PE: A Poincare Explanation Method for Fast Text Hierarchy Generation

Anonymous EMNLP submission

Abstract

The black-box nature of deep learning models in NLP hinders their widespread application. The research focus has shifted to Hierarchical Attribution (HA) for its ability to model feature interactions. Recent works model non-contiguous combinations with a time-costly greedy search in Euclidean spaces, neglecting underlying linguistic information in feature representations. In this work, we introduce a novel method, namely Poincare Explanation (PE), for modeling feature interactions with hyperbolic spaces in a time efficient manner. Specifically, we take building text hierarchies as finding spanning trees in hyperbolic spaces. First we project the embeddings into hyperbolic spaces to elicit inherit semantic and syntax hierarchical structures. Then we propose a simple yet effective strategy to calculate Shapley score. Finally we build the the hierarchy with proving the constructing process in the projected space could be viewed as building a minimum spanning tree and introduce a time efficient building algorithm. Experimental results demonstrate the effectiveness of our approach.

1 Introduction

Deep learning models have been ubiquitous in Natural Language Processing (NLP) areas accompanied by the explosion of the parameters, leading to increased opaqueness. Consequently, a series of interpretability studies have emerged (Abnar and Zuidema, 2020; Geva et al., 2021; He et al., 2022), among them feature attribution methods stand out owing to fidelity and loyalty axioms and straightforward applicability (Guidotti et al., 2018).

Previous feature-based works are limited to single words or phrases (Miglani et al., 2020). However, Mardaoui and Garreau (2021) point out that LIME's (Ribeiro et al., 2016) performance on simple models is not plausible [1]. To model feature interactions, Hierarchical Attribution (HA) (Chen et al., 2020; Ju et al., 2023) has been introduced, with a attribution-then-cluster stage in which constructs feature interaction process by distributing text group scores at different levels[2]. From bottom to the up, HA categorizes all words into different clusters, ending with a tree structure.

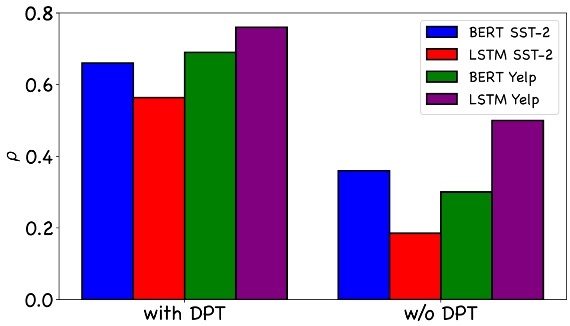


Figure 1: Pearson correlation ρ results from Jin et al. (2020) with BERT and LSTM on SST-2 and Yelp datasets. A higher correlation coefficient indicates a stronger ability of the method to identify important words.

However, building feature hierarchies is not a trivial thing. Existing methods have three following problems. **P-1**: Detecting contiguous text spans to replace all possible interactions (Singh et al., 2019; Chen et al., 2020). Only using spans might lose long-range dependencies in text (Vaswani et al., 2017). For example, in the positive example "*Even in moments of sorrow, certain memories can evoke happiness*", ("*Even*", "*sorrow*") is vital and non-adjacent. **P-2**: Current algorithms estimating the importance of feature combinations are accompanied by lengthy optimization processes (Ju et al., 2023; Chen et al., 2020). For example, HE (Ju et al., 2023) estimates the importance of words using LIME algorithm and then enumerates word combinations to construct the hierarchy, with a cu-

[1] A figure illustration is provided in Appendix E.

[2] A vivid HA example is provided in Appendix D.

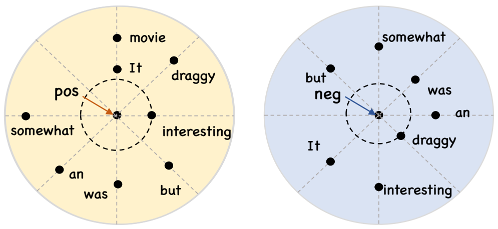


Figure 2: Left: The projection illustration for positive example "*It was an interesting but somewhat draggy movie*." The centre represents the prototype for the positive label. Right: A negative example "*It was a draggy but somewhat interesting movie*." The center point stands for the negative label.

bic time complexity[3]. ASIV (Lu et al., 2023) uses directional Shapley value to model the direction of feature interactions, while estimating Shapley value requires exponential time. **P-3**: Previous methods cannot model the linguistic information including syntax and semantic information. Syntax and semantics can help to construct a hierarchical tree. For syntax, Jin et al. (2020) build hierarchies directly on Dependency Parsing Trees (DPT) and compute Pearson Correlation (i.e.ρ). The results in Figure 1 demonstrate syntax could contribute to building explainable hierarchies by reaching a higher correlation. For semantic, we take Figure 2 as an example, the hierarchy in hyperbolic space has already achieved preliminary interpretability with the proximity corresponding the polarity.

As the input text length continues to increase, efficiently modeling the interaction of non-contiguous features has become a key challenge in promoting HA. Building a hierarchical attribution tree based on the input text is essentially a *hierarchical clustering* problem. The definition is as follows: given words and their pairwise similarities, the goal is to construct a hierarchy over clusters (word groups). PE approaches this problem by following three steps. First, to model linguistic hierarchical information, we project word embeddings into hyperbolic spaces to uncover hidden semantics and syntax structures. Next, inspired by cooperative game theory (Owen, 2013), we regard words as players and clusters as coalitions and introduce a simple yet effective strategy to estimate the Shapley score contribution. Finally we calculate pairwise similarities and propose an algorithm that conceptualizes the bottom-up clustering process as generating a minimum spanning tree.

Our contributions are summarized as follows:

- We propose a method, PE, using hyperbolic geometry for generating hierarchical explanations, revealing the feature interaction process.
- PE introduces a fast algorithm for generating hierarchical attribution trees that model non-contiguous feature interactions.
- We evaluate the proposed method on three datasets with BERT (Devlin et al., 2019), and the results demonstrate the effectiveness.

[3]For convenience of comparison, we ignore the time taken by linear regression in LIME algorithm and detailed discussion is in Section 6.

2 Related Work

Feature importance explanation methods mainly assign attribution scores to features (Qiang et al., 2022; Ferrando et al., 2022; Modarressi et al., 2023). Methods can be classified into two categories: single-feature explanation type and multi-feature explanation type.

