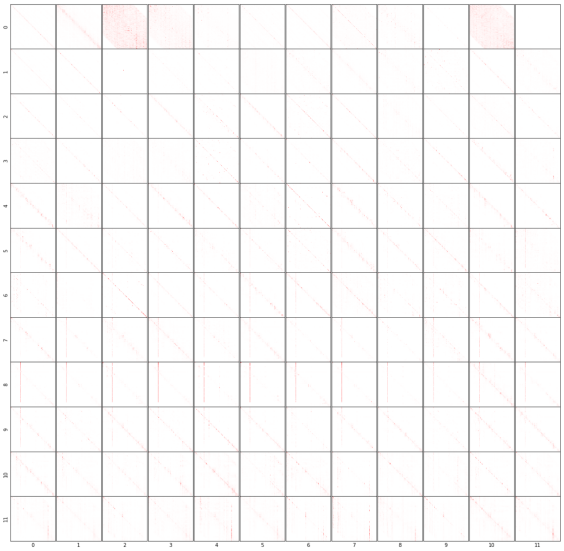
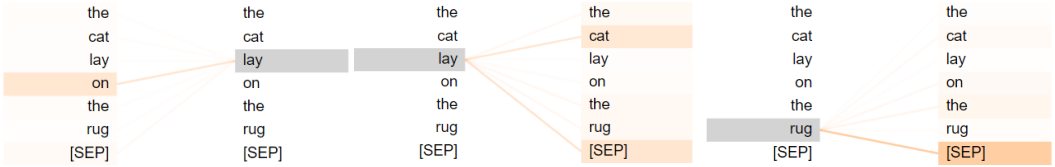


Figure 2: An example of the 2D attention heatmap over all layers (y-axis) and all heads (x-axis)

on two scales, example-wide and dataset-wide, where the dataset wide aggregates and averages the attention for all examples.

3.1.2 In-depth Analysis of Attentions Between Tokens

As mentioned, it is unfeasible to use an attention plot to identify relations between two tokens so supporting visualizations would be needed. One method is to identify the tokens that the model deems the most interesting, the ones with the most attention.



(a) Tokens attending to "lay" (b) Tokens attended by "lay" (c) Tokens attended by "rug"

Figure 3: The "attended to" and "attending" distinction consistent with our methodology

Attention is two-fold; tokens are attending to and attended to by other tokens. A token attending to another token means that the information from the other token is collected, creating attention weights and allowing the model to create associations between the attending token and the other token. Different tokens attending to the same tokens can have different attention weights, so stronger or weaker associations are created. For instance, the token "on" is most associated with the token "lay" and has higher attention weights than the tokens "rug" and "lay" (Fig 3a).

The "attended to" tokens receive attention scores from the attending tokens. Over all "attended to" tokens, the total attention weights add to 1; however, how the attention weights are distributed is dependent on the attending token. For instance, "cat" receives high attention from the token "lay", but does not receive high attention from "rug" (Fig 3b and 3c).

The top-k "attended to" tokens Since some tokens will receive more attention than others, we wish to find out which tokens consistently receive the most attention in our input. For instance, the tokens "cat" and [SEP] are the most attended to by token "lay" but [SEP] is also highly attended to by rug, suggesting a token of interest (Fig 3a). As such, we find, for every token x in our input sequence, the group of k tokens that token x attends to the most. We call this group of k tokens the "attended tokens" as these tokens are being *attended* to the most by x. To get these tokens, we perform an argsort on the token's attention scores to get the indices (position in the input) of the

highest k attention scores. We repeat this process for every layer and head [2]. Having the positions of the top-k tokens allows easy access to the attention scores and the corresponding token, which we store in a Pandas dataframe.

Groups of top-k attended tokens can have tokens in agreement meaning that multiple tokens may give high attentions to the same tokens. These are tokens of interest as it suggests these tokens gather a lot of attention. For every input token, we check their top-k attended and get the set of tokens. We count how many times each token appears in a token's top-k attended. [3] The more often a particular token appears, the more interesting the model thinks the token is, as the token is at the center of attention. We can apply this locally, within a single layer or head, or over all of them. On the other hand, a token with multiple instances throughout the text will have more tokens attending to it, increasing the probability that the token appears in a top-k attended. This may cause the count to be inflated. As such, we run through all the tokens again, but instead of counting how many times a token appears in a top-k (by token type), we count by the token's position or the specific instance of a token.

The top-k "attending" tokens We also find the top-k tokens that are attending to an arbitrary token x. [4] Doing so allows us to identify which tokens are giving x the most attention and are associating themselves the strongest. For instance, the token "on" is giving token "lay" the most attention out of all the tokens (Fig. 3b). We appropriately call this group of k tokens the "attending tokens" as these tokens are *attending* to x. We apply the methods used for the attended tokens now on the attending tokens. [5] A higher count suggests that the particular token is notable, but rather than the token being in the center of attention, it seems to be giving the most attention.

The natural step after obtaining these sets is to find the intersection. This tells us which tokens and instances both gather lots of attention, and do a lot of attending.

3.1.3 Total Attention on a Given Word

While we can get an idea of what tokens have the highest attention based on which tokens are attended to the most by other tokens, we can also get the summed attention weights of a token x by adding up the attention weights for all tokens that attend to x. [6] The amount of attention a token gets from other tokens does not necessarily total 1, as it is dependent on the amount of attention this token receives from other tokens. The higher the amount, the more interesting the model believes the token is. We would not be able to tell how many tokens or which tokens consider this token to be of high interest, whether its a very small set of tokens with massive attention weights or many tokens with decent attention weights, but we will be able to tell if this token gathers a lot of attention.

Given summed attention weights for each token, we visualize these weights by mapping a colour that increases in saturation the higher the attention weight. We overlay the colour on top of the input text, creating a heatmap, allowing us to read the full text while getting an estimation of how much or little attention the model gives that token. We take advantage of a simple, but effective method implemented by Yang and Zhang [2018]. Furthermore, given that we have a full matrix for all layers and heads, we can isolate different layer configurations as desired.

3.2 Attribution Visualization

To get the attribution scores for each token, we use Captum's implementation of Integrated Gradients which incorporates the Gauss-Legendre method of integration [Kokhlikyan et al., 2020, Gauss, 2011]. We run for 500 steps, using the padding token as a baseline. This is a very large number of steps and causes the runtime to be upwards of five minutes per example. For the methods that require us to iterate through the entire dataset, we compromise to strike a balance between precision and runtime. We revert to 50 steps, which is a sufficient number to adequately capture the representation of each token over the entire dataset without taking days to run.

[2] See Algorithm 1 in Appendix A.5

[3] See Algorithm 2 in Appendix A.5

[4] See Algorithm 3 in Appendix A.5

[5] The implementation is the same as Algorithm 2 in Appendix A.5. We simply have to replace the top-k "attended tokens" with top-k "attending tokens"

[6] See Algorithm 4 in Appendix A.5

3.2.1 Attribution Heatmap

Using Captum's visualization tool, we create a heatmap that displays the attribution scores of each token similar to the attention overlay. [Kokhlikyan et al., 2020]. The attribution score for each input token is converted into a colour that increases in saturation the higher the magnitude of the score. Each token of the input is overlayed with its corresponding colour to represent its attribution score. To distinguish between positive and negative attributions, the positive attributions are set to green and the negative attributions are set to red.

3.2.2 An In-depth Look at Token Attributions

The attribution scores and their respective tokens are compiled to a Pandas dataframe that displays the attribution scores sorted from highest to lowest, the corresponding tokens, and the token's position in the input. To display the most negative attributions, the dataframe is re-sorted from lowest score to highest. Two identical tokens in different positions could have high positive and negative attributions, making it difficult to tell if it was positively or negatively influencing the prediction overall. Thus, attribution scores are also aggregated by their token, giving us a token's example-wide influence. We take it a step further by aggregating and averaging over an entire dataset, allowing us to identify which tokens were found to have consistently large positive or negative attribution scores.

3.3 Attentions vs Attributions

The prior methods demonstrate how we can use these attributions and attentions for model interpretability, but for this section, we assess the faithfulness and plausibility of the model's attention outputs by comparing the list of normalized attention weights for each token with their normalized absolute attribution scores. This gives us an idea of whether we can use the model's attention outputs as independent explanations, or if they are meant to support other methods such as attributions. Furthermore, these methods are most suitable for confirming whether scaling by head importance and masking is beneficial to the model. We compare these for two different attention outputs, the final hidden layer of the model before the linear classifier, and the aggregate attention over all layers of the model.

