Locally Distributed Activation Vectors for Guided Feature Attribution

Housam K. B. Babiker[1], **Mi-Young Kim**[2], **Randy Goebel**[1]
[1] Department of Computing Science, University of Alberta
[2] Department of Science, Augustana Faculty, University of Alberta
Alberta Machine Intelligence Institute
Edmonton, Alberta, Canada
{khalifab, miyoung2, rgoebel}@ualberta.ca

Abstract

Explaining the predictions of a deep neural network (DNN) is a challenging problem. Many attempts at interpreting those predictions have focused on attribution-based methods, which assess the contributions of individual features to each model prediction. However, attribution-based explanations do not always provide faithful explanations to the target model, e.g., noisy gradients can result in unfaithful feature attribution for back-propagation methods. We present a method to learn explanations-specific representations while constructing deep network models for text classification. These representations can be used to faithfully interpret black-box predictions, i.e., highlighting the most important input features and their role in any particular prediction. We show that learning specific representations improves model interpretability across various tasks, for both qualitative and quantitative evaluations, while preserving predictive performance.

1 Introduction

Deep neural network (DNN) models have become crucial tools in natural language processing (NLP) and define state-of-the-art on a large variety of tasks. However, DNN models are often considered "black boxes," whose predictions are difficult to interpret and understand. An immediate consequence is that quantifying the contribution of individual features is a challenging fundamental task in NLP and explainable AI research. In most related work, whether implicitly or explicitly, an explanation's role in NLP text classification is to reveal which words and phrases are the most salient for the final prediction (Bastings and Filippova, 2020). From this viewpoint, a popular approach for explaining a prediction is to use attribution methods, which justify the prediction of a pre-trained deep network, i.e., the explanation approximates the feature attribution $w.r.t.$ the predicted class. However, attribution techniques, also included in the class of methods called post-hoc methods, might not always provide explanations that are faithful to the underlined model because of the instability of the explanations. This is largely because they often rely on heuristic rules; how those rules mimic the predictive calculation of the black box has the limitation of correlation with expected model behavior (Rudin, 2018). For instance, (Alvarez-Melis and Jaakkola, 2018) showed that the explanations of two very close prediction points varied significantly in a simulated setting. We know that faithful explanations are essential, especially in high-stakes domains. If the explanations are wrong, we cannot trust the black box model. In addition, explanations are supposed to be faithful to what the model actually computes, so they may not meet the end user's expectations. We define a faithful explanation in the context of NLP as follows: an explanation method is *faithful* if it is capable of identifying the most salient/meaningful features used by the model to make a prediction. The way humans arrive at a decision can be different from a black-box model. This limitation makes it difficult to enforce the idea that an explanation must follow the user's expectation, e.g., as suggested by human annotation, which can be completely different from the predictive model behavior.

In a nutshell, our goal is to uncover faithful feature attributions from deep networks, thus to reveal, as accurately as possible, the most influential features used by the network to make a prediction using a bottom-up approach. To do so, we need to focus on learning representations to support feature attribution. So we optimize a deep network model for both faithful attribution and high prediction accuracy. As a result, we construct a model that can learn meaningful representations to explain class predictions *without* using post-hoc methods. Our guided model is based on learning an activation vector for each class. This vector is intended to capture the salient features for each class. Finally,

Proceedings of the 29th International Conference on Computational Linguistics, pages 994–1005
October 12–17, 2022.

the activation vector is used to explain the model's prediction. Our contributions are as follows: (1) We propose a method to learn feature attribution concurrently while training a black-box, in order to faithfully explain the black-box; (2) Our method achieves better explanation and is capable of identifying the most salient words; (3) Our method shows that it does not trade off interpretability against classification accuracy; (4) We also propose a method that can be used for hypothesis testing and measuring importance of phrases.

2 Related work

Existing work on interpreting predictive models tackles the problem from the following five directions.

Propagation-based methods This line of work relies on a back-propagation algorithm to compute the gradient of the output of the model's prediction with respect to the input vector. The result is then used to construct a saliency map, which masks irrelevant features from the input (Simonyan et al., 2013; Denil et al., 2014). (Bach et al., 2015) proposed ϵ-LRP, which is another technique for feature attribution. It focuses on redistributing the prediction score until the input layer is reached. An improvement on these gradient-based methods was proposed by (Sundararajan et al., 2017). Their approach integrates overall gradients using a linear interpolation between a baseline input (all zero embeddings) and the target input.

Model-agnostic methods Another method for feature attribution is the so-called model-agnostic approach. Local Interpretable Model-agnostic Explanation (LIME) (Ribeiro et al., 2016) is a perturbation-based method for feature attribution. It approximates the information flow of a given black-box in the neighborhood of the input with an interpretable classifier (e.g., a linear classifier) model. One issue with LIME is that it relies on a Gaussian distribution for sampling and ignores the correlation between features. (Lundberg and Lee, 2017) proposed to use Shapley values to quantify the importance of a given feature. They also proposed a sampling strategy, "kernel SHAP" for approximating Shapley values. Both approaches focus on feature attribution, and treat features as independent from one another.

Learning-based attribution methods Another line of work has focused on learning feature attributions. For example, (Chen et al., 2018a) employed mutual information to learn essential features from a classifier. However, this technique assumes access to the output model. As a result, it learns the attribution score from a pre-trained model, while we learn feature attribution concurrently when training a black-box model.

