Explainable AI Techniques: Using Transformers to detect Online Toxicity

Thomas GEORGES

April 16, 2025

Contents

1 Introduction

Transformer-based language models such as BERT and MobileBERT have shown remarkable accuracy in text classification tasks, including the detection of toxic content on social media platforms. However, their complexity and depth render them opaque "black boxes," making it challenging to understand why a particular input is labeled as toxic. Due to the sheer sensitivity of the topic, and the consequences that an error can cause, having easy to understand, accurate Explainable AI (XAI) methods is of paramount importance. This report will showcase two major categories: local methods and global methods.

2 Preliminaries: Transformer Architecture and Attention

A transformer layer consists primarily of multi-head self-attention and a feed-forward network, with residual connections and layer normalization. The attention mechanism is a representation of the importance of each element of the input when attempting to reconstruct other elements. This allows the model to access any information from previous inpits without any threshold imposed by functions like in RNN or LSTM architectures. Now we will see a detailed explanation of how attention is constructed.

Since our goal is to detect online toxicity, we will from now on refer to an input as a "token" since it is the main method of segmentation of text that is employed.

For a sequence of N tokens, let $Q, K \in \mathbb{R}^{N\times d_k}$ be the query and key matrices, and $V \in \mathbb{R}^{N\times d_v}$ the value matrix. The attention mechanism computes a weighted sum of values, where weights derive from scaled dot-products of queries and keys:

$$\text{Attention}(Q, K, V) = \text{softmax}\big(\tfrac{QK^\top}{\sqrt{d_k}}\big), V. \tag{1}$$

The matrix $QK^\top/\sqrt{d_k}$ returns raw attention scores for each token pair, scaled by $\sqrt{d_k}$ to prevent large dot-products. Applying the softmax row-wise ensures each token's attention distribution sums to one, producing an $N \times N$ weight matrix that linearly combines rows of V.

There have been more advanced definitions of attention such as multi-head attention, but our goal was just to give a broad definition of the concept.

In the original Transformer architecture, the model is divided into two distinct components: the *encoder* and the *decoder*. Both components employ self-attention mechanisms, but their purposes and configurations differ:

- **Encoder:** Takes as input the sequence of token embeddings and processes it through L identical layers, each comprising a bidirectional self-attention sublayer followed by a position-wise feed-forward sublayer (with residual connections and layer normalization). We call it bi-directional because the attention can access eevery element of the entry vector to reconstruct the meaning of any given token, which would not be the case for attention used in generativeAI tools where we want ro generate new texts and thus do not want to predict the next token using future tokens. The encoder therefore *encodes* the entire source text into a rich semantic embedding.
- **Decoder:** Responsible for autoregressive generation or sequence-to-sequence decoding, it also has L layers, but each layer contains three sublayers in this order:
  1. A *masked* self-attention, which this time is masked since the goal is to generate new tokens. (ensuring tokens are generated left-to-right).
  2. An *encoder-–decoder cross-attention*, in which each target position attends to the full set of encoder outputs, integrating source-side context into the generation process.
  3. A position-wise feed-forward sublayer, again with residual connections and layer normalization.

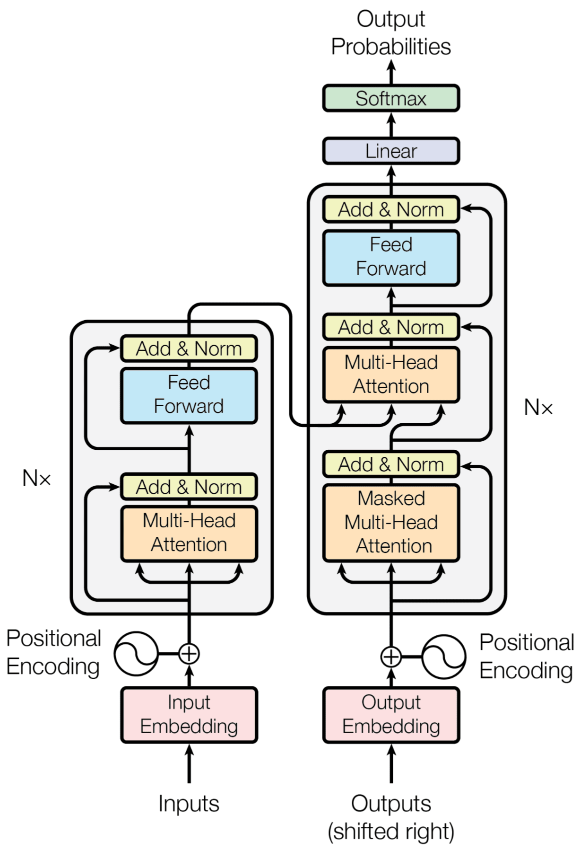


Figure 1: Visual representation of the basic encoder-decoder architecture of a transformer