To compare the two, we use three metrics: Cosine Similarity, Kendall-Tau coefficient, and Rank-Biased-Overlap (RBO) [7]. We use cosine similarity to identify how similar the attention and attribution values are for all tokens in our input. This allows us to figure out whether the model's attention scores are comparable to the attribution scores. However, two tokens with similar attribution and attention scores could be ranked very differently depending on how the scores were distributed before it was normalized. The rank refers to how high the attention or attribution scores are compared to the scores of other tokens in their respective lists. This is important as not only do we want to know how similar the values of the attention and attribution scores are but also if the amount of attention the model places on a token accurately reflects the amount of influence the token has compared to other input tokens. As such, we use Kendall-Tau and RBO to calculate the similarity based on the rank of the token.

3.3.1 Selective Examination

We apply the above metrics over the entire list of attention and attribution scores; however, the value in examining tokens with below-average scores is much lower than tokens with high attention and/or attribution. It is much more interesting and valuable to examine the tokens with the highest attention or influence. We calculate the Jaccard Similarity to find out how agreeable the tokens in the top 5 percent of attention and attribution scores are[Jaccard, 1912]. We calculate this based on the position number of the token as we want to know based on the instance of the token, rather than the type of token. The Jaccard Similarity outputs 0 if there are no tokens that appear in the 95^{th} percentile of both lists, and 1 if the two lists are identical.

3.4 Masking Tokens

When examining the top-k tokens and the token relationships for attributions and attentions, unimportant or exceptionally general tokens may gather a lot of attention and/or importance. These

[7] A brief summary of these metrics are found in Appendix A.3

tokens include stopwords and punctuations found in any piece of text, which are not very useful in interpreting the model as they are not task-specific. By masking out these tokens when visualizing and interpreting, it helps isolate the keywords that make this model stand out for its task, and how the keywords relate to each other.

4 Experiment Set-up

4.1 Dataset

We create and test our visualizations on a subset of the data used in the paper "Long Document Classification from Local Word Glimpses via Recurrent Attention Learning" [He et al., 2019]. It is a compilation of research papers from 11 different fields; however, we only use papers from two fields. These were Artificial Intelligence (A.I.) and Programming Languages (P.L) which we labeled as positive and negative respectively. The papers categorized as both A.I. papers and P.L. papers were removed from our data. We use Pandas to compile each paper and their respective label into a single .csv file, which is used to create a Huggingface dataset, which we call "papers". The dataset contains a total of 5350 items, 2722 A.I. papers and 2628 P.L. papers, and contains on average over 5000 words, more than the Longformer's limit. We randomly partition the dataset into an 80/20 split for training and validation respectively with a good distribution of A.I. and P.L. papers (2177 and 2103 respectively) in our training set to avoid class imbalance.

An additional dataset was used to test our visualizations after using the papers dataset. We use a very small set of 12 mock clinical notes predicting suicidality and non-suicidality, designating them positive and negative respectively. The dataset contains a more unbalanced distribution of 9 positive examples and 3 negative examples and contains an average length of 1550 words.

4.2 Configurations

We used the default configuration of the base Longformer model designed by Beltagy et al. [2020]. It allows for a maximum of 4096 tokens for any given input sequence. The default configuration is implemented with 12 hidden layers and 12 attention heads, where each attention head has a sliding attention window of 512 tokens. The model features a dropout rate of 0.1 at each hidden layer. Lastly, we use the default tokenizer for the Longformer model which uses byte-level Byte-Pair Encoding [Sennrich et al., 2015].

4.3 Fine-tuning Parameters

The model was trained with the above model configuration using Pytorch Huggingface with an NVIDIA Geforce RTX 3070 with 8GB GPU memory. [8] Each example in the dataset was truncated to a maximum length of 2048 tokens. Long sequences can be very computationally expensive, so we train for only 2 epochs. We use a learning rate of $2e^{-5}$ with a weight decay of 0.01 using the AdamW optimizer [Loshchilov and Hutter, 2017]. To minimize the amount of GPU memory we use during training, a batch size of 1 is used alongside mixed precision training, gradient accumulation, and gradient checkpointing [Micikevicius et al., 2017, Sohoni et al., 2019].

5 Results

We explore the results of fine-tuning our model on the papers dataset and analyze the results of applying our visualization methods to the model. [9]

5.1 Model Performance

Our validation set contains 1070 total examples. On evaluation, our model obtains a validation accuracy of 0.98 which is a strong performance, and sufficient to design and test our visualizations on.

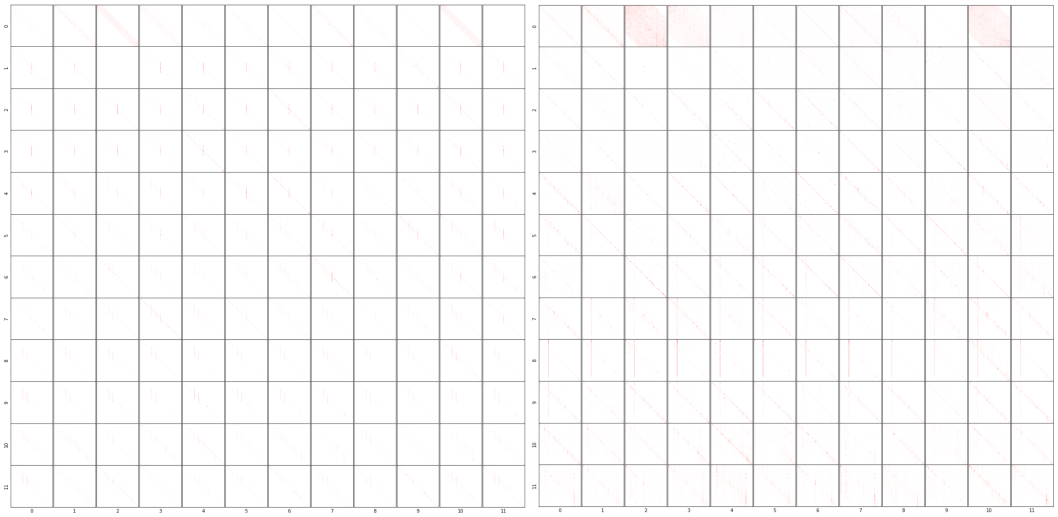
[8] A different hardware setup was used to visualize the predictions. Details are in Appendix A.2

[9] We explore the visualizations for the mock clinical notes in Appendix A.4

5.2 Visualization Analysis

Our results are mainly focused on exploring and analyzing the results of applying the visualizations on the papers dataset.

5.2.1 Attention Visualizations



(a) (b)