All three of these approaches are post-hoc techniques, and they are not always reliable in providing faithful explanations to the model's prediction because their explanations do not always have any relation with the actual behaviour of the model.

Rationale-based methods In addition to the three aobve noted methods, there are other types of interpretability methods for NLP text classification called rationale-based methods (Lei et al., 2016; Bastings et al., 2019; Bashier et al., 2020). These methods attempt to extract a subset of text features as the "rationale" for an explanation, then feed them to a black-box to make the final prediction. Rationale-based methods rely on using a complex heuristic function to extract rationales from text, and then use another complex (black-box) model to classify the rationales. In our work, we rely only on simple high-dimensional vectors to explain the prediction faithfully without using complex functions to pre-identify constellations of text as rationales.

Disentanglement representations Our work is also different from existing disentanglement representations. For instance, (Higgins et al., 2016) tackled a completely different problem, i.e., learning independent factors in the highly non-linear latent manifold for a given dataset, by using a variation auto encoder. In this paper we focus on building disentanglement representations at the embedding layer for feature attribution. Similarly (John et al., 2019) focused on disentangling the latent space of deep neural networks for text generation, which is again a different objective from our work. (Sha and Lukasiewicz, 2021) employed disentanglement representations instead of adversarial training for style transfer, which is also different from our current work. In a nutshell, we build a disentanglement representation to learn feature attribution concurrently while training the deep neural network classifier.

3 Locally distributed activation vectors

Our focus, like traditional post-hoc methods, is on feature importance. We present our model **L**ocally **D**istributed **A**ctivation **V**ector, which is an effective method for learning distributed-activation-vectors concurrently

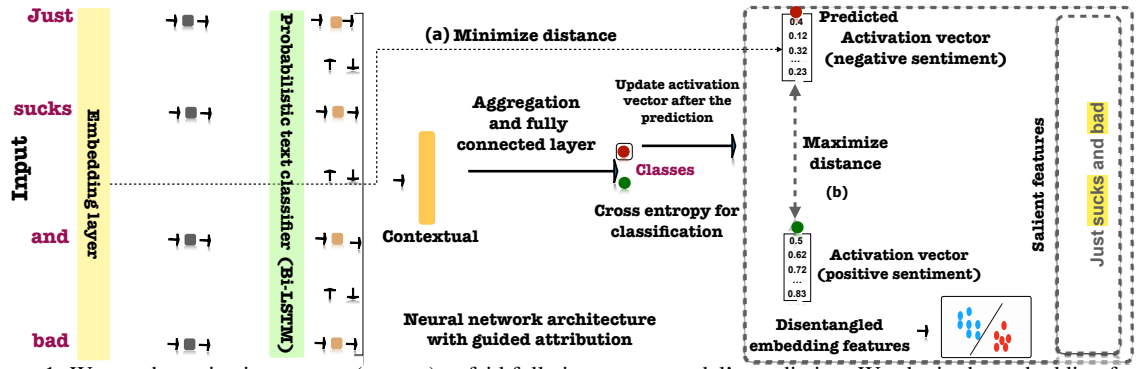


Figure 1: We use the activation vectors (LDAVs) to faithfully interpret a model's prediction. We obtain the embedding features and then feed the result to the deep neural network for classification. During training, we minimize the cosine distance between the activation vector of the predicted class and the corresponding sentence vector (see dotted line (a)). In addition, we also maximize the distance between activation vectors (see (b)).

while training deep neural networks for text classification (i.e., learning a new representation to support feature attribution).

A locally distributed activation vector (LDAV, activation vector) is simply a one-dimensional vector that encodes the knowledge learned by a deep neural network for a text classification task, with a focus on interpreting the predictions (see Figure 1). We can use the activation vectors with any black-box models, including Transformer, GRU, and LSTM methods. Each activation vector records the prediction knowledge of the deep neural network for a particular class. The goal is to alter the optimization problem to learn LDAVs that will be used to interpret model predictions. We can also use an LDAV to conduct hypothesis testing on the role of attributes in any classification, for example *whether blood pressure is a critical factor in predicting kidney disease.*

For notation, we denote scalars with italic lowercase letters (e.g., x), vectors with bold lowercase letters (e.g., $\boldsymbol{x}$), and matrices with bold uppercase letters (e.g., $\boldsymbol{W}$). In the text classification task, an input sequence $\boldsymbol{x}_1, ..., \boldsymbol{x}_l \in \mathbb{R}^d$, where l is the length of the input text and d is the vector dimension, is mapped to a distribution over class labels using a parameterized deep neural network (e.g., BILSTM). The output $\boldsymbol{y}$ is a vector of class probabilities, and the predicted class $\hat{y}$ is a categorical outcome. To faithfully interpret the deep neural network's prediction using relative importance, we rely on information encoded by the LDAV. The model learns k distributed activation vectors $\boldsymbol{z}_j$ ($j = 1, 2, ..., k$), where the prediction knowledge of each $\hat{y}$ is represented using $\boldsymbol{z}_{\hat{y}} \in \mathbb{R}^d$ and k represents the number of classes. During deep neural network training, we concurrently update each $\boldsymbol{z}_j$. Our intuition is that the "locally distributed activation vector" for a given class is trained to emulate the average word embedding of all of the instances that are predicted for that class, while being maximally different from the LDAVs of the other classes. In general, for text classification, we feed $\boldsymbol{x}_1, ..., \boldsymbol{x}_l$ to the representation layer (e,g, a LSTM) to obtain the context vector $\boldsymbol{h}$. The model predicts the label by feeding $\boldsymbol{h}$ to an output layer.

