PEACH: Pretrained-Embedding Explanation across Contextual and Hierarchical Structure

Feiqi Cao[1] , **Soyeon Caren Han**[1,2] and **Hyunsuk Chung**[3]
[1]School of Computer Science, University of Sydney
[2]School of Computing and Information Systems, University of Melbourne
[3]FortifyEdge
fcao0492@uni.sydney.edu.au, caren.han@unimelb.edu.au, david.chung@fortifyedge.com

Abstract

In this work, we propose a novel tree-based explanation technique, PEACH (Pretrained-embedding Explanation Across Contextual and Hierarchical Structure), that can explain how text-based documents are classified by using any pretrained contextual embeddings in a tree-based human-interpretable manner. Note that PEACH can adopt any contextual embeddings of the PLMs as a training input for the decision tree. Using the proposed PEACH, we perform a comprehensive analysis of several contextual embeddings on nine different NLP text classification benchmarks. This analysis demonstrates the flexibility of the model by appling several PLM contextual embeddings, its attribute selections, scaling, and clustering methods. Furthermore, we show the utility of explanations by visualising the feature selection and important trend of text classification via human-interpretable word-cloud-based trees, which clearly identify model mistakes and assist in dataset debugging. Besides interpretability, PEACH outperforms or is similar to those from pretrained models. Code and Appendix are in https://github.com/adlnlp/peach.

1 Introduction

Large Pretrained Language Models (PLMs), like BERT, RoBERTa, or GPT, have made significant contributions to the advancement of the Natural Language Processing (NLP) field. Those offer pretrained continuous representations and context models, typically acquired through learning from co-occurrence statistics on unlabelled data, and enhance the generalisation capabilities of downstream models across various NLP domains. PLMs successfully created contextualised word representations and are considered word vectors sensitive to the context in which they appear. Numerous versions of PLMs have been introduced and made easily accessible to the public, enabling the widespread utilisation of contextual embeddings in diverse NLP tasks.

However, the aspect of human interpretation has been rather overlooked in the field. Instead of understanding how PLMs are trained within specific domains, the decision to employ PLMs for NLP tasks is often solely based on their state-of-the-art performance. This raises a vital concern: *Although PLMs demonstrate state-of-the-art performance, it is difficult to fully trust their predictions if humans cannot interpret how well they understand the context and make predictions.* To address those concerns, various interpretable and explainable AI techniques have been proposed in the field of NLP, including feature attribution-based [Sha *et al.*, 2021; Ribeiro *et al.*, 2018; He *et al.*, 2019], language explanation-based [Ling *et al.*, 2017; Ehsan *et al.*, 2018] and probing-based methods [Sorodoc *et al.*, 2020; Prasad and Jyothi, 2020]. Among them, feature attribution based on attention scores has been a predominant method for developing inherently interpretable PLMs. Such methods interpret model decisions locally by explaining the prediction as a function of the relevance of features (words) in input samples. However, these interpretations have two main limitations: it is challenging to trust the attended word or phrase as the sole responsible factor for a prediction [Serrano and Smith, 2019; Pruthi *et al.*, 2020], and the interpretations are often limited to the input feature space, requiring additional methods for providing a global explanation [Han *et al.*, 2020; Rajagopal *et al.*, 2021]. Those limitations of interpretability are ongoing scientific disputes in any research fields that apply PLMs, such as Computer Vision (CV). However, the CV field has relatively advanced PLM interpretability strategies since it is easier to indicate or highlight the specific part of the image. While several post-hoc methods [Zhou *et al.*, 2018; Yeh *et al.*, 2018] give an intuition about the black-box model, decision tree-based interpretable models such as prototype tree [Nauta *et al.*, 2021] have been capable of simulating context understanding, faithfully displaying the decision-making process in image classification, and transparently organising decision rules in a hierarchical structure. However, the performance of these decision tree-based interpretations with neural networks is far from competitive compared to state-of-the-art PLMs [Devlin *et al.*, 2019; Liu *et al.*, 2020].

For NLP, a completely different question arises when we attempt to apply this decision tree interpretation method: *What should be considered a node of the decision tree in NLP tasks?* The Computer Vision tasks typically use the specific image segment and indicate the representative patterns as a node of the decision tree. However, in NLP, it is too risky to use a single text segment (word/phrase) as a representative of decision rules due to semantic ambiguity. For example,

if we have a single word 'party' as a representative node of the global interpretation decision tree, its classification into labels like *'politics'*, *'sports'*, *'business'* and *'entertainment'* would be highly ambiguous.

In this work, we propose a novel decision tree-based interpretation technique, PEACH (Pretrained-embedding Explanation Across Contextual and Hierarchical Structure), that aims to explain how text-based documents are classified using any pretrained contextual embeddings in a tree-based human-interpretable manner. We first fine-tune PLMs for the input feature construction and adopt the feature processing and grouping. Those grouped features are integrated with the decision tree algorithms to simulate the decision-making process and decision rules, by visualising the hierarchical and representative textual patterns. In this paper, our main contributions are as follows:

- Introducing PEACH, the first model that can explain the text classification of any pretrained models in a human-understandable way that incorporates both global and local interpretation.
- PEACH can simulate the context understanding, show the text classification decision-making and transparently arrange hierarchical-structured decision rules.
- Conducting comprehensive evaluation to present the preservability of the model prediction behaviour, which is perceived as trustworthy and understandable by human judges compared to widely-used benchmarks.