2.1 Single-Feature Explanation

Earlier researches focus on single feature attribution (Ribeiro et al., 2016; Sundararajan et al., 2017; Kokalj et al., 2021). For example, LIME (Ribeiro et al., 2016) aims to fit the local area of the model by linear regression with sampled data points ending with linear weights as attribution scores. Gradient&Input (Grad$\times$Inp) (Shrikumar et al., 2017b) combines the gradient norm with Shapley value (Shapley et al., 1953). Deeplift (Shrikumar et al., 2017a) depends on activation difference to calculate attribution scores. IG (Sundararajan et al., 2017; Sanyal and Ren, 2021; Enguehard, 2023) uses path integral to compute the contribution of the single feature to the output. It is noticeable that IG is the unique path method to satisfy the completeness and symmetry-preserving axioms. There exist several variants of IG. DIG (Sanyal and Ren, 2021) regards similar words as interpolation points to estimate the integrated gradients value. SIG (Enguehard, 2023) computes the importance of each word in a sentence while keeping all other words fixed. However, scoring individual features is incompatible with interactions between features.

2.2 Multi-Feature Explanation

Multi-feature explanation methods aim to model feature interactions in deep learning architectures.

For example, Dhamdhere et al. (2020) proposes a variant of Shapley value to measure the interactions. Zhang et al. (2021a) defines the multivariant Shapley value to analyze interactions between two sets of players. Enouen and Liu (2022) proposes a sparse interaction additive network to select feature groups. Tsang et al. (2020) proposes an Archipelago framework to measure feature attribution and interaction through ArchAttribute and ArchDetect. Lu et al. (2023) proposes ASIV to model asymmetric higher-order feature interactions. To illustrate the feature interplay process completely, the explanation of feature interaction could be articulated within a hierarchical framework. HEDEG (Chen et al., 2020) designs a top-down model-agnostic hierarchical explanation method, with neglecting non-contiguous interactions. Ju et al. (2023) addresses the connecting rule limitation in HEDGE, and proposes a greedy algorithm , HE, for generating hierarchical explanations, which is time-costly. And they all neglect linguistic information including syntax and semantics.

3 Background

We first give a review of hyperbolic geometry.

Poincare ball A common representation model in hyperbolic space is the Poincare ball, denoted as $(\mathcal{B}_c^m, g_{\boldsymbol{x}}^{\mathcal{B}})$, where c is a constant greater than 0. $\mathcal{B}_c^m = \{\boldsymbol{x} \in \mathbb{R}^m \mid c\|\boldsymbol{x}\|^2 < 1\}$ is a Riemannian manifold, and $g_{\boldsymbol{x}}^{\mathcal{B}} = (\lambda_{\boldsymbol{x}}^c)^2 \boldsymbol{I}_m$ is its metric tensor, $\lambda_{\boldsymbol{x}}^c = 2/(1 - c\|\boldsymbol{x}\|^2)$ is the conformal factor and c is the negative curvature of the hyperbolic space. PE uses the standard Poincare ball with $c = 1$. The distance for $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{B}_c^m$ is:

$$d_{\mathcal{B}}(\boldsymbol{x}, \boldsymbol{y}) = 2\tanh^{-1}\|-\boldsymbol{x} \oplus_c \boldsymbol{y}\|, \qquad (1)$$

where $\oplus_c$ denotes the *Möbius addition*. We use $\otimes_c$ to denote the *Möbius matrix multiplication*. The *Möbius addition* for $\boldsymbol{x}$, $\boldsymbol{y} \in \mathbb{R}^m$ is defined as (Demirel, 2013):

$$\boldsymbol{x} \oplus_c \boldsymbol{y} = \frac{(1 + 2\langle \boldsymbol{x}, \boldsymbol{y}\rangle + \|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 - \|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 + 2\langle \boldsymbol{x}, \boldsymbol{y}\rangle + \|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2}. \qquad (2)$$

Given a linear projection $\boldsymbol{A} : \mathbb{R}^m \to \mathbb{R}^p$ and $\boldsymbol{x} \in \mathcal{B}_c^m$, then the *Möbius matrix multiplication*is defined as (Demirel, 2013):

$$\boldsymbol{A} \otimes_c \boldsymbol{x} = \tanh\left(\frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\tanh^{-1}(\|\boldsymbol{x}\|)\right)\frac{\boldsymbol{A}\boldsymbol{x}}{\|\boldsymbol{A}\boldsymbol{x}\|}. \qquad (3)$$

Cooperative Game Theory We use N to denote a set of players (i.e. token set). A game is a pair $\Gamma = (N, v)$ and $v : 2^N \to \mathbb{R}$ is the characteristic function. A coalition is any subset of N. In a cooperative game, players can form coalitions, and each coalition $S \subseteq N$ has a value $v(S)$.

4 Methodology

This section provides a detailed introduction to the three parts of PE. First, we need to score each feature; then, based on these scores, we construct a hierarchy. In Section 4.1, we consider semantic and syntax factors. Besides we facilitate feature Shapley contribution calculation in Section 4.2. In Section 4.3, we combine these factors to score each feature and propose a fast algorithm for constructing the hierarchy.

4.1 Poincare Projection

In this paper, we choose Probing (Hewitt and Manning, 2019) to recover information from embeddings. Namely, we train two matrices to project the Euclidean embeddings to hyperbolic spaces. For a classification task, given a sequence $X_i = \{x_j\}_{1 \le j \le n}$ and a trained model f, n is the sequence length. $\hat{y}$ represents the predicted label, and $f(\cdot)$ represents the model's output probability for the predicted label.