3.1 Objective function

Unlike traditional attribution methods for text classification, our optimization objective now includes new terms for model interpretability. The loss function for the deep neural network is defined as follows:

3.1.1 Cross-entropy

Traditional text classification models employ cross-entropy loss to penalize incorrect classification as:

$$\mathcal{L}_1 = -\frac{1}{k}\sum_{i=1}^{k} \bar{\boldsymbol{y}}_i \log(\boldsymbol{y}_i), \qquad (1)$$

where $\bar{\boldsymbol{y}}$ is the one-hot encoded vector. For example, $\bar{\boldsymbol{y}} = [0, 1, 0]$ indicates that the input belongs to the second class.

3.1.2 Towards faithful interpretations

We use back-propagation to learn the LDAV activation vector $\boldsymbol{z}_{\hat{y}}$ during a model's training. Our goal is to minimize the distance between each feature $\boldsymbol{x}_i$ that triggers the class $\hat{y}$ and the activation vector $\boldsymbol{z}_{\hat{y}}$. As a result, semantically important words will have a short distance to the activation vector and vice versa. To faithfully model distance between $\boldsymbol{x}_i$ and its corresponding $\boldsymbol{z}_{\hat{y}}$, we propose the following hybrid distance approach:

Term 1. This term minimizes the cosine distance between the sentence vector of $\boldsymbol{x}$ and the corresponding $\boldsymbol{z}_{\hat{y}}$, i.e., it minimizes the distance in high dimensional space as follows:

$$\mathcal{L}_2 = \rho_1 \left(1 - \frac{\hat{\boldsymbol{x}} \cdot \boldsymbol{z}_{\hat{y}}}{\|\hat{\boldsymbol{x}}\| \, \|\boldsymbol{z}_{\hat{y}}\|} \right), \tag{2}$$

where $\hat{\boldsymbol{x}}$ is the sentence vector obtained using a pooling operation (i.e., calculating the average of the embedding vectors) of all word vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_l$ and ρ_1 is a weight coefficient. This term attempts to quantify the semantic similarity between the input and corresponding LDAV.

Term 2. We maximize the distance between the activation vectors so that each $\boldsymbol{z}_{\hat{y}}$ has a short distance from words contributing to $\hat{y}$ and a long distance from words contributing to other classes. This ensures that words closer to their corresponding $\boldsymbol{z}_{\hat{y}}$ have a higher importance $w.r.t.$ the predicted class and vice versa. We maximize the pairwise squared distance of $\boldsymbol{z}_1 ... \boldsymbol{z}_k$ (similar to traditional clustering techniques). We denote this loss as $\mathcal{L}_3$, which is the sum over distances. ρ_2 is a weight coefficient.

$$\mathcal{L}_3 = \rho_2 \left(\sum_{i}^{k} \sum_{j}^{k} \left(\boldsymbol{z}_i - \boldsymbol{z}_j \right)^2 \right) \tag{3}$$

Overall, the optimization objective forces the network to learn features where unrelated words are orthogonal and features that have semantic relatedness are co-linear. The final loss is defined as $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 - \mathcal{L}_3$.

3.2 LDAV score

We now have a new representation that we can use to faithfully interpret the classifier's prediction. Our problem is now simpler; we want to quantify the contribution of $\boldsymbol{x}_i$ to the model's prediction $\hat{y}$ using $\boldsymbol{z}_{\hat{y}}$, by calculating the distance between x_i and $\boldsymbol{z}_{\hat{y}}$. This contribution value is called the LDAV score. We initially propose to use a Euclidean measure and calculate the distance as follows:

$$\boldsymbol{\alpha}\left(\boldsymbol{x}_i, \boldsymbol{z}_{\hat{y}}\right) = \sqrt{\sum_{j=1}^{d} \left((\boldsymbol{z}_{\hat{y}})_j - (\boldsymbol{x}_i)_j \right)^2} \tag{4}$$

The LDAV score is calculated as follows:

$$\texttt{LDAV\_score}(\boldsymbol{x}_i, \boldsymbol{z}_{\hat{y}}) = -\left(\frac{\boldsymbol{\alpha}(\boldsymbol{x}_i, \boldsymbol{z}_{\hat{y}}) - \mu}{\sigma} \right), \tag{5}$$

where μ and σ are the mean and standard deviation (std) of $\boldsymbol{\alpha}(\boldsymbol{x}_1, \boldsymbol{z}_{\hat{y}}), \boldsymbol{\alpha}(\boldsymbol{x}_2, \boldsymbol{z}_{\hat{y}}) ..., \boldsymbol{\alpha}(\boldsymbol{x}_l, \boldsymbol{z}_{\hat{y}})$, respectively. LDAV score is the normalized contribution score of x_i on the prediction of $\hat{y}$. A higher score indicates higher word importance. This score explains the contribution of a word $w.r.t.$ the model's prediction. A good feature attribution method would be capable of quantifying the importance of each variable $w.r.t.$ the model prediction. Semantically related features (e.g., 'excellent,' in a positive movie review) will have a short distance from the corresponding LDAV.

