BUILDING HIERARCHICAL INTERPRETATIONS IN NATURAL LANGUAGE VIA FEATURE INTERACTION DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The interpretability of neural networks has become crucial for their applications in real-world with respect to reliability and trustworthiness. Existing explanation generation methods usually provide important features by scoring their individual contributions to the model prediction and ignore the interactions between features, which eventually provide a bag-of-words representation as an explanation. In natural language processing, this type of explanations is challenging for human users to understand the meaning of an explanation and draw the connection between explanation and model prediction, especially for long texts. In this work, we focus on detecting the interactions between features and propose a novel approach to build a hierarchy of explanations based on feature interactions. The proposed method is evaluated with three neural classifiers, LSTM, CNN, and BERT, on two benchmark text classification datasets. The generated explanations are assessed by both automatic evaluation measurements and human evaluators. Experiments show the effectiveness of the proposed method in providing explanations that are both faithful to models, and understandable to humans.

1 INTRODUCTION

Deep neural networks have become a significant component in natural language processing (NLP), achieving state-of-the-art performance in various NLP tasks, such as text classification (Kim, 2014), question answering (Rajpurkar et al., 2016), and machine translation (Bahdanau et al., 2014). However, the lack of understanding on their decision making leads them to be characterized as black-box models and increases the risk of applying them in real-world applications (Lipton, 2016). Producing interpretable decisions has been a critical factor on whether people will trust and use the neural network models (Ribeiro et al., 2016).

Most of existing work on local explanation generation for NLP focuses on producing word-level explanations (Ribeiro et al., 2016; Lei et al., 2016; Plumb et al., 2018), where a local explanation consists of a set of words extracted from the original text. Figure 1 presents an example sentence with its sentiment prediction and corresponding word-level explanation generated by LIME (Ribeiro et al., 2016). Although the LIME explanation captures a negative sentiment word `waste`, it presents the explanation in a bag-of-words format. Without resorting to the original text, it is difficult for us to understand the contribution of word `a` and `of`, as both of them have no sentiment polarity. The situation will become even more serious when this type of explanations are extracted from longer texts.

In this work, we present a novel method to construct hierarchical explanations of a model prediction by capturing the interaction between features. Ultimately, our method is able to produce a hierarchical structure as illustrated in Figure 1. Produced by the proposed method, this example provides a comprehensive picture of how different granularity of features interacting with each other for model prediction. With the hierarchical structure, this example tells us how the words and phrases are combined and what are the contributions of words and phrases to the final prediction. For example, the contribution of the phrase `of good` is dominated by the word `waste`, which eventually leads to the right prediction.

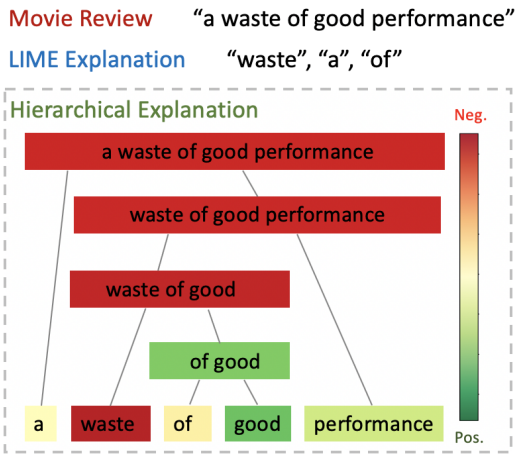


Figure 1: A NEGATIVE movie review `a waste of good performance` with a LIME explanation and a hierarchical explanation, where the color of each block represents the importance of the corresponding word/phrase with respect to the model prediction.

To capture feature interactions, we adopt the interacted Shapley value (Lundberg et al., 2018), an extension of Shapley value (Shapley, 1953) from cooperative game theory, to measure the interactions between features. Based on the interaction scores, we propose a top-down method, called INTERSHAPLEY, to segment a text recursively into phrases and then words eventually. The proposed method is evaluated on text classification tasks with three typical neural network models: long short term memory networks (Hochreiter & Schmidhuber, 1997, LSTM) and convolutional neural networks (Kim, 2014, CNN), and a state-of-the-art model BERT (Devlin et al., 2018) on some benchmark datasets. The comparison of our method is against several competitive baselines from prior work on explanation generation, including Leave-one-out (Li et al., 2016), Contextual Decomposition (CD) (Murdoch et al., 2018) and its hierarchical extension (ACD) (Singh et al., 2019), L- and C-Shapley (Chen et al., 2018), and LIME (Ribeiro et al., 2016).

Our contribution of this work is three-fold: (1) we propose an effective method to calculate feature importance and extend the Shapley value to measure feature interactions; (2) we design a top-down segmentation algorithm to build hierarchical interpretations based on feature interactions; (3) we compare the proposed method with several competitive baselines via both automatic and human evaluations, and show the INTERSHAPLEY method outperforms the existing methods on both word- and phrase-level explanations.

2 RELATED WORK

Over the past years, many different research directions have been explored to build interpretable models or improve the interpretability of neural networks: (1) tracking the inner-workings of neural networks to understand their decision-making, such as contextual decomposition (CD) (Murdoch et al., 2018; Godin et al., 2018); (2) decoding interpretable knowledge (e.g., contextual information, latent representations) from networks to explain model predictions, such as gradient-based interpretation methods (Hechtlinger, 2016; Sundararajan et al., 2017) and attention-based methods (Ghaeini et al., 2018; Xie et al., 2017); (3) modifying neural network architectures to make their output more explainable, such as retreating to simpler models (Alvarez-Melis & Jaakkola, 2018) or incorporating strong regularizers (Yuan et al., 2019); and (4) developing explainable model-agnostic methods to learn the behaviors of neural network models and explain them, such as LIME (Ribeiro et al., 2016) and KernelSHAP (Lundberg & Lee, 2017). The first three directions have limited capacity in real-world applications, as they require deep understanding of neural network architectures (Murdoch et al., 2018) or only work with specific models (Alvarez-Melis & Jaakkola, 2018). On the other hand, model-agnostic methods (Ribeiro et al., 2016; Lundberg & Lee, 2017) generate explanations solely based on model predictions and can be used even without much expertise on machine learning and deep learning. In this work, we also focus on model-agnostic interpretations.

