Generalized Attention Flow: Feature Attribution for Transformer Models via Maximum Flow

Anonymous authors
Paper under double-blind review

Abstract

This paper introduces Generalized Attention Flow, a novel feature attribution method for Transformer models that addresses the limitations of existing approaches. By generalizing Attention Flow and substituting attention weights with an arbitrary Information Tensor, the method leverages attention weights, their gradients, maximum flow, and the barrier method to generate more accurate feature attributions. The proposed approach demonstrates important theoretical properties and resolves issues associated with previous methods that rely solely on simple aggregation of attention weights. Comprehensive benchmarking in NLP sequence classification tasks reveals that a specific variant of Generalized Attention Flow consistently outperforms state-of-the-art feature attribution methods across most evaluation scenarios, offering a more accurate explanation of Transformer model outputs.

1 Introduction

Feature attribution methods are essential for building interpretable machine learning models. These methods assign a score to each input feature, reflecting its importance to the model's output, thereby facilitating the understanding of model predictions.

The rise of Transformer models with self-attention mechanisms has necessitated feature attribution methods for interpreting these models (Vaswani et al., 2017; Bahdanau et al., 2016; Devlin et al., 2019; Sanh et al., 2020; Dosovitskiy et al., 2020; Kobayashi et al., 2021). Initially, attention weights were considered potential feature attributions, but recent studies have questioned their effectiveness in explaining deep neural networks (Abnar & Zuidema, 2020; Clark et al., 2019; Jain & Wallace, 2019; Serrano & Smith, 2019). Consequently, various post hoc methods have been developed to obtain feature attributions for Transformer models.

Recent advancements in XAI have introduced numerous gradient-based methods, including Grads and AttGrads (Barkan et al., 2021), which leverage saliency to interpret Transformer outputs. Qiang et al. (2022) proposed AttCAT, integrating features, their gradients, and attention weights to quantify input influence on model outputs. However, many of these techniques still focus primarily on gradients of attention weights, inheriting limitations of earlier attention-based approaches.

Layer-wise Relevance Propagation (LRP) (Bach et al., 2015; Voita et al., 2019) transfers relevance scores from output to input. Chefer et al. (2021a;b) proposed a comprehensive method enabling information propagation through all Transformer components. However, this approach relies on specific LRP rules, limiting its applicability across various Transformer architectures.

Many existing methods for evaluating feature attributions in Transformers fail to capture pairwise interactions among features. This limitation arises from the independent computation of importance scores, which neglects feature interactions. For example, when calculating gradients of attention weights, they propagate directly from the output to the individual input feature, ignoring interactions. Additionally, many methods used to compute feature attributions in Transformers violate key axioms such as symmetry, sensitivity, efficiency, and linearity (Shapley, 1952; Sundararajan et al., 2017; Sundararajan & Najmi, 2020) (Sec. 3.5).

Recently, Abnar & Zuidema (2020) introduced Attention Flow to overcome these limitations in XAI methods. Attention Flow considers attention as capacities in a maximum flow problem, determining feature attributions based on the solution. This approach naturally captures the influence of attention mechanisms, as the paths of high attention through a network correspond to the flow of information from features to outputs. Applicable to any encoder-only Transformer, Attention Flow has demonstrated strong potential to improve model interpretability (Abnar & Zuidema, 2020; Modarressi et al., 2023; Kobayashi et al., 2021).

Subsequently, Ethayarajh & Jurafsky (2021) sought to connect attention flows and XAI by utilizing Shapley values (Shapley, 1952). While they aimed to show that attention flows could be interpreted as Shapley values under certain conditions, they overlooked the issue of non-uniqueness in such flows (Sec. 3.3).

Our contributions. We propose Generalized Attention Flow, which satisfies important theoretical properties and enhanced empirical performance. Specifically, our contributions are:

1. We introduce Generalized Attention Flow, an extension of the previously described Attention Flow. In this approach, feature attributions are generated by using the log barrier method to solve a regularized maximum flow problem within a capacity network formed from the functions applied to attention weights. Instead of defining capacities based solely on attention weights, we suggest using the gradients of these weights (GF) or the product of attention weights and their gradients (AGF) as alternatives.

2. We have addressed the non-uniqueness issue in Attention Flow, which invalidates some of its previously suggested theoretical properties (Ethayarajh & Jurafsky, 2021). Furthermore, we show that non-unique solutions occur frequently in practice. We have introduced barrier regularization to mitigate this issue to ensure a unique solution. As a result, we have demonstrated that feature attributions derived from the regularized maximum flow problem align with Shapley values and satisfy the axioms of efficiency, symmetry, nullity, and linearity (Shapley, 1952; Young, 1985; Chen et al., 2023b).

3. We extensively benchmarked the proposed feature attribution methods, defined using Generalized Attention Flow, against various existing state-of-the-art attribution methods. We found that a type of the proposed attribution methods outperforms previous state-of-the-art methods in terms of explanation performance for classification tasks across most evaluation scenarios, as measured by AOPC (Barkan et al., 2021; Nguyen, 2018; Chen et al., 2020), LOdds (Chen et al., 2020; Shrikumar et al., 2018), and classification metrics.

4. We have developed an open-source Python package for calculating feature attributions using Generalized Attention Flow. This package is distinctively flexible, capable of being applied to any encoder-only Transformer model available in the Hugging Face Transformers package (Wolf et al., 2020). Furthermore, our methods are easily adaptable for various NLP tasks.

2 Preliminaries

2.1 Multi-Head Attention Mechanism

Given the input sequence $\boldsymbol{X} \in \mathbb{R}^{t \times d}$, where d is the dimensionality of the model's input vectors and t is the number of tokens, the multi-head self-attention mechanism computes attention weights for each element in the sequence employing the following steps:

- **Linear Transformation**:

$$\boldsymbol{Q}_i = \boldsymbol{X}\boldsymbol{W}_i^Q, \quad \boldsymbol{K}_i = \boldsymbol{X}\boldsymbol{W}_i^K, \quad \boldsymbol{V}_i = \boldsymbol{X}\boldsymbol{W}_i^V \tag{1}$$

  Here $\boldsymbol{Q}_i, \boldsymbol{K}_i \in \mathbb{R}^{t \times d_k}$ and $\boldsymbol{V}_i \in \mathbb{R}^{t \times d_v}$, where d_k and d_v represent the dimensionality of the key vector and value vector respectively, and i represents the index of the attention head.

- **Scaled Dot-Product Attention**:

$$\text{Attention}_i(\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i) = \widetilde{\boldsymbol{A}}_i \boldsymbol{V}_i \tag{2}$$

  where the matrix of attention weights $\widetilde{\boldsymbol{A}}_i \in \mathbb{R}^{t\times t}$ is defined as:

$$\widetilde{\boldsymbol{A}}_i = \text{softmax}\left(\frac{\boldsymbol{Q}_i \boldsymbol{K}_i^T}{\sqrt{d_k}}\right) \tag{3}$$

- **Concatenation and Linear Projection**:

$$\text{MultiHead}(\boldsymbol{X}) = \text{Concat}(\text{Attention}_1, \text{Attention}_2, \ldots, \text{Attention}_h)\boldsymbol{W}^O \tag{4}$$

  where $\text{MultiHead}(\boldsymbol{X}) \in \mathbb{R}^{t\times d}$ and $\boldsymbol{W}^O \in \mathbb{R}^{h\cdot d_v \times d}$.

For a Transformer with l attention layers, the attention weights at each layer can be defined as multi-head attention weights:

$$\widehat{\boldsymbol{A}} = \text{Concat}(\widetilde{\boldsymbol{A}}_1, \widetilde{\boldsymbol{A}}_2, \ldots, \widetilde{\boldsymbol{A}}_h) \in \mathbb{R}^{h\times t\times t} \tag{5}$$

Extending this to a Transformer architecture itself, the Transformer attention weights $\boldsymbol{A}$ can be defined as:

$$\boldsymbol{A} = \text{Concat}(\widehat{\boldsymbol{A}}_1, \widehat{\boldsymbol{A}}_2, \ldots, \widehat{\boldsymbol{A}}_l) \in \mathbb{R}^{l\times h\times t\times t} \tag{6}$$

where $\widehat{A}_j \in \mathbb{R}^{h\times t\times t}$ is the multi-head attention weight for the j-th attention layer.

2.2 Minimum-Cost Circulation & Maximum Flow Problem

Definition 2.1 (Minimum Cost Circulation). *Given a network $G = (V, E, \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{c})$ with $|V| = n$ vertices and $|E| = m$ edges, where c_{ij} is the cost, $l_{i,j}$ and $u_{i,j}$ are respectively the lower and upper capacities (demands) for the edge $(i, j) \in E$, circulation is a function $f : E \to \mathbb{R}^{\geq 0}$ s.t.*

$$\begin{aligned} &l_{ij} \leq f_{ij} \leq u_{ij}, && \forall (i,j) \in E \\ &\sum_{j:(i,j)\in E} f_{ij} - \sum_{j:(j,i)\in E} f_{ji} = 0, && \forall i \in V. \end{aligned} \tag{7}$$

The min-cost circulation problem is to find a circulation f minimizing the cost function $\sum_{(i,j)\in E} c_{ij} f_{ij}$.

The minimum-cost circulation problem can be algebraically written as the following primal-dual linear programming (LP) problem (Van Den Brand et al., 2021; Chen et al., 2023a):

$$\text{(Primal)} \quad \underset{\substack{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{0} \\ l_e \leq f_e \leq u_e \forall e \in E}}{\arg\min} \; \boldsymbol{c}^\top \boldsymbol{f} \quad \text{i.e.} \quad \underset{\substack{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{0} \\ \boldsymbol{l} \leq \boldsymbol{f} \leq \boldsymbol{u}}}{\arg\min} \; \boldsymbol{c}^\top \boldsymbol{f}, \quad \text{(Dual)} \quad \underset{\boldsymbol{B}\boldsymbol{y} + \boldsymbol{s} = \boldsymbol{c}}{\arg\max} \sum_i \min\left(l_i s_i, u_i s_i\right) \tag{8}$$

where $\boldsymbol{B}_{m\times n}$, is the edge-vertex incidence matrix. For a directed graph, the entries of the matrix $\boldsymbol{B}$ are defined as follows:

$$\boldsymbol{B}_{ev} = \begin{cases} -1, & \text{if vertex } v \text{ is the tail of edge } e, \\ 1, & \text{if vertex } v \text{ is the head of edge } e, \\ 0, & \text{if edge } v \text{ is not incident to vertex } e. \end{cases}$$

Remark 2.1. *The maximum flow problem can be considered as a specific minimum-cost circulation problem. Here, $\boldsymbol{B}$ is an edge-vertex incidence matrix of the input graph after we added to it an edge $e(t, s)$ that connects the target t to the source s and its lower capacity $l_{t,s}$ be 0 and its upper capacity $u_{t,s}$ be $\|\boldsymbol{u}\|_1$. Also, the cost vector c is a vector in which $c_{t,s} = -1$ and $c_e = 0$ for all other edges $e \in E$ (Cormen et al., 2009).*

2.3 Barrier Methods for Constrained Optimization

Consider the following optimization problem:

$$f^* = \underset{\substack{\alpha(\boldsymbol{f})=\mathbf{0} \\ \beta(\boldsymbol{f})\leq\mathbf{0}}}{\arg\min}\ \xi(\boldsymbol{f}) \tag{9}$$

where h represents a convex inequality constraint, g represents an affine equality constraint, and $\boldsymbol{f}^*$ denote the optimal solution.