4.1.1 Label Aware Semantic Probing

In this subsection, we extract the semantics from the embeddings through probing. We project the embeddings into a hyperbolic space using a transformation matrix. In this space, the distribution of examples with different semantics will change according to their semantic variations. First, we feed the sequence X_i into a pre-trained language model to obtain the contextualized representations $\boldsymbol{E}_i \in \mathbb{R}^{n \times d_{in}}$, with d_{in} denotes the output dim. Next, the sentence embedding $\boldsymbol{s}_i \in \mathbb{R}^{d_{in}}$ is obtained by the hidden representations of the special tag (e.g.[CLS]), which is the first token of the sequence and used for classification tasks. Our probing matrix consists of two types: $\boldsymbol{A}_{se}, \boldsymbol{A}_{sy} \in \mathbb{R}^{d_{in} \times d_{out}}$ (d_{out} denotes the projection dim) for probing label-aware semantic information and syntax information. For semantics, we can obtain the projected representation:

$$\boldsymbol{s}_i^{se} = \boldsymbol{A}_{se} \otimes_c \boldsymbol{s}_i. \qquad (4)$$

Also we can obtain the token presentation:

$$\boldsymbol{e}_j^{se} = \boldsymbol{A}_{se} \otimes_c \boldsymbol{e}_j. \qquad (5)$$

To train the probing matrices, we draw inspiration from prototype networks (Snell et al., 2017), assuming that there exist k centroids representing labels in the hyperbolic space. The closer a point is to a centroid, the higher the probability that it belongs to that category. Specifically, instead of using mean pooling to calculate the prototypes, we directly initialize the prototype embeddings in hyperbolic space, denoted as $\boldsymbol{\omega} = \{\boldsymbol{c}_k\}$ ($\boldsymbol{c}_k$ is the k-th label centroid). Given a distance $d_{\mathcal{B}}$, the prototypes produce a distribution over classes for a point $\boldsymbol{x}$ based on a softmax over distances to prototypes in the embedding space:

$$\mathcal{P}(y = k \mid \boldsymbol{\omega}) = \frac{\exp(-d_{\mathcal{B}}(\boldsymbol{s}_i^{se}, \boldsymbol{c}_k))}{\sum_{k'} \exp(-d_{\mathcal{B}}(\boldsymbol{s}_i^{se}, \boldsymbol{c}_{k'}))}. \quad (6)$$

We minimize the negative log-probability $J(\boldsymbol{\omega}) = -log\mathcal{P}(y = k \mid \boldsymbol{\omega})$ of the true class k via RiemannianAdam (Kochurov et al., 2017).

4.1.2 Syntax Probing

Similarly, in this subsection, we obtain syntax through probing. The difference is that for syntax, we focus on tokens. In the projected hyperbolic space, the distance of the token embeddings from the origin and the distance between tokens correspond to the depth of the tokens and their distance in the DPT respectively. We project word embeddings first:

$$\boldsymbol{e}_j^{sy} = \boldsymbol{A}_{sy} \otimes_c \boldsymbol{e}_j, \quad (7)$$

where $\boldsymbol{e}_j = \boldsymbol{E}_{j,:}$. How to parameterize a dependency tree from dense embeddings is non-trivial. Following Hewitt and Manning (2019), we define two metrics to measure the deviation from the standard: using the distance between two words in embedding space to represent the distance of word nodes in the dependency tree, and using the distance of a word from the origin to represent the depth of the word node. We use the following two loss functions:

$$\mathcal{L}_{\text{dis}} = \frac{1}{n^2} \sum_{j,j' \in [n]} |d_{DPT}(x_j, x_{j'}) - d_{\mathcal{B}}(\boldsymbol{e}_j^{sy}, \boldsymbol{e}_{j'}^{sy})^2|, \quad (8)$$

$$\mathcal{L}_{\text{dep}} = \frac{1}{n} \sum_{j \in [n]} |d_{DPT}(x_j) - d_{\mathcal{B}}(\boldsymbol{e}_j^{sy}, \boldsymbol{0})^2|. \quad (9)$$

where $d_{DPT}(x_j, x_{j'})$ and $d_{DPT}(x_j)$ represent the distance of words and the depth of words respectively. And $d_{\mathcal{B}}(\boldsymbol{e}_j^{sy}, \boldsymbol{0})$ denotes the distance between $\boldsymbol{e}_j^{sy}$ and the origin in the projected hyperbolic space.

4.2 Shapley Contribution Estimation

According to cooperative game theory, we regard the input as a set of players N, where each element of the set corresponds to a word, and the process of hierarchical clustering is viewed as a game, with clusters containing more than two words considered a coalition. Following Zhang et al. (2021b), we define the characteristic function as $v = f$. Given a game $\Gamma = (N, v)$, a fair payment scheme rewards each player according to its contribution. The Shapley value removes the dependence on ordering by taking the average over all possible orderings for fairness. The Shapley value of player j in a game is as follows:

$$\phi_j = \frac{1}{|N|!} \sum_{\pi \in \Pi(N)} [v(Q_j^\pi \cup \{j\}) - v(Q_j^\pi)], \quad (10)$$

where $\Pi(N)$ is the set of all permutations of the players, Q_j^π is the set of players preceding player j (i.e. token j) in permutation π. $v(S)$ is the value that the coalition of players $S \subseteq N$ can achieve together. In practical, Monte Carlo sampling is used:

$$\hat{\phi}_j = \frac{1}{R} \sum_{r=1}^{R} v(Q_j^{\pi_r} \cup \{j\}) - v(Q_j^{\pi_r}) \quad (11)$$

where π_r denotes the r-th permutation in $\Pi(N)$. Unfortunately, Monte Carlo sampling methods can exhibit slow convergence (Mitchell et al., 2022).

It is noticeable that attention mechanism of Transformer is permutation invariant (Vaswani et al., 2017; Xilong et al., 2023), and the sinusoidal position embedding is only related to the specific position, not to the word. Moreover, after being trained with a Language Modeling task, the model has the ability to fill in the blanks based on context. Therefore, we assume that it is unnecessary to enumerate exponential combinations of words and the contribution of preceding permutation set (e.g.$\pi(< r)$) is included in larger subsequent permutation sets (e.g.$\pi(r)$). Therefore, we directly calculate contribution as follows:

$$\begin{aligned}\tilde{\phi}_j &= v(N) - v(N \setminus \{j\}) \\ &= f(X) - f(X \setminus \{x_j\})\end{aligned} \quad (12)$$

where $N \setminus \{j\}$ denotes the player set excluding player j and $X \setminus \{x_j\}$ denotes the input excluding token x_j.

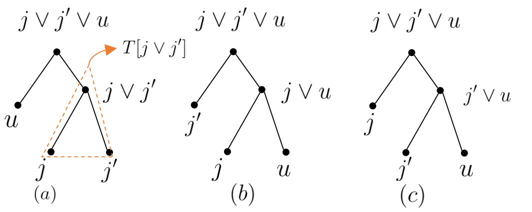


Figure 3: Three different binary tree types rooted from $j \vee j' \vee u$.