2 PEACH

The primary objective of PEACH (Pretrained-embedding Explainable model Across Contextual and Hierarchical Structure) is to identify a contextual and hierarchical interpretation model that elucidates text classifications using any pretrained contextual embeddings. To accomplish this, we outline the construction process, including input representation, feature selection and integration, tree generation, and *interpretability and visualisation of our tree-based model*.

Preliminary Setup Before delving into the components of the proposed explanation model PEACH, it is crucial to distinguish between the pretrained model, fine-tuned model, and its contextual embedding. Appendix B illustrates the pretraining and fine-tuning step. The pretraining step involves utilising a large amount of unlabelled data to learn the overall language representation, while the fine-tuning step further refines this knowledge and generates better-contextualised embeddings on task-specific labelled datasets. Fine-tuned contextual pretrained embeddings typically serve as a valuable resource for representing the text classification capabilities of various deep learning models. Therefore, PEACH leverages these contextualised embeddings to explain the potential outcomes of a series of related choices using a contextual and hierarchical decision tree.

2.1 Input Embedding Construction

We first wrangle the extracted pretrained contextual embeddings in order to demonstrate the contextual understanding capability of the fine-tuned model. Note that we use fine-tuned contextual embedding as input features by applying the following two steps.

Step 1: Fine-Tuning
Given a corpus with n text documents, denoted as $T = \{t_1, t_2, \ldots, t_n\}$, each t_a represents a document instance from the textual dataset. Each document can be from any text-based corpus, such as news articles, movie reviews, or medical abstracts, depending on the specific task. Each document can be represented as a semantic embedding by pretrained models. To retrieve the contextual representation, we firstly tokenise t_a for $a \in [1, n]$ with the pretrained model tokeniser and fine-tune the PLMs on the tokenised contents of all documents, with the goal of predicting the corresponding document label. Then, we extract the d-dimensional embedding of the PLMs [CLS] token as the contextualised document embedding $e_a \in E$ for $a \in [1, n]$ ($d = 768$).

Step 2: Feature Processing
By using all the embeddings in E as row vectors, we construct a feature matrix $M \in \mathbb{R}^{n*d}$. This feature matrix can be represented as $[c_1\ c_2 \ldots\ c_d]$ where each c_i corresponds to the column feature vector along the i-th dimension, which contains embedding values for the i-th dimension from all document embeddings. To optimise the utilisation of the embedding feature matrix in decision tree training, we experiment with various feature selection methods, including the following statistical approaches and a deep learning approach, to extract the most informative features.

Statistical Approaches We employ statistical approaches to extract informative features from the column features of M. We calculate the **correlation** between each pair of dimensions using Pearson correlation coefficient. For dimensions i and j ($i, j \in [1, d]$), the correlation $R_{i,j}$ is computed as:

$$R_{i,j} = Pearson(c_i, c_j) \tag{1}$$

for each $i, j \in [1, d]$.

Dimensions with high Pearson correlation values indicate similar semantic features during fine-tuning. To identify those similar dimensions, we use a percentile v to find the correlation threshold. The threshold t is calculated as

$$t = P_v(R) \tag{2}$$

which gives us the v-th percentile value in R. Using this threshold, we cluster the 768 dimensions and take the average to reduce the number of features we will have eventually.

We divide the set of column features $\{c_1, c_2, \ldots, c_d\}$ into m exclusive clusters. Starting from c_1, we find all the dimensions having correlations greater than t with c_1 and collect them as a new cluster C_1. Among the remaining dimensions, we take the first column feature (e.g. $c_k \notin C_1$) as the new cluster centre and find all the dimensions correlating greater than t with c_k, and consider them as the new cluster C_2. This process is repeated iteratively until all dimensions are assigned to a certain cluster. In this way, we re-arranged all the dimensions into a set of clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ where the column vectors in each cluster have correlations greater than t with the cluster centre.

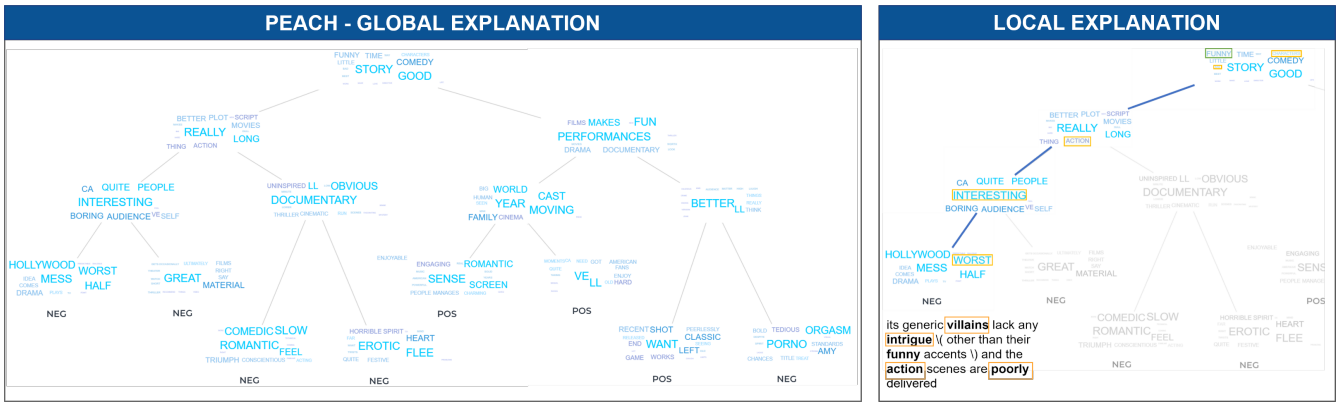


Figure 1: A PEACH is a globally interpretable model that faithfully explains the pretrained models' reasoning using the pretained contextual embedding (left, on MR dataset). Additionally, the decision-making process for a single prediction can be detailed presented (right, partially shown). The detailed description can be found in Section 5.

