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

Behrooz Azarkhalili[1,2] **Maxwell Libbrecht**[1]

[1]Computing Science Department, Simon Fraser University
[2]Life Language Processing Lab, University of California, Berkeley
{bazarkha, maxwell_libbrecht}@sfu.ca

Abstract

This paper introduces Generalized Attention Flow (GAF), a novel feature attribution method for Transformer-based models to address the limitations of current approaches. By extending Attention Flow and replacing attention weights with the generalized Information Tensor, GAF integrates attention weights, their gradients, the maximum flow problem, and the barrier method to enhance the performance of feature attributions. The proposed method exhibits key theoretical properties and mitigates the shortcomings of prior techniques that rely solely on simple aggregation of attention weights. Our comprehensive benchmarking on sequence classification tasks demonstrates that a specific variant of GAF consistently outperforms state-of-the-art feature attribution methods in most evaluation settings, providing a more reliable interpretation of Transformer model outputs.

1 Introduction

Feature attribution methods are essential to develop interpretable machine and deep learning models. These methods assign a score to each input feature, quantifying its contribution to the model's output and thereby enhancing the understanding of model predictions.

The rise of Transformer models with self-attention mechanism has driven the need for feature attribution methods for interpreting these models (Vaswani et al., 2017; Bahdanau et al., 2016; Devlin et al., 2019; Sanh 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 and Zuidema, 2020; Jain and Wallace, 2019; Serrano and Smith, 2019). Consequently, various post hoc techniques have been developed to compute feature attributions in 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. Yet, many of these techniques still focus primarily on the gradients of attention weights and inherit the 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. (2021b,a) proposed a comprehensive methodology enabling information propagation through all Transformer components. Yet, this approach relies on specific LRP rules, limiting its applicability across various Transformer architectures.

Many existing methods to evaluate 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 applied to compute feature attributions in Transformers violate pivotal axioms such as symmetry, sensitivity, efficiency, and linearity (Shapley, 1952; Sundararajan et al., 2017; Sundararajan and Najmi, 2020) (Sec. 3.5).

Abnar and Zuidema (2020) recently introduced Attention Flow to overcome these limitations in XAI methods. Attention Flow considers attention weights as capacities in a maximum flow problem and compute feature attributions using its 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

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 19954–19974

any **encoder-only** Transformer, Attention Flow has demonstrated strong potential to improve model interpretability (Abnar and Zuidema, 2020; Modarressi et al., 2023; Kobayashi et al., 2020, 2021).

Subsequently, Ethayarajh and Jurafsky (2021) attempted to bridge attention flows and XAI by leveraging Shapley values (Shapley, 1952, 2016). While their goal was to demonstrate that Attention Flows can be interpreted as Shapley values under specific conditions, they overlooked the issue of non-uniqueness in such flows (Sec. 3.3).

Our contributions. In this work, we propose Generalized Attention Flow (GAF), a method that not only satisfies crucial theoretical properties but also demonstrates improved empirical performance. The primary contributions of our work are:

1. We proposed Generalized Attention Flow, which generates feature attributions by utilizing the log barrier method to solve a regularized maximum flow problem within a capacity network derived from functions applied to attention weights. Rather than defining capacities solely based on attention weights, we will introduce alternatives using the gradients of these weights (GF) or the product of attention weights and their gradients (AGF).

2. We address the non-uniqueness issue in Attention Flow, which previously undermined some of its proposed theoretical properties (Ethayarajh and Jurafsky, 2021), and demonstrate that non-unique solutions are frequent in practice. To resolve this, we introduce barrier regularization, proving that feature attributions obtained from the regularized maximum flow problem are Shapley values and satisfy the axioms of efficiency, symmetry, nullity, and linearity (Shapley, 1952, 2016; Young, 1985; Chen et al., 2023b).

3. We conduct extensive benchmarking of the proposed attribution methods based on Generalized Attention Flow, comparing them against various state-of-the-art attribution techniques. Our results show that a specific variant of the proposed method outperforms previous methods 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 to compute feature attributions leveraging Generalized Attention Flow. This package is highly flexible, and can compute the feature attributions of any **encoder-only** Transformer model available in the Hugging Face Transformers package (Wolf et al., 2020). Moreover, our methods are easily adaptable for a variety of 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**:

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

  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**:

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

  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**:

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

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

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 (or demands) for the edge $(i, j) \in E$, a 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 compute the 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):

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

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 the 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:

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

where β represents a convex inequality constraint, α 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 the region S is nonempty and convex, we introduce the 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:

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

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 models, 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 neural network (Ferrando and Voita, 2024; Mueller, 2024).

First, attention weights 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 and Zuidema, 2020; Ferrando and Voita, 2024). Additionally, the gradient of attention weights captures the flow of information during back-propagation, quantifying how changes in the output influence the attention weights throughout the network during training (Barkan et al., 2021). Therefore, a combined view of attention weights

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., 2021b,a).

Our Generalized Attention Flow generlize this foundation by leveraging 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 above insights, we present three aggregation functions to define information tensors (Barkan et al., 2021; Chefer et al., 2021b).

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 $\times$ 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.

3.2 Generalized Attention Flow

In Generalized Attention Flow, we use the attention mechanism for feature attribution by developing a network flow representation of a Transformer or other attention-based model. We will 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 compute 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 $\bar{\boldsymbol{A}} \in \mathbb{R}^{l \times t \times t}$. Using the information tensor $\bar{\boldsymbol{A}}$, we can 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 will substitute the obtained matrices into the primal form of eq. 8 to compute the desired optimal flow.

To clarify Algorithm 1 and Algorithm 2 further, we detail the process of constructing the layered attribution graph $\mathcal{G}$, which has an adjacency matrix of shape $(2 + t \times (l+1), 2 + t \times (l+1))$, serving as the input for the maximum flow problem. Nodes at layer $\ell \in \{1, \ldots, l\}$ and token $i \in \{1, \ldots, t\}$ are designated as $v_{\ell,i}$. The following guidelines outline the process to define the upper and lower bound capacities:

- To connect nodes $v_{1,i}$ to the super-target node v_{st}, we define $\boldsymbol{u}[0, i] = u_\infty$ for $1 \leq i \leq t$.
- 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 $\ell \in \{1, \ldots, l\}$, $i \in \{1, \ldots, t\}$, and $j \in \{1, \ldots, t\}$, 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 $1 \leq i \leq t$.
- The lower-bound capacity is defined as $\boldsymbol{l} = \boldsymbol{0}$.

Fig. 1a and Fig. 1b depict schematic graphs generated using the information tensor $\bar{\boldsymbol{A}} \in \mathbb{R}^{3 \times 3 \times 3}$, with Algorithm 1 and Algorithm 2, respectively. While both algorithms solve the same network flow problem by constructing graphs containing a super-source and a super-target, the second algorithm differs from the first in two key aspects. First, the positions of the super-source and super-target are swapped in the second graph, such 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 relative to the first.