Model-agnostic interpretations. Model-agnostic methods are applicable for any black-box models, generating explanations solely based on model predictions. Li et al. (2016) proposed a method, called Leave-one-out, to probe the black-box model by erasing a given word and observing the change in the probability on the predicted class. Another work, LIME (Ribeiro et al., 2016), estimated individual word contributions locally by linear approximation based on perturbed examples. A line of most related works are Shapley-based methods, such as SHAP (Lundberg & Lee, 2017), L-Shapley and C-Shapley (Chen et al., 2018), where the variants of Shapley value (Shapley, 1953) are used to evaluate feature importance. They mainly focus on the challenge of reducing computational complexity of Shapley value by approximations (Kononenko et al., 2010; Datta et al., 2016).

Hierarchical interpretations. Previous work on interpreting neural networks mainly focuses on word-level interpretations, where the top-ranked words are selected based on their contributions to the prediction. There are a few works on building hierarchical interpretations. For example, agglomerative contextual decomposition (ACD) (Singh et al., 2019) uses the word-level CD scores as the joining metric determining which features are clustered together. Unlike our method, this work is model-dependent, and proposes a bottom-up clustering method to aggregate features with respect to their interactions. Besides, Lundberg et al. (2018) extented SHAP values to SHAP interaction values to calculate the interactions between features along a given tree structure. Similarly, Chen & Jordan (2019) suggests to utilize a linguistic tree structure to capture the contributions beyond individual features for text classification. The difference with our work is that both methods from (Lundberg et al., 2018; Chen & Jordan, 2019) require hierarchical structures given, while our method constructs the structures without resorting external resources.

3 METHOD

In this section, we first introduce a new definition of feature importance score in subsection 3.1, which forms the basis of the feature interaction detection method discussed in subsection 3.2. Then, subsection 3.3 presents a novel method of building a hierarchy of explanations by recursively optimizing the feature interaction score on a given text.

3.1 FEATURE IMPORTANCE SCORE

In this paper, the feature importance score is usually understood as the contribution of a subset of features to each model prediction. It is essential in building a hierarchy of explanations where interaction scores built on top of it are extensively used. Considering a set of features $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, and a subset of it written as $\boldsymbol{x}_S$, the feature importance score of $\boldsymbol{x}_S$ is defined as follows,

$$f(\boldsymbol{x}_S) = g_{\hat{y}}(\boldsymbol{x}_S) - \max_{y' \neq y, y' \in \mathcal{Y}} g_{y'}(\boldsymbol{x}_S), \tag{1}$$

where $g_y(\boldsymbol{x})$ is the model output with respect to label y, $\mathcal{Y}$ is the possible label set, and $\hat{y}$ is the predicted label obtained by the model, i.e., $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} g_y(\boldsymbol{x})$; S is an integer set used to select features with subscript values matching those in S, i.e. $\boldsymbol{x}_S = \{\boldsymbol{x}_i | i \in S\}$. More concretely, in this work, $\boldsymbol{x}$ consists of a sequence of words (sentence or text), $\boldsymbol{x}_i$ is the i-th word in $\boldsymbol{x}$.

The feature importance score defined in Equation 1 measures the difference between the probability on the predicted class and the probability of the second most probable class. It measures how far the prediction is to the prediction boundary, hence the confidence of classifying $\boldsymbol{x}_S$ into the predicted label $\hat{y}$. Particularly in text classification, it can be interpreted as the contribution to a specific class $\hat{y}$. For a given text $\boldsymbol{x}$, $\boldsymbol{x}_S$ is usually considered as a consecutive subsequence of words for interpretable explanations. To compute $g_{\hat{y}(\boldsymbol{x}_S)}$ in Equation 1, it is a common practice that words outside S are replaced with the special token $\langle \texttt{pad} \rangle$ (Chen et al., 2018; Shrikumar et al., 2017; Lundberg & Lee, 2017) in order to keep the same length between $\boldsymbol{x}$ and $\boldsymbol{x}_S$. The effectiveness of Equation 1 as a feature importance score is shown in subsection 4.1.

3.2 FEATURE INTERACTION SCORE

By considering the prediction process as a coalition game (Lundberg et al., 2018), where each player (feature) contributes fairly, the feature interactions can be measured consistently, meaning the computed scores are in line with human intuition and are robust when model changes (Lundberg et al.,

2018). Such scores often have the form of Shapley values. Motivated by the Shapley interaction score (Fujimoto et al., 2006; Lundberg et al., 2018), and combined with our model-agnostic feature importance score in subsection 3.1, the feature interaction between the p-th and q-th features in $\boldsymbol{z}$ is computed as follows,

$$\psi_(p, q, \boldsymbol{z}) = \sum_{S \subseteq \mathcal{N} \setminus \{p,q\}} \frac{|S|!(n - |S| - 1)!}{2(n-1)!} \gamma_{p,q}(\boldsymbol{z}, S), \tag{2}$$

where $\boldsymbol{z}$ is a set of features, and a feature in $\boldsymbol{z}$ can be a word or a set of words; $\mathcal{N}$ is the index set of features from $\boldsymbol{z}$; $|S|$ is the size of the set S. Additionally, $\gamma_{p,q}(\boldsymbol{z}, S)$ is the interaction between feature $\boldsymbol{z}_p$ and $\boldsymbol{z}_q$ considering the contributions from other features in $\boldsymbol{z}_S$, which is defined as

$$\gamma_{p,q}(\boldsymbol{z}, S) = f(\boldsymbol{z}_{S \cup \{p,q\}}) - f(\boldsymbol{z}_{S \cup \{p\}}) - f(\boldsymbol{z}_{S \cup \{q\}}) + f(\boldsymbol{z}_S). \tag{3}$$

The total interaction between $\boldsymbol{z}_p$ and $\boldsymbol{z}_q$ is the sum of $\psi_(p, q, \boldsymbol{z})$ and $\psi_(q, p, \boldsymbol{z})$ (Lundberg et al., 2018). Since $\psi_(p, q, \boldsymbol{z})$ is symmetric with respect to p and q, the total interaction between the p-th element and q-th element in $\boldsymbol{z}$ is $\phi(p, q, \boldsymbol{z}) = 2\psi_(p, q, \boldsymbol{z})$.

3.3 Building Hierarchical Interpretation

To build a hierarchical interpretation, we adopt a top-down segmentation approach recursively splitting a feature set into two subsets with the minimum interaction score. In other words, a feature set is partitioned into two subsets with strong intra-interactions and weak inter-interactions in each step. The constructed hierarchy of explanations consists of multiple levels and each level is a set of feature subsets.