4.3 Minimum Spanning Tree

Our goal is to identify a hierarchy tree T that aligns with semantic similarities, syntax similarities, and the contributions of individual elements. Building upon Dasgupta (2016), we use the following cost:

$$C_D(T;e) = \sum_{j,j' \in [n]} e_{jj'} |leaves(T[j \vee j'])|, \quad (13)$$

where $e_{j,j'}$ denotes the pairwise similarities, $leaves(T[j \vee j'])$ is leaves of $T[j \vee j']$, which is the subtree rooted at $j \vee j'$, $j \vee j'$ is the parent node of j and j' as shown in Figure 3. Due to the unfolding dilemma of $leaves(T[i \vee j])$ process, we adopt following expansion by Wang and Wang (2018):

$$\begin{aligned} C_D(T;e) = &\sum_{jj'u \in [n]} [e_{jj'} + e_{ju} + e_{j'u} \\ &- e_{jj'u}(T)] + 2 \sum_{jj'} e_{jj'}, \end{aligned} \quad (14)$$

where

$$\begin{aligned} e_{jj'u}(T) = &e_{jj'} \mathbb{1}[\{j,j' \mid u\}] + e_{ju} \mathbb{1}[\{j,u \mid j'\}] \\ &+ e_{j'u} \mathbb{1}[\{j',u \mid j\}], \end{aligned} \quad (15)$$

where $\{j,j' \mid u\}$ means the $j \vee j'$ is the descendant of $j \vee j' \vee u$, illustrated in Figure 3. The same for $\{j,u \mid j'\}$ and $\{j',u \mid j\}$.

We aim to find the binary tree T^*:

$$T^* = \underset{\text{all binary trees } T}{\operatorname{argmin}} \; C_D(T;e). \quad (16)$$

Directly optimizing this cost presents a combinatorial optimization problem. We introduce the following decomposition:

$$\begin{aligned} e_{jj'} = &-\tilde{\phi}(j \vee j') + \alpha_1 d_{\mathcal{B}}(\boldsymbol{e}_j^{se}, \boldsymbol{e}_{j'}^{se}) \\ &+ \frac{1}{2}\alpha_2 (d_{\mathcal{B}}(\boldsymbol{e}_j^{sy}, \mathbf{0}) + d_{\mathcal{B}}(\boldsymbol{e}_{j'}^{sy}, \mathbf{0})), \end{aligned} \quad (17)$$

where α_1,$\alpha_2 \in [0,1]$.

Under that we prove the optimal tree T^* is a like-minimum spanning tree of Equation14.[4] The proof can be found in Appendix A. Ultimately we introduce the following decoding algorithm:

Algorithm 1 Building Algorithm

Input: Label hyperbolic embeddings $\boldsymbol{E}^{se} = \{\boldsymbol{E}_1^{se}, \cdots, \boldsymbol{E}_n^{se}\}$, syntax hyperbolic embeddings $\boldsymbol{E}^{sy} = \{\boldsymbol{E}_1^{sy}, \cdots, \boldsymbol{E}_n^{sy}\}$
Output: Binary tree T with n leafs

```
T ← ({x_j} : x_j ∈ X)
Initialize a PriorityQueue ϒ
ϒ ← {(x_j, x_j') : pairs sorted by e_jj'}
while ϒ ≠ ∅ do
    x_j, x_j' ← ϒ.front, ϒ.pop
    if x_j and x_j' not in T then
        T ← T ∪ {x_j ∨ x_j'}
        ϒ.push(x_i ∨ x_j)
    end if
end while
```

5 Experiments

5.1 Experimental Setups

Datasets To evaluate the effectiveness of PE, we perform comprehensive experiments on three representative text classification datasets: "Rotten Tomatoes" (Pang and Lee, 2005), "TREC" (Li and Roth, 2002), "Yelp" (Zhang et al., 2015). Detailed statistics are in Table 1.

Datasets	Train/Dev/Test	C	L
Rotten Tomatoes	10K/2K/2K	2	64
TREC	5000/452/500	6	64
Yelp	10K/2K/1K	2	256

Table 1: Statistics of three datasets. C: number of classes, L: average text length

Metrics Following prior literature (DeYoung et al., 2020), we use AOPC metric, which is the average difference of the change in predicted class probability before and after removing top K words.

$$\text{AOPC} = \frac{1}{n} \sum_{K} (f(x_i) - f(\tilde{x}_i^K)) \quad (18)$$

Higher is better. And we evaluate two different strategies: *del* and *pad*. Concretely, We assign

[4] The difference from the original minimum spanning tree is located in the last paragraph of Appendix A.

values to words through the following formula:

$$\text{score}_i = \tilde{\phi}(j) - \beta_1 d_{\mathcal{B}}(\boldsymbol{e}_j^{se}, \boldsymbol{c}_k) - \beta_2 d_{\mathcal{B}}(\boldsymbol{e}_j^{sy}, \mathbf{0}), \tag{19}$$

where $\boldsymbol{c}_k$ is the prototype of predicted label k in the semantic hyperbolic space, $\mathbf{0}$ is the origin in the syntactic hyperbolic space, $\beta_1, \beta_2 \in [0, 1]$.

Infrastructures All experiments are processed on one 15 core 2.6GHz CPU (Intel(R) Xeon(R) Platinum 8358P) and one RTX3090 GPU.

Baselines We compare PE with three hierarchical attribution methods: HEDGE (Chen et al., 2020), HE_{LIME} , HE_{LOO} (Ju et al., 2023) and three feature interaction methods: SOC (Jin et al., 2020), Bivariate Shapley (BS)(Masoomi et al., 2022) and ASIV (Lu et al., 2023).

5.2 General Experimental Results

We first evaluate our method using the AOPC metric across three datasets, as shown in Tables 2 and 3. **Firstly**, our method, PE, consistently surpasses the baseline in binary and multiclass tasks for both short and long texts. For instance, PE outperforms HE_{LOO} by 0.235 in Table 2 and by 0.067 in Table 3 of AOPC_{del},20%, Rotten Tomatoes / Yelp setting. **Second**, in comparison to recent works such as SOC and HE_{LOO}, our method's primary advantage lies in its computation efficiency. We conduct an analysis comparing the average time of various approaches to construct HA trees. The results in Table 3 indicate that PE substantially outperforms its counterparts in terms of speed, being twice as fast as SOC and six times faster than HE_{LIME}.

5.3 Ablation Study

We conduct ablation experiments with three modified baselines from PE: PE w/o prob corresponding $\tilde{\phi}(i) = 0$, PE w/o semantic corresponding $\beta_1 = 0$ and PE w/o syntax corresponding $\beta_2 = 0$.