The interior of the constraint region is defined as $S = \{\boldsymbol{f} \mid \alpha(\boldsymbol{f}) = 0,\ \beta(\boldsymbol{f}) < 0\}$. Assuming S is nonempty and convex, we introduce a barrier function $\psi(\boldsymbol{f})$ on S that is continuous and approaches infinity as $\boldsymbol{f}$ approaches to the boundary of the region, specifically $\lim_{\beta(\boldsymbol{f})\to 0^-} \psi(\boldsymbol{f}) = \infty$. One common example of barrier functions is the log barrier function, which is represented as $\log(-\beta(\boldsymbol{f}))$.

Given a barrier function $\psi(\boldsymbol{f})$, we can define a new objective function $\xi(\boldsymbol{f}) + \mu\psi(\boldsymbol{f})$, where μ is a positive real number, which enables us to eliminate the inequality constraints in the original problem and obtain the following problem:

$$\boldsymbol{f}^*_\mu = \underset{\alpha(\boldsymbol{f})=\mathbf{0}}{\arg\min}\, \xi(\boldsymbol{f}) + \mu\psi(\boldsymbol{f}) \tag{10}$$

Theorem 2.1. *For any strictly convex barrier function $\psi(\boldsymbol{f})$, convex function $\xi(\boldsymbol{f})$, and $\mu > 0$, there exists a unique optimal point $\boldsymbol{f}^*_\mu$. Furthermore, $\lim_{\mu\to 0} \boldsymbol{f}^*_\mu = \boldsymbol{f}^*$, indicating that for any arbitrary $\epsilon > 0$, we can select a sufficiently small $\mu > 0$ such that $\|\boldsymbol{f}^*_\mu - \boldsymbol{f}^*\| < \epsilon$ (van den Brand et al., 2023).*

3 Methods

3.1 Information Tensor

In Transformer-based networks, information propagation occurs through pathways facilitated by the attention mechanism. These pathways can be conceptualized as routes within a graph structure, where tokens are represented by nodes and computations are denoted by edges. The capacities of these edges correspond to meaningful computational quantities that reflect the flow of information through the network (Ferrando & Voita, 2024; Mueller, 2024).

Attention scores can represent the flow of information through the neural network during the feed-forward phase of training, quantifying the importance of different input parts in generating the output (Abnar & Zuidema, 2020; Ferrando & Voita, 2024). Additionally, the gradient of attention scores captures the flow of information during back-propagation, reflecting how changes in the output influence the attention mechanism throughout the network (Barkan et al., 2021). A combined view of attention scores and their gradients can simultaneously represent information circulation during both feed-forward and back-propagation, offering a comprehensive perspective on the network's information dynamics (Barkan et al., 2021; Qiang et al., 2022; Chefer et al., 2021a;b).

Our Generalized Attention Flow builds on this foundation, using an information tensor $\bar{\boldsymbol{A}} \in \mathbb{R}^{l\times t\times t}$ to aggregate Transformer attention weights $\boldsymbol{A}$, as defined in eq. 6. Based on the insights above, we propose three aggregation functions to generate information tensors (Barkan et al., 2021; Chefer et al., 2021a):

1. **Attention Flow (AF):** $\bar{\boldsymbol{A}} := \mathbb{E}_h(\boldsymbol{A})$
2. **Attention Grad Flow (GF):** $\bar{\boldsymbol{A}} := \mathbb{E}_h(\lfloor\nabla\boldsymbol{A}\rfloor_+)$
3. **Attention × Attention Grad Flow (AGF):** $\bar{\boldsymbol{A}} := \mathbb{E}_h(\lfloor\boldsymbol{A}\odot\nabla\boldsymbol{A}\rfloor_+)$

Here, $\lfloor x\rfloor_+ = \max(x, 0)$, $\odot$ represents the Hadamard product, $\nabla\boldsymbol{A} := \frac{\partial y_t}{\partial \boldsymbol{A}}$ where y_t is the model's scalar output, and $\mathbb{E}_h$ denotes the mean across attention heads.

Algorithm 1 Backward Information Capacity

Input: $\bar{\boldsymbol{A}}_{l\times t\times t}$: An information tensor.
Output: Tuple: $(\mathcal{A}, \boldsymbol{l}, \widetilde{\boldsymbol{l}}, \boldsymbol{u}, \widetilde{\boldsymbol{u}}, ss, st)$

```
function GET_BACKWARD_CAPACITY(Ā)
    ▷ Initialization
    l, t, _ ← Ā.shape()
    β_min ← min(Ā > 0)
    β ← −⌊log_10(β_min)⌋
    γ ← 10^β
    Q_tl ← t * (l + 1) + 2
    l ← zeros(Q_tl, Q_tl)
    u ← zeros(Q_tl, Q_tl)
    u_∞ ← t
    ▷ Fill super-source → First Layer
    for i in range(t) do
        u[i + 1][0] ← u_∞
    end for
    ▷ Fill Last Layer → super-target
    for i in range(t) do
        u[−1][−i − 2] ← u_∞
    end for
    ▷ Fill j-th Layer to (j + 1)-th Layer
    for j in range(l) do
        start ← t * j + 1
        mid ← t * (j + 1) + 1
        end ← t * (j + 2) + 1
        u[mid:end , start:mid] ← Ā_[j,:,:]
    end for
    ▷ Get Integral Version of Capacities
    l̃ ← int(γ * l)
    ũ ← int(γ * u)
    ▷ Get Adjacency Matrix
    𝒜 ← 𝕀_(u>0)
    ▷ Get super-source and super-target
    ss, st ← t * (l + 1) + 1, 0
end function
```

Algorithm 2 Forward Information Capacity

Input: $\bar{\boldsymbol{A}}_{l\times t\times t}$: An information tensor.
Output: Tuple: $(\mathcal{A}, \boldsymbol{l}, \widetilde{\boldsymbol{l}}, \boldsymbol{u}, \widetilde{\boldsymbol{u}}, ss, st)$

```
function GET_FORWARD_CAPACITY(Ā)
    ▷ Initialization
    l, t, _ ← Ā.shape()
    β_min ← min(Ā > 0)
    β ← −⌊log_10(β_min)⌋
    γ ← 10^β
    Q_tl ← t * (l + 1) + 2
    l ← zeros(Q_tl, Q_tl)
    u ← zeros(Q_tl, Q_tl)
    u_∞ ← t
    ▷ Fill super-source → First Layer
    for i in range(t) do
        u[0][i + 1] ← u_∞
    end for
    ▷ Fill Last Layer → super-target
    for i in range(t) do
        u[−i − 2][−1] ← u_∞
    end for
    ▷ Fill j-th Layer to (j + 1)-th Layer
    for j in range(l) do
        start ← t * j + 1
        mid ← t * (j + 1) + 1
        end ← t * (j + 2) + 1
        u[start:mid , mid:end] ← Ā^T_[j,:,:]
    end for
    ▷ Get Integral Version of Capacities
    l̃ ← int(γ * l)
    ũ ← int(γ * u)
    ▷ Get Adjacency Matrix
    𝒜 ← 𝕀_(u>0)
    ▷ Get super-source and super-target
    ss, st ← 0, t * (l + 1) + 1
end function
```

3.2 Generalized Attention Flow

In Generalized Attention Flow, we leverage the attention mechanism for feature attribution by defining a network flow representation of a Transformer or other attention-based model. We assign capacities to the edges of this graph corresponding to information tensor defined in Sec. 3.1. We then solve the maximum flow problem to evaluate the optimal flow passing through any output node (or, more generally, any node in any layer) to any input node. The flow traversing through an input node (token) indicates the importance or attribution of that particular node (token).

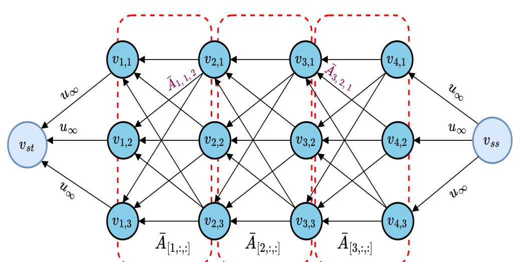
To determine the maximum flow from all output nodes to all input nodes, we leverage the concept of multi-commodity flow (App. A.2 and App. B). This involves the introduction of a super-source node ss and a super-target node st with a large capacity u_∞. The connectivity between layers and capacities between nodes are established using the information tensors, effectively forming a layered graph (App. B).

To formalize the generating of the information flow, consider a Transformer with l attention layers, an input sequence $X \in \mathbb{R}^{t\times d}$, and its information tensor $\bar{\boldsymbol{A}} \in \mathbb{R}^{l\times t\times t}$. We construct the layered attribution graph $\mathcal{G}$ with its adjacency matrix $\mathcal{A}$, its edge-vertex incidence matrix $\boldsymbol{B}$, lower capacity matrix $\boldsymbol{l}$ and its integral version $\widetilde{\boldsymbol{l}}$, upper capacity matrix $\boldsymbol{u}$ and its integral version $\widetilde{\boldsymbol{u}}$ employing either Algorithm 1 or Algorithm 2. Afterward, we substitute the vectorized version of the obtained matrices into the primal form of eq. 8 to evaluate the desired optimal flow.

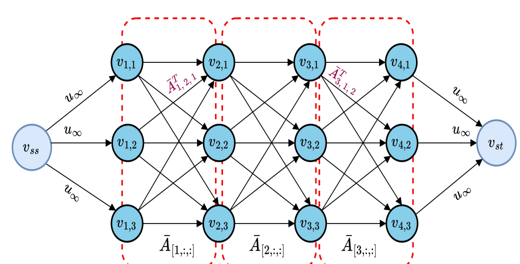
To enhance comprehension of Algorithm 1 and Algorithm 2, we explain the process of constructing the layered attribution graph $\mathcal{G}$ which has an adjacency matrix with shape $(2 + t * (l + 1), 2 + t * (l + 1))$ and serves as an input for the maximum flow problem. Designating nodes at layer $\ell \in \{1, \dots, l\}$ and token $i \in \{1, \dots, t\}$ as $v_{\ell,i}$, the guidelines for defining the upper and lower-bound capacities are as follows:

- To connect nodes $v_{1,i}$ to the super-target node v_{st}, we define $\boldsymbol{u}[0, i] = u_\infty$ for $1 \le i \le t$.
- The upper-bound capacity from node $v_{\ell+1,i}$ to node $v_{\ell,j}$ is defined as $\boldsymbol{u}[\mathrm{I}_{i,\ell+1}, \mathrm{I}_{j,\ell}] = \bar{\boldsymbol{A}}_{\ell,i,j}$ for $\ell \in \{1, \dots, l\}$, $i \in \{1, \dots, t\}$, and $j \in \{1, \dots, t\}$, where $\mathrm{I}_{i,\ell+1} = i + t * \ell$ and $\mathrm{I}_{j,\ell} = j + t * (\ell - 1)$.
- To connect the super-source node v_{ss} to nodes $v_{l+1,i}$, we define $\boldsymbol{u}[t * l + i, 1 + t * (l + 1)] = u_\infty$ for $1 \le i \le t$.
- The lower-bound capacity is defined as $\boldsymbol{l} = \boldsymbol{0}$.