Formally, considering the feature set at the ℓ-th level of the hierarchy, $\boldsymbol{z}^\ell = (\boldsymbol{z}_1^\ell, \ldots, \boldsymbol{z}_{n_\ell}^\ell)$, we can obtain various candidate sets $\hat{\boldsymbol{z}}^{\ell+1}$ for the next layer by splitting one element $\boldsymbol{z}_i^\ell, \forall i \in \{1, \ldots, n_\ell\}$ at a time, where n_ℓ is the size of $\boldsymbol{z}^\ell$. For example, one $\hat{\boldsymbol{z}}^{\ell+1}$ could be $\hat{\boldsymbol{z}}^{\ell+1} = (\boldsymbol{z}_1^\ell, \ldots, \hat{\boldsymbol{z}}_{i_1}^{\ell+1}, \hat{\boldsymbol{z}}_{i_2}^{\ell+1}, \ldots, \boldsymbol{z}_{n_\ell}^\ell)$, where $\boldsymbol{z}_i^\ell$ is partitioned into $\hat{\boldsymbol{z}}_{i_1}^{\ell+1}$ and $\hat{\boldsymbol{z}}_{i_2}^{\ell+1}$, and the size of it is $n_\ell + 1$. Of course, such partition is not unique and the number of potential partitions depends on the size of $\boldsymbol{z}^\ell$. The best feature set at level $\ell + 1$ satisfies

$$\boldsymbol{z}^{\ell+1} = \operatorname*{argmin}_{\hat{\boldsymbol{z}}^{\ell+1}} \phi(i_1, i_2, \hat{\boldsymbol{z}}^{\ell+1}), \tag{4}$$

$$\text{s.t. } \hat{\boldsymbol{z}}_{i_1}^{\ell+1} \cup \hat{\boldsymbol{z}}_{i_2}^{\ell+1} = \boldsymbol{z}_i^\ell. \tag{5}$$

Equation 4 finds the best partition between $\hat{\boldsymbol{z}}_{i_1}^{\ell+1}$ and $\hat{\boldsymbol{z}}_{i_2}^{\ell+1}$ whose interaction score is minimum. At level-zero, there is only one feature set with all input words included, $\boldsymbol{z}^0 = \boldsymbol{x}$. By splitting $\boldsymbol{z}^0$ into two subsets $\boldsymbol{z}_1^1$ and $\boldsymbol{z}_2^1$, we can obtain the feature set at level-one, $\boldsymbol{z}^1 = [\boldsymbol{z}_1^1, \boldsymbol{z}_2^1]$. This process stops when all elements in a layer are word-level features. The top-down segmentation method is named as INTERSHAPLEY.

Note that in generally, the solution to Equation 4 and Equation 5 has exponential computational complexity. In NLP, features (i.e. words) are usually arranged in order, meaning that i_1 and i_2 in Equation 5 are adjacent. With the aid of Shapley value approximation (Chen et al., 2018), where only k neighbors are involved in computation, the computational complexity measured in number of model evaluations will be $\mathcal{O}(4^k n \log n)$. So the method scales well with input text size n. Also note that opposite to the top-down method, we can also combine words with the strongest interaction into phrases, then phrases into larger phrases, resulting in the bottom-up method. In some cases, the bottom-up method cannot capture the phrase having opposite sentiment polarity until the last step. Detailed comparisons will be shown in Appendix A.

4 Experiments

The proposed method is evaluated on text classification tasks with three typical neural network models, a long short-term memories (LSTM) (Hochreiter & Schmidhuber, 1997), a convolutional neural networks (CNN) (Kim, 2014), and BERT (Devlin et al., 2018), on SST (Socher et al., 2013) and IMDB (Maas et al., 2011) datasets. The detailed experimental setup is in Appendix B. We

employ both automatic and human evaluations on word-level and phrase-level explanations, to show the ability of INTERSHAPLEY in interpreting neural network models.

Table 1 shows the best performance of the models on both datasets in our experiments.

	Dataset	
Models	SST	IMDB
LSTM	0.842	0.870
CNN	0.850	0.901
BERT	0.924	0.930

Table 1: The classification accuracy of different models on the SST and IMDB datasets.

4.1 WORD-LEVEL EVALUATION

We adopt two evaluation metrics from prior work on evaluating word-level explanations: the area over the perturbation curve (AOPC) (Nguyen, 2018; Samek et al., 2016) and log-odds scores (Chen et al., 2018). These two metrics measure local fidelity by deleting or masking words in the decreasing order of their importance scores and comparing the probability change on the predicted label.

AOPC. Word-level explanations provide a sequence of words according to their importance scores. AOPC is calculated as the average change in prediction probability on the predicted class over all of the test data by deleting top $k\%$ words. Higher AOPCs are better, which means that the deleted words are important for model prediction.

$$\text{AOPC}(k) = \frac{1}{N}\sum_{i=1}^{N}\{p(\hat{y} \mid \boldsymbol{x}_i) - p(\hat{y} \mid \tilde{\boldsymbol{x}}_i^{(k)})\}, \tag{6}$$

where $\hat{y}$ is the predicted label, N is the number of samples, $p(\hat{y} \mid \cdot)$ is the probability on the predicted class, and $\tilde{\boldsymbol{x}}_i^{(k)}$ is constructed by dropping the top $k\%$ word features from $\boldsymbol{x}_i$.

Log-odds. The log-odds score is calculated by averaging the difference of negative logarithmic probabilities on the predicted class over all of the test data before and after masking the top $r\%$ features with zero paddings. Under this metric, lower log-odds scores are better.

$$\text{Log-odds}(r) = \frac{1}{N}\sum_{i=1}^{N}\log\frac{p(\hat{y} \mid \tilde{\boldsymbol{x}}_i^{(r)})}{p(\hat{y} \mid \boldsymbol{x}_i)}, \tag{7}$$

The notations are the same as in Equation 6 with the only difference that $\tilde{\boldsymbol{x}}_i^{(r)}$ is constructed by replacing the top $r\%$ word features with the special token $\langle \texttt{pad} \rangle$ in $\boldsymbol{x}_i$.