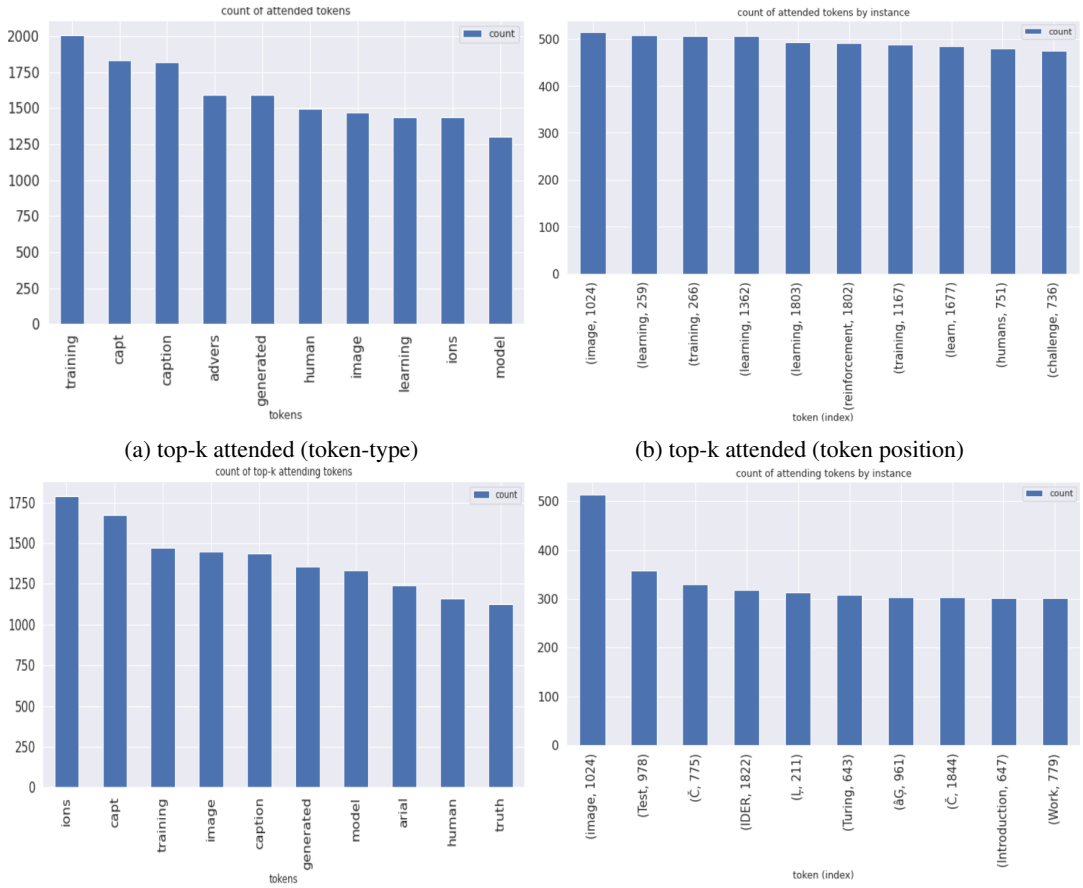
Figure 4: The attention output heatmap for all heads (x-axis) and layers (y-axis). (a) The attention heatmap for the false positive example. (b) The averaged aggregate attention for all examples in the dataset

Figure 4 shows the attention heatmap for a randomly selected, correctly predicted, positive example from the dataset and the aggregate attention output. Looking at the aggregate, we see that the sliding window attention becomes prominent in the later layers when all tokens begin to attend to all other tokens in their window. Interestingly, many heads in the first layer also have high attention across all of its tokens. The tokens in the early-middle layers seem to direct most of their attention to the tokens in the middle of the text. This pattern is reflected in our single example, although, not as prominent as the aggregate.

Using the same positive example, when counting by token type, the token "training" appears the most times in a token's top-k attended; but all of the tokens in the top 10 are domain-specific keywords or suffixes such as "-ions" (Fig 5a). This tells us that these keywords are common throughout the model and gather a lot of attention from many tokens across all heads and layers. Counting by token positions shows many instances of the tokens that appear in the top 10 by token types such as "learning" and "training", identifying high agreement between the most individually attended tokens, and the most attended tokens overall. (Fig 5b).

The agreement of tokens between our top-k attended by token type and token position tells us that not only are these tokens the most attended across all attention layers and heads, meaning the information gained from these tokens is the most relevant when passing to the next hidden layer, but each instance of the token is also highly attended to. This implies that these tokens have the largest effect overall on the model for this particular example, as well as the largest effect on every other token in the attention window learns.

The same agreement between token type and token position is not found for our group of most attending tokens. When we consider only the type of token, there are a lot of domain-specific keywords that are the most attending (Fig 5c). However, when counting based on the token position, there is a big difference in the most attending tokens as only the token "image" is in agreement. Furthermore, by token position, only the token "image" appears to attend to all other tokens across all hidden layers and attention heads as there is a substantial drop between the most attending and second most attending tokens (Fig 5d).



(a) top-k attended (token-type)

(b) top-k attended (token position)

(c) top-k attending (token-type)

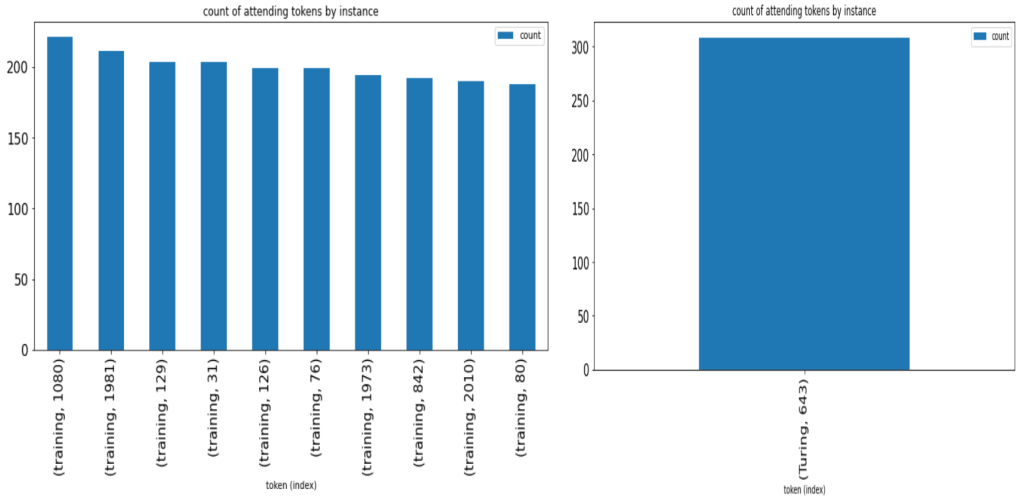
(d) top-k attending (token position)

Figure 5: The 10 tokens with the most appearances in a surrounding token's top-k attended and attending over all layers and heads. (Left) The number of appearances is based on the token. (Right) The number of appearances based on the token position (instance).

The lack of agreement in our attending tokens between the token type and position suggests that the 10 most attending by token type create the strongest associations with other tokens throughout the example, but the individual tokens only create strong associations with a moderate amount of surrounding tokens. For instance, the token "training" is the third most attending token in the example, but there are no instances of the token in the top 10 by token position (Fig 5). Many instances of "training" are in the example, but each only has a moderately high count (Fig 6a). However, since there are so many instances that strongly attend to a high number of tokens in their window, it is not surprising that the token "training" is strongly attending to many tokens throughout the example.

In the same vein, the tokens that only appear in the most attending tokens by token position indicate that each instance is associating themselves with a lot of their surrounding tokens across all layers and heads; but, either other instances of the token do not attend well to others, or there are very few instances in the example. For instance, the token "Turing" is the 6th most attending token by token position and is strongly attending to a lot of surrounding tokens. However, there is only one instance of the token over the entire example so, over the entire example, "Turing" is not attending to a lot of tokens (Fig 6b). A token like "training", which has many instances despite each instance being less attending, will attend to more tokens in the example (Fig 5 and 6).

We see from our plots of the top-k attended and attending tokens that there are many tokens that are in common by token type (Fig 7a). Taking the intersection between the 10 top-k attended and top-k attending by token type, we find that there are 8 common tokens. These token types that get the most attention also attend very well to other tokens making these tokens significant for the model's learning.



(a) Training (b) Turing

Figure 6: The number of times each instance of "training" and "Turing" appears in a surrounding token's top-k attending tokens

	tokens	count_attended	count_attending
0	training	2009.0	1473.0
1	capt	1833.0	1671.0
2	caption	1821.0	1436.0
3	generated	1591.0	1356.0
4	human	1497.0	1160.0
5	image	1471.0	1450.0
6	ions	1436.0	1790.0
7	model	1300.0	1335.0

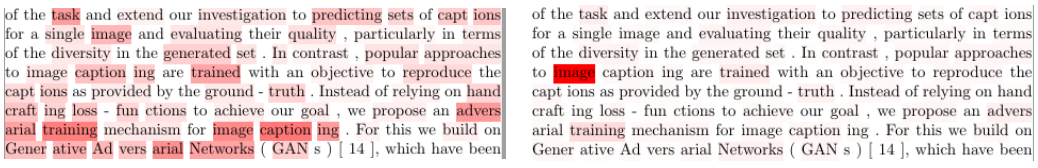
(a) intersection of the tokens by token-type

	token_x	token_position	count_attended	count_attending
0	image	1024	514.0	514.0

(b) intersection of the tokens by token position

Figure 7: The tokens at the intersection of the top-k attended and top-k attending by token-type and token-position

On the other hand, only one token appears in the intersection when counted by token position, the token "image", but it seems to have the most attention and be the most attending (Fig 7b).