In addition to Pearson, we explore **K-means Clustering** as an alternative method to cluster the dimension vectors. The K-means aims to minimise the objective function given by:

$$L(M) = \sum_{i=1}^{m} \sum_{j=1}^{d} (||v_i - c_j||)^2 \tag{3}$$

where v_i is the cluster centre for each cluster C_i. After clustering, we merge the features in each cluster as a single feature vector by taking their average since they exhibit high correlation. By combining the representations from each cluster, we obtain the final feature matrix $F \in \mathbb{R}^{n*m}$. This successfully reduces the number of features from the original embedding dimension d to the number of clusters m. The resulting feature matrix F can be directly used as input for training decision trees using ID3/C4.5/CART algorithms.

Deep Learning Approach We also apply a Convolutional Neural Network (CNN) to extract the input feature matrix F from the initial embedding feature matrix E. Our CNN consists of two blocks, each comprising a 1D convolutional layer followed by a 1D pooling layer. The network is trained to predict the document class based on the output of the last pooling layer, minimising the cross-entropy loss. The filter of each layer reduces the dimension according to the following way, ensuring the last pooling layer has dimension m:

$$D_{out} = \frac{D_{in} - f + 2p}{s} - 1 \tag{4}$$

where D_{in} is the input feature dimension of the convolution/pooling layer, D_{out} is the output feature dimension of the convolution/pooling layer, f is the filter size, p is the padding size, and s is the stride size for moving the filter.

2.2 Decision Tree Generation

We apply the feature matrix F to construct various types of decision trees. In this section, we describe several traditional decision tree training algorithms that we adopted for our model. The **ID3** algorithm calculates the information gain to determine the specific feature for splitting the data into subsets. For each input feature $f_i \in F$ that has not been used as a splitting node previously, the information gain (IG) is computed as follows to split the current set of data instances S:

$$\begin{aligned} IG(S, f_i) &= H(S) - \sum_{t \in T} p(t) H(t) \\ &= H(S) - H(S|f_i) \end{aligned} \tag{5}$$

where $H(S)$ is entropy of the current set of data S, T is the subsets of data instances created from splitting S by f_i and $p(t)$ is the proportion of the number of elements in t to the number of elements in S and $H(t)$ is the entropy of subset t. The feature with the maximum information gain is selected to split S into two different splits as the child nodes. The **C4.5** algorithm calculates the gain ratio to select the specific feature to split the data into subsets. It is similar to ID3 but instead of calculating information gain, it calculates the gain ratio to select the splitting feature. The gain ratio is calculated as follows:

$$GainRatio(S, f_i) = \frac{IG(S, f_i)}{SplitInfo(S, f_i)} \tag{6}$$

where

$$SplitInfo(S, f_i) = \sum_{t \in T} p(t) log_2(t) \tag{7}$$

The **CART** algorithm calculates the Gini index to select the specific feature for splitting the data into subsets. The Gini index is defined as:

$$Gini(S, f_i) = 1 - \sum_{x=1}^{n} (P_x)^2 \tag{8}$$

where P_x is the probability of a data instance being classified to a particular class. The feature with the smallest Gini is selected as the splitting node. The **Random Forest (RF)** algorithm functions as an ensemble of multiple decision trees, where each tree is generated using a randomly selected subset of the input features. The individual trees in the forest can be constructed using any of the aforementioned algorithms.

2.3 Interpretability and Visualisation

PEACH aims to foster global and local interpretability for text classification by arranging hierarchical-structured decision rules. PEACH simulates the context understanding, shows the text classification decision-making process and transparently presents a hierarchical decision tree. For the decision tree, the leaves present the class distributions, the paths from the root to the leaves represent the learned classification rules, and the nodes contain representative parts of the textual corpus. This section explains the way of representing the node in the tree structure and which way of visualisation presents a valuable pattern for human interpretation.

Interpretable Prototype Node: The aim of the decision tree nodes in PEACH is to visualise the context understanding and the most common words in the specific decision path, and simulate the text classification decision-making process. Word clouds are great visual representations of word frequency that give greater prominence to words that appear more frequently in a source text. These particular characteristics of word clouds would be directly aligned with the aim of the decision tree nodes. For each node in the tree, we collect all the documents going through this specific node in their decision path. These documents are converted into lowercase, tokenised, and the stopwords in these documents are removed. Then, Term Frequency Inverse Document Frequency (TFIDF) of the remaining words is calculated and sorted. We take the 100 distinct words with the top TFIDF values to be visualised as a word cloud. This gives us an idea of the semantics that each node of the tree represents and how each decision path evolves before reaching the leaf node (the final class).