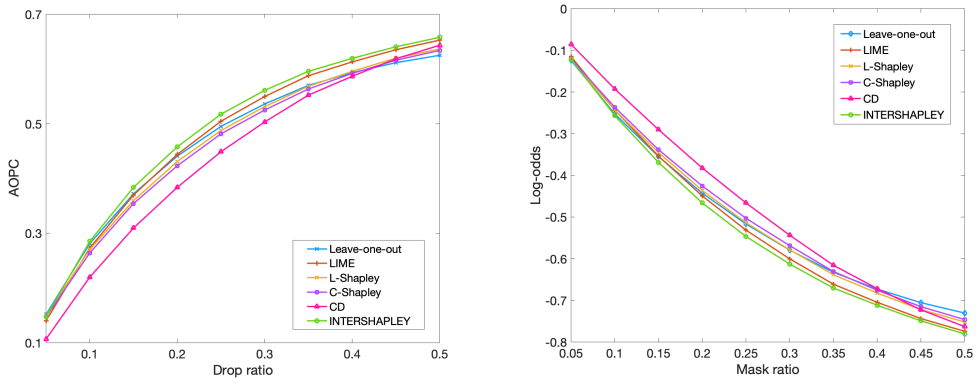
4.1.1 RESULTS

Figure 2 shows the results of AOPCs and log-odds scores of LSTM model on the SST dataset. INTERSHAPLEY achieves the best performance on both evaluation metrics. Together with the other results in Appendix C, this evaluation demonstrates the effectiveness of our method in extracting keywords via feature importance scores, which lays a solid foundation for feature interaction detection.

The L- and C-Shapley perform worse on the IMDB dataset than on the SST dataset, because a small window size (2 as recommended) may cause relatively large error in approximating Shapley values for long sentences. The performance of LIME drops significantly on BERT comparing to the other two models, as Figure 9 and 10 show, which suggests that this locally linear approximation may not be capable of deep neural network models. We also observed an interesting phenomenon that the simplest baseline Leave-one-out can achieve relatively good performance, even better than L- and C-Shapley, and CD. And we suspect that is because the criteria of Leave-one-out for picking keywords matches the evaluation metrics.

4.2 PHRASE-LEVEL EVALUATION

This section presents some evaluation on phrase-level explanations extracted with the proposed methods. Take the hierarchy in Figure 1 as an example, the set of phrases extracted from this hierarchical structure include {`waste of good performance`, `waste of good`, `of good`}.



(a) AOPCs of LSTM on the SST dataset.

(b) Log-odds of LSTM on the SST dataset.

Figure 2: The AOPC and log-odds scores of LSTM on the SST dataset.

On phrase level, we can use the same way as in Equation 1 to compute importance scores. The quantitative evaluation in section 4.2.1 shows our method successfully captures the *strongest* interactions using a contrastive comparison, and the qualitative analysis in section 4.2.2 illustrates how different levels of features interact with each other while contributing to final predictions.

4.2.1 Quantitative Evaluation

We evaluate the interaction effect of an important phrase by extracting the words within this phrase and then randomly insert them back to the original sentence. For example, in a sentence with n words $\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n)$, if $(x_i, \ldots, x_j)$ is the most important phrase, then for every word in $(x_i, \ldots, x_j)$, we randomly pick a position in $(x_1, \ldots, x_{i-1}, x_{j+1}, \ldots, x_n)$ and insert that word back. Let $\bar{\boldsymbol{x}}$ be the newly constructed word sequence in this way. The quantitative evaluation is to compare the difference between $p(\hat{y} \mid \boldsymbol{x})$ and $p(\hat{y} \mid \bar{\boldsymbol{x}})$. Intuitively, by breaking the strong interaction within $(x_i, \ldots, x_j)$, it will lead to a significant drop on the probability of predicted label $\hat{y}$. To obtain a robust evaluation, for each sentence $\boldsymbol{x}_i$, we construct K different word sequences $\{\bar{\boldsymbol{x}}_i^{(k)}\}_{k=1}^{K}$ and compute the average as

$$\text{Cohesion-loss} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{k=1}^{K}(p(\hat{y} \mid \boldsymbol{x}_i) - p(\hat{y} \mid \bar{\boldsymbol{x}}_i^{(k)})), \tag{8}$$

where $\bar{\boldsymbol{x}}_i^{(k)}$ is the k^{th} disturbed sample of $\boldsymbol{x}_i$, and $K = 100$ is the number of perturbations for each sentence. Note that when we pick the most important phrase, we consider the one whose length is no more than 5 as the explanation because a very long clause is uninterpretable. Then we calculate the average length of the most important phrases over all test data as the Average Explanation Length (Ave. Exp. length).

Table 2 shows the results of cohesion-loss of INTERSHAPLEY and ACD with different models on the SST and IMDB datasets. For the IMDB dataset, we tested on a subset with 2000 randomly selected samples due to computation costs. INTERSHAPLEY outperforms ACD with higher cohesion-loss and smaller average length of generated explanations on both datasets with LSTM, which indicates that INTERSHAPLEY can capture more important and concise phrases. Comparing the results of INTERSHAPLEY for different models, the cohesion-loss of LSTM model improves on the IMDB dataset, while BERT performs the opposite. We reason that it is the length constraint on selected phrases that limits BERTs performance, since BERT often utilizes a large range of context for predictions.

4.2.2 Qualitative Analysis

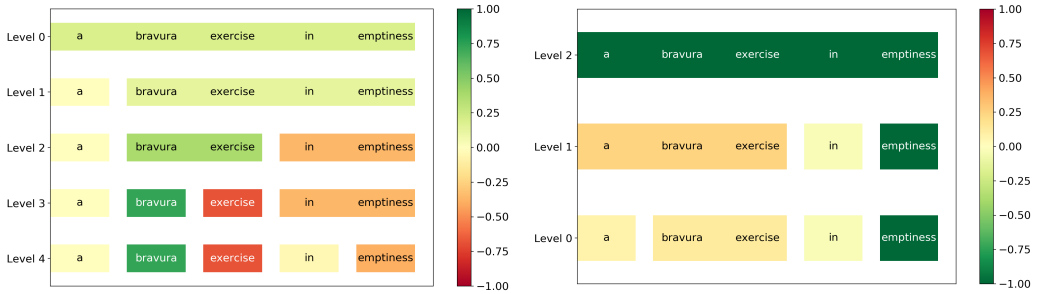
Figure 3 visualizes the hierarchical interpretations of INTERSHAPLEY and ACD for LSTM on a negative movie review from the SST dataset, where red and green colors represent the negative and

Methods	Models	Cohesion-loss		Ave. Exp. length	
		SST	IMDB	SST	IMDB
INTERSHAPLEY	CNN	0.025	0.021	1.63	1.31
	BERT	0.055	0.020	2.7	3.35
	LSTM	0.021	0.039	2.07	2.07
ACD	LSTM	0.007	0.030	3.19	4.83

Table 2: Cohesion-loss and Average Explanation Length (Ave. Exp. length) of INTERSHAPLEY and ACD with different models on SST and IMDB dataset. As ACD is model-dependent, we only compare it on the LSTM model.

positive sentiments respectively. In this case, LSTM makes a wrong prediction as positive. Figure 3(a) shows that INTERSHAPLEY gives correct positive and negative sentiments for `bravura` and `emptiness` respectively, and captures the interaction between them that `bravura exercise` flips the polarity of `in emptiness` to positive, which explains why the model makes the wrong prediction. However, ACD incorrectly marks the two keywords with opposite polarities, and misses the feature interaction, as Figure 3(b) shows.