3.3 Non-uniqueness of Maximum Flow

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

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($\bar{\boldsymbol{A}}$)
▷ Initialization
$l, t, \_ \leftarrow \bar{\boldsymbol{A}}$.shape()
$\beta_{\min} \leftarrow \min(\bar{\boldsymbol{A}} > 0)$
$\beta \leftarrow -\lfloor \log_{10}(\beta_{\min}) \rfloor$
$\gamma \leftarrow 10^{\beta}$
$Q_{tl} \leftarrow t * (l+1) + 2$
$\boldsymbol{l} \leftarrow \text{zeros}(Q_{tl}, Q_{tl})$
$\boldsymbol{u} \leftarrow \text{zeros}(Q_{tl}, Q_{tl})$
$u_{\infty} \leftarrow t$
▷ Fill super-source → First Layer
for i in range(t) **do**
$\boldsymbol{u}[i+1][0] \leftarrow u_{\infty}$
end for
▷ Fill Last Layer → super-target
for i in range(t) **do**
$\boldsymbol{u}[-1][-i-2] \leftarrow u_{\infty}$
end for
▷ Fill j-th Layer to $(j+1)$-th Layer
for j in range(l) **do**
start $\leftarrow t * j + 1$
mid $\leftarrow t * (j+1) + 1$
end $\leftarrow t * (j+2) + 1$
$\boldsymbol{u}$[mid:end , start:mid] $\leftarrow \bar{\boldsymbol{A}}_{[j,:,:]}$
end for
▷ Get Integral Version of Capacities
$\widetilde{\boldsymbol{l}} \leftarrow \text{int}(\gamma * \boldsymbol{l})$
$\widetilde{\boldsymbol{u}} \leftarrow \text{int}(\gamma * \boldsymbol{u})$
▷ Get Adjacency Matrix
$\mathcal{A} \leftarrow \mathbb{I}_{(\boldsymbol{u}>0)}$
▷ Get super-source and super-target
$ss, st \leftarrow t * (l+1) + 1, 0$
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($\bar{\boldsymbol{A}}$)
▷ Initialization
$l, t, \_ \leftarrow \bar{\boldsymbol{A}}$.shape()
$\beta_{\min} \leftarrow \min(\bar{\boldsymbol{A}} > 0)$
$\beta \leftarrow -\lfloor \log_{10}(\beta_{\min}) \rfloor$
$\gamma \leftarrow 10^{\beta}$
$Q_{tl} \leftarrow t * (l+1) + 2$
$\boldsymbol{l} \leftarrow \text{zeros}(Q_{tl}, Q_{tl})$
$\boldsymbol{u} \leftarrow \text{zeros}(Q_{tl}, Q_{tl})$
$u_{\infty} \leftarrow t$
▷ Fill super-source → First Layer
for i in range(t) **do**
$\boldsymbol{u}[0][i+1] \leftarrow u_{\infty}$
end for
▷ Fill Last Layer → super-target
for i in range(t) **do**
$\boldsymbol{u}[-i-2][-1] \leftarrow u_{\infty}$
end for
▷ Fill j-th Layer to $(j+1)$-th Layer
for j in range(l) **do**
start $\leftarrow t * j + 1$
mid $\leftarrow t * (j+1) + 1$
end $\leftarrow t * (j+2) + 1$
$\boldsymbol{u}$[start:mid , mid:end] $\leftarrow \bar{\boldsymbol{A}}^{T}_{[j,:,:]}$
end for
▷ Get Integral Version of Capacities
$\widetilde{\boldsymbol{l}} \leftarrow \text{int}(\gamma * \boldsymbol{l})$
$\widetilde{\boldsymbol{u}} \leftarrow \text{int}(\gamma * \boldsymbol{u})$
▷ Get Adjacency Matrix
$\mathcal{A} \leftarrow \mathbb{I}_{(\boldsymbol{u}>0)}$
▷ Get super-source and super-target
$ss, st \leftarrow 0, t * (l+1) + 1$
end function

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.

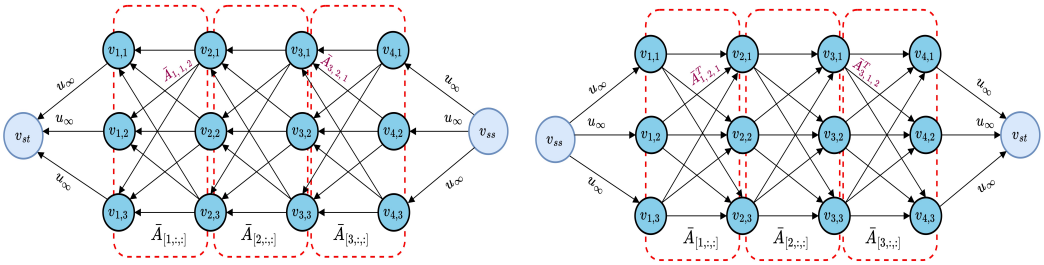
Observation 3.2. If two distinct feasible solutions, denoted f_1 and f_2, exist for a linear programming problem, then any convex combination $\gamma_1 f_1 + \gamma_2 f_2$ forms another feasible solution. Consequently, the maximum flow problem can possess an infinite number of feasible solutions. Additionally, 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 $\vartheta(S) \coloneqq |\boldsymbol{f}^*(S)| = \sum_{i\in S} |f_{\text{out}}(i)|$, where $|f_{\text{out}}(i)|$ denotes the total outflow value of node i. Although Ethayarajh and Jurafsky (2021) claimed that for each node $i \in N$, $\phi_i(\vartheta) = |\boldsymbol{f}^*_{\text{out}}(i)|$ 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 challenge in the maximum flow problem, we reformulate the minimum-cost circulation problem as follows:

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



(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 using Algorithm 1 and Algorithm 2.

where $\beta(\boldsymbol{f}) = (\boldsymbol{f} - \boldsymbol{l})(\boldsymbol{f} - \boldsymbol{u})$. Consequently, the original problem can 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:

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

It is evident that, for any positive μ and a feasible initial solution, the barrier function guarantees that the solution derived from an iterative minimization scheme, such as interior point methods, remains feasible (Bubeck, 2015; Boyd and Vandenberghe, 2004; Mądry, 2019). Moreover, to obtain an ε-approximate solution to eq. 11, it suffices to set $\mu \leq \frac{\varepsilon}{2m}$ and solve the corresponding problem in eq. 12 (Bubeck, 2015; Boyd and Vandenberghe, 2004; Mądry, 2019).

Finally, the Hessian of the objective function in eq. 11 at some point $\boldsymbol{f}$ is equal to the Hessian 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 and Vandenberghe, 2004).

Fig. 2 visually describes our proposed approach for computing feature attributions. Utilizing maximum flow to derive these attributions produces a convex set containing all optimal flows, which makes it unsuitable as a feature attribution method. In contrast, our proposed approach, which utilizes the log barrier method, generate a unique optimal flow and provides an interpretable set of feature attributions.

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 four essential axioms, as proved 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 $\phi(\vartheta) \in \mathbb{R}^n$ can be computed by averaging the marginal contribution of each feature over all possible feature combinations:

$$\phi_i(\vartheta) = \sum_{S \subseteq N \setminus \{i\}} \binom{n-1}{|S|}^{-1} \left(\vartheta(S \cup \{i\}) - \vartheta(S)\right) \tag{14}$$

Shapley values are the unique feature attributions that satisfy four fairness-based axioms: efficiency (completeness), symmetry, linearity (additivity), and nullity (Shapley, 1952, 2016; Young, 1985) (App. A.3). Initially, a value function based on model accuracy was proposed (Lundberg and Lee, 2017), but since then, various alternative payoff functions have been introduced (Jethani et al., 2022; Sundararajan and Najmi, 2020), each providing distinct feature importance scores.