Visualisation Filter: The more valuable visualisation pattern the model presents, the better the human-interpretable models are. We apply two valuable token/word types in order to enhance the quality of visualisation of the word cloud node in the decision tree. First, we adopt Part-of-Speech (PoS) tagging, which takes into account which grammatical group (Noun, Adjective, Adverb, etc.) a word belongs to. With this PoS tagging, it is easy to focus on the important aspect that each benchmark has. For example, the sentiment analysis dataset would consider more emotions or polarities so adjective or adverb-based visualisation would be more valuable. Secondly, we also apply a Named Entity Recognition (NER), which is one of the most common information extraction techniques and identifies relevant nouns (person, places, organisations, etc.) from documents or corpus. NER would be a great filter for extracting valuable entities and the main topic of the decision tree decision-making process.

Word Cloud - Document Word Matching: To facilitate the visualisation of local interpretations, which aims to elucidate the decision-making process for specific document examples, we employ two methods to align word elements in word clouds with the actual document for classification. The first method involves an exact string match. When a word in the word clouds exactly corresponds to a word token in the original document, we highlight all instances of that word in both the word clouds and the original document using a green colour. The second method employs WordNet synonym matching, leveraging the WordNet from the nltk[1] library. We explore the document for words that belong to possible synonym sets of the words in the word clouds, highlighting them in yellow if a match is found.

3 Evaluation Setup[2]

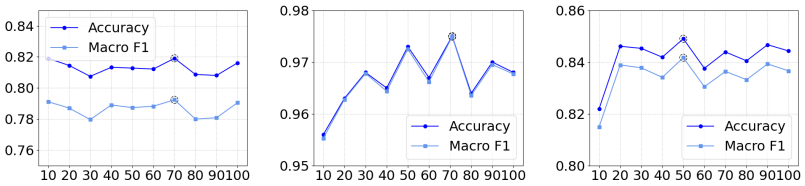
3.1 Datasets[3]

We evaluate PEACH with 5 state-of-the-art PLMs on 9 benchmark datasets. Those datasets encompass five text classification tasks, including Natural Language Inference (NLI), Sentiment Analysis (SA), News Classification (NC), Topic Analysis (TA), and Question Type Classification (QC). **1) Natural Language Inference (NLI)** *Microsoft Research Paraphrase (MSRP)* [Dolan *et al.*, 2004] contains 5801 sentence pairs with binary labels. The task is to determine whether each pair is a paraphrase or not. The training set contains 4076 sentence pairs and 1725 testing pairs for generating decision trees. During PLM finetuning, we randomly split the training set with a 9:1 ratio so 3668 pairs are used for training and 408 pairs are used for validation. *Sentences Involving Compositional Knowledge (SICK)* [Marelli *et al.*, 2014] dataset consists of 9840 sentence pairs that involve compositional semantics. Each pair can be classified into three classes: entailment, neutral or contradiction. The dataset has 4439, 495, and 4906 pairs for training, validation and testing sets. For both MSRP and SICK, we combine each sentence pair as one instance when finetuning PLMs. **2) Sentiment Analysis (SA)** The sentiment analysis datasets in this study are binary for predicting positive or negative movie reviews. *Standford Sentiment Treebank (SST2)* [Socher *et al.*, 2013] has 6920, 872 and 1821 documents for training, validation and testing. The *MR* [Pang and Lee, 2005] has 7108 training and 3554 training documents. *IMDB* [Maas *et al.*, 2011] and 25000 training and 25000 training documents. For MR and IMDB, since no official validation split is provided, the training sets are randomly split into a 9:1 ratio to obtain a validation set for finetuning PLMs. **3) News Classification (NC)** The *BBCNews* is used to classify news articles into five categories: entertainment, technology, politics, business, and sports. There are 1225 training and 1000 testing instances, and we further split the 1225 training instances with 9:1 ratio to get the validation set for finetuning PLMs. *20ng* is for news categorization with 11314 training and 7532 testing documents and aims to classify news articles into 20 different categories. Similar to BBCNews, the training set is split with 9:1 ratio to obtain the finetuning validation set. **4) Topic Analysis (TA)** The *Ohsumed* provides 7400 medical abstracts, with 3357 train and 4043 test, to categorise into 23 disease types. **5) Question Type Classification (QC)** The *Text REtrieval Conference (TREC)* Question Classification dataset offers 5452 questions in the training and 500 questions in the testing set. This dataset categorises different natural language questions into six types: abbreviation, entity, description and abstract, human, location, and numbers.