4 Experiments and analysis

We focus on the following objectives: 1) ensure explainability does not affect predictive accuracy, and 2) ensure the constrained optimization problem provides faithful feature attribution. A summary of the datasets is shown in Table 1. In the table, Kaggle-CF means Kaggle-consumer-finance data.

Dataset	Train	Test	Voc.	Length	classes
IMDB (Maas et al., 2011)	25000	25000	10000	50	2
Kaggle-CF (Kaggle, 2016)	60125	6681	52943	60	11
DBpedia (Zhang et al., 2015)	63000	5600	50002	32	15
AG news (Zhang et al., 2015)	102080	25520	59706	20	4

Table 1: A summary of the datasets used in evaluation. Voc. means the vocabulary size.

4.1 Implementation specification

We evaluate our approach on two popular architectures, namely Bi-directional Long Short Term Memory with attention mechanism (BILSTM) (Zhou et al., 2016) and a Transformer architecture (Vaswani et al., 2017). The dimension of the embedding vector, LDAV, and the context vector that we used is 128, based on the cross validation results using $\{64, 128, 256\}$. For training the classifiers, we used the Adam optimizer with a learning rate of 0.0001 based on the cross validation from $\{0.000001, 0.00001, 0.0001\}$ and the batch size of 256 from $\{128, 256, 512\}$. We have tried different values for ρ_1 and ρ_2. We train for a maximum of 250 epochs with early stopping if the validation score has not been improved during 10 consecutive epochs. We report the results based on the average of 5 runs. We compare our LDAV method with seven baseline methods: IntGrad (Sundararajan et al., 2017), SHAP (Lundberg

and Lee, 2017), LIME (Ribeiro et al., 2016), Occlusion (Zeiler and Fergus, 2014), ϵ-LRP (Bach et al., 2015), Grad*Input (Denil et al., 2014) and Saliency (Simonyan et al., 2013).

4.2 Interpretability does not affect predictive accuracy

The proposed constrained optimization problem to support a model's explainability does not sacrifice the classification performance of the deep neural networks (DNNs) as shown in Tables 2 and 3. This is because the constrained optimization problem enforces identification of semantic similarity between sentences, which means sentences in a specific category are close to each other in the embedding space and far from sentences in other categories.

	BILSTM		Proposed	
Dataset	**Accuracy**	**F1 score**	**Accuracy**	**F1 score**
AG news	0.88	0.88	0.88	0.88
DBpedia	0.90	0.84	**0.94**	**0.88**
IMDB	0.79	0.79	**0.81**	**0.81**
Kaggle-CF	0.81	0.67	**0.82**	0.67

Table 2: BILSTM performance on four datasets. The BILSTM is from (Zhou et al., 2016)

	Transformer		Proposed	
Dataset	**Accuracy**	**F1 score**	**Accuracy**	**F1 score**
IMDB	0.76	0.76	**0.78**	**0.78**
Kaggle-CF	0.78	0.59	**0.79**	**0.66**
AG news	0.88	0.88	0.88	0.88
DBpedia	0.91	0.85	**0.94**	**0.88**

Table 3: Transformer's performance on four datasets. The Transformer baseline is from (Vaswani et al., 2017)

4.3 Quantitative evaluation

We evaluate the faithfulness of the feature attribution obtained by previous post-hoc approaches and our approach, and then compare performance. We followed the current practice standard evaluation techniques to evaluate the faithfulness. We note that human evaluation might not be the best metric for evaluating the faithfulness $w.r.t.$ the black-box (Jacovi and Goldberg, 2020). For example, human annotation may not correlate with the salient features used by the deep neural network. Further note that a comparison with human annotation is contrary to the ultimate goal of our technique, as we aim to analyze the model's behavior and deficiencies. We adopt the following four metrics from prior work.

4.3.1 Degradation test

This metric evaluates the faithfulness of the salient features used by the model. We measure the local fidelity by incrementally deleting words according to their attribution score for the predicted class. For each test data instance, we mask the top u words (by using a special token `<pad>`) based on the `LDAV` score that measures word attribution. We then observe any change in the model's prediction compared with the original prediction when no words are removed. We use the following equation as a degradation score:

$$\text{degradation-score}(u) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_u^{(i)} = \hat{y}^{(i)}), \quad (6)$$

where m is the total number of test samples, $\hat{y}^{(i)}$ is the predicted label on the i-th test data when no words are masked, and $\hat{y}_u^{(i)}$ is the predicted label when u words are removed. A higher drop indicates the capture of more informative words, which leads to a better explanation for the model's prediction. This metric has also been used in previous work (Nguyen, 2018).

Figures 2 and 3 show the results of degradation scores in different explanation methods, as we increase the number of masked words. We show only the experimental results on BILSTM using the AG news and DBpedia and the results on Transformer using the other two datasets. The figures show that our method captures informative words for the model's prediction better than traditional attribution methods. For instance, in IMDB, we see a steep decline in the curve when removing the top 6% of important words, meaning that the classifier uses a small percentage of words in IMDB to make predictions on sentiment classification. We can also observe that AG news and DBpedia classifiers use a higher percentage of words for prediction, compared to IMDB and Kaggle-consumer-finance, which implies that they employ a larger context to make a prediction. We arrived at the same conclusion for Transformer tested on DBpedia and AG news.