Fig. 1a and Fig. 1b illustrate schematic graphs generated using the information tensor $\bar{\boldsymbol{A}} \in \mathbb{R}^{3\times3\times3}$ with Algorithm 1 and Algorithm 2, respectively. While both algorithms are identically solving the same network flow problem by creating graphs containing a super-source and a super-target, the second algorithm differs from the first in two key aspects. First, in the second graph, the positions of the super-source and super-target are swapped, meaning that the super-source in the first graph becomes the super-target in the second and vice versa. Second, the direction of the edges in the second graph is reversed compared to the first.



(a) Schematic information flow created via Algorithm 1.



(b) Schematic information flow created via Algorithm 2.

Figure 1: Schematics overview of Generalized Attention Flow created employing Algorithm 1 and Algorithm 2.

3.3 Non-uniqueness of Maximum Flow

The maximum flow problem lacks strict convexity, meaning it does not necessarily have a unique solution. We found that the maximum flow problem associated with the graphs constructed employing Generalized Attention Flow also fails to yield a unique optimal flow (App. C).

Observation 3.1. *It is straightforward to verify that both Algorithm 1 and Algorithm 2 solve the same maximum flow problem. Therefore, determining the maximum flow in graphs generated by either Algorithm 1 or Algorithm 2 is equivalent and yields the same optimal value. However, it's worth noting that the optimal flows associated with them may not necessarily be equivalent, as explained in App. C.*

Observation 3.2. *If two distinct feasible solutions, denoted f_1 and f_2, exist for a linear programming problem, then any convex combination $\gamma_1 f_1 + \gamma_2 f_2$ forms another feasible solution. Consequently, the maximum flow problem can possess an infinite number of feasible solutions. Additionally, due to the non-uniqueness of optimal flows arising from the maximum flow problem, their projections onto any subset of nodes in the graph may also not be unique.*

Corollary 3.1. *Let V be the set of all nodes in a layered attribution graph $\mathcal{G}(\mathcal{A}, \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{c}, ss, st)$, and $N \subseteq V$, with all nodes in N chosen from the same layer. Suppose $\boldsymbol{f}^*$ is the optimal solution of eq. 8, and for every $S \subseteq N$, define the payoff function $\vartheta(S) := |\boldsymbol{f}^*(S)| = \sum_{i \in S} |f_{out}(i)|$, where $|f_{out}(i)|$ denotes the total outflow value of node i. Although Ethayarajh & Jurafsky (2021) claimed that for each node $i \in N$, $\phi_i(\vartheta) = |\boldsymbol{f}^*_{out}(i)|$ represents the Shapley value, these feature attributions are non-unique and cannot be considered Shapley values. In fact, their method for defining feature attributions is not well-defined (Proof in App. E).*

3.4 Log Barrier Regularization of Maximum Flow

To address the non-uniqueness issues in the maximum flow problem, we reformulate the minimum-cost circulation problem as follows:

$$\underset{\substack{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{0} \\ \beta(\boldsymbol{f}) \leq \boldsymbol{0}}}{\arg\min} \ \boldsymbol{c}^\top \boldsymbol{f} \tag{11}$$

where $\beta(\boldsymbol{f}) = (\boldsymbol{f} - \boldsymbol{l})(\boldsymbol{f} - \boldsymbol{u})$. The original problem can, therefore, be approximated using the log barrier function as the following optimization problem:

$$\underset{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{0}}{\arg\min} \, \boldsymbol{c}^\top \boldsymbol{f} + \psi_\mu(\boldsymbol{f}) \tag{12}$$

where the log barrier function is:

$$\psi_\mu(\boldsymbol{f}) = -\mu \sum_{e \in E} \log\left(-\beta(f_e)\right) = -\mu \sum_{e \in E} \left(\log\left(f_e - l_e\right) + \log\left(u_e - f_e\right)\right) \tag{13}$$

It is evident that, as long as $\mu > 0$ and our initial solution is feasible, the barrier function guarantees that any solution obtained through an iterative minimization scheme, like interior point methods, remains feasible (Bubeck, 2015; Boyd & Vandenberghe, 2004; Mądry, 2019). Furthermore, it can be demonstrated that to obtain an ε-approximate solution to eq. 11, it suffices to set $\mu \leq \frac{\varepsilon}{2m}$ and find the optimal solution to the corresponding problem in eq. 12 (Bubeck, 2015; Boyd & Vandenberghe, 2004; Mądry, 2019).

Finally, the Hessian of the objective function in eq. 11 at some point $\boldsymbol{f}$ is equal to the Hessian $\nabla^2 \psi_\mu(x)$ of the barrier function, which is positive definite (assuming $\mu > 0$). This implies that the objective function is strictly convex and, consequently, eq. 12 has a unique feasible solution (Bubeck, 2015; Boyd & Vandenberghe, 2004).

3.5 Axioms of Feature Attributions

In XAI, axioms are core principles that guide the evaluation of explanation methods, ensuring their reliability, interpretability, and fairness. These axioms provide standards to measure the effectiveness and compliance of explanation techniques. Our proposed methods meet five essential axioms, as demonstrated by the following theorem and corollaries.

Definition 3.1 (Shapley values). *For any value function $\vartheta : 2^N \mapsto \mathbb{R}$ where $N = \{1, 2, \ldots, n\}$, Shapley values $\phi(\vartheta) \in \mathbb{R}^n$ is computed by averaging the marginal contribution of each feature over all possible feature combinations as:*

$$\phi_i(\vartheta) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (\vartheta(S \cup \{i\}) - \vartheta(S)) \tag{14}$$

Shapley values are the unique explanation that satisfies four fairness-based axioms of efficiency (completeness), symmetry, linearity (additivity), and nullity (Shapley, 1952; Young, 1985) (App. A.3). Initially, a payoff function based on model accuracy was proposed (Lundberg & Lee, 2017); however, since then, various alternative payoff functions have been introduced (Jethani et al., 2022; Sundararajan & Najmi, 2020), each yielding distinct feature importance scores.

Theorem 3.1 (Log Barrier Regularization of Generalized Attention Flow Outcomes Shapley Values). *Given a layered attribution graph $\mathcal{G}(\mathcal{A}, \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{c}, ss, st)$ defined using either Algorithm 1 or Algorithm 2, let V be the set of all nodes in $\mathcal{G}$, and $N \subseteq V$ such that all nodes in N are chosen from the same layer. Suppose $\boldsymbol{f}^*$ is the optimal unique solution of eq. 12, and for every $S \subseteq N$, define the payoff function $\vartheta(S) := |\boldsymbol{f}^*(S)| = \sum_{i \in S} |f_{out}(i)|$ where $|f_{out}(i)|$ is the total outflow value of a node i. Then, it can be proven that for each node $i \in N$, $\phi_i(\vartheta) = |\boldsymbol{f}^*_{out}(i)|$ represents the Shapley value (Proof in App. E).*

Corollary 3.2. *Theorem 3.1 implies that the feature arbitration obtained by eq. 12 are Shapley values and, consequently, adhere to the axioms of **efficiency**, **symmetry**, **nullity**, **linearity**.*

4 Experiments

In this section, we thoroughly evaluate the effectiveness of our methods for sequence classification. While our approach is versatile and applicable to various NLP tasks, including question answering and named entity recognition, which use encoder-only Transformer architectures, this assessment focuses solely on sequence classification.

4.1 Transformer Models

Transformer models have demonstrated exceptional performance in various NLP tasks including sequence classification, question answering, and named entity recognition. In our evaluations, we used a specific pretrained model from the HuggingFace Hub (Wolf et al., 2020) for each dataset and compared our explanation methods against others to assess their performance (App. F.1).

4.2 Datasets

Our method's assessment involves sequence classification spanning binary classification tasks on datasets including SST2 (Socher et al., 2013), Amazon Polarity (McAuley & Leskovec, 2013), Yelp Polarity (Zhang et al., 2016), and IMDB (Maas et al., 2011), alongside multi-class classification on the AG News dataset (Zhang et al., 2015). To minimize computational overhead, we conducted experiments on a subset of 5,000 randomly selected samples for the Amazon, Yelp, and IMDB datasets while utilizing the entire test sets for other datasets (App. F.1).

4.3 Benchmark Methods

Our experiment compares the methods introduced in Sec. 3.1 with various baseline explanation methods tailored for Transformer models. To evaluate attention-based methods such as RawAtt and Rollout (Abnar & Zuidema, 2020), attention gradient-based methods like Grads, AttGrads (Barkan et al., 2021), CAT, and AttCAT (Qiang et al., 2022), as well as LRP-based methods such as PartialLRP (Voita et al., 2019) and TransAtt (Chefer et al., 2021a), we adapted the repository developed by (Qiang et al., 2022). Additionally, we implemented classical attribution methods such as Integrated Gradient (Sundararajan et al., 2017), KernelShap (Lundberg & Lee, 2017), and LIME (Ribeiro et al., 2016) leveraging the Captum package (Kokhlikyan et al., 2020).

4.4 Evaluation Metric

AOPC: One of the key evaluation metrics employed is the Area Over the Perturbation Curve (AOPC), a measure that quantifies the impact of masking top $k\%$ tokens on the average change in prediction probability across all test examples. The AOPC is calculated as follows:

$$\mathrm{AOPC}(k) = \frac{1}{N}\sum_{i=1}^{N} p\left(\hat{y}|\mathbf{x}_i\right) - p\left(\hat{y}|\tilde{\mathbf{x}}_i^k\right) \tag{15}$$

where N is the number of examples, $\hat{y}$ is the predicted label, $p(\hat{y}|\cdot)$ is the probability on the predicted label, and $\tilde{\mathbf{x}}_i^k$ is constructed by masking the $k\%$ top-scored tokens from $\mathbf{x}_i$. To avoid arbitrary choices for k, we systematically mask $10\%, 20\%, \ldots, 90\%$ of the tokens in order of decreasing saliency, resulting in $\tilde{\mathbf{x}}_i^{10}, \tilde{\mathbf{x}}_i^{10}, \ldots, \tilde{\mathbf{x}}_i^{90}$.

LOdds: Log-odds score is calculated by averaging the difference of negative logarithmic probabilities on the predicted label over all test examples before and after masking $k\%$ top-scored tokens.

$$\mathrm{LOdds}(k) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{p\left(\hat{y}|\tilde{\mathbf{x}}_i^k\right)}{p\left(\hat{y}|\mathbf{x}_i\right)} \tag{16}$$

5 Results

We evaluated various explanation methods by masking the top $k\%$ of tokens across multiple datasets and measuring their AOPC and LOdds scores, as shown in Tab. 1, which presents average scores for different k values. Our findings indicate that the AGF method consistently outperforms others, achieving the highest AOPC and lowest LOdds scores, effectively identifying and masking the most important tokens for model predictions. Furthermore, the GF method also exceeds most baseline methods. Evaluation based on classification metrics also yields consistently similar results (App. D.1 and App. D.2).

Additionally, we assessed similar explanation methods by masking the bottom $k\%$ of tokens across datasets and measuring AOPC and LOdds scores, detailed in Tab. 2. The AGF method achieved the highest LOdds and lowest AOPC across most datasets, highlighting its ability to pinpoint important tokens for model predictions, with the GF method also surpassing many baseline methods in this context.