[1] https://www.nltk.org/howto/wordnet.html

[2] The baseline description can be found in the Appendix B

[3] The statistics can be found in the Appendix B



(a) MSRP (b) BBCNews (c) 20ng

Figure 2: The effects of input feature dimension with PEACH on three datasets, including MSRP, BBCNews, and 20ng

3.2 Implementation Details

We perform finetuning of the PLMs and then construct the decision tree-based text classification model for the evaluation. We initialise the weights from **five base models: bert-base-uncased, roberta-base, albert-base-v2, xlnet-base-cased and the original 93.6M ELMo** for BERT, RoBERTa, ALBERT, XLNet, and ELMo respectively. A batch size of 32 is applied for all models and datasets. The learning rate is set to 5e-5 for all models, except for the ALBERT model of SST2, 20ng and IMDB datasets, where it is set to 1e-5. All models are fine-tuned for 4 epochs, except for the 20ng, which used 30 epochs. To extract the features from learned embedding and reduce the number of input features into the decision tree, we experiment with quantile thresholds of 0.9 and 0.95 for correlation methods. For k-means clustering, we search for the number of clusters from 10 to 100 (step size: 10 or 20), except for IMDB where we search from 130 to 220 (step size: 30). For CNN features, we use kernel size 2, stride size 2, and padding size 0 for two convolution layers. The same hyperparameters are applied to the first pooling layer, except for IMDB where stride size 1 is used to ensure we can have enough input features for the next convolutional block and get enough large number of features from the last pooling layer. Kernel size and stride size for the last pooling layer are adjusted to maintain consistency with the number of clusters used in k-means clustering. Decision trees are trained using the chefboost library with a maximum depth of 95. We used em_core_web_sm model provided by spaCy library to obtain the NER and POS tags for visualization filters. All experiments are conducted on Google Colab with Intel(R) Xeon(R) CPU and NVIDIA T4 Tensor Core GPU. Classification accuracy is reported for comparison.

4 Results

4.1 Overall Classification Performance

We first evaluate the utility of PLMs after incorporating PEACH in Table 1. As shown, our proposed PEACH with PLMs (rows 7-11) outperforms or performs similarly to the fine-tuned PLMs (rows 2-6) in all benchmarks. Note that our model outperforms the baselines in all binary sentiment analysis datasets (SST2, MR, IMDB) when utilising RoBERTa features in our PEACH model. Furthermore, our model outperforms the baselines in more general domain datasets with multiple classes, such as BBC News, TREC, and 20ng, when BERT features are employed in PEACH. We also experimented with various types of PLMs, other than BERT and RoBERTa, on our PEACH, including BERT-based (ALBERT), generative-based (XLNet), and LSTM-based model (ELMo). In general, RoBERTa and BERT performed better across all datasets, except IMDB, where XLNet and its PEACH model showed superior performance. This observation is attributed to the larger corpus of IMDB and requires more features for an accurate explanation.

4.2 Ablation and Parameter Studies[4]

The effects of Feature Processing All three feature processing methods we propose in Section 2.1 work well overall. Table 2 shows that K-means demonstrated the best across most datasets, except for MR and TREC. CNN worked better on the MR dataset. Conversely, Pearson correlation grouping worked better for TREC. The fine-tuned BERT model captured features that exhibited stronger correlations with each other rather than clustering similar question types together.

The effects of Input Dimension We then evaluate the effects of input feature dimension size with PEACH. We selected three datasets, including BBCNews (the largest average document length with a small data size), 20ng (a large number of classes with a larger dataset size) and MSRP (a moderate number of documents and a moderate average length). We present macro F1 and Accuracy to analyse the effects of class imbalance for some datasets. Figure 2 shows there is no difference in the trend for F1 and the trend for accuracy on these datasets; especially the relatively balanced dataset BBCNews provides almost identical F1 and accuracy. The binary dataset MSRP does not lead to large gaps in the performance of different input dimensions, however, for those with more classes (BBCNews and 20ng), there is a noticeable performance drop when using extremely small input dimensions like 10. The performance difference becomes less significant as the input dimension increases beyond 20.

The Maximum Tree Depth Analysis was conducted to evaluate the visualisation of the decision-making pattern. The result can be found in Appendix C.

4.3 Human Evaluation

To assess interpretability and trustability, we conducted a human evaluation[5], 26 human judges[6] annotated 75 samples

[4]The effects of the Decision Tree is in Appendix A

[5]Sample Cases for the Human Evaluation is in the Appendix H

[6]Annotators are undergraduates and graduates in computer science; 6 females and 20 males. The number of human judges and samples is relatively higher than other NLP interpretation papers [Rajagopal *et al.*, 2021]

Model	MSRP	SST2	MR	IMDB	SICK	BBCNews	TREC	20ng	Ohsumed
BERT [Devlin *et al.*, 2019]	0.819	0.909	0.857	0.870	0.853	0.969	0.870	0.855	0.658
RoBERTa [Liu *et al.*, 2020]	0.824	0.932	0.867	0.892	0.881	0.959	0.972	0.802	0.655
ALBERT [Lan *et al.*, 2020]	0.665	0.907	0.821	0.879	0.838	0.917	0.938	0.794	0.518
XLNet [Yang *et al.*, 2019]	0.819	0.907	0.879	0.905	0.727	0.959	0.950	0.799	0.669
ELMo [Peters *et al.*, 2018]	0.690	0.806	0.751	0.804	0.608	0.845	0.790	0.537	0.359
PEACH (BERT)	0.816	0.885	0.853	0.871	0.837	**0.975**	**0.978**	**0.849**	0.649
PEACH (RoBERTa)	**0.819**	**0.938**	**0.872**	**0.893**	**0.877**	0.947	0.968	0.800	**0.651**
PEACH (ALBERT)	0.650	0.885	0.804	0.878	0.834	0.921	0.942	0.783	0.497
PEACH (XLNet)	0.809	0.903	0.790	**0.899**	0.832	0.964	0.966	0.776	0.638
PEACH (ELMo)	0.638	0.632	0.510	0.725	0.565	0.900	0.702	0.164	0.284

Table 1: Classification performance comparison between fine-tuned contextual embeddings and those with PEACH. Best performances among the baselines are underscored, and the best performances among our PEACH variants are bolded.