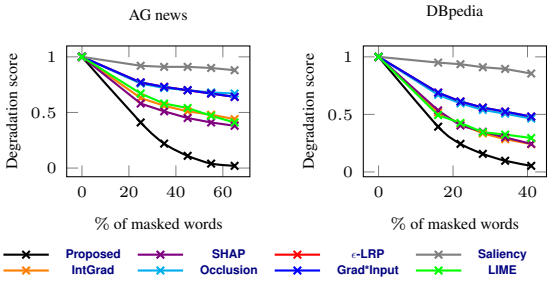


Figure 2: Change of degradation score when words are masked on the BILSTM.

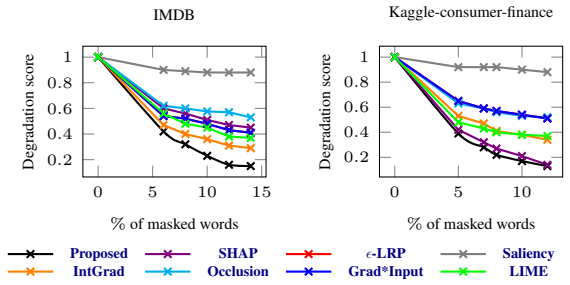


Figure 3: Change of degradation score when words are masked on the Transformer.

4.3.2 Change in log-odds score

In this experiment, we analyze the change in the model's probability of the predicted class when the top u words are masked. Lower log-odds indicate that the masked words are more important in the model prediction. This metric is also used in some previous models' interpretation (Chen et al., 2018b). The log-odds score is defined as follows:

$$\text{Log-odds}(u) = \frac{1}{m}\sum_{i=1}^{m} \log\left(\frac{p(\hat{y}|\boldsymbol{x}_u)_i}{p(\hat{y}|\boldsymbol{x})_i}\right), \quad (7)$$

where $p(\hat{y}|\boldsymbol{x})_i$ is the probability of the predicted class when no tokens are deleted in the test sample i, and $p(\hat{y}|\boldsymbol{x}_u)_i$ is the probability of the predicted class when u words are deleted in the test sample i. Results are shown in Figure 4 and 5.

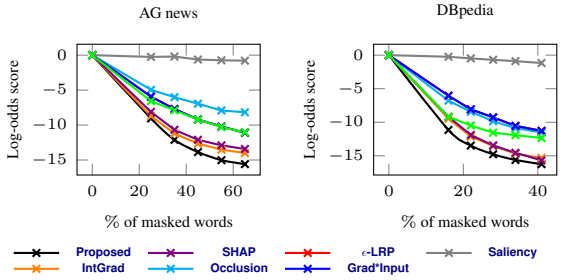


Figure 4: Change of log-odds score when words are masked on the BILSTM.

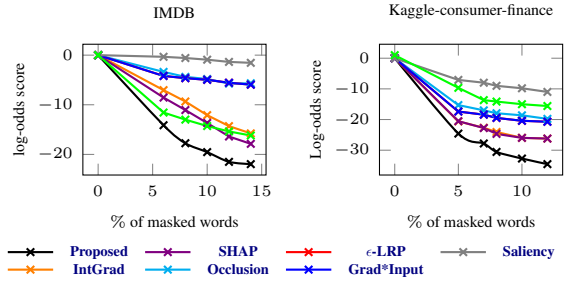


Figure 5: Change of log-odds score when words are masked on the Transformer.

4.3.3 Switching point

The switching point test evaluates the sufficiency of salient words to conform with the model prediction. Words will be masked according to their importance score, e.g., first x_1, second x_2, ..., and last x_n, where x_1 is the word with the highest importance for the predicted class based on the LDAV score and x_n is the word with the lowest importance. For each test, we measure the number of words that need to be deleted before the prediction switches to another class (the switching point), normalized by the number of words in the input, as proposed by (Nguyen, 2018). Our model (LDAV) employs fewer words for classification on DBpedia and AG news with the BILSTM architecture, and on IMDB and Kaggle data with the Transformer (see Table 4). This means our approach performs better than post-hoc methods in capturing salient features.

Transformer			BILSTM		
Method	IMDB	Kaggle	Method	AG news	DBpedia
IntGrad	0.17	0.12	IntGrad	0.13	0.27
SHAP	0.26	0.08	SHAP	0.13	0.27
Occlusion	0.24	0.2	Occlusion	0.19	0.38
e-LRP	0.23	0.18	e-LRP	0.23	0.41
Grad*Input	0.23	0.18	Grad*Input	0.23	0.41
LIME	0.21	0.18	LIME	0.21	0.26
Saliency	0.41	0.42	Saliency	0.72	0.64
LDAV	**0.11**	**0.06**	**LDAV**	**0.12**	**0.24**

Table 4: The % of words that needs to be deleted to change the classifier's prediction. (e.g. 0.11 means 11%.)