This design allows the decoder first to process the already-generated (or shifted) target tokens in an autoregressive fashion, then to *decode* information from the source sequence via cross-attention.

We can see from the construction of each aach mechanism how they can differ and for which tasks each of them is well suited for. The encoder excels at creating a comprehensive embedding of the source text, and the decoder excels at generative or conditional tasks (i.e. translation, summarization, text generation) by attending both to its own previous outputs and to encoder-produced context.

BERT (Bi-directional Encoding fpr Representing Transformers) is by his very name an encoder only model, which makes it well suited for summarization and text analysis, which is good for us since we want to predict if a tweet is toxic or not.

We load a pre-trained model (MobileBERT in our case), and then fine-tunes it using a set of available tweets labelled as toxic or not.

We will now look at multiple tools to check how the model does manage to determine that.

3 Local methods for explaining transformers outputs

Much like methods to study the effect of each point in linear regression, such as the Cook's distance, most local methods aim to switch one or more element from an input and check the prediction' impact.

3.1 LIME: Local Interpretable Model-Agnostic Explanations

LIME approximates the complex model f locally around a specific input x by fitting a simple, interpretable surrogate model g over perturbed samples of x. Key components:

- $\tilde{x}$: set of perturbed inputs obtained by randomly masking or altering tokens in x.
- $\pi_x(\tilde{x})$: proximity kernel weighting perturbations by similarity to x.
- $\Omega(g)$: regularization term enforcing simplicity (e.g., sparsity) of the surrogate.

The proximity kernel is often defined as a decaying function of a distance metric $D(x, \tilde{x})$, such as Hamming or cosine distance:

$$\pi_x(\tilde{x}) = \exp\bigl(-D(x, \tilde{x})^2/\sigma^2\bigr). \tag{2}$$

LIME solves the following optimization:

$$g^* = \arg\min_{g \in \mathcal{G}} \sum_{\tilde{x}} \pi_x(\tilde{x})\bigl(f(\tilde{x}) - g(\tilde{x})\bigr)^2 + \Omega(g). \tag{3}$$

For a linear surrogate $g(\tilde{x}) = w^\top \phi(\tilde{x}) + b$, the learned coefficients w_i quantify each token's contribution to $f(x)$ in the neighborhood defined by π_x. The regularizer $\Omega(g)$ (e.g., $\lambda|w|_1$) ensures only a few tokens receive nonzero weights, enhancing interpretability.

3.2 SHAP: Shapley Additive Explanations

SHAP attributes the model output $f(x)$ to individual input features (tokens) using Shapley values from cooperative game theory. Consider the token set $N = 1, 2, \ldots, N$ and let $f_S(x)$ denote the model's output when only tokens in subset S are present (others masked). The Shapley value for token i is:

$$\phi_i(x) = \sum_{S \subseteq N \setminus i} \frac{|S|!, (N - |S| - 1)!}{N!} \Big[f_{S \cup i}(x) - f_S(x) \Big]. \tag{4}$$

Here, each term $f_{S \cup i}(x) - f_S(x)$ is the marginal contribution of token i when joining coalition S. The Shapley weighting $\frac{|S|!,(N-|S|-1)!}{N!}$ averages this marginal contribution over all possible insertion orders. SHAP satisfies four axioms:

- **Efficiency:** $\sum_i \phi_i(x) = f(x) - f(\emptyset)$ ensures total attribution equals the model output difference.
- **Symmetry:** if two tokens contribute identically, they receive equal ϕ.
- **Dummy:** if a token never changes the output, its ϕ is zero.
- **Additivity:** linear combination of models yields sum of Shapley values.