However, the Yelp dataset poses a unique challenge, as our methods do not perform optimally in terms of AOPC and LOdds metrics. This is likely due to the prevalence of conversational language, slang, and typos in Yelp reviews, which adversely affect the AGF method's performance more than others.

6 Limitations

The primary limitation of our proposed method is the increased running time of the optimization problem in eq. 12 as the number of tokens grows (Lee & Sidford, 2020; van den Brand et al., 2021). Moreover, it's important to note that optimization problems generally cannot be solved in parallel. However, recent advancements have led to the development of almost-linear time algorithm that solves the optimization problem described in eq. 12 (Tab. 7). Additionally, we found that the practical runtime of our method is comparable to other XAI approaches (Tab. 8), indicating that our method is both practical and efficient for obtaining feature attributions in AI models using Transformers.

Table 1: AOPC and LOdds scores of all methods in explaining the Transformer-based model across datasets when we mask **top** $k\%$ tokens. Higher AOPC and lower LOdds are desirable, indicating a strong ability to mark important tokens. Best results are in bold, and differences between AGF and benchmarks are statistically significant according to the ASO test (App. D.3).

Methods	SST2		IMDB		Yelp		Amazon		AG News	
	AOPC↑	LOdds↓	AOPC↑	LOdds↓	AOPC↑	LOdds↓	AOPC↑	LOdds↓	AOPC↑	LOdds↓
RawAtt	0.348	-0.973	0.329	-1.393	0.383	-1.985	0.353	-1.593	0.301	-1.105
Rollout	0.322	-0.887	0.354	-1.456	0.260	-0.987	0.304	-1.326	0.249	-0.983
Grads	0.354	-0.313	0.324	-1.271	0.412	-1.994	0.405	-1.793	0.327	-1.319
AttGrads	0.367	-0.654	0.337	-1.226	0.423	-1.978	0.419	-1.918	0.348	-1.477
CAT	0.369	-1.175	0.332	-1.274	0.417	-1.992	0.381	-1.639	0.325	-1.226
AttCAT	0.405	-1.402	0.371	-1.642	0.431	**-2.134**	0.427	-2.041	0.387	-1.688
PartialLRP	0.371	-1.171	0.323	-1.321	**0.443**	-2.018	0.384	-1.945	0.356	-1.627
TransAtt	0.399	-1.286	0.355	-1.513	0.411	-1.473	0.375	-1.875	0.377	-1.318
LIME	0.362	-1.056	0.347	-1.379	0.361	-1.568	0.358	-1.612	0.349	-1.538
KernelShap	0.382	-1.259	0.367	-1.423	0.385	-1.736	0.374	-1.717	0.351	-1.413
IG	0.401	-1.205	0.350	-1.443	0.409	-1.924	0.434	-2.024	0.393	-1.681
AF	0.371	-1.215	0.313	-1.297	0.398	-1.886	0.388	-1.923	0.352	-1.282
GF	0.412	-1.616	0.491	-1.718	0.396	-1.654	0.421	-2.006	0.366	-1.513
AGF	**0.427**	**-1.687**	**0.498**	**-1.849**	0.429	-1.982	**0.439**	**-2.103**	**0.398**	**-1.693**

Table 2: AOPC and LOdds scores of all methods in explaining the Transformer-based model across datasets when we mask **bottom** $k\%$ tokens. Lower AOPC and higher LOdds are desirable, indicating a strong ability to mark important tokens. Best results are in bold, and differences between AGF and benchmarks are statistically significant according to the ASO test (App. D.3).

Methods	SST2		IMDB		Yelp		Amazon		AG News	
	AOPC↓	LOdds↑	AOPC↓	LOdds↑	AOPC↓	LOdds↑	AOPC↓	LOdds↑	AOPC↓	LOdds↑
RawAtt	0.184	-0.693	0.151	-0.471	0.157	-0.747	0.129	-0.281	0.101	-0.427
Rollout	0.221	-0.773	0.123	-0.425	0.169	-0.734	0.171	-0.368	0.117	-0.471
Grads	0.234	-0.776	0.083	-0.203	0.131	-0.641	0.134	-0.254	0.083	-0.390
AttGrads	0.217	-0.713	0.088	-0.243	0.127	-0.603	0.135	-0.266	0.071	-0.351
CAT	0.247	-0.874	0.099	-0.327	0.134	-0.659	0.126	-0.240	0.104	-0.419
AttCAT	0.143	-0.412	0.041	-0.092	**0.103**	**-0.339**	0.115	-0.148	0.057	-0.219
PartialLRP	0.163	-0.527	0.057	-0.116	0.116	-0.486	0.167	-0.327	0.056	-0.204
TransAtt	0.148	-0.483	0.045	-0.107	0.123	-0.538	0.113	-0.140	0.049	-0.173
LIME	0.173	-0.603	0.076	-0.141	0.143	-0.687	0.158	-0.263	0.075	-0.372
KernelShap	0.197	-0.729	0.039	-0.084	0.135	-0.645	0.174	-0.351	0.067	-0.219
IG	0.150	-0.532	0.026	-0.064	0.130	-0.617	0.134	-0.241	0.052	-0.191
AF	0.199	-0.747	0.061	-0.148	0.153	-0.689	0.388	-1.923	0.106	-0.402
GF	0.154	-0.497	0.034	-0.079	0.149	-0.654	0.130	-0.267	0.090	-0.313
AGF	**0.084**	**-0.263**	**0.014**	**-0.039**	0.121	-0.504	**0.092**	**-0.114**	**0.037**	**-0.134**

7 CONCLUSION

In this study, we propose Generalized Attention Flow, an extension of Attention Flow. The core idea behind Generalized Attention Flow is applying the log barrier method to the maximum flow problem, defined by information tensors, to derive feature attributions. By leveraging the log barrier method, we resolve the non-uniqueness issue in optimal flows originating from the maximum flow problem, ensuring that our feature attributions are Shapley values and satisfy efficiency, symmetry, nullity, and linearity axioms. Additionally, we demonstrate that our approach satisfies the axiom of conservation.

Our experiments across multiple datasets indicate that our proposed AGF method generally outperforms other feature attribution methods in most evaluation scenarios. It could be valuable for future research to explore whether alternative definitions of the information tensor could enhance AGF's effectiveness.

8 Reproducibility

The code used to implement all results presented in this paper is available anonymously in this repository. Comprehensive details for developing the proposed methods can be found in Sec. 3.1, Sec. 3.2, and App. B. Additionally, the pre-trained models, datasets, and further implementation details for our experiments are thoroughly discussed in Sec. 4 and App. F. Proofs supporting our theoretical claims are provided in App. E.

REFERENCES

Samira Abnar and Willem Zuidema. Quantifying Attention Flow in Transformers, May 2020.

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Kyriakos Axiotis, Aleksander Madry, and Adrian Vladu. Faster Sparse Minimum Cost Flow by Electrical Flow Localization. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 528–539, Denver, CO, USA, February 2022. IEEE. ISBN 978-1-66542-055-6. doi: 10.1109/FOCS52979.2021.00059.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130140.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate, May 2016.

Oren Barkan, Edan Hauon, Avi Caciularu, Ori Katz, Itzik Malkiel, Omri Armstrong, and Noam Koenigstein. Grad-SAM: Explaining Transformers via Gradient Self-Attention Maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2882–2887, Virtual Event Queensland Australia, October 2021. ACM. ISBN 978-1-4503-8446-9. doi: 10.1145/3459637.3482126.

Stephen P. Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK ; New York, 2004. ISBN 978-0-521-83378-3.

Sébastien Bubeck. Convex Optimization: Algorithms and Complexity, November 2015.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer Interpretability Beyond Attention Visualization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 782–791, Nashville, TN, USA, June 2021a. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.00084.

Hila Chefer, Shir Gur, and Lior Wolf. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers, March 2021b.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection, May 2020.

Li Chen, Rasmus Kyng, Yang P. Liu, Simon Meierhans, and Maximilian Probst Gutenberg. Almost-Linear Time Algorithms for Incremental Graphs: Cycle Detection, SCCs, \$s\$-\$t\$ Shortest Path, and Minimum-Cost Flow, November 2023a.

Lu Chen, Siyu Lou, Keyan Zhang, Jin Huang, and Quanshi Zhang. HarsanyiNet: Computing Accurate Shapley Values in a Single Forward Propagation, December 2023b.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What Does BERT Look at? An Analysis of BERT's Attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, August 2009. ISBN 978-0-262-03384-8.

E. del Barrio, J. A. Cuesta-Albertos, and C. Matrán. An optimal transportation approach for assessing almost stochastic order, May 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, October 2020.

Rotem Dror, Segev Shlomov, and Roi Reichart. Deep Dominance - How to Properly Compare Deep Neural Models. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2773–2785, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1266.

Kawin Ethayarajh and Dan Jurafsky. Attention Flows are Shapley Value Explanations, May 2021.

Javier Ferrando and Elena Voita. Information Flow Routes: Automatically Interpreting Language Models at Scale, April 2024.

Sarthak Jain and Byron C. Wallace. Attention is not Explanation. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357.

Neil Jethani, Mukund Sudarshan, Ian Covert, Su-In Lee, and Rajesh Ranganath. FastSHAP: Real-Time Shapley Value Estimation, March 2022.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Incorporating Residual and Normalization Layers into Analysis of Masked Language Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 4547–4568, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.373.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for PyTorch, September 2020.

Yin Tat Lee and Aaron Sidford. Path Finding Methods for Linear Programming: Solving Linear Programs in Õ(vrank) Iterations and Faster Algorithms for Maximum Flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 424–433, Philadelphia, PA, USA, October 2014. IEEE. ISBN 978-1-4799-6517-5. doi: 10.1109/FOCS.2014.52.

Yin Tat Lee and Aaron Sidford. Solving Linear Programs with Sqrt(rank) Linear System Solves, August 2020.

Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions, November 2017.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Aleksander Mądry. GRADIENTS AND FLOWS: CONTINUOUS OPTIMIZATION APPROACHES TO THE MAXIMUM FLOW PROBLEM. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pp. 3361–3387, Rio de Janeiro, Brazil, May 2019. WORLD SCIENTIFIC. ISBN 978-981-327-287-3 978-981-327-288-0. doi: 10.1142/9789813272880_0185.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pp. 165–172, Hong Kong China, October 2013. ACM. ISBN 978-1-4503-2409-0. doi: 10.1145/2507157.2507163.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. GlobEnc: Quantifying Global Token Attribution by Incorporating the Whole Encoder Layer in Transformers. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 258–271, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.19.

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. DecompX: Explaining Transformers Decisions by Propagating Token Decomposition. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2649–2664, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.149.

Aaron Mueller. Missed Causes and Ambiguous Effects: Counterfactuals Pose Challenges for Interpreting Neural Networks. 2024. doi: 10.48550/ARXIV.2407.04690.

Dong Nguyen. Comparing Automatic and Human Evaluation of Local Explanations for Text Classification. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1069–1078, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1097.

Yao Qiang, Deng Pan, Chengyin Li, Xin Li, Rhongho Jang, and Dongxiao Zhu. AttCAT: Explaining Transformers via Attentive Class Activation Tokens. In *Advances in Neural Information Processing Systems*, October 2022.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier, August 2016.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter, February 2020.