As shown in Figure 4, both PE and variants outperform w/o prob baselines, demonstrating our approach's effectiveness in directly calculating contributions in Equation 12. Moreover, we observe that both in *del* and *pad* settings, the utility of estimating contribution is more striking than the other two components in Equation 19. The reason may be that context has a greater impact on output than single semantics and syntax. It is noticeable that syntax slightly outperforms semantics, we hypothesis that the reason might be related to the nature of the tasks in the TREC dataset, as the labels tend to associate with syntactic structures (Li and Roth, 2002).

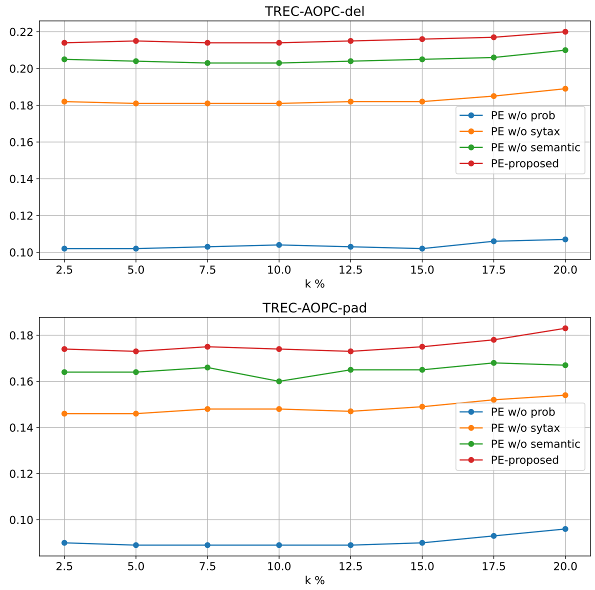


Figure 4: Evaluation results of Ablation Study.

5.4 Case Study

For qualitative analysis, we present two typical examples from the Rotten Tomatoes dataset to illustrate the role of PE in modeling the interaction of discontinuous features and we show more examples in Appendix B. In the first example, we compare the results of PE and HE_{LOO} in interpreting BERT model. Figure 5 provides two hierarchical explanation examples for a positive and negative review, each generated by PE and HE_{LOO} respectively. In Figure 5(a), it can be seen that PE accurately captures the combination of words with positive sentiment polarity: *delightful*, *out*, and *humor*, and captures the key combination of *out* and *humor* at step 1. Additionally, this example includes a word with negative polarity: *stereotypes*, where it can be observed that HE_{LOO} captures its combination with *in* and *delightful*, missing the combination with *out* and *humor*. In Figure 5(b), PE captures the combination of *slightest* and *wit* in the first phase and complements it with the combination of *lacking* at step 2. HE captures the combination of *combination* and *animation* at step 1, and it adds *lacking* at step 2. We can infer that PE is able to capture the feature combination more related to the label at a shallow level, which demonstrates the effectiveness of our method.

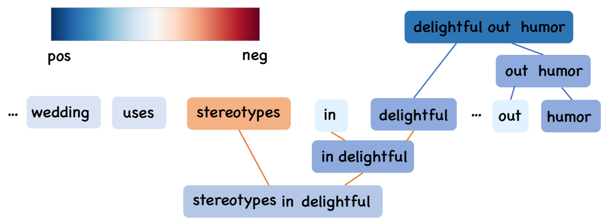
Additionally, to more vividly demonstrate the role of semantics and syntax in building hierar-

Datasets / Methods	Rotten Tomatoes						TREC					
	AOPC$_{del}$			AOPC$_{pad}$			AOPC$_{del}$			AOPC$_{pad}$		
	10%	20%	Avg	10%	20%	Avg	10%	20%	Avg	10%	20%	Avg
SOC	0.102	0.117	0.110$_{\pm 0.003}$	0.149	0.153	0.151$_{\pm 0.002}$	0.074	0.087	0.081$_{\pm 0.001}$	0.097	0.099	0.098$_{\pm 0.001}$
HEDGE	0.087	0.134	0.111$_{\pm 0.011}$	0.084	0.194	0.139$_{\pm 0.009}$	0.068	0.079	0.074$_{\pm 0.004}$	0.095	0.101	0.098$_{\pm 0.008}$
HE$_{LIME}$	0.075	0.195	0.135$_{\pm 0.005}$	0.076	0.193	0.135$_{\pm 0.009}$	0.063	0.072	0.068$_{\pm 0.003}$	0.059	0.066	0.063$_{\pm 0.007}$
HE$_{LOO}$	0.062	0.117	0.090$_{\pm 0.004}$	0.061	0.119	0.090$_{\pm 0.004}$	0.081	0.092	0.087$_{\pm 0.001}$	0.075	0.086	0.081$_{\pm 0.005}$
BS	0.109	0.121	0.116$_{\pm 0.013}$	0.103	0.185	0.144$_{\pm 0.009}$	0.099	0.104	0.102$_{\pm 0.003}$	0.097	0.105	0.101$_{\pm 0.005}$
ASIV	0.101	0.113	0.107$_{\pm 0.005}$	0.098	0.181	0.140$_{\pm 0.008}$	0.093	0.106	0.199$_{\pm 0.006}$	0.092	0.113	0.103$_{\pm 0.003}$
PE	**0.304**	**0.352**	**0.328**$_{\pm 0.011}$	**0.364**	**0.313**	**0.339**$_{\pm 0.003}$	**0.214**	**0.220**	**0.217**$_{\pm 0.007}$	**0.183**	**0.174**	**0.179**$_{\pm 0.004}$

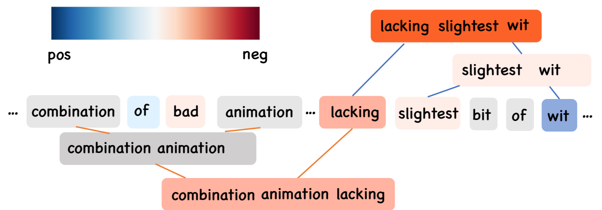
Table 2: AOPC comparison results of PE with baselines on the Rotten Tomatoes and TREC dataset.