Model	MSRP	SST2	MR	IMDB	SICK	BBCNews	TREC	20ng	Ohsumed
PEACH (Pearson)	0.817	0.913	0.862	0.892	0.867	0.972	**0.978**	0.845	0.645
PEACH (K-means)	**0.819**	**0.938**	0.869	**0.893**	**0.877**	**0.975**	0.974	**0.849**	**0.651**
PEACH (CNN)	0.817	0.936	**0.872**	0.890	0.870	0.974	0.974	0.801	0.621

Table 2: The effects of feature processing approach.

from MR, SST2, TREC, and BBC News. Judges conducted the pairwise comparison between local and global explanations generated by PEACH against two commonly used text classification-based interpretability models, LIME [Ribeiro *et al.*, 2016] and Anchor [Ribeiro *et al.*, 2018]. The judges were asked to choose the approach they trusted more based on the interpretation and visualisation provided. We specifically considered samples, where both LIME and PEACH or Anchor and PEACH predictions were the same, following [Wan *et al.*, 2021]. Among 26 judges, our PEACH explanation evidently outperforms the baselines by a large margin. All percentages in the first column of all four datasets are over 84%, indicating that the majority of annotators selected our model to be better across interpretability and trustability. The last column ('Agree') represents results from the Fleiss' kappa test used to assess inter-rater consistency, and all the agreement scores are over 0.7 which shows a strong level of agreement between annotators. Several judges commented that the visualisation method of PEACH allows them to see the full view of the decision-making process in a hierarchical decision path and check how the context is trained in each decision node. This indicates a higher level of trust in PEACH than in the saliency technique commonly employed in NLP.

5 Analysis and Application

5.1 Visualisation Analysis

Figure 1 in Section 1 shows the sample interpretations by PEACH on MR, a binary dataset predicting positive or negative movie reviews. The global explanation in Figure 1(left) faithfully shows the entire classification decision-making behaviour in detail. Globally, the decision tree and its nodes cover various movie-related entities and emotions, like DOCUMENTARY, FUN, FUNNY, ROMANTIC, CAST, HOLLYWOOD, etc. In addition, final nodes (leaves) are passed via the clear path with definite semantic words, distinguishing between negative and positive reviews. In addition to the global interpretation, our PEACH can produce a local explanation (Figure 1, right). By applying generated decision rules to the specific input text, a rule path for the given input simulates the decision path that can be derived to the final classification (positive or negative). We also present how PEACH can visualise the interpretation by comparing the successful and unsuccessful pretrained embeddings in Figure 3. While the successful one (RoBERTa with MR - Figure 3 left) shows a clear and traceable view of how they can classify the positive samples by using adjectives like moving and engaging, the unsuccessful one (ELMo with MR - Figure 3 right) has many ambiguous terminologies and does not seem to understand the pattern in both positive and negative classes, e.g. amusing is in the negative class. More visualisations on different datasets and models are shown in Appendix D and E.

5.2 Application: PEACH

We developed an interactive decision tree-based text classification decision-making interpretation system for different PLMs. This would be useful for human users to check the feasibility of their pretrained and/or fine-tuned model. The user interface and detailed description are in Appendix F.

5.3 Case-Study: High-risk Medical Domain

Furthermore, illustrated in Appendix G, we undertook an additional case study focused on the high-risk text classification domain, specifically in tasks such as medical report analysis. This endeavour aimed to illustrate how interpretability can be effectively integrated into specialised domains with heightened sensitivity and complexity. By applying our visualization techniques and interpretative tools to the intricacies of medical text analysis, we sought to demonstrate the adaptability and utility of our approach in enhancing transparency and understanding within critical and specialised areas of text classification.

MR				SST2				TREC				BBCNews			
PEACH	**LIME**	**Tie**	**Agree**	**PEACH**	**LIME**	**Tie**	**Agree**	**PEACH**	**LIME**	**Tie**	**Agree**	**PEACH**	**LIME**	**Tie**	**Agree**
92.3	1.3	6.4	0.83	88.4	1.8	9.8	0.78	96.1	1.2	2.7	0.81	84.6	3.9	11.5	0.75
PEACH	**Anchor**	**Tie**	**Agree**	**PEACH**	**Anchor**	**Tie**	**Agree**	**PEACH**	**Anchor**	**Tie**	**Agree**	**PEACH**	**Anchor**	**Tie**	**Agree**
94.6	2.1	3.3	0.85	87.2	2.5	10.3	0.76	95.4	1.8	2.8	0.83	85.3	3.5	11.2	0.71

Table 3: Human evaluation results. Pairwise comparison between PEACH with LIME and Anchor across the interpretability. The 'Agree' column shows the Fleiss' Kappa results.