4.3.4 Comprehensiveness

Here we use another alternative metric to evaluate our approach, called ERASER (DeYoung et al., 2019). This metric can be also used to evaluate faithfulness of the explanation. It measures the degree to which the words in the explanation influence the prediction. It provides two different terms for faithfulness: comprehensiveness and sufficiency. Due to page limits, we report only the comprehensiveness result here. The comprehensiveness evaluates if all tokens needed to make a prediction are selected. Let f_θ denote a deep network using LDAVs and parameterized by θ. A new input is created $\tilde{\boldsymbol{x}}$ such that $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{r}$, where $\boldsymbol{r}$ is the salient words selected based on the LDAV score. Let $f_\theta(\boldsymbol{x})_j$ be the prediction probability of our model on the input $\boldsymbol{x}$ for class j. The comprehensiveness is calculated as $f_\theta(\boldsymbol{x})_j - f_\theta(\tilde{\boldsymbol{x}})_j$. A higher score implies that the removed words are more influential in the prediction.

Table 5 shows the results of comprehensiveness in term of Area Over the Perturbation

Transformer			BILSTM		
Method	IMDB	Kaggle	Method	AG news	DBpedia
IntGrad	0.122	0.008	IntGrad	0.014	0.009
SHAP	0.146	0.01	SHAP	0.011	0.01
Occlusion	0.065	0.01	Occlusion	0.009	0.005
e-LRP	0.081	0.005	e-LRP	0.012	0.005
Grad*Input	0.081	0.005	Grad*Input	0.012	0.005
LIME	0.113	0.007	LIME	0.012	0.028
Saliency	0.008	0.001	Saliency	0.001	0.001
LDAV	**0.151**	**0.011**	**LDAV**	**0.0179**	**0.02**

Table 5: Comprehensiveness scores of different explanation techniques with the Transformer and BILSTM in terms of AOPC.

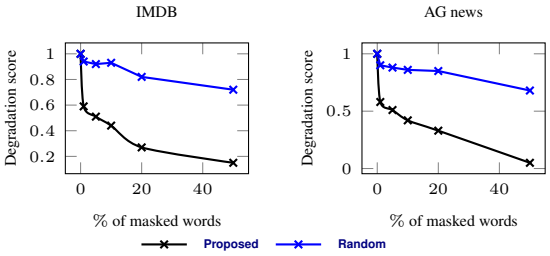


Figure 6: Degradation score on the RoBERTa model.

Curve (AOPC) scores of different attribution techniques. The comprehensiveness was calculated at different percentages, $10\%, 13\%, 16\%, 20\%, 23\%$ for (IMDB, Kaggle consumer finance) and $15\%, 21\%, 28\%, 34\%, 40\%$ for (DBpedia, AG news), and the AOPC is reported. Since DBpedia and AG news employed a larger context in prediction, we used higher percentages for these two datasets. LDAV outperforms the traditional feature attribution techniques, achieving the highest scores in comprehensiveness.

4.4 LDAV for pre-trained transformers

We also show that LDAVs can be used with pre-trained language transformer models. We evaluate the effectiveness of LDAVs on two datasets: IMDB and AG news, when a pre-trained model is used. We use the RoBERTa encoder (Liu et al., 2019), which is a robustly optimized version of BERT. We incorporate LDAVs into the RoBERTa encoder and make the optimization trainable in an end-to-end fashion by modifying the objective function to learn LDAVs along with the classification task. The hidden layer is fine-tuned for the downstream classification task. The model was trained on an NVIDIA GeForce RTX 3070 8 GB GDDR6. We used two metrics here, degradation score and comprehensiveness (using different percentages $1\%, 5\%, 10\%, 20\%, 50\%$). The results in Table 6 and Figure 6 show that our method captures the influential features used by the model in the pre-trained transformer. For instance, we showed that removing $\sim 4\%$ of the words can significantly affect the predictive power of the model.

	Random	Proposed		Random	Proposed
IMDB	0.011	0.047	**AG news**	0.021	0.036

Table 6: Comprehensiveness in terms of AOPC on RoBERTa.

4.5 Natural language inference

We also evaluate our approach on a structured classification task, i.e., natural language inference (NLI). Given a premise sentence $\boldsymbol{x}^{(p)}$ and a hypothesis sentence $\boldsymbol{x}^{(h)}$, the objective is to predict their relation $\hat{y}$, which can be one of the following: $\{$neutral, contradiction, entailment$\}$. We use the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) for model training. The dataset consists of $408,579$ samples for training and $9,824$ for testing. For building the predictive model, we use the Decomposable Attention network (DA) (Parikh et al., 2016).

DA+LDAV Similar to other tasks, we create an LDAV for each of the three classes. During the training, we update the deep network following our proposed method. Because we have two inputs (premise, hypothesis), Equation 2 will be modified to consider information from both sentences when learning the LDAVs. We first compute the premise sentence vector $\hat{\boldsymbol{x}}^{(p)}$ for $\boldsymbol{x}^{(p)}$, and the hypothesis sentence vector $\hat{\boldsymbol{x}}^{(h)}$ for $\boldsymbol{x}^{(h)}$.

Inspired by the idea of (Conneau et al., 2017), to extract relations between $\hat{\boldsymbol{x}}^{(p)}$ and $\hat{\boldsymbol{x}}^{(h)}$, we use the element-wise product. We compute the element-wise product $\bar{\boldsymbol{x}}^{(p,h)} = \hat{\boldsymbol{x}}^{(p)} * \hat{\boldsymbol{x}}^{(h)}$ and minimize the cosine distance between $\bar{\boldsymbol{x}}^{(p,h)}$ and the corresponding LDAV vector.