Exact computation requires summing over 2^N subsets; practical implementations (e.g., Kernel SHAP) approximate via weighted regression using the same kernel π_x as in LIME.

3.3 Integrated Gradients and GradCAM

Integrated Gradients (IG) attribute feature importance by integrating the model's gradients along a straight-line path from a baseline input x' (e.g., zero embeddings) to the input x. Denote the model's scalar output as $F(x)$ (e.g., the probability of toxicity). The IG for the ith input dimension is:

$$IG_i(x) = (x_i - x'i) \int 0^1 \frac{\partial F\big(x' + \alpha(x - x')\big)}{\partial x_i}, d\alpha. \tag{5}$$

This integral accumulates the sensitivity of F to feature i over intermediate points. IG satisfies:

- **Completeness:** $\sum_i IG_i(x) = F(x) - F(x')$.
- **Sensitivity:** non-zero change in x_i leads to non-zero attribution if F depends on i.
- **Implementation Invariance:** attributions depend only on F's input-output mapping.

In practice, approximate the integral via Riemann summation with m steps:

$$IG_i(x) \approx (x_i - x'i) \frac{1}{m} \sum k = 1^m \frac{\partial F\big(x' + \frac{k}{m}(x - x')\big)}{\partial x_i}. \tag{6}$$

GradCAM adapts Class Activation Mapping to transformer attention heads. Let $A_h^{(\ell)} \in \mathbb{R}^{N \times N}$ be the attention matrix for head h at layer ℓ. Compute the gradient-based weight:

$$\alpha_h^{(\ell)} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\partial F}{\partial (A_h^{(\ell)})ij}, \tag{7}$$

averaging gradients over all token pairs. Then combine heads:

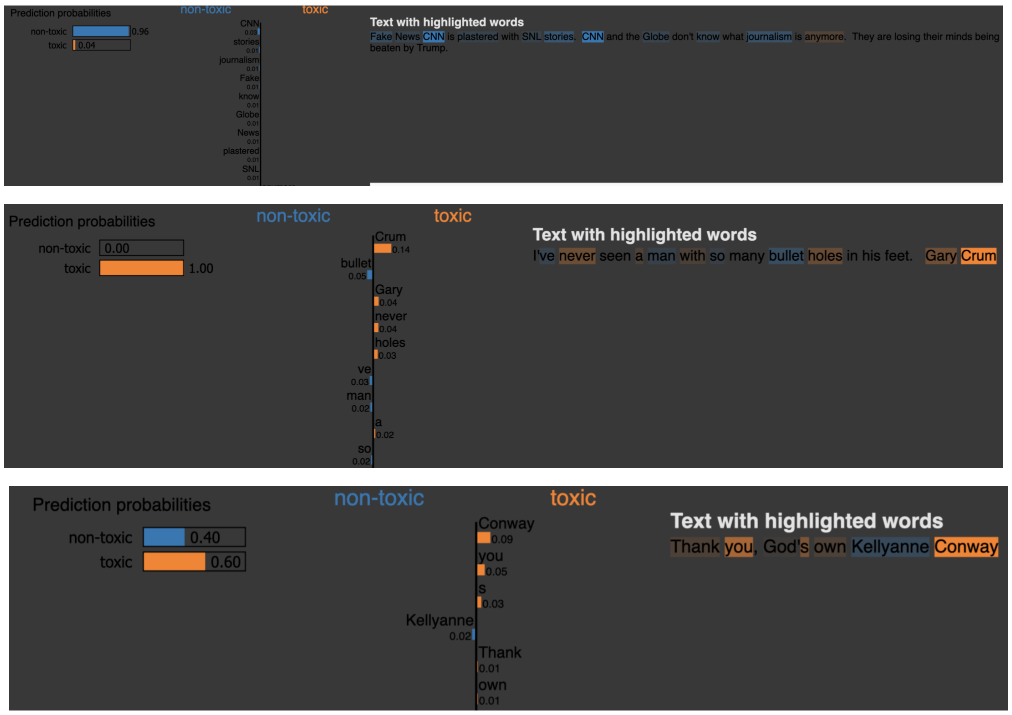
$$M^{(\ell)} = \mathrm{ReLU}\Big(\sum h = 1^H \alpha_h^{(\ell)}, A_h^{(\ell)} \Big), \tag{8}$$

where ReLU retains only positive contributions. Captum implements this via hooks that capture forward attentions and backward gradients.