Sofia Serrano and Noah A. Smith. Is Attention Interpretable? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2931–2951, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1282.

Lloyd S. Shapley. A Value for N-Person Games. Technical report, RAND Corporation, March 1952.

Avanti Shrikumar, Jocelin Su, and Anshul Kundaje. Computationally Efficient Measures of Internal Neuron Importance, July 2018.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

Mukund Sundararajan and Amir Najmi. The Many Shapley Values for Model Explanation. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9269–9278. PMLR, November 2020.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic Attribution for Deep Networks, June 2017.

Dennis Ulmer, Christian Hardmeier, and Jes Frellsen. Deep-significance - Easy and Meaningful Statistical Significance Testing in the Age of Neural Networks, April 2022.

Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P. Liu, Richard Peng, and Aaron Sidford. Faster Maxflow via Improved Dynamic Spectral Vertex Sparsifiers, December 2021.

Jan Van Den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pp. 859–869, Virtual Italy, June 2021. ACM. ISBN 978-1-4503-8053-9.

doi: 10.1145/3406325.3451108.

Jan van den Brand, Yang P. Liu, and Aaron Sidford. Dynamic Maxflow via Dynamic Interior Point Methods. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pp. 1215–1228, New York, NY, USA, June 2023. Association for Computing Machinery. ISBN 978-1-4503-9913-5. doi: 10.1145/3564246.3585135.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing, July 2020.

H. P. Young. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72, June 1985. ISSN 1432-1270. doi: 10.1007/BF01769885.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification, April 2016.

A PRELIMINARIES

A.1 MAXIMUM FLOW

Definition A.1 (Network Flow). *Given a network $G = (V, E, s, t, \boldsymbol{u})$, where s and t are the source and target nodes respectively and u_{ij} is the capacity for the edge $(i, j) \in E$, a flow is characterized as a function $f : E \to \mathbb{R}^{\geq 0}$ s.t.*

$$
\begin{aligned}
& f_{ij} \leq u_{ij} \quad \forall (i,j) \in E \quad \textit{(capacity constraints)} \\
& \sum_{j:(i,j)\in E} f_{ij} - \sum_{j:(j,i)\in E} f_{ji} = 0, \quad \forall i \in V, i \neq s, t \quad \textit{(flow conservation constraints)}
\end{aligned} \tag{17}
$$

We define $|f_{\text{out}}(i)|$ to be the total outflow value of a node i and $|f_{\text{in}}(i)|$ to be the total inflow value of a node i. For a given set $K \subseteq V$ of nodes, we define $|f(K)| = \sum_{i\in K} |f_{\text{out}}(i)|$ for every flow f. The value of a flow in a given network $G = (V, E, s, t, \boldsymbol{u})$ is denoted as $|f| = \sum_{v:(s,v)} f_{sv} - \sum_{v:(v,s)} f_{vs} = |f_{\text{out}}(s)| - |f_{\text{in}}(s)|$, and a maximum flow is identified as a feasible flow with the highest attainable value.

A.2 MULTI-COMMODITY MAXIMUM FLOW

The multi-commodity flow problem is an important variant of the maximum flow problem. This problem involves multiple source-sink pairs, unlike the standard maximum flow problem, which only considers one source and one sink. The goal is to find multiple optimal flows, denoted by $f^1(\cdot,\cdot), \ldots, f^r(\cdot,\cdot)$, where each $f^k(\cdot,\cdot)$ represents a feasible flow from the source s_k to the sink t_k. The objective is to ensure that all capacity constraints are satisfied, which are represented by the equation:

$$
\sum_{k=1}^{r} f^k(i,j) \leq u(i,j) \quad \forall (i,j) \in E \tag{18}
$$

Such a flow is known as a "multi-commodity" flow. A multi-commodity maximum flow problem is to maximize the function $\sum_{k=1}^{r} \sum_{v:(v,s_k)} f^k(s_k, v)$.

To solve the problem of multi-commodity maximum flow, we can simplify it by transforming it into a standard maximum flow problem. This can be achieved by introducing two new nodes, a "super-source" node ss and a "super-target" node st. The "super-source" node ss should be connected to all the original sources s_i through edges of finite capacities, while the "super-target" node st should be connected to all the original sinks t_i with edges of finite capacities:

- Each outgoing edge from the "super-source" node ss to each source node s_i gets assigned a capacity that is equal to the total capacity of the outgoing edges from the source node s_i.
- Each incoming edge from an original "super-target" node st to each sink node t_i gets assigned a capacity that is equal to the total capacity of the incoming edges to the sink node t_i.

It is easy to demonstrate that the maximum flow from ss to st is equivalent to the maximum sum of flows in a feasible multi-commodity flow in the original network.

A.3 SHAPLEY VALUES

The Shapley value, introduced by Shapley (1952), concerns the cooperative game in the coalitional form (N, ϑ), where N is a set of n players and $\vartheta : 2^N \to \mathbb{R}$ with $\vartheta(\emptyset) = 0$ is the characteristic (payoff) function. In the game, the marginal contribution of the player i to any coalition S with $i \notin S$ is considered as

$\vartheta(S \cup i) - \vartheta(S)$. These Shapley values are the only constructs that jointly satisfy the efficiency, symmetry, nullity, and additivity axioms (Shapley, 1952; Young, 1985):

Efficiency: The Shapley values must add up to the total value of the game, which means $\sum_{i \in N} \phi_i(\vartheta) = \vartheta(N)$.

Symmetry: If two players are equal in their contributions to any coalition, they should receive the same Shapley value. Mathematically, if $\vartheta(S \cup \{i\}) = \vartheta(S \cup \{j\})$ for all $S \subseteq N \backslash \{i, j\}$, then $\phi_i(\vartheta) = \phi_j(\vartheta)$.

Nullity (Dummy): If a player has no impact on any coalition, their Shapley value should be zero. Mathematically, if $\vartheta(S \cup \{i\}) = \vartheta(S)$ for all $S \subseteq N \backslash \{i\}$, then $\phi_i(\vartheta) = 0$.

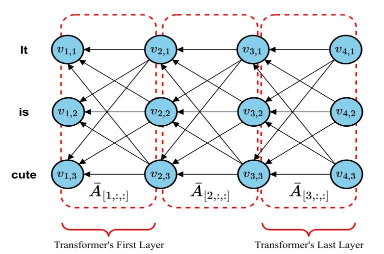
Linearity: If the game $\vartheta(\cdot)$ is a linear combination of two games $\vartheta_1(\cdot), \vartheta_2(\cdot)$ for all $S \subseteq N$, i.e. $\vartheta(S) = \vartheta_1(S) + \vartheta_2(S)$ and $(c \cdot \vartheta)(S) = c \cdot \vartheta(S), \forall c \in \mathbb{R}$, then the Shapley value in the game ϑ is also a linear combination of that in the games ϑ_1 and ϑ_2, i.e. $\forall i \in N, \phi_i(\vartheta_1) = \phi_i(\vartheta_1) + \phi_i(\vartheta_2)$ and $\phi_i(c \cdot \vartheta) = c \cdot \phi_i(\vartheta)$.

Considering these axioms, the attribution of a player j is uniquely given by (Shapley, 1952; Young, 1985):

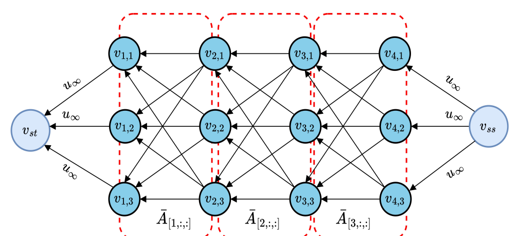
$$\phi_j(\vartheta) = \sum_{S \subseteq N \backslash \{j\}} \frac{|S|!(n - |S| - 1)!}{n!} \left(\vartheta(S \cup \{j\}) - \vartheta(S)\right) \tag{19}$$

where the difference $\vartheta(S \cup \{j\}) - \vartheta(S)$ represents the i-th feature's contribution to the subset S, and the summation represents a weighted average across all subsets that do not include i.

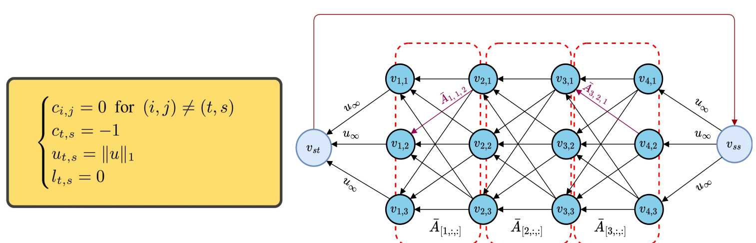
Initially, a payoff function based on model accuracy was suggested (Lundberg & Lee, 2017). Since then, various alternative coalition functions have been proposed (Jethani et al., 2022; Sundararajan & Najmi, 2020), each resulting in a different feature importance score. Many of these alternative approaches are widely used and have been shown to outperform the basic SHAP method (Lundberg & Lee, 2017) in empirical studies.



(a) Network flow with multiple sources and targets.



(b) Network flow including super-source and super-target.



(c) Network flow defined for MCC problem.

Figure 2: Initial network flow to be used in our proposed method, the multi-commodity flow with multiple sources and targets, and the network flow for MCC problems.

B Network Flow Generation

Fig. 2 will detail the process of defining a graph network and its parameters using Algorithm 1 for use in our proposed method. It is worth noting that the same method can be used for Algorithm 2. To solve the maximum flow or MCC problem within this graph network, we must compute network flow with multiple sources and targets, assigning all nodes in the first and last layers of transformers as sources and targets, respectively (Fig. 2a).

To solve this problem, we leverage the concept of multi-commodity flow (multiple-sources multiple-targets maximum flow) by introducing a super-source node ss and a super-target node st (Fig. 2b). To define the upper-bound and lower-bound capacities of this new graph network, we utilize the procedure defined in Sec. 3.2. In the last step, we add a new edge from the super-target node st to the super-source node ss and define the cost vector, upper-bound capacities, and lower-band capacities according to Fig. 2c. Subsequently, we can input all derived parameters into eq. 12, solve the optimization problem, and evaluate feature attributions.

C Non-uniqueness of Maximum Flow

Fig. 3 visually describes our proposed approach for computing feature attributions. Using maximum flow to derive these attributions produces a convex set containing all optimal flows, which makes it unsuitable as a feature attribution technique. In contrast, our proposed approach, which utilizes the log barrier method, generate a unique optimal flow and provides an interpretable set of feature attributions.