(a) (b)

Figure 8: An excerpt from the tokenized positive class example with normalized summed attention overlayed on the token for two different configurations. (a) The summed attention in layer 12. (b) The summed attention averaged over all layers.

Figure 8 shows the summed attention overlay using an excerpt of the positive class example. Tokens with higher attention have a darker red colour overlayed on their token such as the token "image". This is not a perfect replication of the original text as there are some instances where the words are split up into different tokens such as captions being split into "capt" and "tions". However, the text

can still be read while we understand what tokens the model gives the highest attention to, as well as how each token compares to others. For instance, consider the tokens "training" and "image". These tokens have higher attention compared to the token "mechanism" as the colour overlayed is much darker in comparison. Since attention evolves with each hidden layer, we expect the attentions to change depending on the layer. The attention overlay illustrates this as tokens in different layers have different summed attentions (Fig 8a and 8b).

5.2.2 Attributions Visualizations

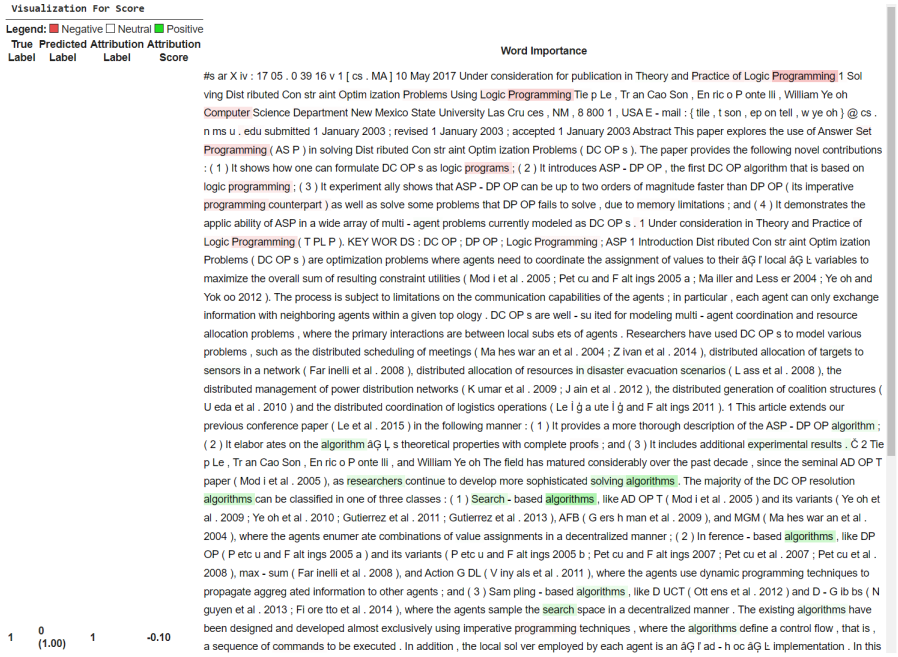


Figure 9: The Attribution Overlay on a randomly selected false negative example from the papers dataset

Figure 9 illustrates the attribution overlay view on a randomly selected false negative example. The top of the visualization contains headers for each column and the legend mapping the colour to the attributions. The headers include the true label of the input, the predicted label, and the probability of prediction. The attribution label displays the class you chose to represent positive attributions. In our case, it is class "1" which represents an A.I. Paper. The attribution score identifies the magnitude of the attribution scores for the entire text, where the higher the attribution, the more the model is influenced to predicting a class. We see in our example that the attribution score is -0.1, suggesting a negative prediction, reflected by our model's predicting negative despite a positive example.

We immediately notice that tokens such as "algorithm" are very positively attributed, while words such as "programming" are negatively attributed (Fig 9). Furthermore, it appears that many instances of the words "algorithm" or its derivations are among the highest positive attributions. We see the same for the negative attributions, with the token "programming" and its derivations. By looking at our table, we confirm that the many instances of "algorithm" and "programming" are highly attributed, not only on a per token basis (Fig 10a and 10b), but also an example wide basis (Fig 10c and 10d). We find that these top tokens are consistent with the attributions aggregated over the entire dataset (Fig 10e and 10f). Tokens such as "algorithms", and "programming" are prevalent for all examples in the dataset.

The tables also assist in identifying which of the two tokens are more strongly attributed. Consider the tokens "search" and "researcher" which show similarly high attributions, making it difficult to tell which has larger attributions based on colour saturation alone (Fig 11). Using the table, we find that "search" is a little higher than "researcher" (0.153 vs 0.135).

	Word	index	attribution
0	algorithms	720	0.377713
1	algorithms	695	0.280959
2	algorithms	808	0.227175
3	algorithms	704	0.206849
4	search	948	0.161278
5	Search	717	0.153566
6	solving	694	0.147626
7	researchers	688	0.135309
8	algorithm	616	0.135011
9	algorithms	908	0.118957

(a)

	Word	index	attribution
0	Programming	32	-0.336014
1	Programming	48	-0.268116
2	programs	178	-0.198515
3	Programming	136	-0.176396
4	Computer	68	-0.167327
5	Programming	286	-0.150109
6	programming	229	-0.146876
7	programming	200	-0.125191
8	of	30	-0.117793
9	Logic	47	-0.112481

(b)

	tokens	attribution
0	algorithms	1.350910
1	algorithm	0.269703
2	search	0.161278
3	Search	0.153566
4	researchers	0.135309
5	solving	0.130131
6	agents	0.109138
7	and	0.082688
8	the	0.071412
9	experimental	0.071383

(c)

	tokens	attribution
0	Programming	-1.018337
1	programming	-0.341622
2	Logic	-0.293834
3	programs	-0.198517
4	Computer	-0.167329
5	Practice	-0.089079
6	counterpart	-0.071546
7	Set	-0.067201
8	Problems	-0.065523
9	OP	-0.064579

(d)

	tokens	attribution
0	learning	0.270925
1	neural	0.159887
2	data	0.128880
3	AI	0.081176
4	training	0.080792
5	dataset	0.071329
6	algorithms	0.058013
7	human	0.048780
8	intelligence	0.046269
9	datasets	0.042661

(e)

	tokens	attribution
0	programming	-0.120735
1	program	-0.088943
2	programs	-0.082175
3	languages	-0.072262
4	language	-0.053761
5	code	-0.052443
6	software	-0.038568
7	compiler	-0.032076
8	Programming	-0.031495
9	syntax	-0.029020

(f)

Figure 10: Tables of the 10 largest attributions. (a)(b) The most positive and negative attributions per single token. (c)(d) The aggregated positive and negative attributions of the example. (e)(d) The average aggregated positive and negative attributions over the dataset.

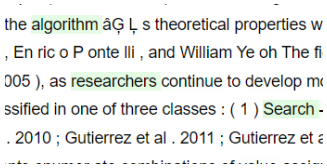


Figure 11: An example of three similarly attributed, but difficult to tell apart tokens

There can be situations where certain tokens dominate the top 10 attribution scores. For example, "programming" and "algorithms" take up over half of the top 10 tokens in their respective lists (Fig 10a and 10b). The example-wide attribution score lets us confirm the influence of these tokens, but also brings to the user's attention some tokens that are not as individually important but play an important role in influencing the prediction (Fig 10c and 10d). For instance, the token "experimental" is the 10th highest positive attributed token for this example but is not in the 10 highest individually attributed tokens. Lastly, since the top tokens are consistent between example and dataset, it tells us that the model is not demonstrating any strange behavior in this example such as attributing tokens with high positive attributions over the dataset as negative (Fig 10e and 10f).

5.2.3 Attention-Attribution Similarities

Table 1 displays the mean cosine similarities, Kendall-Tau, and RBO between the summed attention scores and the attributions of our Longformer Model over the entire dataset. We find that the masked attributions and attentions present the highest similarities for both the final layer of attention and the sum of all attention layers. Masking reveals that there are high overlaps between the attribution scores and the summed attention scores for the important tokens regardless of scaling. We see an average increase of 32.03 percent for cosine similarity, 405.03 percent increase for Kendall-Tau, and 79.51 percent increase for RBO. Scaling, however, only seems to be more effective for layer 12. For all attention layers averaged, scaling actually seems to be marginally detrimental (Tab 1).