Theorem 3.1 (Log Barrier Regularization of Generalized Attention Flow Outcomes Shapley Values). Given a layered attribution graph $\mathcal{G}(\mathcal{A}, \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{c}, ss, st)$ which has been defined using either of 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. Now, suppose $\boldsymbol{f}^*$ is the optimal unique solution of eq. 12, and for every $S \subseteq N$, define the payoff function $\vartheta(S) := |\boldsymbol{f}^*(S)| = \sum_{i \in S} |f_{\text{out}}(i)|$ where $|f_{\text{out}}(i)|$

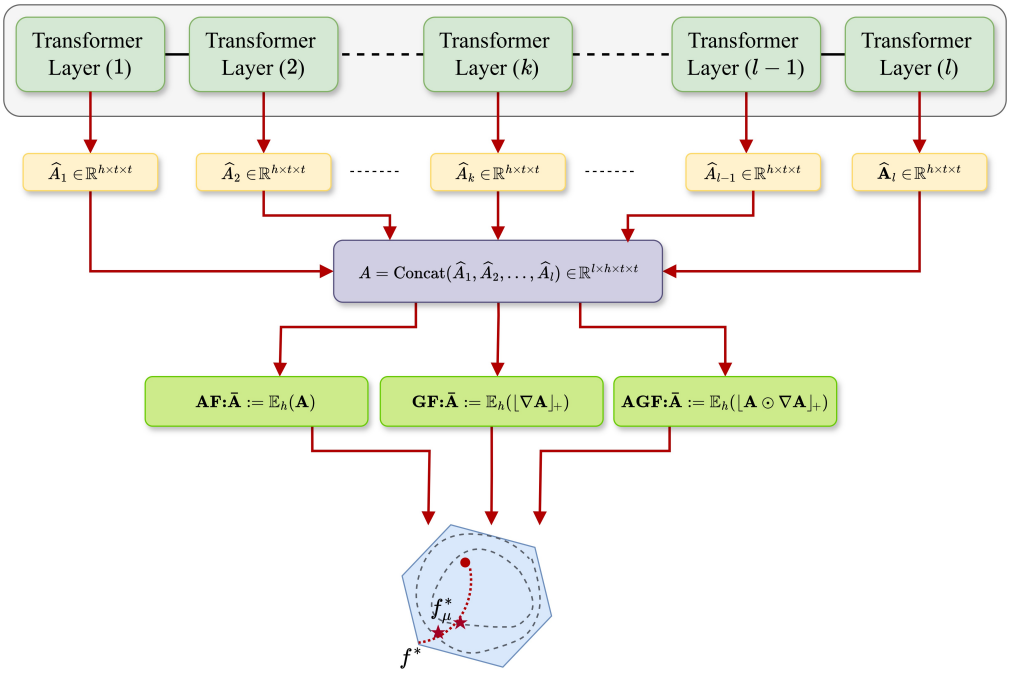


Figure 2: Overview of how the proposed method computes the unique optimal flow using the log barrier method, attention weights, and their gradients in Transformers.

is the total outflow value of a node i. Then, it can be proven that for each node $i \in N$, $\phi_i(\vartheta) = |\boldsymbol{f}^*_{\text{out}}(i)|$ represents the Shapley value (Proof in App. E).

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

4 Experiments

In this section, we comprehensively evaluate the effectiveness of our methods for NLP 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 only on sequence classification.

4.1 Transformer Models

In our evaluations, we use a specific pre-trained model from the HuggingFace Hub (Wolf et al., 2020) for each dataset and compare our explanation methods against others to assess their performance (App. F.1).

4.2 Datasets

Our method's assessment encompasses sequence classification spanning binary classification tasks on datasets including SST2 (Socher et al., 2013), Amazon Polarity (McAuley and 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 conduct the experiments using a subset of 5,000 randomly selected samples from the Amazon, Yelp, and IMDB datasets, while utilizing the full test sets for the other datasets (App. F.1).

4.3 Benchmark Methods

Our experiments compare the methods introduced in Sec. 3.1 with several well-known explanation methods tailored for Transformer models. To evaluate attention-based methods such as RawAtt and Rollout (Abnar and 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., 2021b), we adapted the repository developed by Qiang et al. (2022). We also implement GlobEnc (Modarressi et al., 2022) and DecompX (Modarressi et al., 2023) using their repositories. Moreover, we implemented classical attribution methods such as Integrated Gradient (Sundararajan et al., 2017), KernelShap (Lundberg and Lee, 2017), and LIME (Ribeiro et al., 2016) using the Captum package (Kokhlikyan et al., 2020).

4.4 Evaluation Metric

AOPC: One of the important 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:

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

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 defined 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 decreasing saliency order, resulting in $\tilde{\mathbf{x}}_i^{10}, \tilde{\mathbf{x}}_i^{10}, \ldots, \tilde{\mathbf{x}}_i^{90}$.

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

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

5 Results

We assessed various explanation methods by masking the top $k\%$ of tokens across multiple datasets and measuring their AOPC and LOdds scores. Tab. 1 presents the average scores for different k values, proving that the AGF method consistently outperforms others by achieving the highest AOPC and lowest LOdds scores, effectively identifying and masking the most important tokens for model predictions. Additionally, the GF method surpasses most baseline approaches. Evaluations based on classification metrics further confirm these findings. (App. D.1, App. D.2).

Additionally, we assessed the aforementioned 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.

In contrast, the Yelp dataset presents 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.

5.1 Masking Top-Ranked Tokens

In this evaluation scenario, higher AOPC and lower LOdds scores are desirable, indicating that masking most important tokens has maximum impact on model predictions.

In the SST2 dataset, the AGF method achieves the highest AOPC and the lowest LOdds scores, significantly surpassing the performance of the next closest competitor, GF. In the IMDB dataset, the gap in performance becomes even more pronounced, with AGF exhibiting markedly superior AOPC and LOdds scores relative to alternative methods. Both AGF and GF methods outperformed the next best approach, AttCAT, across AOPC and LOdds metrics.

In the Amazon dataset, AGF again leads with the highest AOPC score and the lowest LOdds score, followed by IG and AttCAT. For the AG News dataset, AGF continues to demonstrate superior performance compared to alternative attribution methods.

Although AGF exhibits strong performance on the Yelp dataset, it is slightly outperformed by the PartialLRP and AttCAT methods. This suggests that the informal language and typographical errors often present in Yelp reviews may present particular challenges for our method.

In addition, the GF method also demonstrates robust performance across datasets, consistently outperforming many methods and ranking among the top three methods for most datasets.

5.2 Masking Bottom-Ranked Tokens

In this evaluation scenario, lower AOPC and higher LOdds scores are desirable, indicating that masking less important tokens has minimal impact on model predictions.

AGF achieves the most desirable metrics across most datasets. On the SST2 dataset, AGF recorded the lowest AOPC and highest LOdds, significantly outperforming the next best method, AttCAT. In the IMDB dataset, AGF again exhibits superior performance compared to IG and other baseline methods.

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
DecompX	0.396	-1.115	0.343	-1.411	0.401	-1.887	0.391	-1.912	0.396	-1.385
GlobEnc	0.373	-1.095	0.330	-1.323	0.388	-1.826	0.367	-1.861	0.373	-1.496
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 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	0.184	-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
DecompX	0.161	-0.578	0.077	-0.134	0.121	-0.557	0.141	-0.227	0.073	-0.311
GlobEnc	0.175	-0.592	0.086	-0.161	0.139	-0.638	0.152	-0.253	0.078	-0.345
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**

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).

In the Amazon dataset, AGF maintained its lead with substantially better performance than the next best method, TransAtt. Similarly, for the AG News dataset, AGF exhibits the best performance, outperforming TransAtt and other competitive methods.

However, consistent with our observations from masking top-ranked tokens, the Yelp dataset presents unique challenges for our methods. While AGF maintains a competitive performance overall, it falls short compared to AttCAT and PartialLRP when assessing the performance on bottom-ranked tokens.

6 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.