Datasets	Yelp				
	AOPC$_{del}$		AOPC$_{pad}$		$\bar{t}$
Methods	10%	20%	10%	20%	
HEDGE	0.077	0.084	0.074	0.089	70.312$_{\pm 0.074}$
HE$_{LIME}$	0.056	0.075	0.065	0.076	20.383$_{\pm 0.054}$
HE$_{LOO}$	0.040	0.071	0.059	0.064	16.201$_{\pm 0.079}$
PE	**0.110**	**0.138**	**0.112**	**0.143**	**2.230**$_{\pm 0.042}$

Table 3: AOPC and time efficiency comparision results of PE and baselines on the Yelp dataset. $\bar{t}$ denotes the average time of building HA tree per input in seconds.



(a) A positive example "*My big fat greek wedding uses stereotypes in a delightful blend of sweet romance and lovingly dished out humor.*"



(b) A negative example "*Just another combination of bad animation and mindless violence lacking the slightest bit of wit or charm.*"

Figure 5: PE,HE$_{LOO}$ for BERT on two examples from the Rotten Tomatoes dataset. The subtree in the upper right corner is generated by PE and the lower is produced by HE$_{LOO}$.

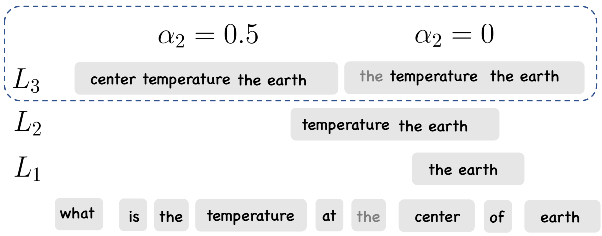
chical explanations, we illustrate with two examples from the TREC dataset. As shown in Figure 6(a), when $\alpha_2 = 0.5$, at the level L_3, PE combines *center*, *temperature*, *the*, *earth* together. However, when $\alpha_2 = 0$, PE combines *the*, *temperature*, *the*, *earth* together. In the dependency parse tree of the sentence *what is the temperature of the center of the earth*, *the* distance to root is greater than *center*. This indicates that incorporating syntactic information is meaningful for constructing convincing hierarchical explanations.

6 Analysis of Time Complexity

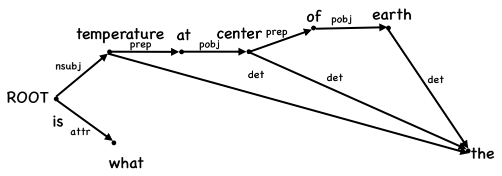
In this section, we delve into the time complexity associated with HA methods, which can be divided into two parts: the complexity of generating attribution scores, denoted as $\mathbf{O}_{attr}$, and the complexity of generating the hierarchy from the scores, denoted as $\mathbf{O}_{hierarchy}$. As shown in Table 4, we elaborate on the time complexity of various methods. For score computation, HEDGE utilizes the Monte Carlo sampling algorithm, with the number of samples denoted by M_1, leading to a time complexity of $O(nM_1)$. HE$_{LOO}$ uses the LOO algorithm (Lipton, 2018), with a time complexity of $O(n^2M_1)$, where M_2 is the maximum number of iterations of the LOO algorithm. HE$_{LIME}$ method employs the LIME algorithm, with ridge regression solving complexity of $O(n^3M_2)$, and M_2 is the number of sampled instances. The time complexity of PE for solving scores is $O(n^2)$.

Methods	$\mathbf{O}_{attr}$	$\mathbf{O}_{hierarchy}$
HEDGE (2020)	$O(nM_1)$	$O(n^3)$
HE$_{LOO}$ (2023)	$O(n^2M_2)$	$O(n^3)$
HE$_{LIME}$ (2023)	$O(n^3M_3)$	$O(n^3)$
PE (ours)	$O(n^2)$	$O(n^2logn)$

Table 4: Comparison results of HA methods about capturing non-contiguous interactions and their time complexity. The relationship between the number of samples in the table and the value of n is: $M_1 \gg M_2 > M_3 \gg n$.



(a) An example "*What is the temperature at the center of the earth?*", which the predicted label is numeric value.



(b) A dependency parsing tree generated by Spacy (Honnibal and Montani, 2017).

Figure 6: PE for BERT on the example from the TREC dataset. The cluster on the left side of the third level L_3 is the results for $\alpha_2 = 0.5$, and the right side is the result for $\alpha_2 = 0$.

7 Conclusion

In this paper, we introduce PE, a computationally efficient method employing hyperbolic geometry for modeling feature interactions. More concretely, we use two hyperbolic projection matrices to embed the semantic and syntax information and devise a simple strategy to estimate the contributions of feature groups. Finally we design an algorithm to decode the hierarchical tree in an $O(n^2 logn)$ time complexity. Based on the experimental results of three typical text classification datasets, we demonstrate the effectiveness of our method.

8 Limitations

The limitations of our work include: 1) Although our method boasts low time complexity, the use of the probing method to train additional model parameters, including two Poincare projection matrices, somewhat limits the generalizability of our approach. 2) In our experiments, we decompose the weights of the edges of the HA tree according to Equation 17. Whether there exists a optimal decomposition formula remains for future investigation.

References

Samira Abnar and Willem Zuidema. 2020. Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Sanjoy Dasgupta. 2016. A cost function for similarity-based hierarchical clustering. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*.

Oğuzhan Demirel. 2013. A characterization of möbius transformations by use of hyperbolic triangles. *Journal of Mathematical Analysis and Applications*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL2018*.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Kedar Dhamdhere, Ashish Agarwal, and Mukund Sundararajan. 2020. The shapley taylor interaction index. In *Proceedings of the 37th International Conference on Machine Learning*.

Joseph Enguehard. 2023. Sequential integrated gradients: a simple but effective method for explaining language models. In *Findings of the Association for Computational Linguistics: ACL 2023*.

James Enouen and Yan Liu. 2022. Sparse interaction additive networks via feature interaction detection and sparse selection. In *Advances in Neural Information Processing Systems*.

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Ronald L Graham and Pavol Hell. 1985. On the history of the minimum spanning tree problem. *Annals of the History of Computing*.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. 2020. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. In *International Conference on Learning Representations*.

Yiming Ju, Yuanzhe Zhang, Kang Liu, and Jun Zhao. 2023. A hierarchical explanation generation method based on feature interaction detection. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Max Kochurov, Rasul Karimov, and Serge Kozlukov. 2017. Geoopt: Riemannian optimization in pytorch. In *International Conference on Machine Learning, GRLB Workshop*.

Enja Kokalj, Blaž Škrlj, Nada Lavrač, Senja Pollak, and Marko Robnik-Šikonja. 2021. BERT meets shapley: Extending SHAP explanations to transformer-based classifiers. In *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*.