If we only look at the top 5 percent of tokens, we find that the mean Jaccard similarity for the final and all attention layers are very low. Immediately the difference between the masked and unmasked similarities is apparent as there is only a marginal difference between similarities, unlike the large difference found when comparing all tokens (Tab 2). Overall, the low similarities reveal that the

Table 1: The three similarity measures averaged over the dataset with respect to different attention layers

Attention layer	Is scaled?	Is Masked?	Cosine Similarity	Kendall-Tau	RBO
Attention Layer 12	True	True	.411	.742	.778
		False	.227	.168	.594
	False	True	.426	.738	.776
		False	.120	.157	.589
All Attention Layers	True	True	.296	733	.769
		False	.120	.134	.581
	False	True	.213	.736	.774
		False	.170	.131	.581

Table 2: The mean Jaccard Similarity for the 95th percentile for tokens over the entire dataset with respect to different attention layers

Attention layer	Is scaled?	Is Masked?	Jaccard Sim
Attention Layer 12	True	True	.197
		False	.183
	False	True	.199
		False	.186
All Attention Layers	True	True	.178
		False	.188
	False	True	.196
		False	.198

tokens with the highest attention and attribution scores are not similar, suggesting that the high similarity of the masked inputs is attributed to the other 95 percent of tokens.

6 Discussion

The difficulty with visualizing Longformer predictions is simply the fact that there are so many tokens that attempting to see how a single token attends to all other tokens, like visualizations for BERT, is unfeasible [Vig, 2019, Clark et al., 2019]. Using our ensemble of methods to interpret and visualize attention, we manage to address our wishes to gain a model-wide understanding of attention. Through the use of the Attention Heatmap, we assessed how attention patterns change with heads and layers. To some degree, this has been achieved previously, just requiring some modification of the attention output [Li et al., 2021] [10]. The attention overlay further assists this task by visualizing the distribution of attention for any layer and head combination of your choice to see how attentions change or evolve.

To properly get an in-depth look at attention between tokens, our approach to some extent is similar to BERTviz as it assesses the attended and attending attentions [Vig, 2019]. Our approach only takes a very small subset of each token's relations by obtaining their top-k attended and attending tokens. We can still identify the most important token relationships but do not have to deal with hundreds of mundane relationships for every token. Interestingly, our most plausible method of obtaining a model-wide understanding stems from counting the number of times tokens appear in the top-k attended and attendings, which were originally meant to be token-specific. Our most attended and attending tokens (Fig 5) over all layers and heads demonstrate strong domain-specificity, indicating a high plausibility of the attention outputs being a source of explanation. Of course, plausibility itself is a subjective concept and will change depending on the interpreter [Ding and Koehn, 2021, Bibal et al., 2022].

[10] Recall Section 3.1.0

Visualizing saliency through attribution overlay proves to be surprisingly effective with some accompanying tables [Kokhlikyan et al., 2020]. The attribution overlay identifies important and influential tokens while being able to naturally read the input text. This type of overlay has been used for other Transformer Neural Networks, but the number of tokens passed in those examples is much lower in comparison [Ding and Koehn, 2021]. This not only makes viewing those pieces of text short but also makes comparing tokens easier. With a much larger text, comparing tokens can be difficult. Creating an accompanying table assists with this comparison immensely. Furthermore, having an example-wide attribution score summarizes the attributions much better than a token-by-token basis. Long documents tend to have multiple instances of tokens which could all have high attributions. Recalling figure 10, the tokens "programming" and "algorithms" dominate the per-token attribution table. Other tokens that also have large attributions such as "logic" become overshadowed in comparison. The example-wide attribution score not only solidifies the influence of the common, highly attributed tokens but helps bring important tokens to the user's attention to assist with interpreting the Longformer's prediction.

However, the tables do not solve all our problems. A limitation of the attribution overlay, and by extension the attention overlay, is that comparisons can still be difficult without the ability to efficiently obtain token positions or scores. This was likely not a problem for evaluating on the IMDB dataset given the short length of the examples; however, with a large input example, identifying a token's position can be problematic, especially if there are a lot of duplicate tokens [Kokhlikyan et al., 2020, Maas et al., 2011].

The low similarities between unmasked attention and attribution scores suggest that the natural (unmasked and unscaled) attention outputs of the model fall in line with previous work by Jain and Wallace [2019], who found that attention weights display a low correlation with gradient-based importance. We extended this by assessing how post-processing methods (scaling and masking) affect the correlation, where we found that scaling did not positively impact the similarity, but masking was effective. The masked attentions do not perfectly correlate with the attributions, but the high degree of similarity suggests that it could have a similar degree of faithfulness and plausibility as attributions. This suggests that our interpretations and the accompanying visualizations of the masked attention may not only help visualize the model's architecture but also identify the reason for the model's prediction.

A caveat is that the attention weights require task-specific knowledge if being used to explain model predictions. Unlike attribution scores, attention weights are not signed so we don't know which class the attention helps predict, thus requiring human conjecture. While the "attention weights cannot be directly used as a casual explanation" given their lack of sign [Grimsley et al., 2020], it still seems to adequately identify the importance of tokens.

7 Conclusion

We created and adapted an ensemble of visualization and interpretations to assist in understanding Longformer model predictions on long documents. We visualize the model's sliding and global attention outputs, and its saliency with respect to an input. Furthermore, we examined whether our attention interpretations and visualizations could be used as an explanation for the model's prediction by examining the correlation between attention and attribution scores. We find out that only by applying a mask do we get similar results as established faithful and plausible explanations such as gradient-based saliency methods.

7.1 Future Work

The attention visualizations refactored the Longformer's sliding window attention and global attention into a full attention matrix. While this makes creating visualizations simpler, it substantially increases the memory requirement. We recommend adapting these visualizations to use the un-altered Longformer attentions. Additionally, implementing stronger interactively such as being able to click on the tokens to get attention and attribution scores in the attention and attribution overlay will greatly improve ease of interpretation. Lastly, we created visualizations for interpreting binary classification. Modifications and extensions will be required for visualizing other NLP tasks.

References

J. Bastings and K. Filippova. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? 2020. doi: 10.48550/ARXIV.2010.05607. URL https://arxiv.org/abs/2010.05607.

I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

A. Bibal, R. Cardon, D. Alfter, R. Wilkens, X. Wang, T. François, and P. Watrin. Is attention explanation? an introduction to the debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.269. URL https://aclanthology.org/2022.acl-long.269.

G. Chrysostomou and N. Aletras. Enjoy the salience: Towards better transformer-based faithful explanations with word salience, 2021. URL https://arxiv.org/abs/2108.13759.

K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. 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, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

S. Ding and P. Koehn. Evaluating saliency methods for neural language models, 2021. URL https://arxiv.org/abs/2104.05824.

C. F. Gauss. *METHODUS NOVA INTEGRALIUM VALORES PER APPROXIMATIONEM INVENIENDI*, volume 3 of *Cambridge Library Collection - Mathematics*, page 165–196. Cambridge University Press, 2011. doi: 10.1017/CBO9781139058247.008.

C. Grimsley, E. Mayfield, and J. R. S. Bursten. Why attention is not explanation: Surgical intervention and causal reasoning about neural models. In *LREC*, 2020.

J. He, L. Wang, L. Liu, J. Feng, and H. Wu. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718, 2019. doi: 10.1109/ACCESS.2019.2907992.

P. Jaccard. The distribution of the flora in the alpine zone. *The New Phytologist*, 11(2):37–50, 1912. ISSN 0028646X, 14698137. URL http://www.jstor.org/stable/2427226.

A. Jacovi and Y. Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.386. URL https://aclanthology.org/2020.acl-main.386.

S. Jain and B. C. Wallace. Attention is not explanation, 2019. URL https://arxiv.org/abs/1902.10186.

M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. ISSN 00063444. URL http://www.jstor.org/stable/2332226.

N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020. URL https://arxiv.org/abs/2009.07896.

R. Li, W. Xiao, L. Wang, H. Jang, and G. Carenini. T3-vis: a visual analytic framework for training and fine-tuning transformers in nlp, 2021. URL https://arxiv.org/abs/2108.13587.