We use element-wise product to encode the interaction between the premise and hypothesis sentences which capture information from both. In general, it can catch similarities or discrepancies. The performance of the DA predictor with LDAV was relatively similar to the original DA achieving an accuracy of $\sim 84\%$. To calculate the attribution score of each word in premise and hypothesis, we first predict the relation and then use the corresponding LDAV of the predicted class. For instance, to compute the attribution score for the token $\boldsymbol{x}_0^{(p)}$ using Equation 5: (1) We find a new vector for the token defined as $\bar{\boldsymbol{x}}_0^{(p)} = \boldsymbol{x}_0^{(p)} * \hat{\boldsymbol{x}}^{(h)}$ so that we could estimate the attribution score given the hypothesis sentence, (2) we apply Equation 5 using $\bar{\boldsymbol{x}}_0^{(p)}$. We use the same approach for the hypothesis tokens.

Result. Results shown in Figure 7 in terms of degradation score and log-odds demonstrate the effectiveness of our approach in more structured/complex tasks such as NLI. LDAV outperforms traditional post-hoc explanation methods by faithfully finding the most salient features used by the model to predict the relation. Similar to previous experiments, we have also used the comprehensiveness metric on the DA network using different percentages ($10\%, 20\%, 25\%, 30\%, 35\%$) in Table 7, and showed that our proposed method has best captured the salient features.

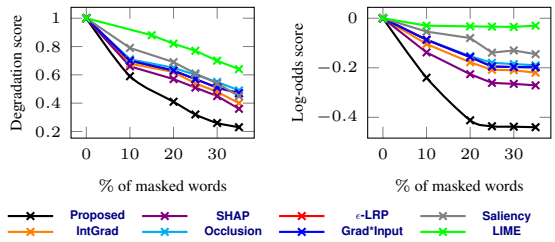


Figure 7: Change of degradation score and log-odds when words are masked on the DA network. (SNLI dataset). Lower values are better.

Method	AOPC	Method	AOPC	Metho	AOPC	Method	AOPC
IntGrad	0.136	Grad*Input	0.13	SHAP	0.19	Saliency	0.09
Occlusion	0.13	LIME	0.02	ϵ-LRP	0.13	**LDAV**	**0.34**

Table 7: Comprehensiveness in terms of AOPC. For each method, AOPC is used to evaluate the features identified to be supportive of predicted relation (positive evidence).

4.6 Ablation study

Loss terms. We conducted an ablation study to understand the impact of each loss term on model interpretability. This experiment identifies the minimum number of words required to switch the prediction to another class (similar to the experiment in Table 4). However, here we remove one term from the optimization objective, and then evaluate the effectiveness of our method in explaining the prediction. Results shown in Table 8 demonstrate the effectiveness of the proposed loss terms. The values in Table 8 denotes the minimum percentage of words required to be removed from the input so that the prediction changes to another class. For instance, 0.63 means we need to remove 63% of the words in the input to switch the prediction. However, when we use the proposed LDAV method, we will only need to remove 23% of the input.

Loss	Deletion	Loss	Deletion	Loss	Deletion
Remove L_2	0.63	**Remove** L_3	0.66	**No Removal (LDAV)**	**0.23**

Table 8: Ablation study for the proposed loss terms.

4.7 Qualitative results

Instead of visualizing salient words for qualitative analysis, we take a different approach by testing the hypothesis. For example, consider a binary classifier for kidney disease identification. A doctor can be interested in understanding whether or not `low blood pressure` or the combination `low blood pressure+heart disease` has a high correlation with kidney disease. This kind of analysis allows users to test different sets of hypotheses when using a model. This solution supports the consideration of evaluating any combination of features without feeding it to the classifier. Note that a feature can be a single word or a phrase.

To test a hypothesis, we only require the corresponding LDAV and the embedding vectors of the string. We can then compute the LDAV score of the string: a higher score with a specific LDAV vector indicates that the features within the string are more salient/discriminative for the model to trigger prediction of that class. In Table 9, we conduct a similar analysis on the AG news dataset trained using a BILSTM. We can see that `sentiment analysis` is correlated with the "business news" class based on the high LDAV score. However, *sentiment classification* is correlated with the "science/tech" class. Another interesting observation is that the model encodes the perspective that *corona virus* is correlated with "business news" and "world news," and the highest contribution goes to the "business news." However, `corona virus infection` is correlated with the "science/tech" class, most probably due to the word `infection`. The LDAV score for phrases is calculated using the mean-pooling of the embedding vectors of all the words.

Sentence	world	sports	business	science/tech
Corona virus	0.66	-1.14	**1.72**	-0.79
Corona virus infection	-0.65	-0.73	0.35	**1.67**
Sentiment classification	-0.58	-1.22	0.36	**1.42**
Sentiment analysis	-0.3	-1.49	**1.06**	-0.79

Table 9: LDAV scores on AG news for hypothesis testing.