Xiaolei Lu, Jianghong Ma, and Haode Zhang. 2023. Asymmetric feature interaction for interpreting model predictions. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Dina Mardaoui and Damien Garreau. 2021. An analysis of lime for text data. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*.

Aria Masoomi, Davin Hill, Zhonghui Xu, Craig P Hersh, Edwin K. Silverman, Peter J. Castaldi, Stratis Ioannidis, and Jennifer Dy. 2022. Explanations of black-box models based on directional feature interactions. In *International Conference on Learning Representations*.

Vivek Miglani, Narine Kokhlikyan, Bilal Alsallakh, Miguel Martin, and Orion Reblitz-Richardson. 2020. Investigating saturation effects in integrated gradients. *Human Interpretability Workshop at ICML.*

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. 2022. Sampling permutations for shapley value estimation. *J. Mach. Learn. Res.*

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)*.

Guillermo Owen. 2013. *Game theory*. Emerald Group Publishing.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*.

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. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Soumya Sanyal and Xiang Ren. 2021. Discretized integrated gradients for explaining language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Lloyd S Shapley et al. 1953. A value for n-person games.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017a. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2017b. Not just a black box: Learning important features through propagating activation differences.

Chandan Singh, W. James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*.

Jake Snell, Kevin Swersky, and RichardS. Zemel. 2017. Prototypical networks for few-shot learning. In *Neural Information Processing Systems,Neural Information Processing Systems*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*.

Michael Tsang, Sirisha Rambhatla, and Yan Liu. 2020. How does this interaction affect me? interpretable attribution for feature interactions. In *Advances in Neural Information Processing Systems*.

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 *NIPS2017*.

Dingkang Wang and Yusu Wang. 2018. An improved cost function for hierarchical cluster trees.

Zhang Xilong, Liu Ruochen, Liu Jin, and Liang Xuefeng. 2023. Interpreting positional information in perspective of word order. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Die Zhang, Hao Zhang, Huilin Zhou, Xiaoyi Bao, Da Huo, Ruizhao Chen, Xu Cheng, Mengyue Wu, and Quanshi Zhang. 2021a. Building interpretable interaction trees for deep nlp models.

Die Zhang, Hao Zhang, Huilin Zhou, Xiaoyi Bao, Da Huo, Ruizhao Chen, Xu Cheng, Mengyue Wu, and Quanshi Zhang. 2021b. Building interpretable interaction trees for deep nlp models. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*.

A Proof

First, we prove that the conclusion holds for $n = 3$, and we generalize to the case of $n > 3$ using induction.

Notation Due to the specificity of the binary tree we are solving for, a unique candidate tree can correspond to a node permutation π. For a tree with n leaves, we define π_n as the corresponding permutation.

We denote the constructed permutation π_n^* and prefix permutation π_m^* in Algorithm 1.

Base Case We here start the discussion from the left case in Figure 10. The cost can be expanded into:

$$
\begin{aligned}
C_D(\pi_3^*; e) &= \sum_{ijk}(e_{ik} + e_{jk}) + 2\sum_{ij} e_{ij} \\
&= \sum_{ijk} 2e_{ij} + e_{ik} + e_{jk}
\end{aligned}
\tag{20}
$$

Notice that e_{ij} is smallest among e_{ij}, e_{ik}, e_{jk} and among $\{i, j \mid k\}$, $\{i, k \mid j\}$, $\{j, k \mid i\}$, only one will hold true. We can conclude that π_3^* is the solution that minimizes the cost.

Induction Step We assume that the tree corresponding to the permutation π_m has the smallest cost. To prove that π_{m+1} is also the smallest. We use a proof by contradiction to demonstrate that π_{m+1} corresponds to the tree with the smallest cost. We define the tree's level as $L_1, \cdots, L_{n-1}$ in Figure 10. Firstly, we introduce the following lemma:

Lemma We denote the γ-th step permutation produced in Algorithm 1 as π_γ^*, and its corresponding tree cost as $C(\pi_\gamma^*)$. Now, if we swap the nodes at level L_s and L_t, $s < t$, and the resulting sequence $\pi_\gamma^{*\prime}$, then $C(\pi_\gamma^{*\prime}) > C(\pi_\gamma^*)$.

Proof. We consider the cost after the swap as three parts: the triples that do not include s and t, the part of the triples that include s and the part that include t, denoted as C_1,C_2 and C_3. For ease of proof, we denote the sequence to the left of s as $A = \pi^*_{\gamma,1:s-1}$, and the sequence between s and t as $B = \pi^*_{\gamma,s+1:t-1}$. Obviously C_1 remains unchanged, as for C_2, before and after the swap:

$$
C_2 = \sum_{i,j\in A,s} e(\cdot) + \sum_{i\in A,s,j\in B} e(\cdot) + \sum_{s,i,j\in B} e(\cdot), \tag{21}
$$

$$
C_2' = \sum_{i,j\in A,s} e(\cdot) + \sum_{i\in A,j\in B,s} e(\cdot) + \sum_{i,j\in B,s} e(\cdot) \tag{22}
$$

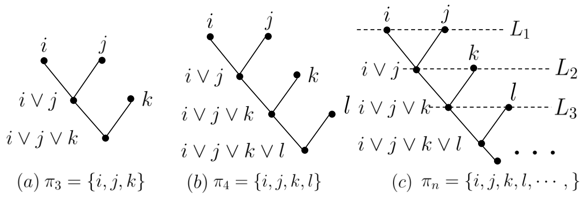


Figure 7: Examples for π_3, π_4 and π_n.