Y. Li, R. M. Wehbe, F. S. Ahmad, H. Wang, and Y. Luo. Clinical-longformer and clinical-bigbird: Transformers for long clinical sequences, 2022a. URL https://arxiv.org/abs/2201.11838.

Z. Li, X. Wang, W. Yang, J. Wu, Z. Zhang, Z. Liu, M. Sun, H. Zhang, and S. Liu. A unified understanding of deep nlp models for text classification, 2022b. URL https://arxiv.org/abs/2206.09355.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2017. URL https://arxiv.org/abs/1711.05101.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu. Mixed precision training, 2017. URL https://arxiv.org/abs/1710.03740.

P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz. Importance estimation for neural network pruning, 2019. URL https://arxiv.org/abs/1906.10771.

J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein. Explainable prediction of medical codes from clinical text, 2018. URL https://arxiv.org/abs/1802.05695.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units, 2015. URL https://arxiv.org/abs/1508.07909.

N. S. Sohoni, C. R. Aberger, M. Leszczynski, J. Zhang, and C. Ré. Low-memory neural network training: A technical report, 2019. URL https://arxiv.org/abs/1904.10631.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks, 2017. URL https://arxiv.org/abs/1703.01365.

J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal. Generating token-level explanations for natural language inference, 2019. URL https://arxiv.org/abs/1904.10717.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

A. Vaswani, S. Bengio, E. Brevdo, F. Chollet, A. N. Gomez, S. Gouws, L. Jones, L. Kaiser, N. Kalchbrenner, N. Parmar, R. Sepassi, N. Shazeer, and J. Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018. URL http://arxiv.org/abs/1803.07416.

J. Vig. A multiscale visualization of attention in the transformer model, 2019. URL https://arxiv.org/abs/1906.05714.

J. Vig and Y. Belinkov. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy, Aug. 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4808. URL https://aclanthology.org/W19-4808.

W. Webber, A. Moffat, and J. Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28:20, 11 2010. doi: 10.1145/1852102.1852106.

Q. Xie, X. Ma, Z. Dai, and E. Hovy. An interpretable knowledge transfer model for knowledge base completion, 2017. URL https://arxiv.org/abs/1704.05908.

J. Yang and Y. Zhang. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018. URL http://aclweb.org/anthology/P18-4013.

A Appendix

A.1 Attention Summary

Attention gets a query Q, a key K, and a value V for every word in the input. These are combined into their own matrices and compute the scaled dot-product of the queries and keys before finally multiplying by the values, creating attention weights between all pairs of words [Vaswani et al., 2017]. Furthermore, each Transformer contains multiple hidden layers and attention heads, with each layer passing along the attention weights to the next, and each attention head calculating its own attention weights.

A.2 Visualization Set-up

We used a different setup for visualizations than for fine-tuning. Instead of using a local setup, the visualizations were created and run using Google Colab. Google Colab uses an NVIDIA Tesla T4 with 16GB of GPU memory. For consistency with our fine-tuning process, each example was truncated to a maximum length of 2048 tokens. We use 3 examples from our dataset. The first two examples were randomly selected from the pool of false negatives and false positives. The last example was the example with the highest predicted probability for the positive class.

A.3 Metrics Explanation

A.3.1 Cosine Similarity

Cosine Similarity measures the similarity of two vectors by getting the dot product of the two vectors (in this case our attention and attributions) and dividing by the product of their lengths. The formula for Cosine Similarity is as follows, where A and B are the two lists of scores:

$$\text{CosineSimiliarty(A,B)} = \frac{A \cdot B}{\|A\| \, \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2}$$

We are using the normalized attention and absolute attribution scores and directly calculating the difference between the two vectors. The range of the output for cosine similarity is from [-1, 1], where -1 means the two arrays are completely different and 1 means they are identical.

A.3.2 Kendall-Tau

Kendall-Tau coefficient calculates similarity coefficients by assigning a rank to all items in both lists, then finding the number of concordant pairs subtracted by the number of discordant pairs, all divided by total number of pairs [Kendall, 1938]. To briefly describe concordant pairs: given a pair of observations (X_i, Y_i) and (X_j, Y_j), where X is the attention vector, Y is the attribution vector, and $i < j$ are the indices of the vector representing the token; if either $(X_i > X_j)$ and $(Y_i > Y_j)$ is true or $(X_i < X_j)$ and $(Y_i < Y_j)$ is true, then they are concordant. The coefficient scale is from [-1, 1], where -1 means the two arrays are completely different and 1 means they are identical.

A.3.3 RBO

Similar to Kendall-Tau, RBO also computes similarity coefficients. The main difference between Kendall-Tau and RBO is that RBO places more emphasis on the higher ranking items[Webber et al., 2010]. However, if we wish to identify the commonality between tokens with the most influence and attention, placing more emphasis on the higher-ranking items seems to be a good approach. Unlike Kendall-Tau, instead of assigning a rank to every item in both arrays, we must sort the scores in both arrays from highest to lowest and represent the scores by using their corresponding positions from the original arrays. The token with the highest attribution or attention rank, represented by its position in the input, is first in their respective lists indicating the highest rank. The formula for RBO is as follows:

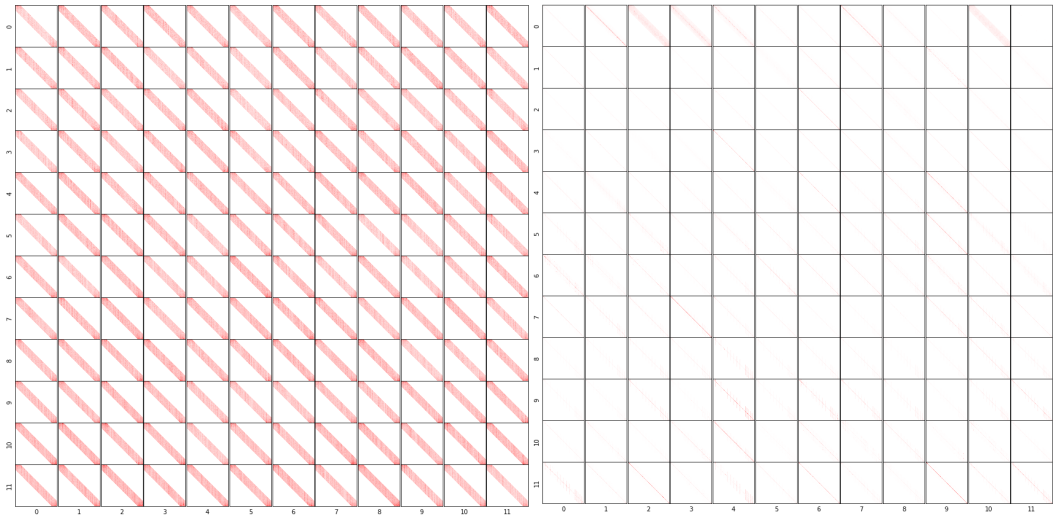
$$RBO(S, T, p) = (1 - p) \sum_{d=1}^{\infty} p^{d-1} \cdot A_d \tag{3}$$

S and T are the two arrays of items, in our case token positions, ranked from highest to lowest and d is the depth of the rankings where 1 is highest rank. A_d is the size of the intersection of common token positions in S & T up to depth d, or indices (0 to $d-1$), divided by d. p is a weight parameter where the lower the value of p, the more focus on the higher ranks. The RBO coefficient scale is from [0, 1], where 0 means the two arrays are completely different, and 1 means they are the same.

A.4 Experiment 2: Fake clinical notes

Here we explore and analyze the results of running our visualizations on our small mock clinical notes dataset.