Figure 4 visualizes INTERSHAPLEY for LSTM and BERT on a positive movie review, on which BERT makes a correct prediction, while LSTM makes a wrong prediction. The comparison between Figure 4(a) and Figure 4(b) shows the difference of feature interactions within the two models and explains why a model makes the correct/wrong prediction. For example, Figure 4(b) illustrates that BERT captures the key phrase `not a bad` in level 1, and thus makes the positive prediction, while LSTM (as shown in 4(a)) misses the interaction between `not` and `bad`, and the phrases `not a` and `bad journey` both push the model making the negative prediction. Due to the page limitation, more examples are presented in Appendix D.

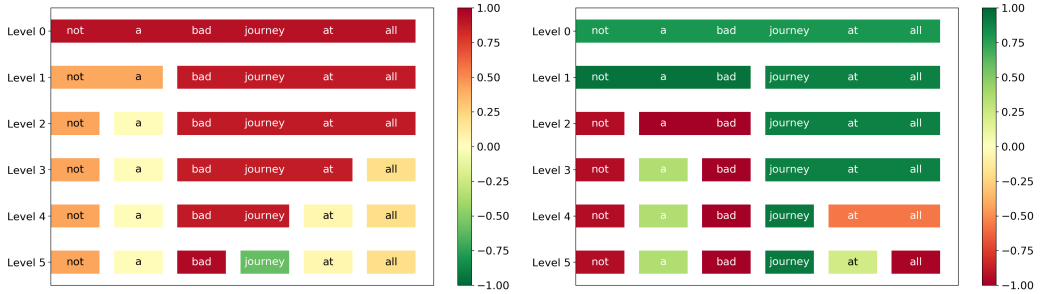


(a) INTERSHAPLEY for LSTM on SST. (b) ACD for LSTM on SST.

Figure 3: INTERSHAPLEY and ACD for LSTM on a negative movie review, where the importance scores of INTERSHAPLEY and CD scores are normalized for comparing. LSTM makes wrong prediction (positive). Red and green colors represent the negative and positive sentiments respectively.

4.3 HUMAN EVALUATION

We employ 10 human annotators to do human evaluation on Amazon Mechanical Turk (AMT). The most important feature (with the highest importance score) is picked from the hierarchy as the explanation, with its length being limited to no more than five words for the ease of human understanding. We evaluate the explanation by asking human annotators to guess the output of a model based on the explanation and the input text (Nguyen, 2018). For each example, human annotators choose a label from "Negative", "Positive", "N/A" based on their understanding of the explanation, where "N/A" means that the explanation may be some irrelevant words/phrase, and annotators cannot figure out the model's prediction. An example interface is shown in Appendix E. We measure the number of human annotations that are coherent with the actual model predictions,



(a) INTERSHAPLEY for LSTM on SST. (b) INTERSHAPLEY for BERT on SST.

Figure 4: INTERSHAPLEY for LSTM and BERT on a positive movie review from the SST dataset, on which BERT makes correct prediction (positive), while LSTM makes wrong prediction (negative). Red and green colors represent the negative and positive sentiments respectively.

and define the *coherence score* as the ratio between the coherent annotations and the total number of examples.

We randomly pick 60 movie reviews from IMDB dataset, half of which are used in one of the following experiments. First, we compare the explanations obtained from INTERSHAPLEY and ACD with LSTM model. For the same model, higher coherence score means that the explanations are more human understandable. Second, we apply INTERSHAPLEY to the three neural models, and compare the corresponding explanations. For the same interpretation method, a more interpretable mode usually can achieve higher coherence score.

4.3.1 RESULTS

Table 3 shows the coherence scores of the two interpretation methods. INTERSHAPLEY outperforms ACD with much higher coherence score, which means that the important features captured by INTERSHAPLEY are highly consistent with human comprehension in explaining model predictions. Table 4 shows the accuracy and coherence scores of different models. INTERSHAPLEY achieves relatively high coherence scores on all of the three neural networks, which validates its ability in interpreting black-box models. Although BERT can achieve higher prediction accuracy than other two models, its coherence score is lower, which illustrates that the interpretability of more complex model is even worse.

Methods	Coherence Score
INTERSHAPLEY	0.88
ACD	0.53

Table 3: Human evaluation of INTERSHAPLEY and ACD with LSTM model on the IMDB dataset.

Models	Accuracy	Coherence scores
LSTM	0. 87	0.90
CNN	0.90	0.91
BERT	0.97	0.83

Table 4: Human evaluation of INTERSHAPLEY with different models on the IMDB dataset.

5 CONCLUSION

In this paper, we proposed an effective method, INTERSHAPLEY, building model-agnostic hierarchical interpretations via detecting feature interactions. In this work, we mainly focus on sentiment classification task. We test INTERSHAPLEY with three different neural network models on two benchmark datasets, and compare it with several competitive baseline methods. The superiority of INTERSHAPLEY is approved by both automatic and human evaluations.

REFERENCES

David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Jianbo Chen and Michael I Jordan. Ls-tree: Model interpretation when the data are linguistic. *arXiv preprint arXiv:1902.04187*, 2019.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data. *arXiv preprint arXiv:1808.02610*, 2018.

Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pp. 598–617. IEEE, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55 (1):72–99, 2006.

Reza Ghaeini, Xiaoli Z Fern, and Prasad Tadepalli. Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*, 2018.