Even though GradCAM is not very well-spread to study transformers, there are some packages that can show its results, like the package captum. In practice however, the results are rather noisy, and we prefer to leverage the added smoothing by the integral from Integrated Gradient.

4 Results

We will now focus on the results from the LIME methodology, since they are the most easily interpretable through the LimeTextExplainer library in Python. Results are available below.



Fake News CNN is plastered with SNL stories. CNN and the Globe don't know what journalism is anymore. They are losing their minds being beaten by Trump.

I've never seen a man with so many bullet holes in his feet. Gary Crum

Thank you, God's own Kellyanne Conway

Figure 2: Three examples that highlights each token contribution to the gener tion of the output.

We see mixed results: The third outpus, which correctly label the tweet as toxic (even though it is more snarky than really toxic, but that's a larger debate) has identified some sources of toxicity, like Conway. However, the first name of the main source of Toxicity, Kellyanne is this time labelled as a source of nox-toxicity (in Blue). This, combined with the top example where the model incorrectly calls a model non-toxic, even though it clearly is highly inflammatory shows the limit of the model and the methodology. Some tokens like CNN which is labelled as a source of non-toxicity due to its respectability is in this tweet leveraged to attack CNN and the "other medias".. Furtgermore, even though each word independently is not inflammatory in nature, them being aggregated in this way represents a clear attack and should be labelled toxic. But the model doesn't see it that way.

The example in the middle, which is a comment intended toward Senator Gary Crum, shows the limitation of the model: each family word is presented as a major source of toxicity due to its ad-hominem nature, but thr word "bullet", which is the third major source of attack, is perceived as a source of non-toxicity! This is because LIME, and other local methods are working on a per-token basis. They only examine the effect of removing one token, and not a series of tokens, while we know of the auto-regressive nature of language: each word can only be interpreted in context with the previous ones.

This is why we will now focus on methods to explain the global relationships between one token and the others.

5 Global mechanisms for Explaining Transformers

5.1 Attention Rollout

Attention rollout aggregates layer-wise attentions to capture multi-hop token influence [?]. Steps:

1. **Head Averaging:** for each layer ℓ, average across H heads:

$$A^{(\ell)} = \frac{1}{H} \sum_{h=1}^{H} A_h^{(\ell)}. \tag{9}$$

2. **Residual Addition:** add identity to account for skip connections:

$$\widetilde{A}^{(\ell)} = A^{(\ell)} + I_N. \tag{10}$$

3. **Layer-wise Multiplication:** multiply from layer 1 to L:

$$R = \widetilde{A}^{(1)} \times \widetilde{A}^{(2)} \times \cdots \times \widetilde{A}^{(L)}. \tag{11}$$

   Entry R_{ij} quantifies the total influence of token j on token i through all layers.