Our experiments conducted on various datasets confirm that our proposed AGF (and GF) method generally outperforms other feature attribution methods in most evaluation scenarios. This strong and consistent performance of AGF (and GF) across both evaluation scenarios, masking the top and bottom tokens, demonstrates their robustness and effectiveness in identifying the importance of tokens in model predictions. The future research can explore weather alternative definitions of the information tensor enhance AGF's effectiveness.

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 and Sidford, 2020; van den Brand et al., 2021). Moreover, it's important to note that optimization problems generally cannot be solved in parallel.

Although recent theoretical advancements have developed almost-linear time algorithms to solve the optimization problem described in eq. 12 (Tab. 7), their computational cost is still significant, particularly when we have long input sequences. Nevertheless, we found that the practical runtime of our method is comparable to other XAI methods (Tab. 8).

References

Samira Abnar and Willem Zuidema. 2020. Quantifying Attention Flow in Transformers. *Preprint*, arXiv:2005.00928.

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. 2017. A Rewriting System for Convex Optimization Problems. https://arxiv.org/abs/1709.04494v2.

Kyriakos Axiotis, Aleksander Madry, and Adrian Vladu. 2022. Faster Sparse Minimum Cost Flow by Electrical Flow Localization. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 528–539, Denver, CO, USA. IEEE.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural Machine Translation by Jointly Learning to Align and Translate. *Preprint*, arXiv:1409.0473.

Oren Barkan, Edan Hauon, Avi Caciularu, Ori Katz, Itzik Malkiel, Omri Armstrong, and Noam Koenigstein. 2021. Grad-SAM: Explaining Transformers via Gradient Self-Attention Maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2882–2887, Virtual Event Queensland Australia. ACM.

Stephen P. Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, Cambridge, UK ; New York.

Sébastien Bubeck. 2015. Convex Optimization: Algorithms and Complexity. *Preprint*, arXiv:1405.4980.

Hila Chefer, Shir Gur, and Lior Wolf. 2021a. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers. *Preprint*, arXiv:2103.15679.

Hila Chefer, Shir Gur, and Lior Wolf. 2021b. Transformer Interpretability Beyond Attention Visualization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–791, Nashville, TN, USA. IEEE.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection. *Preprint*, arXiv:2004.02015.

Li Chen, Rasmus Kyng, Yang P. Liu, Simon Meierhans, and Maximilian Probst Gutenberg. 2023a. Almost-Linear Time Algorithms for Incremental Graphs: Cycle Detection, SCCs, s-t Shortest Path, and Minimum-Cost Flow. *Preprint*, arXiv:2311.18295.

Lu Chen, Siyu Lou, Keyan Zhang, Jin Huang, and Quanshi Zhang. 2023b. HarsanyiNet: Computing Accurate Shapley Values in a Single Forward Propagation. *Preprint*, arXiv:2304.01811.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition*, 3rd edition. The MIT Press.

E. del Barrio, J. A. Cuesta-Albertos, and C. Matrán. 2017. An optimal transportation approach for assessing almost stochastic order. *Preprint*, arXiv:1705.01788.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Preprint*, arXiv:1810.04805.

Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Preprint*, arXiv:1603.00943.

Rotem Dror, Segev Shlomov, and Roi Reichart. 2019. Deep Dominance - How to Properly Compare Deep Neural Models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, Florence, Italy. Association for Computational Linguistics.

Kawin Ethayarajh and Dan Jurafsky. 2021. Attention Flows are Shapley Value Explanations. *Preprint*, arXiv:2105.14652.

Javier Ferrando and Elena Voita. 2024. Information Flow Routes: Automatically Interpreting Language Models at Scale. *Preprint*, arXiv:2403.00824.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Neil Jethani, Mukund Sudarshan, Ian Covert, Su-In Lee, and Rajesh Ranganath. 2022. FastSHAP: Real-Time Shapley Value Estimation. *Preprint*, arXiv:2107.07436.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is Not Only a Weight: Analyzing Transformers with Vector Norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

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

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for PyTorch. *Preprint*, arXiv:2009.07896.

Yin Tat Lee and Aaron Sidford. 2014. 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*, pages 424–433, Philadelphia, PA, USA. IEEE.

Yin Tat Lee and Aaron Sidford. 2020. Solving Linear Programs with Sqrt(rank) Linear System Solves. *Preprint*, arXiv:1910.08033.

Scott Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. *Preprint*, arXiv:1705.07874.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Aleksander Mądry. 2019. GRADIENTS AND FLOWS: CONTINUOUS OPTIMIZATION APPROACHES TO THE MAXIMUM FLOW PROBLEM. In *Proceedings of the International Congress of Mathematicians (ICM 2018)*, pages 3361–3387, Rio de Janeiro, Brazil. WORLD SCIENTIFIC.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 165–172, Hong Kong China. ACM.

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2023. DecompX: Explaining Transformers Decisions by Propagating Token Decomposition. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2649–2664, Toronto, Canada. Association for Computational Linguistics.

Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. GlobEnc: Quantifying Global Token Attribution by Incorporating the Whole Encoder Layer in Transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 258–271, Seattle, United States. Association for Computational Linguistics.

Aaron Mueller. 2024. Missed Causes and Ambiguous Effects: Counterfactuals Pose Challenges for Interpreting Neural Networks. In *ICML 2024 Workshop on Mechanistic Interpretability*.

Dong Nguyen. 2018. Comparing Automatic and Human Evaluation of Local Explanations for Text Classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana. Association for Computational Linguistics.

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

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *Preprint*, arXiv:1602.04938.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

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

L. S. Shapley. 2016. 17. A Value for n-Person Games. In *17. A Value for n-Person Games*, pages 307–318. Princeton University Press.

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

Avanti Shrikumar, Jocelin Su, and Anshul Kundaje. 2018. Computationally Efficient Measures of Internal Neuron Importance. *Preprint*, arXiv:1807.09946.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

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

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. *Preprint*, arXiv:1703.01365.

Dennis Ulmer, Christian Hardmeier, and Jes Frellsen. 2022. Deep-significance - Easy and Meaningful Statistical Significance Testing in the Age of Neural Networks. *Preprint*, arXiv:2204.06815.

Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P. Liu, Richard Peng, and Aaron Sidford. 2021. Faster Maxflow via Improved Dynamic Spectral Vertex Sparsifiers. *Preprint*, arXiv:2112.00722.

Jan Van Den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. 2021. 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*, pages 859–869, Virtual Italy. ACM.

Jan van den Brand, Yang P. Liu, and Aaron Sidford. 2023. Dynamic Maxflow via Dynamic Interior Point Methods. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 1215–1228, New York, NY, USA. Association for Computing Machinery.

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

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

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. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *Preprint*, arXiv:1910.03771.

H. P. Young. 1985. Monotonic solutions of cooperative games. *International Journal of Game Theory*, 14(2):65–72.

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

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2016. Character-level Convolutional Networks for Text Classification. *Preprint*, arXiv:1509.01626.

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.

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

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 maximum flow problem aims to generalize the maximum flow problem by considering multiple source-sink pairs instead of a single pair. The objective of this new problem is to determine multiple optimal flows, $f^1(\cdot,\cdot), \ldots, f^r(\cdot,\cdot)$, where each $f^k(\cdot,\cdot)$ represents a feasible flow from source s_k to sink t_k

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

Therefore, the multi-commodity maximum flow problem aims to maximize the objective function $\sum_{k=1}^{r} \sum_{v:(v,s_k)} f^k(s_k, v)$.