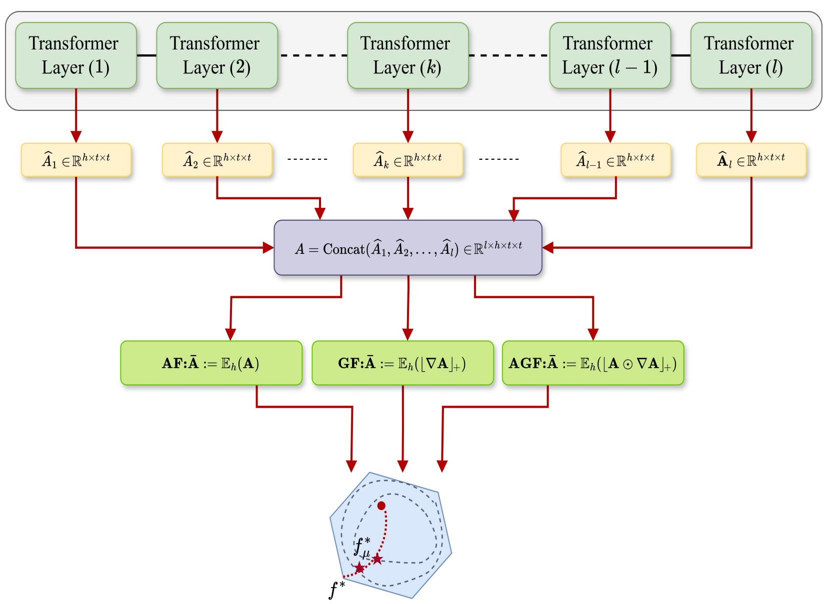
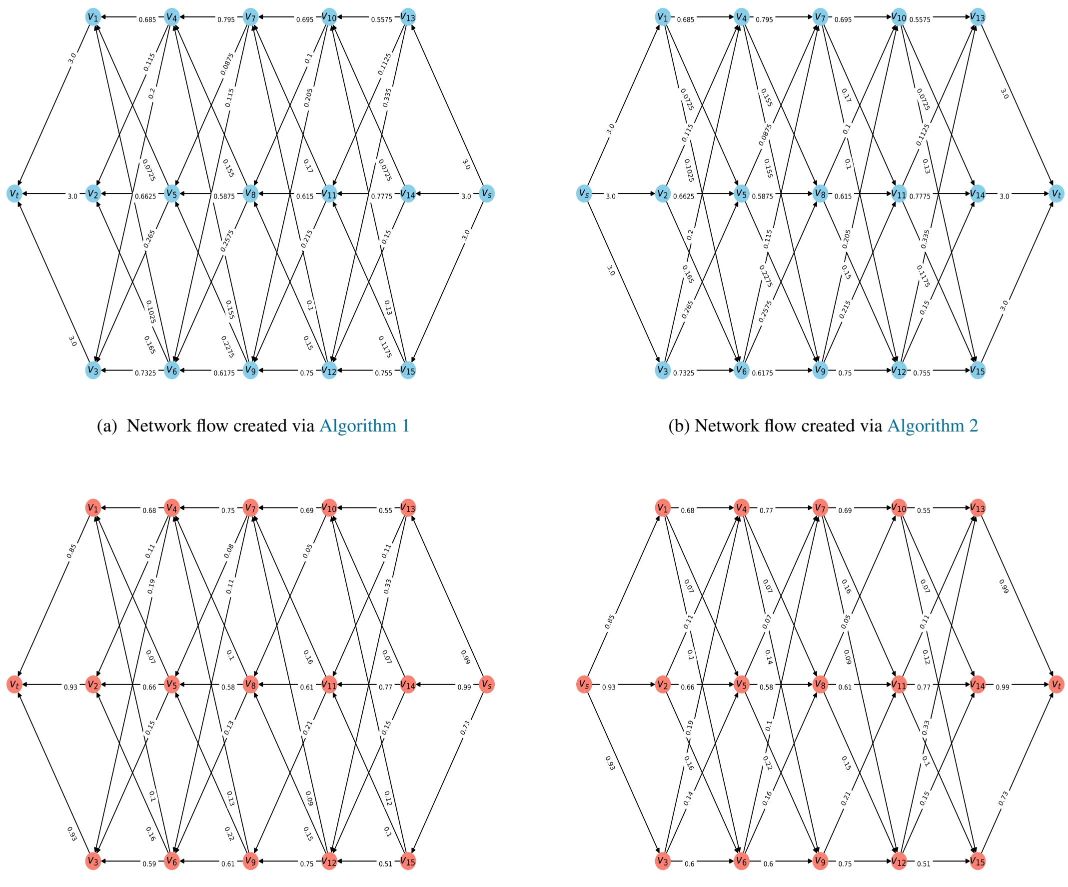


Figure 3: Overview of how the proposed method evaluates the unique optimal flow computed using the log barrier method, attention weights, and their gradients in Transformers.

Fig. 4 shows the capacities and optimal flows obtained by solving maximum flows on the network, constructed with the synthetic information tensor $\bar{\boldsymbol{A}} \in \mathbb{R}^{4\times 3\times 3}$ as input, using Algorithm 1 and Algorithm 2. While the maximum flows are the same for both algorithms, their optimal flows differ. Notably, significant differences in flows between node pairs $\{v_4, v_8\}$ and $\{v_3, v_5\}$ are visible in Fig. 4c and Fig. 4d. Additionally, we evaluated the maximum flow and its optimal flow generated by both algorithms across various combinations of token numbers t and Transformer layers l. Our findings indicate that the optimal flows from the two algorithms do not coincide in any scenario.



(a) Network flow created via Algorithm 1

(b) Network flow created via Algorithm 2

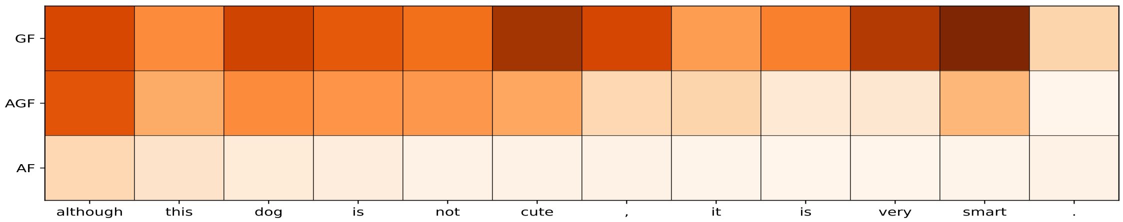
(c) Maximum flow in the graph created from Algorithm 1

(d) Maximum flow in the graph created from Algorithm 2

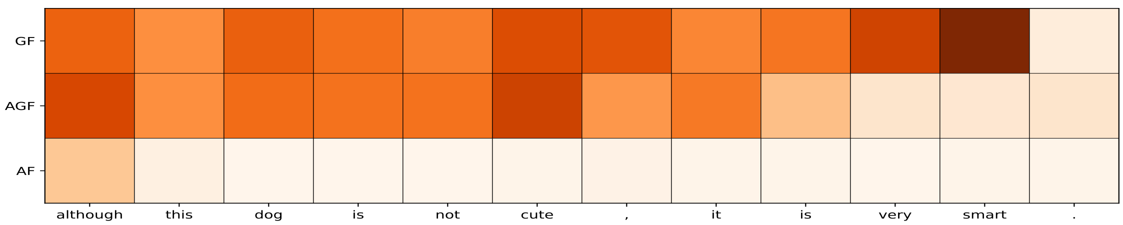
Figure 4: Network flow, maximum flow, and residual flow created by Algorithm 1 and Algorithm 2. The optimal flows and residual flows evaluated using Algorithm 1 and Algorithm 2 are different.

Fig. 5a and Fig. 5b display the normalized feature attributions evaluated across three information tensors introduced in Sec. 3.1 for sentiment analysis of the sentence "although this dog is not cute, it is very smart." employing both Algorithm 1 and Algorithm 2. Across each of the three information tensors, the resulting optimal flows and their corresponding normalized attributions differ depending on whether Algorithm 1 or Algorithm 2 is used.

The layer-wise normalized feature attributions, obtained through the same process, are displayed in Fig. 6. For each information tensor type and layer, the resulting optimal flows and their normalized attributions differ based on whether Algorithm 1 or Algorithm 2 is utilized. We also computed the optimal flow and its feature attributions for various input sentences using both algorithms for each of the information tensors AF, GF, and AGF. Our findings reveal that the optimal flows and corresponding feature attributions generated by Algorithm 1 and Algorithm 2 differ for all sentences.

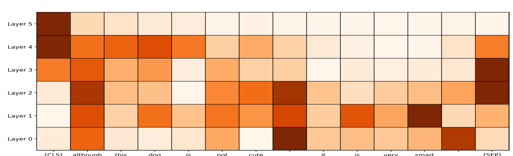


(a) Normalized feature attributions for Transformer's input layer evaluated by Algorithm 1 for different information tensors.

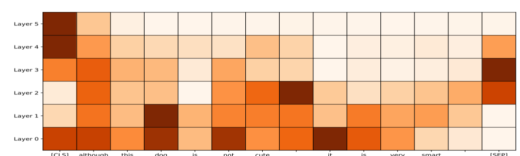


(b) Normalized feature attributions for Transformer's input layer evaluated by Algorithm 2 different information tensors.

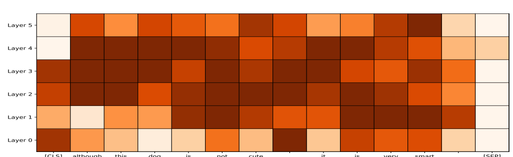
Figure 5: Normalized feature attributions for Transformer's input layer and different information tensors.



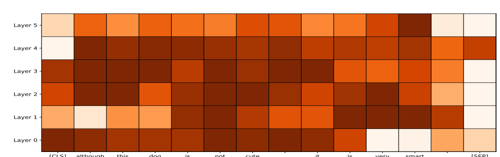
(a) AF method:Algorithm 1



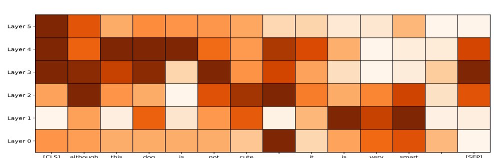
(b) AF method:Algorithm 2.



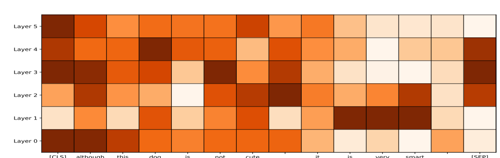
(c) GF method:Algorithm 1.



(d) GF method:Algorithm 2.



(e) AGF method:Algorithm 1.



(f) AGF method:Algorithm 2.

Figure 6: Normalized feature attributions for all Transformer layers evaluated by Algorithm 1 and Algorithm 2.

D RESULTS

D.1 QUALITATIVE VISUALIZATIONS

This section visually examines the feature attributions derived from our proposed methods, applied to information tensors as defined in Sec. 3.1. Fig. 7 illustrates feature attributions obtained from our proposed methods applied to two graphs generated by either Algorithm 1 or Algorithm 2. Remarkably, our approaches consistently yield identical results for both graphs. The outcomes vividly demonstrate the superiority of AGF over AF and GF, offering more insightful and reasonable feature attributions. Specifically, both AGF and GF effectively highlight the importance of tokens like ’smart’ and ’cute’, while assigning lower values to less significant tokens such as ’this’, ’it’, and ’and’. In contrast, AF fails to capture the expected feature attribution of ’smart’ and tends to produce an almost uniform distribution for feature attributions.