4. **Row Normalization (optional):**

$$R_{i,:} \leftarrow \frac{R_{i,:}}{\sum_{j=1}^{N} R_{ij}}. \tag{12}$$

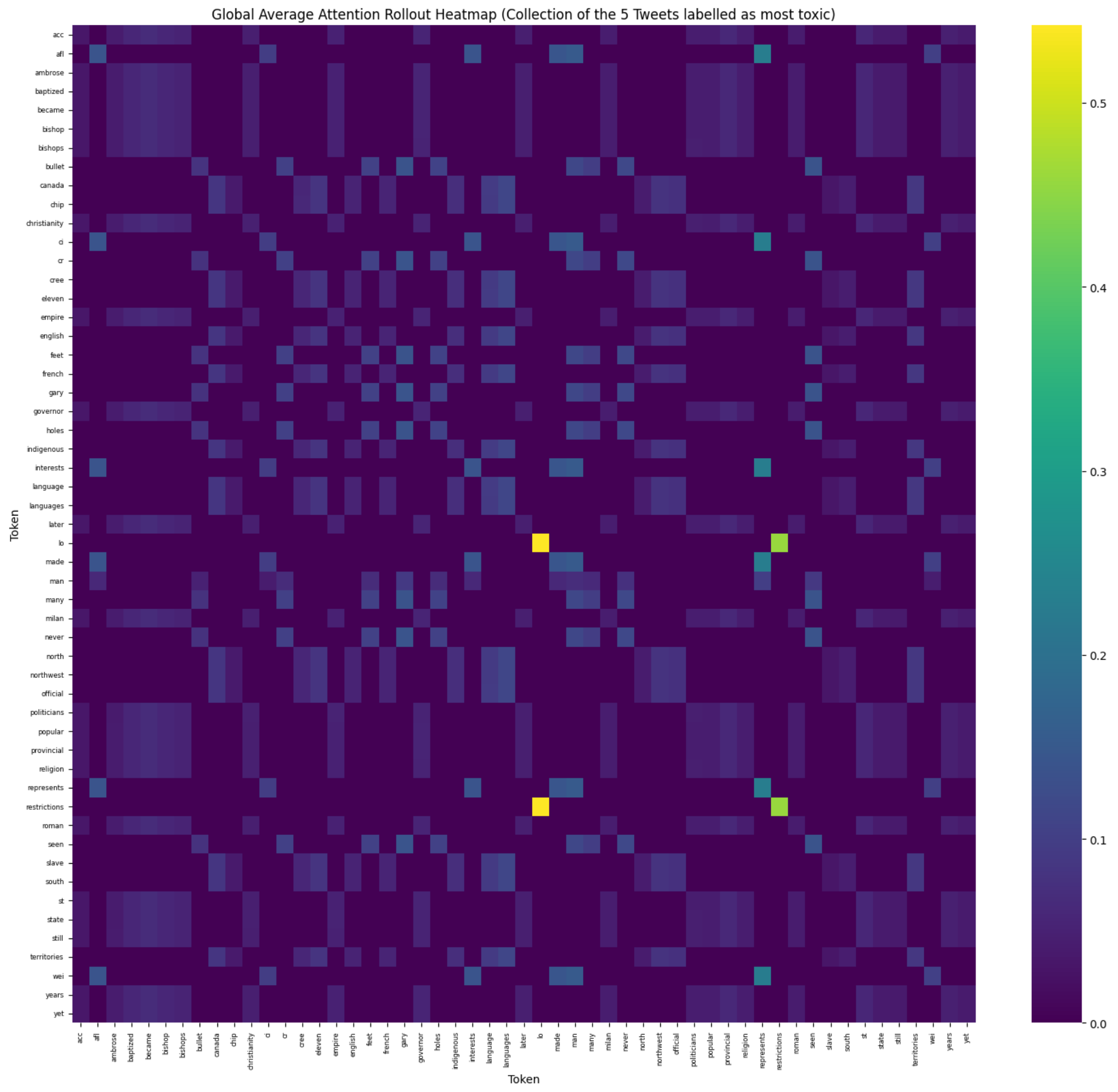
Averaging R over multiple inputs yields a global influence matrix for corpus-wide interpretability.

We will now visualize the attention rollout matrix by representing a heatmap. We first select a couple of relevant tweets, look at the tokens involved, compute the attention rollout matrix for each tweet, and then concatenate/aggregate them to form the aggregate attention matrix. This step is very important, as the attention matrix is only valid for each context of a tweet. That's why we can't compute one single attention matrix. Here are a quick example of one potential attention matrix for the top 5 most toxic tweets according to the model.
Even though this is a heatmap like one produced from GradCAM, the source of it is very different. The GradCAM comes from a local perturbation inside the tweet itself, pinpointing the reasons behind the model's decision, while the attention rollout is a representation of how each token's influence flows through the model when generating other tokens.