To solve the multi-commodity maximum flow problem, we can easily transform 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 with finite capacities, while the "super-target" node st should be connected to all the original sinks t_i through edges with finite capacities:

- Each outgoing edge from the "super-source" node ss to each source node s_i is 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 is 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 within 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 value (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):

$$\phi_i(\vartheta) = \sum_{S \subseteq N \backslash \{i\}} \binom{n-1}{|S|}^{-1} (\vartheta(S \cup \{i\}) - \vartheta(S)) \tag{19}$$

The difference term $\vartheta(S \cup \{j\}) - \vartheta(S)$ represents the i-th feature's contribution to the subset S, while the summation provides a weighted average across all subsets excluding i.

Initially, a value (payoff) function based on model accuracy was suggested (Lundberg and Lee, 2017). Since then, various alternative coalition functions have been proposed (Jethani et al., 2022; Sundararajan and 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 and Lee, 2017) in empirical studies.

B Network Flow Generation

Fig. 2 will detail the process of defining a graph network and its parameters in our proposed method using Algorithm 1. It is worth noting that the same procedure can be defined using Algorithm 2. To solve the maximum flow or MCC problem within this graph network, we should 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. 3a).

To solve this problem, we utilize the concept of multi-commodity maximum flow (multiple-sources multiple-targets maximum flow) by introducing a super-source node ss and a super-target node st (Fig. 3b). 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 compute the cost vector, upper-bound capacities, and lower-band capacities according to Fig. 3c. Finally, we can input all derived parameters into eq. 12, solve the optimization problem, and calculate feature attributions.

C Non-uniqueness of Maximum Flow

Fig. 4 depicts the capacities and optimal flows obtained by solving maximum flow problem on the network, defined with the synthetic information tensor $\bar{\boldsymbol{A}} \in \mathbb{R}^{4\times3\times3}$ as input, using Algorithm 1 and Algorithm 2. While the optimal values are the same for both algorithms, their optimal flows differ. Notably, significant differences in flows between node pairs $\{\boldsymbol{v_4}, \boldsymbol{v_8}\}$ and $\{\boldsymbol{v_3}, \boldsymbol{v_5}\}$ are visible in Fig. 4c and Fig. 4d. Additionally, we evaluated the optimal value 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.

Fig. 5a and Fig. 5b demo the normalized feature attributions computed for three information tensors introduced in Sec. 3.1 for sentiment analysis of the sentence "although this dog is not cute, it is very smart." using both Algorithm 1 and Algorithm 2. Fig. 5a and Fig. 5b corroborate that the resulting optimal flows and their corresponding normalized attributions for the three information tensors differ depending on whether Algorithm 1 or Algorithm 2 is applied.

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 have 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 results show that the optimal flows and their corresponding feature attributions calculated using either Algorithm 1 or Algorithm 2 differ for all sentences.

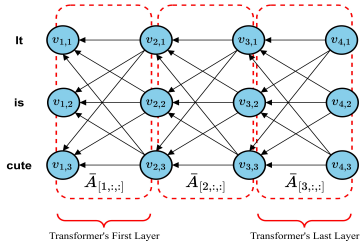
D Results

D.1 Qualitative Visualizations

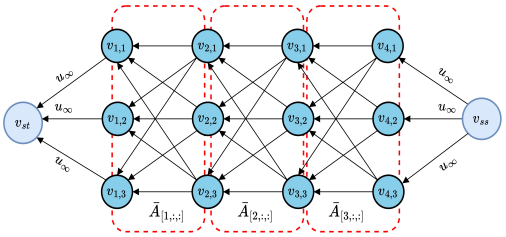
This section renders a visual analysis of the feature attributions computed by our proposed methods defined in Sec. 3.1. Fig. 7 displays the feature attributions derived from our methods using graph networks generated by Algorithm 1 or Algorithm 2. Notably, both approaches create identical results for both graphs. The results clearly confirm the superior performance of AGF over AF and GF, yielding more informative feature attributions. In fact, both AGF and GF accurately underscore the importance of tokens such as 'smart' and 'cute', while assigning lower values to the less important tokens like 'this', 'it', and 'and'. However, AF fails to capture the expected attribution for 'smart' and tends to distribute attributions almost uniformly.

D.2 Additional Results

Fig. 8 demonstrates a comprehensive comparison between the performance of the different feature attribution methods under varying corruption rates across three datasets: IMDB, Amazon, and Yelp. The evaluation is based on two key metrics, AOPC and LOdds, which assess how effectively each method identifies important tokens that influence model predictions. It is evident that our proposed AGF method outperform other methods by achiev-

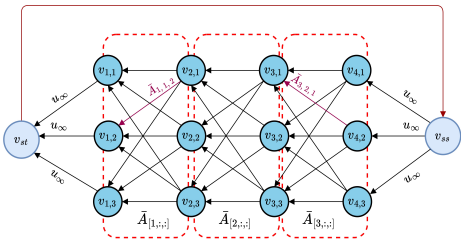


(a) Network flow with multiple sources and targets.



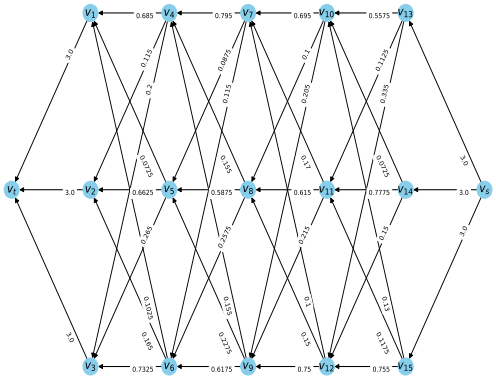
(b) Network flow including super-source and super-target.

$$\begin{cases} c_{i,j} = 0 \text{ for } (i,j) \neq (t,s) \\ c_{t,s} = -1 \\ u_{t,s} = \|u\|_1 \\ l_{t,s} = 0 \end{cases}$$

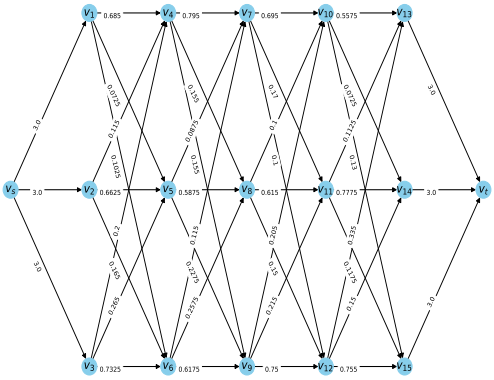


(c) Network flow defined for MCC problem.

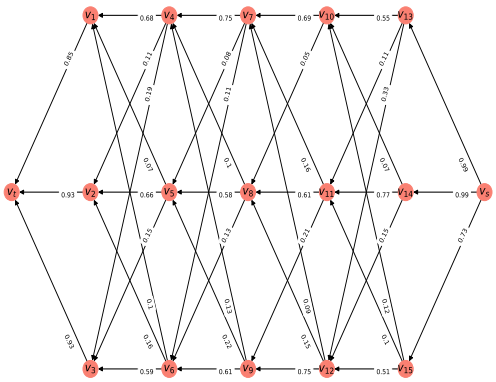
Figure 3: Initial network flow to be used in our proposed method, the multi-commodity flow with multiple sources and targets, and the network flow for MCC problems.