A.4.1 Attention Analysis



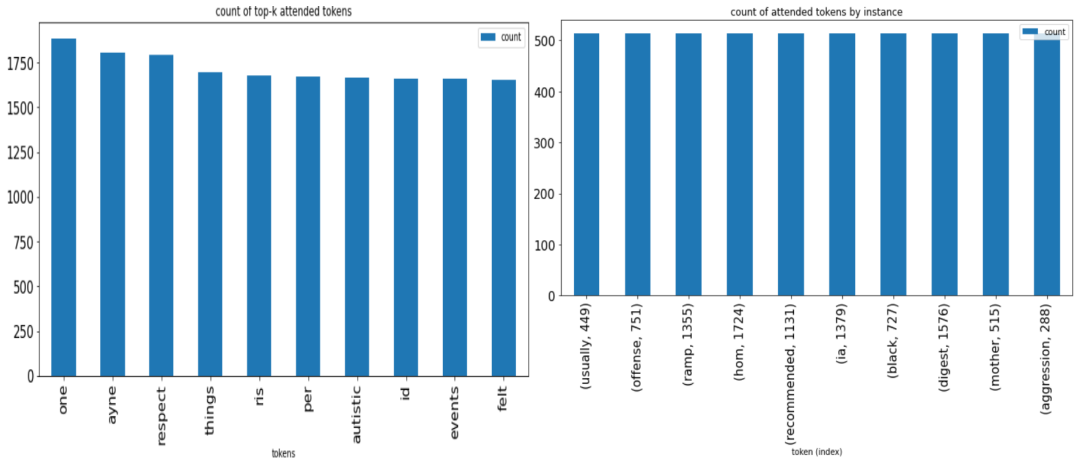
(a) (b)

Figure 12: The attention output heatmap for all heads (x-axis) and layers (y-axis). (a) The attention heatmap for a single example. (b) The averaged aggregate attention heatmap over all examples in the dataset

Figure 12 finds that the attention patterns for a randomly selected example and the aggregate are substantially different. The single example illustrates a strong sliding attention shape for all heads and layers, telling us that each token has high attention weights to all other tokens in its window. The attention heatmaps are normalized so more emphasis is placed on the higher attention weights. Given that the sliding window attention shape is super prominent throughout all layers and heads for the selected example, it suggests that are few select attention weights are very low which makes the rest of the attention stand out in comparison. The aggregate attention heatmap shows a much narrower and more focused attention pattern, suggesting that the rest of the examples are very focused in their attentions, attending mainly to a smaller portion of surrounding tokens, as opposed to the wide coverage of tokens in this example.

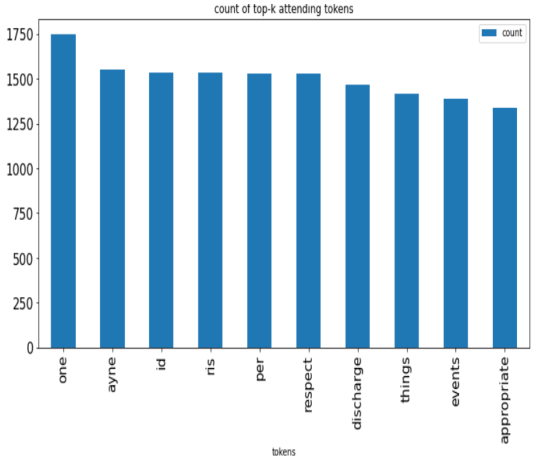
Figure 13a and 13b indicate that the type of token that gets the most attention from other tokens is not in agreement with the individual tokens that have the most attention. As for the attending tokens side, we see results similar to the attended side, where there is little agreement between counting by tokens, and by their positions (Fig 13c and 13d).

The lack of agreement suggests that information gathered from the tokens in the token type count has the most significance on the model across the entire example, but is not necessarily on all of the tokens in their attention window. Likewise, the information gathered from the tokens from the token position count has the most significance on the other tokens within their attention window, but the other instances may not have the same significance or there may not be any other instances; therefore, its overall significance is not as high. (Fig 13).

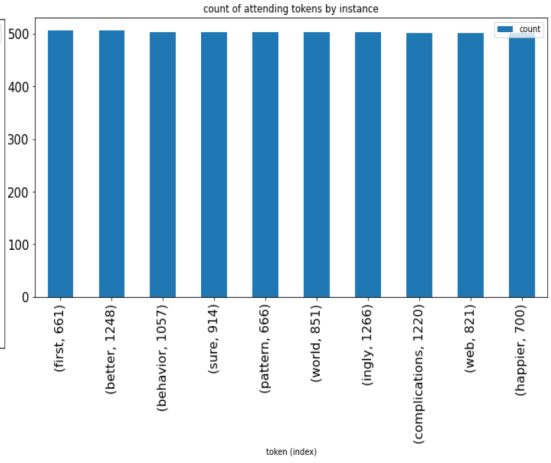


(a) top-k attended

(b) top-k attended (token position)



(c) top-k attending



(d) top-k attending (token position)

Figure 13: The 10 tokens with the most appearances in a surrounding token's top-k attended and attending over all layers and heads. (Left) The number of appearances is based on the token. (Right) The number of appearances based on the token position (instance).

When we look at the intersection between the most attended and most attending tokens, we find that there is a high agreement between the top-k attended and top-k attending when counting by the type of token, as there are eight common tokens (Fig 14a). However, this behavior is not reciprocated when counting by the position of the token (Fig 14b).

Unlike the results found for the papers dataset (Fig 8), the attention overlay shows a negligible difference between the summed attention of our two different attention layers (Fig 15). Recalling our 2D attention heatmap, there was a very clear sliding window attention pattern which suggested many tokens receiving a lot of attention (Fig 12a). Using our attention overlay, figure 15 finds that nearly every token is heavily saturated for both the final hidden layer and all hidden layers averaged, meaning that they are receiving a lot of attention from the tokens in their attention window, similar to the patterns shown by the 2D heatmap.

Furthermore, figure 15 finds that there are no visible differences between the summed attention over the final layer compared to the summed, averaged attention over all layers. Alluding back to the 2D heatmap, the attention patterns looked the same for all heads and layers, which would mean that the attention for layer 12 and the average attention over all layers would be roughly the same, reflected in our attention overlay.

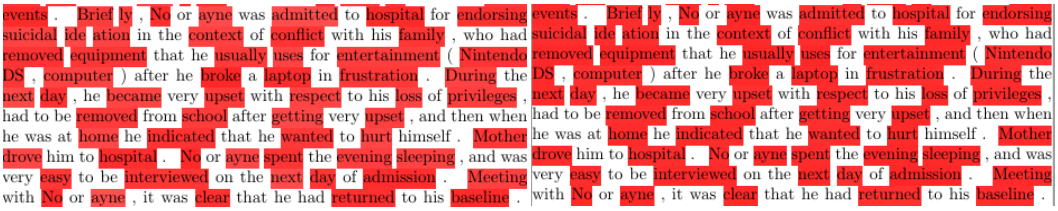
	tokens	count_attended	count_attending
0	one	1884.0	1811.0
1	ayne	1816.0	1536.0
2	respect	1802.0	1525.0
3	things	1715.0	1404.0
4	per	1677.0	1611.0
5	id	1674.0	1583.0
6	ris	1668.0	1588.0
7	events	1644.0	1432.0

(a) Intersection of the tokens by token-type

	token_x	token_position	count_attended	count_attending
0	unit	1164	512.0	503.0

(b) Intersection of the tokens by token position

Figure 14: The tokens at the intersection of the top-k attended and top-k attending by token-type and token-position



(a) (b)

Figure 15: An excerpt from the tokenized positive class example with normalized summed attention overlayed on the token for two different configurations. (a) The summed attention in layer 12. (b) The summed attention averaged over all layers.

A.4.2 Attribution Visualizations

Figure 16 presents the attribution overlay for an excerpt of our randomly selected fake clinical note. We find that our overall attribution score is negative and our predictions reflect it with a 0.54 probability of our model predicting negative despite the true label being positive.

The tokens we find in the top individual attributions are mostly reflected in the top example-wide aggregate attributions for the positive attributions. On the other hand, for the negative attributions, only "plan" and "would" appear in the example-wide attributions. Furthermore, many tokens in the top positive attributions can be found in the top dataset-wide attributions. The negative attributions do not demonstrate this same pattern (Fig 17e and 17f).

Overall, the tokens that appear to get the highest attribution scores are domain-related and do not have any single token that dominates the highest attribution scores (Fig 17). The inconsistency with the negatively attributed tokens suggests that there are either very few instances of that token, or the other instances of the tokens are not important to the model. Furthermore, since the tokens that are the most negatively attributed in the example are not the most important in the dataset, it implies that the negatively attributed tokens are example specific, or the model deems it not important for the rest of the examples (Fig 17b, 17d and 17f). The positively attributed tokens are generally consistent but we do see an interesting result from the dataset-wide positive attributions. The tokens "mother" and "female" are highly attributed to suicidality, which suggests a possible gender bias in our model and dataset (Figure 17e).