Fréderic Godin, Kris Demuynck, Joni Dambre, Wesley De Neve, and Thomas Demeester. Explaining character-aware neural networks for word-level prediction: Do they discover linguistic rules? *arXiv preprint arXiv:1808.09551*, 2018.

Yotam Hechtlinger. Interpretation of prediction models using the input gradient. *arXiv preprint arXiv:1611.07634*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18, 2010.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 107–117, 2016.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pp. 142–150. Association for Computational Linguistics, 2011.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *arXiv preprint arXiv:1801.05453*, 2018.

Dong Nguyen. 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)*, pp. 1069–1078, 2018.

Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. Model agnostic supervised local explanations. In *Advances in Neural Information Processing Systems*, pp. 2515–2524, 2018.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 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*, pp. 1135–1144. ACM, 2016.

Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28), 1953.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153. JMLR. org, 2017.

Chandan Singh, W. James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=SkEqro0ctQ`.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328. JMLR. org, 2017.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. An interpretable knowledge transfer model for knowledge base completion. *arXiv preprint arXiv:1704.05908*, 2017.

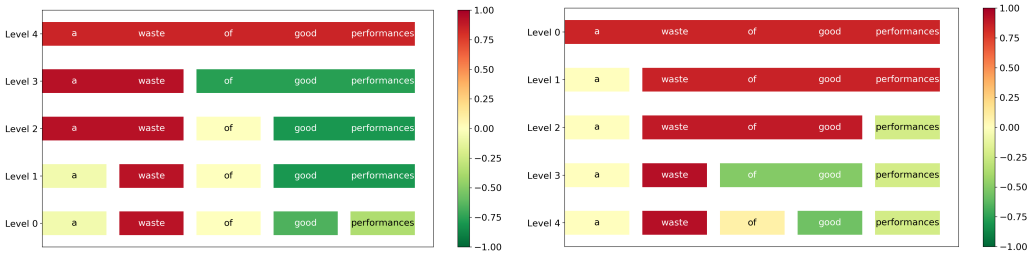
Hao Yuan, Yongjun Chen, Xia Hu, and Shuiwang Ji. Interpreting deep models for text analysis via optimization and regularization methods. In *AAAI*, 2019.

Dataset	Classes	Average length	Train	Dev.	Test
SST-2	2	19	6920	872	1821
IMDB	2	268	22500	2500	25000

Table 5: Summary statistics for the datasets.

A Comparison between Top-down and Bottom-up Approaches

Given the sentence `a waste of good performance` for example, Figure 5 shows the hierarchical interpretations for the LSTM model using the bottom-up and top-down approaches respectively. Figure 5(a) shows that the interaction between `waste` and `good` can not be captured until the last (top) layer, while the important phrase `waste of good` can be extracted in the intermediate layer by top-down algorithm. We can see that `waste` flips the polarity of `of good` to negative, causing the model predicting negative as well.



(a) Bottom-up clustering (b) Top-down segmentation

Figure 5: Hierarchical interpretations for the LSTM model using the bottom-up and top-down approaches respectively. Red and green colors represent the negative and positive sentiments respectively.

B Experimental Setup

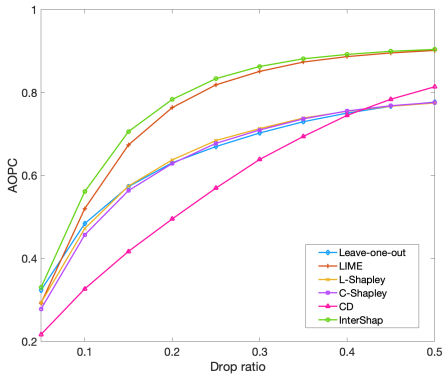
Models. In this work, we use a recurrent neural network (RNN) with uni-directional one-layer LSTM Hochreiter & Schmidhuber (1997), a CNN model with single convolutional layer Kim (2014), and BERT Devlin et al. (2018) which has achieved remarkable performance on a variety of NLP tasks including text classification. The CNN model includes a single convolutional layer with filters of the window sizes ranging from 3 to 5. The LSTM is also single layer with 300 hidden states. Both models are initialized with 300-dimensional pretrained word embeddings Mikolov et al. (2013). We use the pre-trained BERT model[1] with 12 transformer layers, 12 self-attention heads, and the hidden size of 768, and fine-tune it in different downstream tasks to achieve the best performance.

Datasets. We use two benchmark datasets on text classification: the SST-2 corpus (Socher et al., 2013) and the IMDB corpus (Maas et al., 2011). Table 5 shows the summary statistics of the datasets. SST-2 is the binary-class version of the Stanford sentiment treebank and IMDB is a balanced dataset with 25000 training examples and 25000 test examples. We split 90% of the training examples for training, and the rest as the development set.

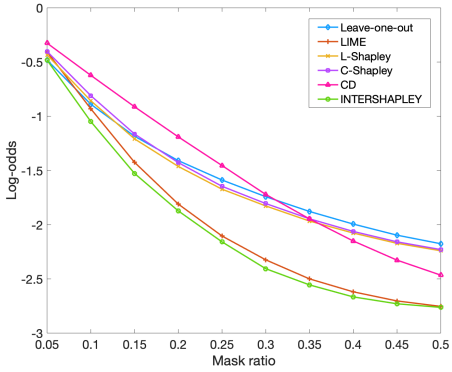
Competitive Baselines. We compare our method with some competitive baselines, including Leave-one-out (Li et al., 2016), CD (Murdoch et al., 2018), L- and C-Shapley (Chen et al., 2018), and LIME (Ribeiro et al., 2016) for word-level explanation generation, and ACD (Singh et al., 2019) for phrase-level explanation generation.

[1] `https://github.com/huggingface/pytorch-transformers`

C OTHER RESULTS OF AOPCS AND LOG-ODDS

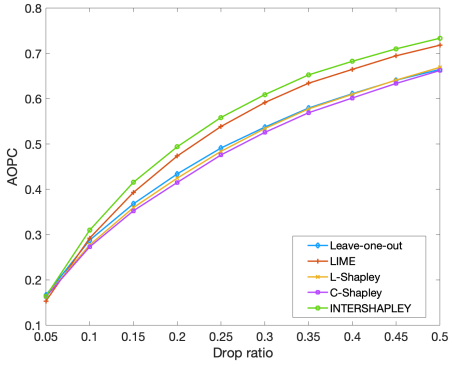


(a) AOPCs of LSTM on IMDB dataset.