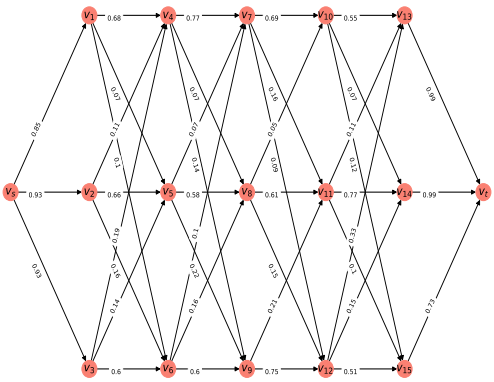
(a) Network flow generated via Algorithm 1



(b) Network flow generated via Algorithm 2

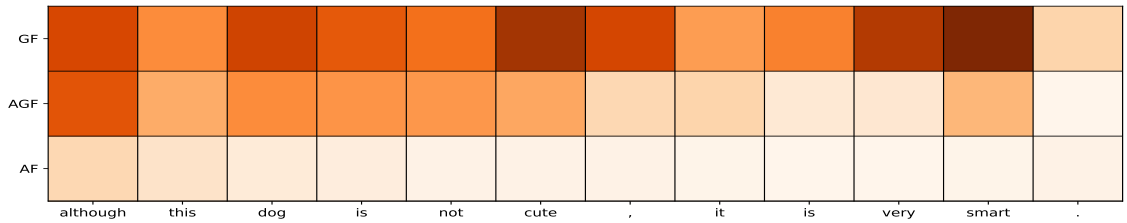


(c) Optimal flow generated via Algorithm 1

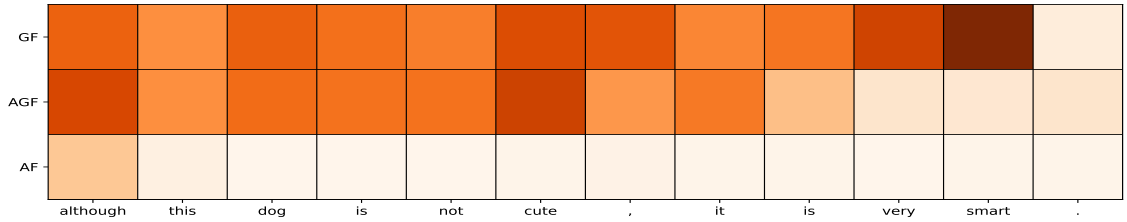


(d) Optimal flow generated via Algorithm 2

Figure 4: Network flows and optimal flows generated by Algorithm 1 and Algorithm 2. The optimal flows computed using Algorithm 1 and Algorithm 2 are not equivalent.

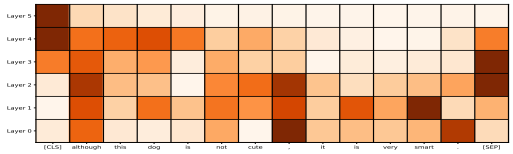


(a) Normalized feature attributions for Transformer's input layer generated by Algorithm 1 for different information tensors.

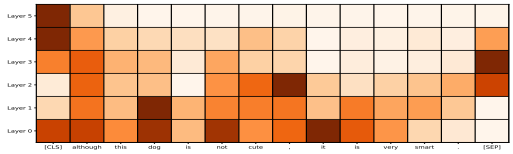


(b) Normalized feature attributions for Transformer's input layer generated by Algorithm 2 for different information tensors.

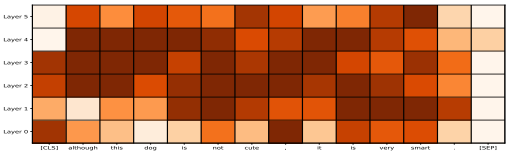
Figure 5: Normalized feature attributions for Transformer's input layer and different information tensors.



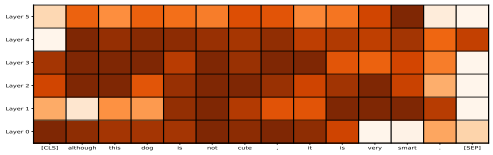
(a) AF method:Algorithm 1



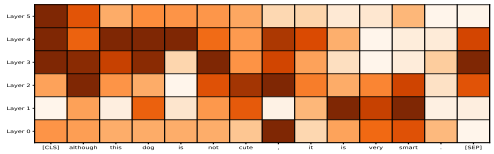
(b) AF method:Algorithm 2.



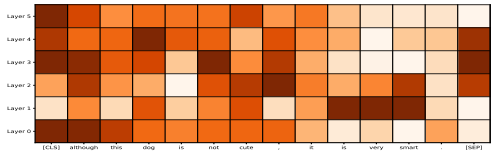
(c) GF method:Algorithm 1.



(d) GF method:Algorithm 2.



(e) AGF method:Algorithm 1.



(f) AGF method:Algorithm 2.

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

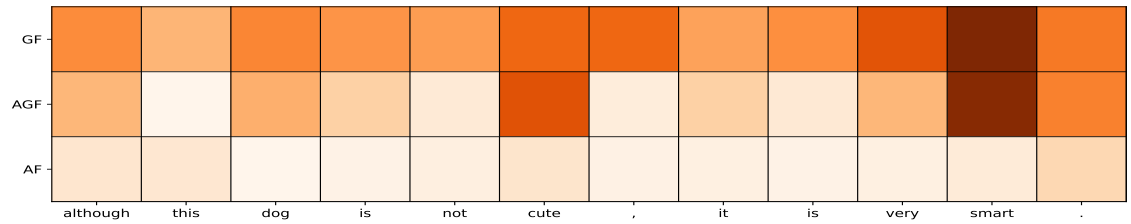


Figure 7: Visualizations of the feature attributions generated by running our proposed method on the three proposed information tensors on the showcase example.

ing the highest average AOPC and LOdds scores, notably for the IMDB and Amazon datasets. This consistently strong performance confirms AGF's robustness in identifying the most important tokens in the model's decision-making process.

We also evaluated various feature attribution

methods using classification metrics, with results summarized in Tab. 3. This table represents the average Accuracy, F1, Precision, and Recall scores across multiple k values. On the SST2 dataset, AGF and GF, alongside KernelShap, achieved the highest performance. For IMDB, AGF and GF, with Integrated Gradients (IG), outperformed other methods. On Amazon, AGF and GF, combined with TransAtt, led the competition. On AG News, AGF and AF, paired with AttGrads, demonstrated superior results. The strong performance of AGF across multiple datasets and metrics underscores its versatility and reliability to identify important tokens, confirming its effectiveness as a feature attribution method in NLP models.

However, the Yelp dataset presents a distinct challenge where our proposed methods, including AGF, do not consistently achieve optimal results across all evaluation metrics. This performance gap is likely due to the unique characteristics of the Yelp dataset, which often contains informal language, colloquialisms, and typographical errors. The occurrence of linguistic noises in Yelp reviews is significantly higher than in other datasets like IMDB or Amazon. These textual noises introduce complexity that AGF, in its current configuration, may find challenging to address effectively.

Fig. 9 compares feature attribution methods that were assessed using a model trained on the SST2 dataset, focusing on the aforementioned sentence. All methods predict a positive sentiment for the example presented. Our methods, AGF and GF, effectively identify the most important tokens, such as 'cute' and 'smart' (highlighted with dark orange shading), which play a pivotal role in the positive sentiment prediction. Some other methods, including Grads, LIME, RawAtt, and PartialLRP, also demonstrate some capability to identify important tokens. However, methods like AF, AttCAT, CAT, Rollout, KernelShap, and IG struggle to accurately identify these important tokens.