A.4.3 Attention Attribution Similarities

Table 3 shows high cosine similarities for all configurations but low Kendall-Tau and RBO scores for the unmasked scores. Similar to the papers dataset, masking tokens in the notes dataset has high similarities between the attributions and attentions for both the final layer of attention and averaged over all layers, indicating a lot of overlap between the attention and attribution scores of important

1	0 (0.54)	1	-3.46	meetings , and was receptive to the idea of me meeting with his parents first . when parents came in for a meeting , they shared that his behavior had declined since march , with a reduction of ris per id one from a higher dose which caused significant o cul ogy ric events , down to 1 mg which has not led to any o cul ogy ric events but has been less helpful with respect to containing his aggression . we explored options with respect to treating this and parents selected the treatment plan below , which was our recommended treatment plan . parents had no concerns in taking no or ayne home and felt that he had returned to baseline . they were appreci ative of his short stay on the unit but they do miss him and the unit was active with respect to distress by other patients , so they were worried about the influence of fear on no or ayne . they elected to take him home on discharge . no or ayne was discharged easily with no complications on [** 2020 - 12 - 30 **], at approximately 2 p . m . impression if feels like

Figure 16: The attribution overlay on a randomly selected example from the mock clinical notes dataset

	Word	index	attribution
0	family	430	0.063025
1	threat	1619	0.061966
2	ben	214	0.060865
3	name	331	0.057714
4	discharged	1205	0.056924
5	name	318	0.056194
6	happen	2018	0.054360
7	aggression	1098	0.053639
8	mg	1993	0.053250
9	take	1948	0.051977

(a)

	Word	index	attribution
0	plan	1966	-0.078297
1	plan	1122	-0.069454
2	computer	448	-0.069208
3	day	530	-0.068755
4	cycl	248	-0.066530
5	easier	1953	-0.066436
6	diagnosis	268	-0.064512
7	misogyn	1527	-0.061235
8	diagnosis	286	-0.060768
9	would	646	-0.059901

(b)

	tokens	attribution
0	mg	0.453486
1	one	0.281195
2	name	0.226609
3	discharge	0.196559
4	significant	0.173714
5	ric	0.172324
6	aggression	0.132329
7	frustration	0.131977
8	discharged	0.127808
9	risk	0.123484

(c)

	tokens	attribution
0	ayne	-0.477179
1	c̄	-0.364867
2	plan	-0.246068
3	would	-0.205324
4	cul	-0.177146
5	context	-0.160468
6	medication	-0.138289
7	daily	-0.130862
8	try	-0.130545
9	things	-0.128669

(d)

	tokens	attribution
0	discharge	0.372953
1	name	0.234457
2	ine	0.160587
3	mg	0.158104
4	mother	0.124459
5	female	0.120808
6	first	0.110170
7	un	0.103989
8	things	0.097076
9	diagnosis	0.066723

(e)

	tokens	attribution
0	c̄	-0.596677
1	hospital	-0.252264
2	name	-0.240356
3	also	-0.195440
4	disorder	-0.129558
5	time	-0.125816
6	able	-0.123044
7	first	-0.117680
8	home	-0.104151
9	admission	-0.101625

(f)

Figure 17: Tables of the 10 largest attributions. (a)(b) The most positive and negative attributions per single token. (c)(d) The aggregated positive and negative attributions of the example. (e)(d) The average aggregated positive and negative attributions over the dataset.

tokens. Unlike the papers dataset, where the similarities between the final attention layer and the attributions were slightly higher when scaling was applied, scaling is marginally detrimental for both the final layer and averaged over all layers.

Looking at only the top 5 percent of tokens, table 4 shows that the notes dataset has lower Jaccard Similarities for all configurations than the papers dataset, which already did not have high Jaccard similarities. The low similarities in the top 5 percent of tokens suggest that the high similarity of the masked inputs is attributed to the other 95 percent of tokens.

Table 3: The three similarity measures averaged over the entire dataset with respect to different attention layers

Attention layer	Is scaled?	Is Masked?	Cosine Similarity	Kendall-Tau	RBO
Attention Layer 12	True	True	.786	.795	.782
		False	.741	.026	.509
	False	True	.788	.798	.783
		False	.747	.029	.510
All Attention Layers	True	True	.788	.800	.784
		False	.753	.044	.516
	False	True	.788	.800	.785
		False	.753	.043	.516

Table 4: The mean Jaccard Similarity for the 95th percentile of tokens over the entire dataset with respect to different attention layers

Attention layer	Is scaled?	Is Masked?	Jaccard Sim
Attention Layer 12	True	True	.079
		False	.027
	False	True	.077
		False	.029
All Attention Layers	True	True	.077
		False	.033
	False	True	.077
		False	.033

A.5 Algorithms

Algorithm 1 Getting each Token's top-k attended

Inputs: Matrix A: (layer, batch, head, seq_len, seq_len)
Output: Matrix B: (layer, batch, head, seq_len, k)

B = new array ▷ shape: (layer, batch head, seq_len)
for $layer$ in $A.num\_layers$ **do**
  batch = new array ▷ shape: (batch, head, seq_len)
  for $batch$ in $A.num\_batches$ **do**
    heads = new array ▷ shape: (head, seq_len)
    for $head$ in $A.num\_heads$ **do**
      tokens = new array ▷ shape: (seq_len)
      for $token$ in $A.seq\_length$ **do**
        topk = A[$layer$, $batch$, $head$, $token$].argsort()
        topk = topk[:k] ▷ Get k values
        tokens.append(topk)
      end for
      heads.append(tokens)
    end for
    batch.append(heads)
  end for
  B.append(batch)
end for
return: B

Algorithm 2 set of all top-k

Inputs: list of tokens from our example
Output: dataframe with all top-k tokens and their count

```
count = { }                                                  ▷ create new dictionary
topklist = ()                                                ▷ create new list

for token in input.tokens do
    topktokens = set(token.topkattended)                     ▷ assume top-k's loaded with the input
                                                             ▷ also works with token positions
                                                             ▷ can substitute topkattended with topkattending
    topklist.append(list(topktokens))
end for

for topk in topklist do
    if topk is in count.keys then
        count[topk] +=1                                      ▷ add one to existing counter
    else
        count[topk] = 1                                      ▷ add token to dictionary with count 1
    end if
end for

count = dataframe(att_count, orient="index") ▷ create dataframe with each key-value pair as row
return: att_count
```

Algorithm 3 Getting each Token's top-k attending

Inputs: Matrix A: (layer, batch, head, seq_len, seq_len)
Output: Matrix B: (layer, batch, head, seq_len, k)

```
B = new array                                                ▷ shape: (layer, batch head, seq_len)
for layer in A.num_layers do
    batch = new array                                        ▷ shape: (batch, head, seq_len)
    for batch in A.num_batches do
        heads = new array                                    ▷ shape: (head, seq_len)
        for head in A.num_heads do
            tokens = new array                               ▷ shape: (seq_len)
            A[layer, batch, head] = A[layer, batch, head].Transpose
            ▷ Swap Rows and Columns
            for token in A.seq_length do
                topk = A[layer, batch, head, token].argsort()
                topk = topk[:k]                              ▷ Get k values
                tokens.append(topk)
            end for
            heads.append(tokens)
        end for
        batch.append(heads)
    end for
    B.append(batch)
end for
return: B
```

Algorithm 4 Token's total attention

Inputs: Matrix A: (layer, batch, head, seq_len, seq_len)
Output: Matrix B: (layer, batch, head, seq_len)

B = new array ▷ shape: (layer, batch head, seq_len)
for $layer$ in $A.num\_layers$ **do**
    batch = new array ▷ shape: (batch, head, seq_len)
    for $batch$ in $A.num\_batches$ **do**
        heads = new array ▷ shape: (head, seq_len)
        for $head$ in $A.num\_heads$ **do**
            summed_attention_scores = $[0_0, 0_i, ...0_{seq\_len}]$ ▷ Array is of shape (seq_len)
            for $token$ in $A.seq\_length$ **do**
                summed_attention_scores += A[$layer$, $batch$, $head$, $token$]
            end for
            heads.append(summed_attention_scores)
        end for
        batch.append(heads)
    end for
    B.append(batch)
end for
return: B