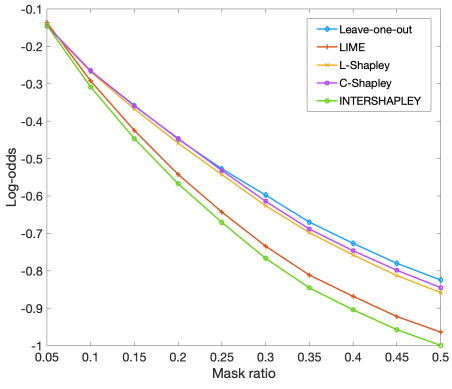


(b) Log-odds of LSTM on IMDB dataset.

Figure 6: The AOPC and log-odds scores of LSTM on IMDB dataset.

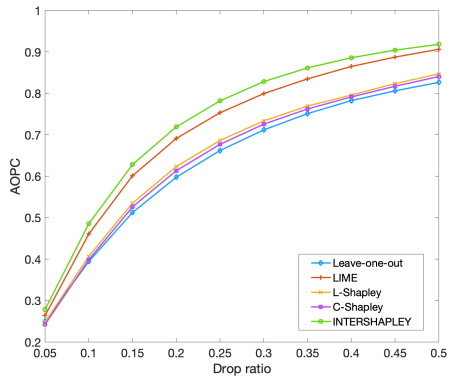


(a) AOPCs of CNN on SST dataset.

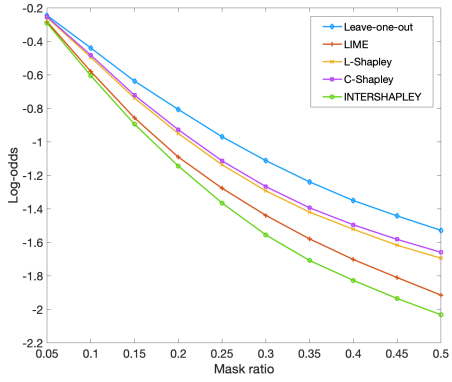


(b) Log-odds of CNN on SST dataset.

Figure 7: The AOPC and log-odds scores of CNN on SST dataset.

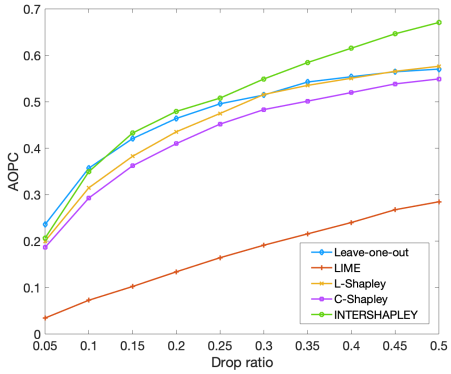


(a) AOPCs of CNN on IMDB dataset.

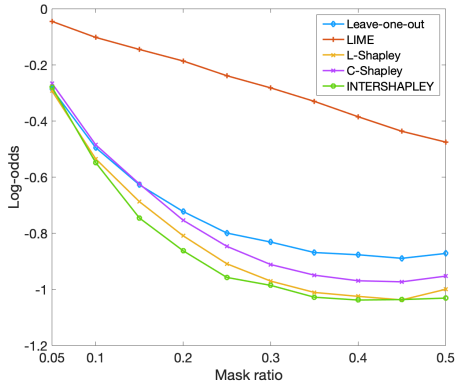


(b) Log-odds of CNN on IMDB dataset.

Figure 8: The AOPC and log-odds scores of CNN on IMDB dataset.

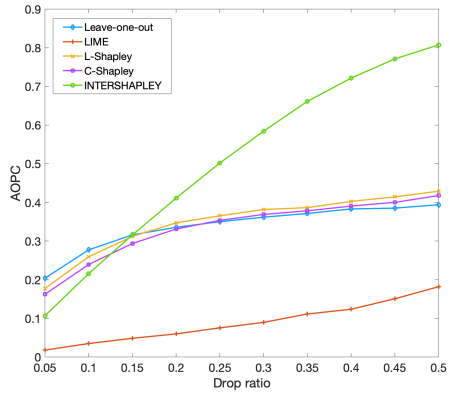


(a) AOPCs of BERT on SST dataset.

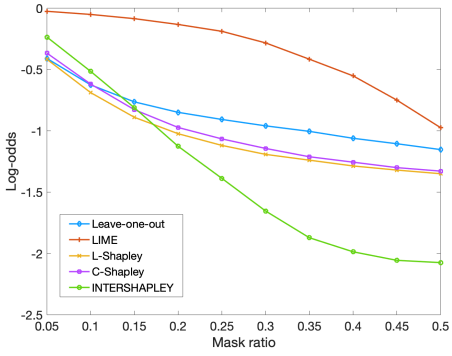


(b) Log-odds of BERT on SST dataset.

Figure 9: The AOPC and log-odds scores of BERT on SST dataset.



(a) AOPCs of BERT on IMDB dataset.



(b) Log-odds of BERT on IMDB dataset.

Figure 10: The AOPC and log-odds scores of BERT on IMDB dataset.

D VISUALIZATION OF HIERARCHICAL INTERPRETATIONS

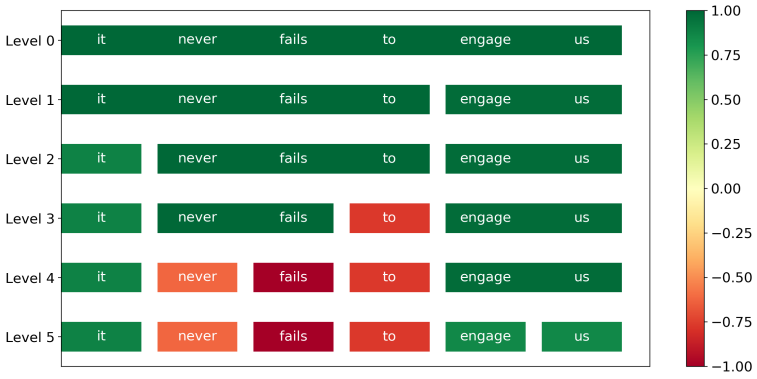


Figure 11: INTERSHAPLEY for BERT on a positive movie review from SST dataset, on which BERT makes correct prediction. Red and green colors represent the negative and positive sentiments respectively.