D.3 Statistical Significance Test

To perform a statistical significance test, we have employed the ASO (Almost Stochastic Order) method (Ulmer et al., 2022; Dror et al., 2019; del Barrio et al., 2017). This method compares the cumulative distribution functions (CDFs) of two score distributions to assess stochastic dominance. Importantly, ASO does not make any assumptions about the score distributions, which allows it to be applied 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 $\epsilon_{\min}$ signifies greater confidence in the superiority of model A. The null hypothesis for the ASO method is defined as follows:

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

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

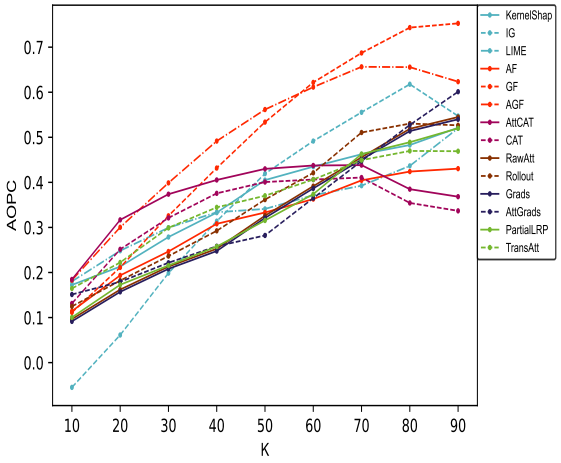
In this research, we conduct 500 independent runs for each method to perform comprehensive statistical tests, comparing the AOPC and LOdds metrics of our top-performing proposed method, AGF, against the top benchmark methods listed in Tab. 1 and Tab. 2, using $\tau = 0.5$. As shown in Tab. 4 and Tab. 5, the AGF method consistently outperforms these benchmark techniques across all datasets, except for the Yelp dataset.

E Proofs

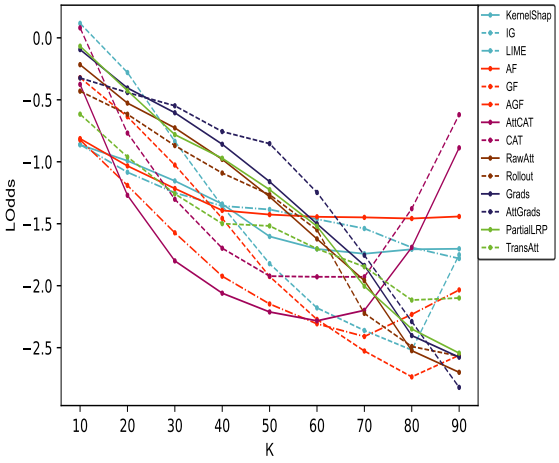
Corollary 3.1. While Shapley values are unique, our findings in Sec. 3.3 and App. C present a critical inconsistency. We demonstrated that the optimal solution of the maximum flow problems defined in eq. 8 is not necessarily unique, thereby disproving the claim that the feature attributions proposed by Ethayarajh and 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 and 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, we can derive corresponding feature attributions from each optimal solution that differ from one another. □

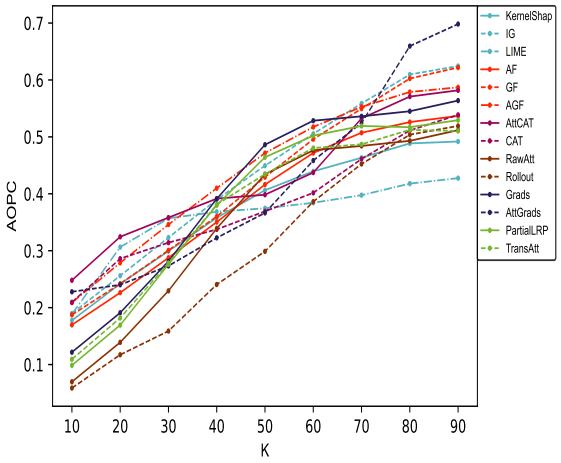
Theorem 3.1. As the optimal flow $\boldsymbol{f}^*$ is computed once for the entire graph and not for each potential subgraph, and the players (tokens) are all disjoint without any connections in S, blocking the flow through one player does not impact the outflow



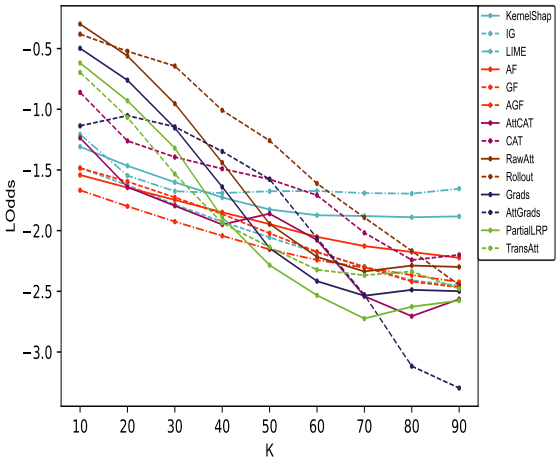
(a) AOPC score for IMDB dataset



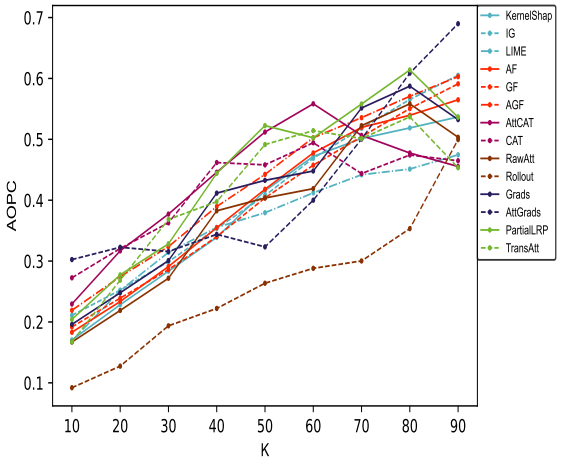
(b) LOdds score for IMDB dataset



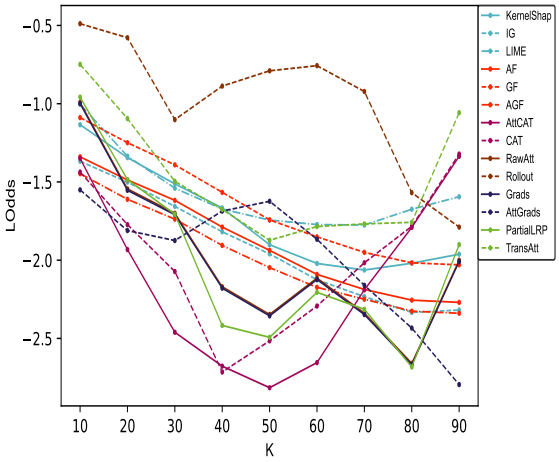
(c) AOPC score for Amazon dataset



(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.

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	**0.62**	**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
DecompX	0.66	0.66	0.66	0.67	0.59	0.60	0.58	0.59	0.73	0.74	0.73	0.72	0.69	0.70	0.69	0.69	0.71	0.69	0.70	0.72
GlobEnc	0.70	0.70	0.70	0.69	0.61	0.62	0.60	0.61	0.70	0.71	0.69	0.71	0.71	0.73	0.71	0.71	0.72	0.72	0.73	0.71
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

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. Best results are in bold.