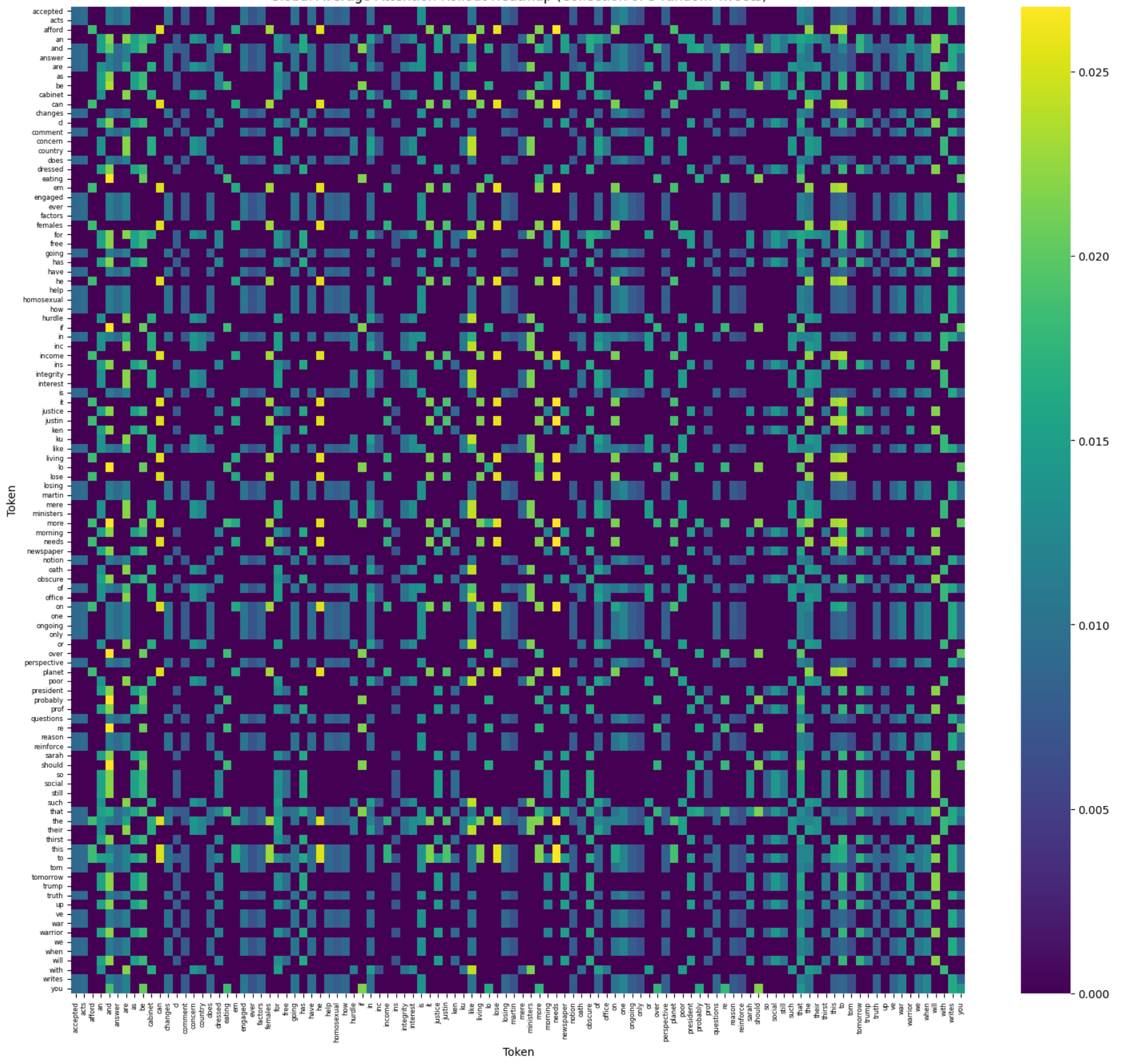
We can see that even though we have cleaned the data to remove all the 1 letter tokens and other tokens like number tokens, the biggest link can be fund between tokens "would", and "win", and there is no discernable link between terms usually marked as toxics (insults, threatening words etc...). that can be due to the model's inability to label toxic tweets as toxic, as we saw previously. The top 5 most toxic tweets according to the model are not inflammatory, so a better model would be able to find the true toxic tweets and plot their representation accordingly. Furthermore, we would need far more truely toxic tweets to be able to represent the links between toxic tokens. This could be plotted with an alternative method, maybe some graph networks. Nevertheless, it remains an elegant form to represent links between tokens in an architecture, as shown in the second graph, this time for 5 random tweets.

6 Conclusion

In this study, we have shown multiple ways to generate representations of the inner-workings of a model aimed at discovering toxicity in online content. The two approaches: local and global are very different and each offers specific information that the other can't access to. However, there isn"t for now any holistic approach that reunites them behing a single method to understand the model in its entirety.



Global Average Attention Rollout Heatmap (Collection of 5 random Tweets)


Token
Token