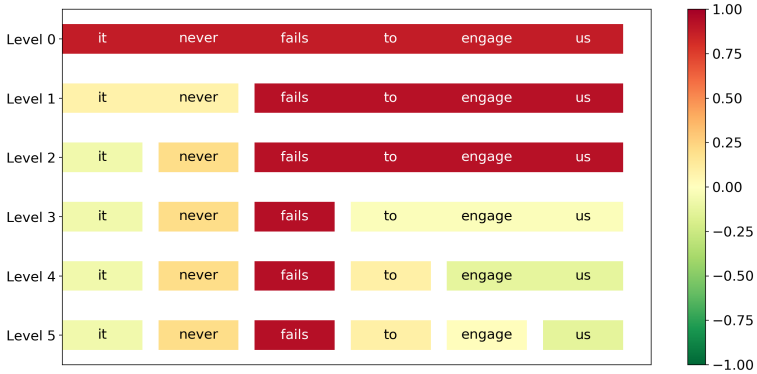


Figure 12: INTERSHAPLEY for LSTM on a positive movie review from SST dataset, on which LSTM makes wrong prediction. Red and green colors represent the negative and positive sentiments respectively.

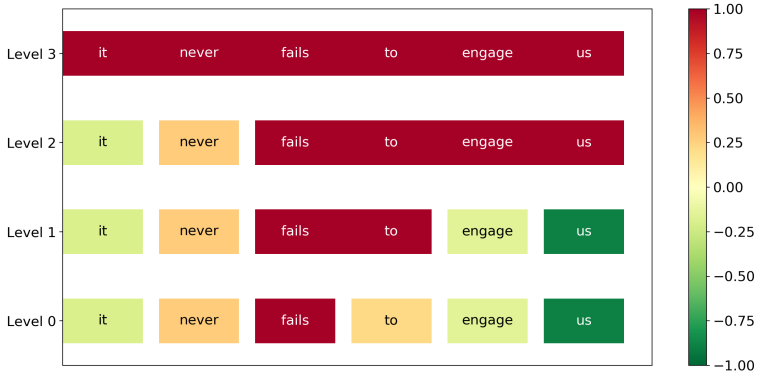


Figure 13: ACD for LSTM on a positive movie review from SST dataset, on which LSTM makes wrong prediction. Red and green colors represent the negative and positive sentiments respectively.

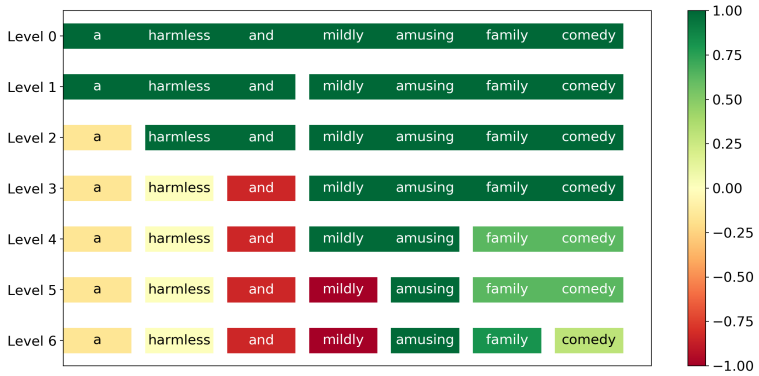


Figure 14: INTERSHAPLEY for BERT on a positive movie review from SST dataset, on which BERT makes correct prediction. Red and green colors represent the negative and positive sentiments respectively.

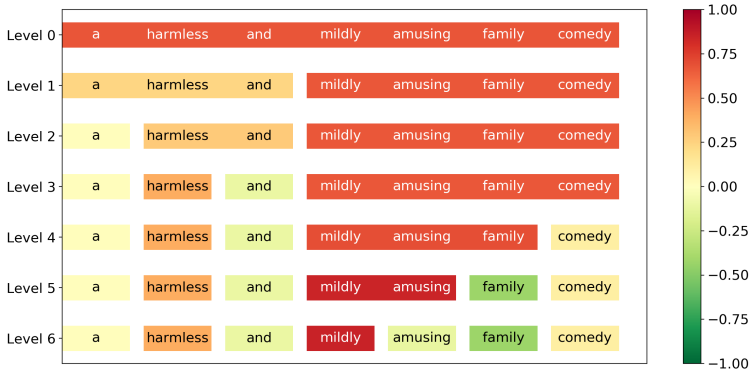


Figure 15: INTERSHAPLEY for LSTM on a positive movie review from SST dataset, on which LSTM makes wrong prediction. Red and green colors represent the negative and positive sentiments respectively.

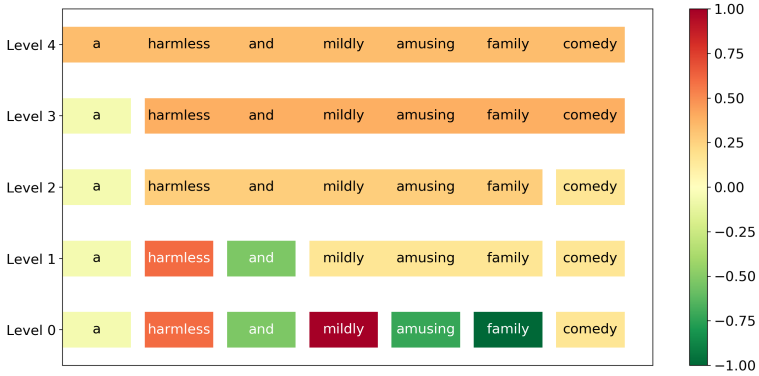


Figure 16: ACD for LSTM on a positive movie review from SST dataset, on which LSTM makes wrong prediction. Red and green colors represent the negative and positive sentiments respectively.

E HUMAN EVALUATION INTERFACE

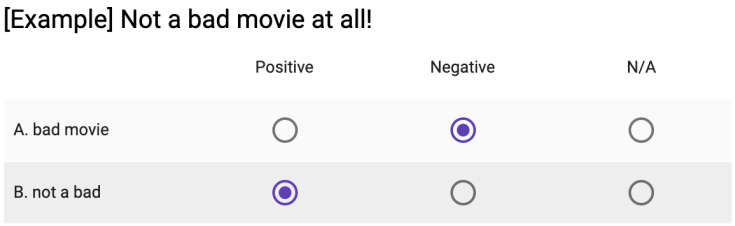


Figure 17: Interfaces of Amazon Mechanical Turk where annotators are asked to guess the model's prediction based on different explanations.