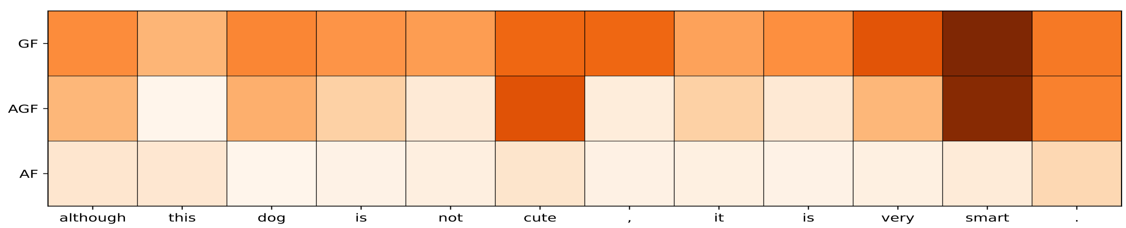


Figure 7: Visualizations of the feature attributions generated by running our proposed method on the three introduced information tensors on the showcase example.

D.2 ADDITIONAL RESULTS

Fig. 8 presents a detailed comparison of the performance dynamics of various feature attribution methods under different corruption rates across three distinct datasets: IMDB, Amazon, and Yelp. The efficacy of these methods is evaluated using two key metrics, AOPC and LOdds, which measure how well each method identifies important tokens that influence model predictions. Notably, our proposed AGF method consistently outperforms the other techniques, maintaining the highest average AOPC and LOdds scores across a range of corruption levels, particularly for both the IMDB and Amazon datasets. This consistent superiority highlights the robustness of AGF in pinpointing the most important tokens, which significantly affect the model’s decision-making process.

We also evaluated various explanation methods by analyzing their performance on classification metrics, with results summarized in Tab. 3, which details the average Accuracy, F1, Precision, and Recall scores across multiple k values. On the SST2 dataset, our proposed methods AGF and GD, in conjunction with KernelShap, achieved the highest overall performance. For the IMDB dataset, AGF and GF, along with Integrated Gradients (IG), showed significant improvement over other methods. In the Amazon dataset, AGF and GF, when combined with TransAtt, outperformed all competitors. For the AG News dataset, AGF and AF, paired with AttGrads, demonstrated superior performance. The consistently strong performance of AGF across diverse datasets and evaluation metrics highlight its versatility and effectiveness in accurately identifying important tokens, reinforcing its reliability as a feature attribution method in NLP models.

However, the Yelp dataset poses a distinct challenge where our proposed methods, including AGF, do not consistently achieve optimal results across all evaluation metrics. This performance discrepancy is likely due to the unique characteristics inherent to the Yelp dataset, which frequently contains a higher concentration of informal language, colloquialisms, and typographical errors. The prevalence of such linguistic noise in Yelp reviews is notably higher compared to other datasets like IMDB or Amazon. These textual irregularities introduce complexity that AGF, with its current configuration, may be less equipped to handle effectively. Consequently, AGF's performance suffers, as it appears to have a lower tolerance for handling noisy or non-standard text inputs, which compromise its ability to accurately attribute features and identify the most influential tokens within these reviews.

Fig. 9 compare the feature attribution methods evaluated on the aforementioned sentence using a model trained on SST2. In all instances, the feature attribution methods predict positive sentiment for the showcased example. Our methods, AGF and GF, effectively capture the most important tokens, such as 'cute' and 'smart' (indicated in dark orange shading), which significantly contribute to the positive sentiment prediction. Some other methods, including Grads, LIME, RawAtt, and PartialLRP, also exhibit some capability in identifying important tokens. However, certain methods like AF, AttCAT, CAT, Rollout, KernelShap, and IG struggle to correctly identify important tokens.

Table 3: The average of F1, Accuracy, Precision, and Recall scores of all methods in explaining the Transformer-based model on each dataset when we mask **top** $k\%$ tokens. Lower scores are desirable for all metrics (indicated by ↓), indicating a strong ability to mark important tokens.

Methods	SST2				IMDB				Yelp				Amazon				AG News			
	F1↓	Acc↓	Prec↓	Rec↓	F1↓	Acc↓	Prec↓	Rec↓	F1↓	Acc↓	Prec↓	Rec↓	F1↓	Acc↓	Prec↓	Rec↓	F1↓	Acc↓	Prec↓	Rec↓
RawAtt	0.75	0.75	0.72	0.79	0.69	0.67	0.70	0.69	0.68	0.72	0.74	0.63	0.67	0.68	0.67	0.66	0.65	0.68	0.68	0.68
Rollout	0.81	0.82	0.80	0.82	0.74	0.67	0.65	0.84	0.80	0.83	0.88	0.74	0.71	0.73	0.71	0.72	0.61	0.63	0.64	0.63
Grads	0.78	0.75	0.72	0.79	0.69	0.67	0.70	0.69	0.68	0.72	0.74	0.63	0.67	0.68	0.67	0.66	0.65	0.68	0.68	0.68
AttGrads	0.78	0.78	0.75	0.82	0.76	0.75	0.78	0.76	0.91	0.91	0.89	0.93	0.79	0.82	0.80	0.78	0.58	0.61	0.60	0.60
CAT	0.68	0.65	0.61	0.76	0.56	0.49	0.51	0.65	0.70	0.70	0.68	0.73	0.68	0.64	0.67	0.70	0.63	0.64	0.64	0.64
AttCAT	0.68	0.65	0.62	0.76	0.57	0.48	0.50	0.64	0.67	0.66	0.65	0.70	0.67	0.62	0.66	0.68	0.64	0.64	0.65	0.64
PartialLRP	0.75	0.75	0.71	0.78	0.66	0.65	0.67	0.67	0.65	0.70	0.71	0.61	0.65	0.66	0.65	0.64	0.65	0.68	0.68	0.68
TransAtt	0.73	0.72	0.69	0.76	0.61	0.58	0.60	0.62	0.62	0.66	0.66	0.60	0.62	0.63	0.63	0.61	0.63	0.66	0.66	0.66
LIME	0.61	0.63	0.62	0.63	0.55	0.55	0.55	0.55	0.72	0.73	0.72	0.73	0.72	0.73	0.72	0.73	0.67	0.67	0.68	0.67
KernelShap	0.53	0.52	0.53	0.53	0.67	0.69	0.68	0.69	0.77	0.78	0.77	0.78	0.67	0.68	0.67	0.68	0.74	0.74	0.75	0.74
IG	0.56	0.57	0.56	0.57	0.48	0.50	0.48	0.50	0.72	0.72	0.72	0.72	0.73	0.74	0.73	0.74	0.67	0.68	0.67	0.67
AF	0.72	0.72	0.71	0.71	0.65	0.68	0.70	0.68	0.71	0.71	0.71	0.70	0.69	0.71	0.71	0.71	0.64	0.62	0.65	0.64
GF	0.56	0.57	0.56	0.56	0.45	.0.48	0.45	0.48	0.70	0.71	0.70	0.71	0.65	0.67	0.66	0.67	0.67	0.68	0.67	0.67
AGF	0.54	0.52	0.54	0.54	0.46	0.47	0.47	0.47	0.70	0.72	0.70	0.70	0.67	0.67	0.67	0.67	0.64	0.63	0.65	0.64

D.3 Statistical Significance Test

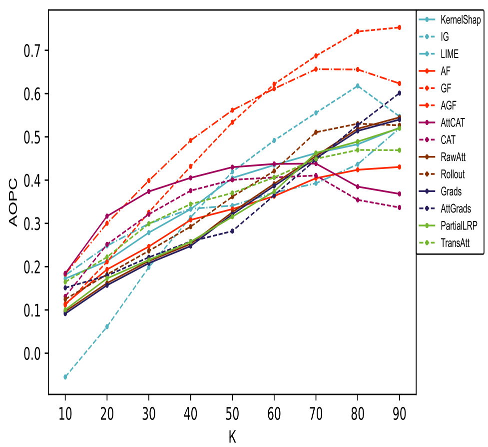
To implement a statistical significance test, we employ the ASO (Almost Stochastic Order) method (Ulmer et al., 2022; Dror et al., 2019; del Barrio et al., 2017), which compares the cumulative distribution functions (CDFs) of two score distributions to determine stochastic dominance. Notably, ASO imposes no assumptions about score distributions, making it applicable to any metric where higher scores indicate better performance.

When comparing model A with model B using the ASO method, we obtain the value $\epsilon_{\min}$, which is an upper bound on the violation of stochastic order. If $\epsilon_{\min} \leq \tau$ (with $\tau \leq 0.5$), model A is considered stochastically dominant over model B, implying superiority. This value can also be interpreted as a confidence score; a lower $\epsilon_{\min}$ suggests greater confidence in model A's superiority. The null hypothesis for ASO is defined as:

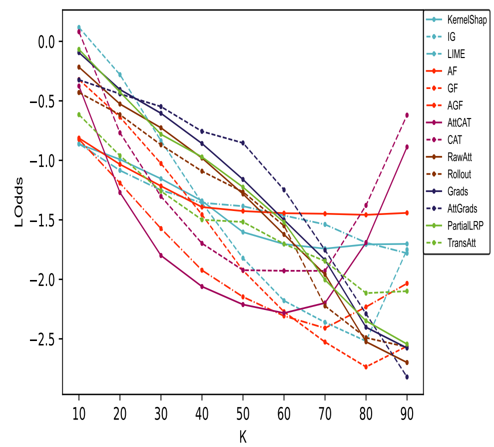
$$H_0 : \epsilon_{\min} \geq \tau \tag{20}$$

where the significance level α is an input parameter that influences $\epsilon_{\min}$.

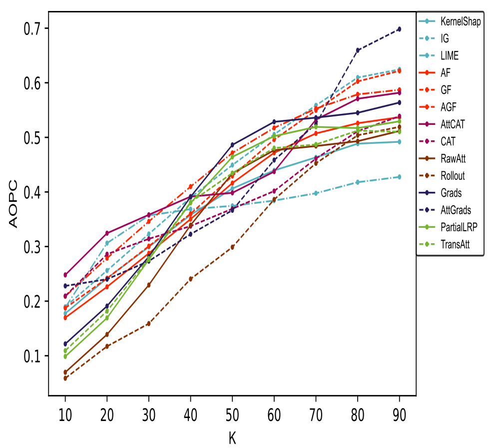
In this study, we conduct 500 independent runs per method to perform comprehensive statistical tests, comparing the AOPC and LOdds metrics between our best-performing proposed method, AGF, and the top benchmark methods outlined in Tab. 1 and Tab. 2, using $\tau = 0.5$. As illustrated in Tab. 4 and Tab. 5, the AGF method stochastically dominates the performance of these benchmark methods across all datasets, with the exception of the Yelp dataset.