By subtracting, we obtain:

$$
\begin{aligned}
C_2' - C_2 = (&\sum_{i\in A,j\in B,s} e(\cdot) - \sum_{i\in A,s,j\in B} e(\cdot)) + \\
&(\sum_{i,j\in B,s} e(\cdot) - \sum_{s,i,j\in B} e(\cdot)) \geq 0.
\end{aligned}
\tag{23}
$$

Similarly we obtain:

$$
\begin{aligned}
C_3' - C_3 = (&\sum_{i\in A,t,j\in B} e(\cdot) - \sum_{i\in A,j\in B,t} e(\cdot)) + \\
&(\sum_{t,i,j\in B} e(\cdot) - \sum_{i,j\in B,t} e(\cdot)) \geq 0.
\end{aligned}
\tag{24}
$$

Now we prove that π_{m+1} is smallest. If π_{m+1} is not the smallest, then the node at the last level can be the smallest by swapping with a previous node. There are two cases: when the swapped node is from the first level (e.g. j), in this case, the difference in cost before and after the swap becomes:

$$
\begin{aligned}
\Delta C = (&\sum_{i\in C,m+1,j\in D} e(\cdot) - \sum_{i\in C,j\in D,m+1} e(\cdot)) + \\
&(\sum_{t,i,j\in D} e(\cdot) - \sum_{i,j\in D,t} e(\cdot)) \geq 0,
\end{aligned}
\tag{25}
$$

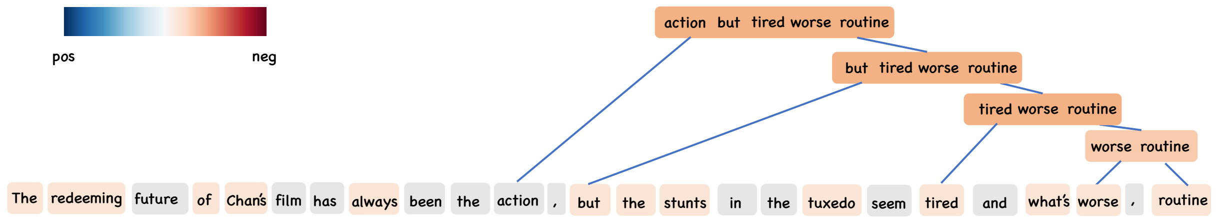
where $C = \pi^*_{m+1,1}$, $D = \pi^*_{m+1,3:m}$. Similarly, when the swapped node is located in other levels, the cost after the swap will not decrease. This means that in $C(\pi_{m+1})$ cannot be smaller through swapping other leaves from different levels, thus π_{m+1} is smallest.

The primary difference is that the edge weights in our graph (Graham and Hell, 1985) are not all known in advance but are dynamically generated.

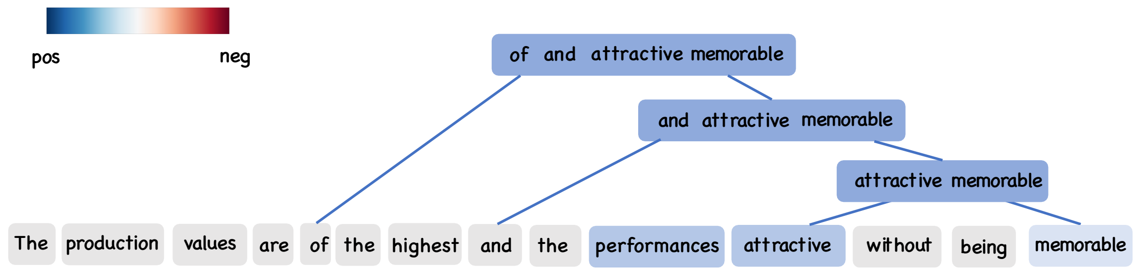
B Visualization

C Implementation Details

In this work, all language models are implemented by Transformers. All our experiments are per-

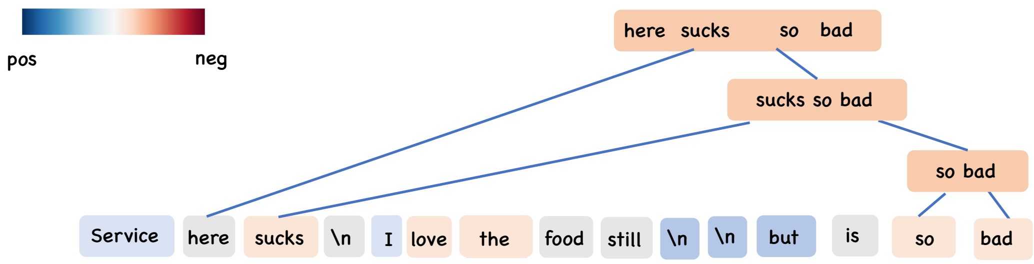


(a) A negative example "*The redeeming feature of Chan's films has always been the action, but the stunts in the tuxedo seem tired and what's worse, routine.*"



(b) A positive example "*The production values are of the highest and the performances attractive without being memorable.*"

Figure 8: PE for BERT on two examples from the Rotten Tomatoes dataset.



(a) A negative example "*Service here sucks \n I love the food still \n\n but the service is so bad.*"

(b) A positive example "*Flavors are great but every time I come this location it is disgusting machines are dirty.*"

Figure 9: PE for BERT on two examples from Yelp dataset.

formed on one A800. The results are reported with 5 random seeds.

For fine tuning the projection matrix $\boldsymbol{P}_c$, we iterate 5 epochs using RiemanianAdam optimizer and learning rate is initialized as 1e-3, the batch size is 32. For fine tuning the projection matrix $\boldsymbol{P}_s$, we use the Penn Treebank dataset we iterate 40 epochs using Adam optimizer and learning rate is initialized as 1e-3. We set d_{out} as 64. We use grid search to search $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$.

D HA Example

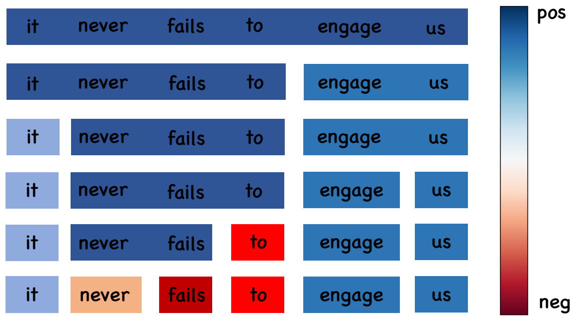


Figure 10: A hierarchy example from HEDGE (Chen et al., 2020). The background color of the words and phrases represents emotional polarity, with cool colors indicating positive and warm colors indicating negative.

E Lime Explanation

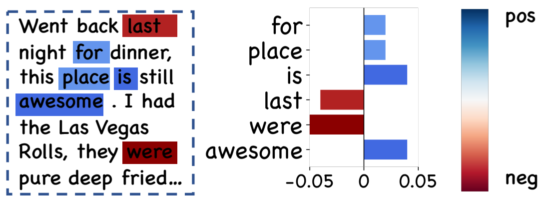


Figure 11: A LIME explanation example from a random forest classifier. It can be observed that two stop words (i.e."is" and "were") are identified as positive and negative emotional polarities, respectively.