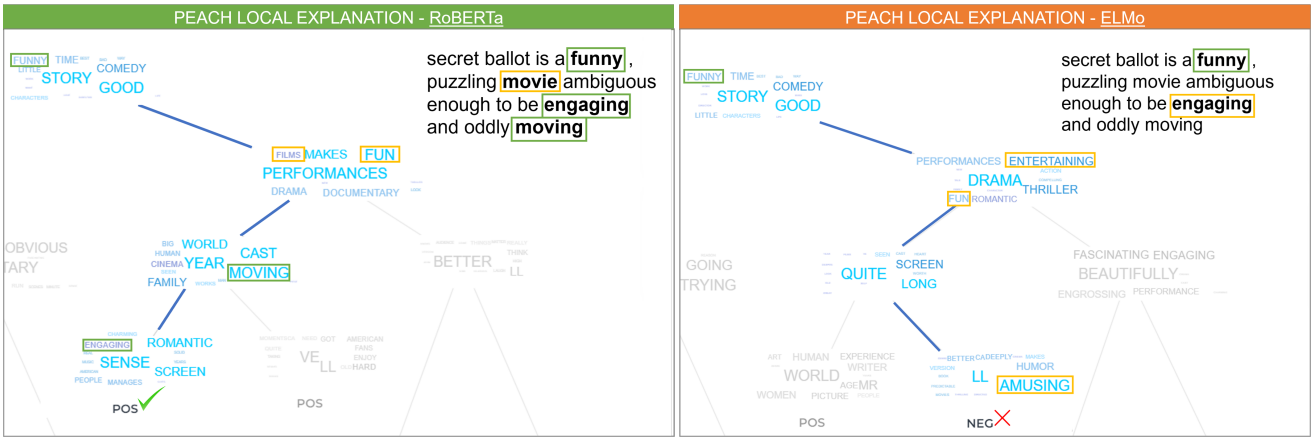


Figure 3: The local explanation decision trees generated for a **positive** review in MR dataset based on fine-tuned RoBERTa embedding compared to fine-tuned ELMo embedding.

6 Related Works

Interpretable Models in NLP Among interpretable NLP, feature attribution-based methods are the most common[7]. There are mainly four types, including Rationale Extraction [Lei *et al.*, 2016; Sha *et al.*, 2021], Input Perturbation [Ribeiro *et al.*, 2016; Ribeiro *et al.*, 2018; Slack *et al.*, 2020], Attention Methods [Mao *et al.*, 2019], and Attribution Methods [He *et al.*, 2019; Du *et al.*, 2019]. Such models locally explain the prediction based on the relevance of input features (words). However, global explainability is crucial to determine how much each feature contributes to the model's predictions of overall data. A few recent studies touched on the global explanation idea and claimed they have global interpretation by providing the most relevant concept [Rajagopal *et al.*, 2021] or the most influential examples [Han *et al.*, 2020] searched from the corpus to understand why the model made certain predictions. Such global explanations do not present the overall decision-making flow of the models.

Tree-structured Model Interpretation The recent studies adopting decision-tree and rules[Quinlan, 2014; Han *et al.*, 2014] into neural networks [Fuhl *et al.*, 2020; Lee and Jaakkola, 2020; Wang *et al.*, 2020a] introduced neural trees compatible with the state-of-the-art of CV and NLP downstream tasks. Such models have limited interpretation or are only suitable to the small-sized dataset. Tree-structured neural models have also been adopted in syntactic or semantic parsing [Nguyen *et al.*, 2020; Wang* *et al.*, 2020b; Yu *et al.*, 2021; Zhang *et al.*, 2021]. Few decision-tree-based approaches show the global and local explanations of black-boxed neural models. NBDT [Wan *et al.*, 2021] applies a sequentially interpretable neural tree and uses parameters induced from trained CNNs and requires WordNet to establish the interpretable tree. ProtoTree [Nauta *et al.*, 2021] and ViT-NeT [Kim *et al.*, 2022] construct interpretable decision trees for visualising decision-making with prototypes for CV tasks. Despite their promise, those have not been adopted in NLP.

[7]Some studies cover the language explanation-based, probing-based or counterfactual explanation-based methods, but the text-based model interpretation methods are dominant by feature attribution-based approaches.

7 Conclusion

We introduce PEACH, a novel tree-based explanation technique for text-based classification using pretrained contextual embeddings. While many NLP applications rely on PLMs, the focus has been on employing them without thoroughly analysing their contextual understanding for specific tasks. We provide a human-interpretable explanation of how text-based documents are classified, using any pretrained contextual embeddings in a hierarchical tree-based manner. The human evaluation also indicates that the visualisation method of PEACH allows them to see the full global view of the decision-making process in a hierarchical decision path, making fine-tuned PLMs interpretable. We hope that PEACH can open avenues for understanding the reasons behind the effectiveness of PLMs in NLP.

Acknowledgements

The two first authors, Feiqi Cao and Soyeon Caren Han have equal contribution. We thank Dr. Nick Lewins for fruitful suggestion, and Prof. Ross Quinlan for inspiring our research.

References

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minnesota, June 2019. Association for Computational Linguistics.

[Dolan *et al.*, 2004] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland, aug 23–aug 27 2004. COLING.

[Du *et al.*, 2019] Mengnan Du, Ninghao Liu, Fan Yang, Shuiwang Ji, and Xia Hu. On attribution of recurrent neural network predictions via additive decomposition. In *The World Wide Web Conference*, pages 383–393, 2019.