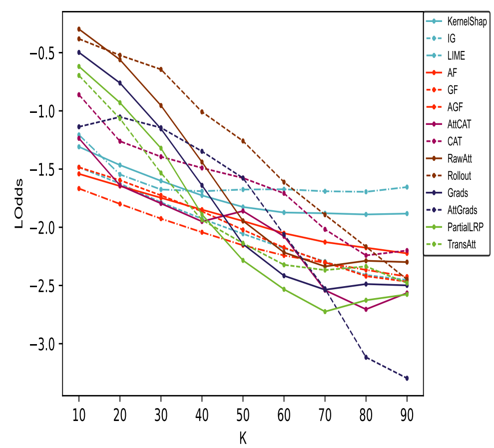
(a) AOPC score for IMDB dataset



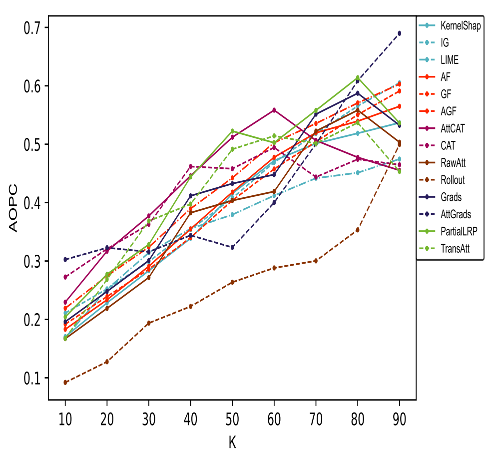
(b) LOdds score for IMDB dataset



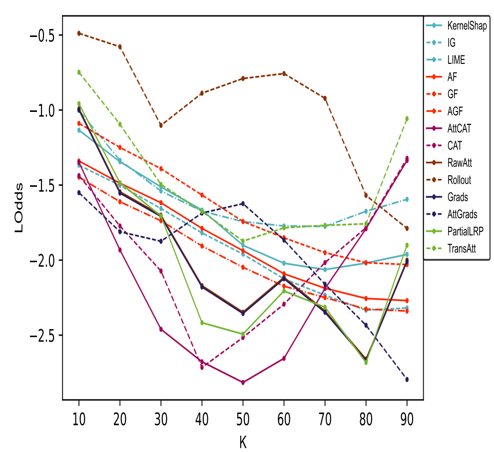
(c) AOPC score for Amazon dataset



(d) LOdds score for Amazon dataset



(e) AOPC score for Yelp dataset



(f) LOdds score for Yelp dataset

Figure 8: AOPC and LOdds scores of different methods in explaining BERT across the varying corruption rates k on IMDB, Amazon, and Yelp datasets. The x-axis illustrates masking the $k\%$ of the tokens in an order of decreasing saliency.

although this dog is not cute , it is very smart .

AGF
although this dog is not cute , it is very smart .

GF
although this dog is not cute , it is very smart .

AF
although this dog is not cute , it is very smart .

Grads
although this dog is not cute , it is very smart .

PartialLRP
although this dog is not cute , it is very smart .

CAT
although this dog is not cute , it is very smart .

AttCAT
although this dog is not cute , it is very smart .

IG
although this dog is not cute , it is very smart .

KernelShap
although this dog is not cute , it is very smart .

Rollout
although this dog is not cute , it is very smart .

RawAtt
although this dog is not cute , it is very smart .

LIME
although this dog is not cute , it is very smart .

TransAtt
although this dog is not cute , it is very smart .

Figure 9: Visualizations of the normalized feature attributions generated by the methods on the binary classification task.

Table 4: ASO test to compare AGF method with the best benchmark method when we mask **top** $k\%$ tokens.

Dataset	SST2		IMDB		Yelp		Amazon		AG News	
	AOPC	LOdds	AOPC	LOdds	AOPC	LOdds	AOPC	LOdds	AOPC	LOdds
$\epsilon_{\min}$	0.054	0.039	0.078	0.061	0.813	0.924	0.053	0.043	0.091	0.076

Table 5: ASO test to compare AGF method with the best benchmark method when we mask **bottom** $k\%$ tokens.

Dataset	SST2		IMDB		Yelp		Amazon		AG News	
	AOPC	LOdds	AOPC	LOdds	AOPC	LOdds	AOPC	LOdds	AOPC	LOdds
$\epsilon_{\min}$	0.049	0.037	0.113	0.085	0.913	0.824	0.062	0.053	0.091	0.067

E PROOFS

Corollary 3.1. While Shapley values are inherently unique, our findings in Sec. 3.3 and App. C expose a critical inconsistency. We demonstrate that the optimal solution of the maximum flow problems defined in eq. 8 is not necessarily unique, thereby disproving the claim that the feature attributions proposed by Ethayarajh & Jurafsky (2021) are Shapley values.

The non-uniqueness of these attributions, as evidenced by our proof, fundamentally conflicts with the defining properties of Shapley values. If these attributions defined by Ethayarajh & Jurafsky (2021) were indeed Shapley values, they would necessarily be unique. However, our observations demonstrate that since the optimal solution of the maximum flow problem is not necessarily unique, for each optimal solution of the maximum flow problem, we can derive corresponding feature attributions that differ from one another.

$\square$

Theorem 3.1. Since the optimal flow $\boldsymbol{f}^*$ is computed once for the entire graph and not for each potential subgraph, and the players (tokens) are all disjoint without any connections in S, blocking the flow through one player does not impact the outflow of any other players. Therefore, for every $S \subseteq N$ where $i \notin S$, we have $|f_{\text{out}}(i)| = v(S \cup \{i\}) - v(S)$. Utilizing the definition of Shapley values in eq. 14, we obtain:

$$
\begin{aligned}
\phi_i(\vartheta) &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!}(\vartheta(S \cup \{i\}) - \vartheta(S)) \\
&= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!}(|f_{\text{out}}(i)|) \\
&= |f_{\text{out}}(i)| \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} \\
&= |f_{\text{out}}(i)|
\end{aligned}
$$

It is also evident that the defined function meets all four fairness-based axioms of efficiency, symmetry, linearity, and additivity. $\square$

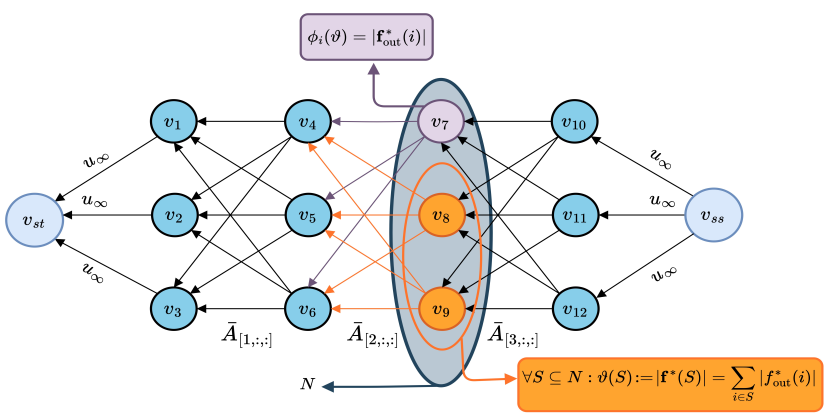


Figure 10: Visualizations of the procedure to define the cooperative game (N, ϑ) using the solution of barrier-regularized maximum flow or its corresponding MCC problem.

F Implementation Details

F.1 Datasets

Tab. 6 illustrates comprehensive statistics of the datasets utilized for the classification task. We randomly extracted $5,000$ sentences from each test section of the datasets, except for those with a test size less than $5,000$, where we retained all samples. Furthermore, we prioritized diversity in our sampling process by incorporating sentences of varying lengths, with an equal distribution between those shorter and longer than the mode size of the test dataset.

Table 6: Statistical information and the pre-trained models employed for each dataset.

Datasets	# Test Samples	# Classes	ℓ_{mode}	ℓ_{min}	ℓ_{max}	ℓ_{avg}	Pre-trained Model
SST2	1,821	2	108	5	256	103.3	textattack/bert-base-uncased-SST-2
Amazon	5,000	2	127	15	1009	404.9	fabriceyhc/bert-base-uncased-amazon_polarity
IMDB	5,000	2	670	32	12988	1293.8	fabriceyhc/bert-base-uncased-imdb
Yelp	5,000	2	313	4	5107	723.8	fabriceyhc/bert-base-uncased-yelp_polarity
AG News	5,000	4	238	100	892	235.3	fabriceyhc/bert-base-uncased-ag_news

F.2 Time Complexity of Proposed Methods

In the minimum-cost circulation problem, we are given a directed graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, upper and lower edge capacities $\boldsymbol{u}, \boldsymbol{l} \in \mathbb{R}^m$, and edge costs $\boldsymbol{c} \in \mathbb{R}^m$. Our objective is to find a circulation $\boldsymbol{f} \in \mathbb{R}^m$ satisfying:

$$\underset{\substack{\boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{0} \\ \boldsymbol{l} \leq \boldsymbol{f} \leq \boldsymbol{u}}}{\arg\min}\ \boldsymbol{c}^\top \boldsymbol{f}$$

where $\mathbf{B} \in \mathbb{R}^{m \times n}$ is the edge-vertex incidence matrix.

To compare running times, we assume that $\widetilde{\boldsymbol{l}}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{c}}$ represent the integral versions of $\boldsymbol{l}, \boldsymbol{u}, \boldsymbol{c}$ obtained from either Algorithm 1 or Algorithm 2, and define $U = \|\widetilde{\boldsymbol{u}}\|_\infty$ and $C = \|\widetilde{\boldsymbol{c}}\|_\infty$. Tab. 7 compares the latest iterative algorithms for solving the maximum flow and minimum-cost circulation problems. While the most efficient algorithm achieves nearly linear running time relative to the number of edges m, the runtime can still be significant with long input sequences. To solve the minimum-cost circulation problem, we implemented the algorithm by (Lee & Sidford, 2014) using CVXPY (Agrawal et al., 2018; Diamond & Boyd, 2016).

Table 7: Overview of recent iterative algorithms for maximum flow and minimum-cost circulation problems.

Year	MCC Bound	Max-Flow Bound	Author
2014	$O\left(m\sqrt{n}\operatorname{polylog}(n)\log^2(U)\right)$	$O\left(m\sqrt{n}\operatorname{polylog}(n)\log^2(U)\right)$	Lee & Sidford (2014)
2022	$O\left(m^{\frac{3}{2}-\frac{1}{762}}\operatorname{polylog}(n)\log(U+C)\right)$	$O\left(m^{\frac{10}{7}}\operatorname{polylog}(n)U^{\frac{1}{7}}\right)$	Axiotis et al. (2022)
2023	$O\left(m^{\frac{3}{2}-\frac{1}{58}}\operatorname{polylog}(n)\log^2(U)\right)$	$O\left(m^{\frac{3}{2}-\frac{1}{58}}\operatorname{polylog}(n)\log^2(U)\right)$	van den Brand et al. (2023)
2023	$O\left(m^{1+o(1)}\log(U)\log(C)\right)$	$O\left(m^{1+o(1)}\log(U)\log(C)\right)$	Chen et al. (2023a)

Experiments were conducted on a computing device running Ubuntu 20.04.4 LTS, equipped with an Intel(R) Xeon(R) Platinum 8368 CPU at 2.40GHz, featuring 12 cores and 24 threads for parallel processing. Graphics processing was handled by an NVIDIA RTX 3090 Ti with 40GB of dedicated memory. The device also had 230GB of system memory, ensuring ample computational resources for efficient and effective experimentation.

Tab. 8 compares the runtime of benchmark methods for analyzing the sentence "although this dog is not cute, it is very smart." Methods like RawAtt and Rollout, which rely on raw attention weights, have the shortest runtime. In contrast, methods requiring complex post-processing to evaluate feature attributions have longer runtime. Our proposed methods' runtime is comparable to others in this latter group.

Table 8: Runtime of all methods for the showcase example.

Methods	Runtime (seconds)
RawAtt	0.123
Rollout	0.154
Grads	1.554
AttGrads	1.571
CAT	1.684
AttCAT	1.660
PartialLRP	1.571
TransAtt	1.620
LIME	1.462
KernelShap	2.342
IG	2.701
AF	2.301
GF	2.305
AGF	2.306