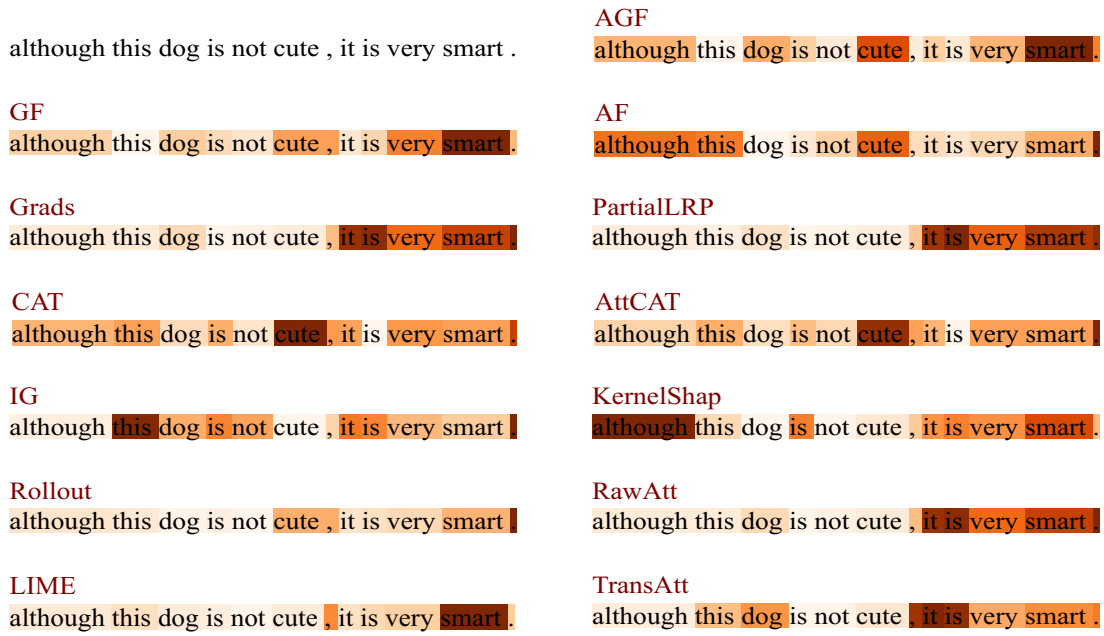


Figure 9: Normalized feature attributions generated by the methods on the binary classification task.

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 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
$\epsilon_{\min}$	0.049	0.037	0.113	0.085	0.913	0.824	0.062	0.053	0.091	0.067

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

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:

$$\begin{aligned}\phi_i(\vartheta) &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (\vartheta(S \cup \{i\}) - \vartheta(S)) \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} (|f_{\text{out}}(i)|) = |f_{\text{out}}(i)|\end{aligned} \tag{21}$$

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

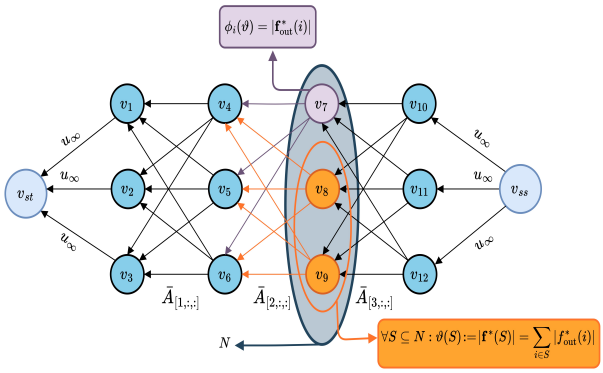


Figure 10: The procedure to define the cooperative game (N, ϑ) employing the solution of barrier-regularized maximum flow or its corresponding MCC problem.

F Implementation Details

F.1 Datasets

Tab. 6 illustrates comprehensive statistics of the datasets utilized for the classification task. We randomly selected $5,000$ 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.

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:

$$\begin{array}{l} \arg\min \boldsymbol{c}^\top \boldsymbol{f} \\ \boldsymbol{B}^\top \boldsymbol{f} = \boldsymbol{0} \\ \boldsymbol{l} \leq \boldsymbol{f} \leq \boldsymbol{u} \end{array}$$

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

To comprehensively compare the computational efficiency between algorithms, we consider $\widetilde{\boldsymbol{l}}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{c}}$ as the integral representations of the parameters $\boldsymbol{l}, \boldsymbol{u}, \boldsymbol{c}$, derived from Algorithm 1 or Algorithm 2 and define $U = \|\widetilde{\boldsymbol{u}}\|_\infty$ and $C = \|\widetilde{\boldsymbol{c}}\|_\infty$. Tab. 7 provides a comparative analysis of state-of-the-art iterative algorithms designed to solve the maximum flow and minimum-cost circulation problems.

While the most efficient algorithms exhibits asymptotically near-linear runtime with respect to the number of edges m, its computational cost can still be significant, particularly when we have long input sequences. To solve the minimum-cost circulation problem efficiently, we have implemented the algorithm proposed by Lee and Sidford (2014) using CVXPY (Agrawal et al., 2017; Diamond and Boyd, 2016), which ensures numerical stability and computational efficiency.

This study has been conducted on a computing device running Ubuntu 20.04.4 LTS. The system is powered by Intel(R) Xeon(R) Platinum 8368 CPUs, which operate at a clock speed of 2.40 GHz. This processor features 12 physical cores and 24 threads, enabling efficient parallel computing and optimized execution of computationally intensive tasks. The graphical computations were handled by an NVIDIA RTX 3090 Ti GPU, equipped with 40 GB of dedicated VRAM, ensuring high-speed processing of deep learning and machine learning workloads. The system is also equipped with 230 GB of dedicated system memory, ensuring smooth and efficient experimentation.

Tab. 8 compares the runtime of all methods to compute feature attributions of each token in the sentence "although this dog is not cute, it is very smart.". Examining the runtime performance in Tab. 8, we can categorize the feature attribution methods into three primary categories based on their computational complexity:

Fast Attention-Based Methods:

RawAtt and Rollout methods exhibit significant computational efficiency by utilizing pre-computed attention weights directly, thereby obviating the necessity for complex post-processing or gradient computations.

DecompX and GlobEnc also show reasonable computational efficiency, balancing computational cost with enhanced analysis capabilities. These methods incorporate model information beyond raw attention while offering optimized algorithmic implementations and reasonable runtimes.

Moderate Gradient Methods:

This tier includes Grads, AttGrads, CAT, AttCAT, PartialLRP, and TransAtt. These methods exhibit moderate computational costs, likely due to their

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

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

Year	MCC Bound	Max-Flow Bound	Author
2014	$O\left(m\sqrt{n}\,\text{polylog}(n)\log^2(U)\right)$	$O\left(m\sqrt{n}\,\text{polylog}(n)\log^2(U)\right)$	Lee and Sidford (2014)
2022	$O\left(m^{\frac{3}{2}-\frac{1}{762}}\,\text{polylog}(n)\log(U+C)\right)$	$O\left(m^{\frac{10}{7}}\,\text{polylog}(n)U^{\frac{1}{7}}\right)$	Axiotis et al. (2022)
2023	$O\left(m^{\frac{3}{2}-\frac{1}{58}}\,\text{polylog}(n)\log^2(U)\right)$	$O\left(m^{\frac{3}{2}-\frac{1}{58}}\,\text{polylog}(n)\log^2(U)\right)$	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)

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

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
DecompX	0.526
GlobEnc	0.413
AF	2.301
GF	2.305
AGF	2.306

Table 8: Runtime of methods for the showcase example.

reliance on gradient computations or layer-wise propagation mechanisms.

Although these methods are significantly slower than attention-based methods, they demonstrate enhanced performance, as shown by our evaluation metrics.

Slow and Complex Methods:

KernelShap, IG, and our proposed methods AF, GF, and AGF are the most computationally intensive approaches. The longer runtimes of these methods reflect their algorithmic complexity: KernelSHAP relies on sampling requirements, IG employs path integration, and our methods utilize optimization-based flow calculations.

AGF and GF methods offer competitive runtimes compared to other methods in this category, while also delivering improved attribution quality. While our proposed methods fall into the category with the highest runtime, their computational costs are reasonable for most scenarios. This trade-off is justifiable given their superior performance across various evaluation metrics.