[Ehsan *et al.*, 2018] Upol Ehsan, Brent Harrison, Larry Chan, and Mark O Riedl. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 81–87, 2018.

[Fuhl *et al.*, 2020] Wolfgang Fuhl, Gjergji Kasneci, Wolfgang Rosenstiel, and Enkeljda Kasneci. Training decision trees as replacement for convolution layers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3882–3889, Apr. 2020.

[Han *et al.*, 2014] Soyeon Caren Han, Hee-Geun Yoon, Byeong Ho Kang, and Seong-Bae Park. Using mcrdr based agile approach for expert system development. *Computing*, 96:897–908, 2014.

[Han *et al.*, 2020] Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online, July 2020. Association for Computational Linguistics.

[He *et al.*, 2019] Shilin He, Zhaopeng Tu, Xing Wang, Longyue Wang, Michael Lyu, and Shuming Shi. Towards understanding neural machine translation with word importance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 952–961, 2019.

[Kim *et al.*, 2022] Sangwon Kim, Jaeyeal Nam, and Byoung Chul Ko. ViT-NeT: Interpretable vision transformers with neural tree decoder. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 11162–11172. PMLR, 17–23 Jul 2022.

[Lan *et al.*, 2020] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[Lee and Jaakkola, 2020] Guang-He Lee and Tommi S. Jaakkola. Oblique decision trees from derivatives of relu networks. In *International Conference on Learning Representations*, 2020.

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

[Ling *et al.*, 2017] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, 2017.

[Liu *et al.*, 2020] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2020.

[Maas *et al.*, 2011] 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*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[Mao *et al.*, 2019] Qianren Mao, Jianxin Li, Senzhang Wang, Yuanning Zhang, Hao Peng, Min He, and Lihong Wang. Aspect-based sentiment classification with attentive neural turing machines. In *IJCAI*, pages 5139–5145, 2019.

[Marelli *et al.*, 2014] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation*, pages 1–8, Dublin, 2014. Association for Computational Linguistics.

[Nauta *et al.*, 2021] M. Nauta, R. van Bree, and C. Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14928–14938, Los Alamitos, CA, USA, 2021. IEEE Computer Society.

[Nguyen *et al.*, 2020] Xuan-Phi Nguyen, Shafiq Joty, Steven Hoi, and Richard Socher. Tree-structured attention with hierarchical accumulation. In *International Conference on Learning Representations*, 2020.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. 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)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[Peters *et al.*, 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, Louisiana, June 2018. Association for Computational Linguistics.

[Prasad and Jyothi, 2020] Archiki Prasad and Preethi Jyothi. How accents confound: Probing for accent information in end-to-end speech recognition systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3739–3753, Online, July 2020. Association for Computational Linguistics.

[Pruthi *et al.*, 2020] Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Graham Neubig, and Zachary C. Lipton. Learning to deceive with attention-based explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4782–4793, Online, July 2020. Association for Computational Linguistics.

[Quinlan, 2014] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[Rajagopal *et al.*, 2021] Dheeraj Rajagopal, Vidhisha Balachandran, Eduard H Hovy, and Yulia Tsvetkov. SELFEXPLAIN: A self-explaining architecture for neural text classifiers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 836–850, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[Ribeiro *et al.*, 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*, pages 1135–1144, 2016.

[Ribeiro *et al.*, 2018] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

[Sha *et al.*, 2021] Lei Sha, Oana-Maria Camburu, and Thomas Lukasiewicz. Learning from the best: Rationalizing predictions by adversarial information calibration. In *AAAI*, pages 13771–13779, 2021.

[Slack *et al.*, 2020] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

[Socher *et al.*, 2013] 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*, pages 1631–1642, Washington, 2013. Association for Computational Linguistics.

[Sorodoc *et al.*, 2020] Ionut-Teodor Sorodoc, Kristina Gulordava, and Gemma Boleda. Probing for referential information in language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4177–4189, Online, July 2020. Association for Computational Linguistics.

[Wan *et al.*, 2021] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Suzanne Petryk, Sarah Adel Bargal, and Joseph E. Gonzalez. {NBDT}: Neural-backed decision tree. In *International Conference on Learning Representations*, 2021.

[Wang *et al.*, 2020a] Ke Wang, Xuyan Chen, Ning Chen, and Ting Chen. Automatic emergency diagnosis with knowledge-based tree decoding. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3407–3414, 2020.

[Wang* *et al.*, 2020b] Ziqi Wang*, Yujia Qin*, Wenxuan Zhou, Jun Yan, Qinyuan Ye, Leonardo Neves, Zhiyuan Liu, and Xiang Ren. Learning from explanations with neural execution tree. In *International Conference on Learning Representations*, 2020.

[Yang *et al.*, 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[Yeh *et al.*, 2018] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[Yu *et al.*, 2021] Jing Yu, Yuan Chai, Yujing Wang, Yue Hu, and Qi Wu. Cogtree: Cognition tree loss for unbiased scene graph generation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1274–1280, 2021. Main Track.

[Zhang *et al.*, 2021] Die Zhang, Hao Zhang, Huilin Zhou, Xiaoyi Bao, Da Huo, Ruizhao Chen, Xu Cheng, Mengyue Wu, and Quanshi Zhang. Building interpretable interaction trees for deep nlp models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14328–14337, May 2021.

[Zhou *et al.*, 2018] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.