4.8 Concept testing

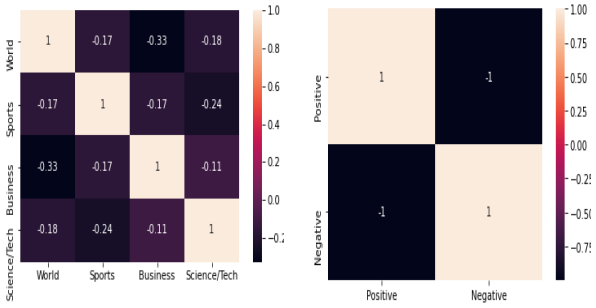
The LDAVs can also help measure whether a neural network model reflects a specific domain or potential bias, i.e., whether the classifier is relying on irrelevant features for making predictions or not. For instance, our model can measure whether the sentiment classifier is using positive lexicon words

as "features" for predicting positive sentiment or not.

Experiment. We apply a mean-strategy for embedding vectors (i.e. calculating the average of the embedding vectors) of the positive sentiment lexicon (Hu and Liu, 2004), in order to get a single concept vector. We construct a concept vector for the negative sentiment lexicon in the same way. We then use the concept vector to calculate the LDAV score $w.r.t.$ each class, using already constructed LDAVs from IMDB. The LDAV of the positive class has the score of -1 $w.r.t.$ the negative sentiment lexicon, while it has the score of 1 $w.r.t.$ the positive sentiment lexicon. Similarly, the LDAV of the negative class has the score of 1 and -1 for the negative sentiment lexicon and the positive lexicon, respectively. The result shows that each constructed LDAV from IMDB captures the positive and negative concepts, respectively.

4.9 How correlated are LDAV vectors?

We have considered whether LDAV vectors are correlated with each other or not. Figure 8 shows the correlation coefficient between LDAVs of classes. All the negative values between different classes imply that each learned vector negatively correlates with others. In conclusion, the model is learning discriminative features that do not correlate or overlap with features from other classes.



(a) AG news. (b) IMDB.

Figure 8: Correlation analysis between LDAVs trained on a BILSTM.

5 Conclusion and future work

We have presented a method to learn locally distributed activation vectors (LDAVs) that can be adapted to faithfully interpret deep network predictions. Our method outperforms traditional post-hoc techniques in revealing the classifier's most discriminative features for a given prediction. It also avoids the often misrepresented trade off between interpretability against classification accuracy. We also showed that LDAV can be used for concept testing and importance measure for phrases. Following this work, we want to extend our approach to other tasks such as Question answering and Name Entity Recognition.

Acknowledgements

We would like to acknowledge the support of the Alberta Machine Intelligence Institute (Amii), and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

David Alvarez-Melis and Tommi S Jaakkola. 2018. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One*, 10(7):e0130140.

Housam Khalifa Bashier, Mi-Young Kim, and Randy Goebel. 2020. Rancc: Rationalizing neural networks via concept clustering. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3214–3224.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of ACL*, pages 2963–2977.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. 2018a. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR.

Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. 2018b. L-shapley and c-shapley: Efficient model interpretation for structured data. *ICLR 2019*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Misha Denil, Alban Demiraj, and Nando De Freitas. 2014. Extraction of salient sentences from labelled documents.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2019. Eraser: A benchmark to evaluate rationalized nlp models. *Computation and Language*.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434.

Kaggle. 2016. Us consumer finance complaints. *Kaggle*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *Proceedings of EMNLP*, pages 107–117.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*, pages 142–150. Association for Computational Linguistics.

Dong Nguyen. 2018. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144. ACM.

Cynthia Rudin. 2018. Please stop explaining black box models for high stakes decisions. *32nd Conference on Neural Information Processing Systems (NIPS 2018), Workshop on Critiquing and Correcting Trends in Machine Learning.*

Lei Sha and Thomas Lukasiewicz. 2021. Multi-type disentanglement without adversarial training. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of International Conference on Machine Learning (ICML)*, page 3319–3328.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pages 818–833. Springer.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.

A Average runtime

We evaluate the computation time of each explanation method on two architectures (BILSTM and Transformer). In Table 10, we compare the average runtime of 500 samples in seconds. SHAP and Occlusion remain expensive compared to other techniques. LDAV achieves the lowest time $\sim 1e^{-4}$ as it only requires feeding the input to the model followed by calculating the LDAV scores. We used Tensorflow running on Ubuntu machine with an Intel Core i7 CPU at 3.60 GHz and Nvidia GPU with 6GB in memory.

Model	Methods	DBpedia	Model	Methods	IMDB
BILSTM	IntGrad	8.8	Transformer	IntGrad	9.0
	Occlusion	191.4		Occlusion	252.7
	SHAP	881.4		SHAP	976.6
	ϵ-LRP	1.3		ϵ-LRP	1.5
	Grad*Input	1.4		Grad*Input	1.7
	Saliency	1.6		Saliecy	1.7
	LIME	0.3		LIME	0.4
	LDAV	**0.0001**		**LDAV**	**0.0002**

Table 10: Average runtime for each input in seconds on two architectures: BILSTM (using DBpedia) and Transformer (using IMDB)

B Analysis of learned representations

To see how well LDAVs capture the semantic difference between classes, we analyze the change of the embedding vectors. In other words, we compare between the embedding vectors without learning LDAVs and the embedding vectors after learning LDAVs. To do so, we perform two experiments: one is to project the average of all word embedding vectors ($\hat{\boldsymbol{x}}$) in each input without learning LDAVs into two dimensions using principal component analysis (PCA). The other is to project $\hat{\boldsymbol{x}}$ after learning LDAVs into two dimensions using PCA. The results of the projections on AG news and IMDB are shown in Figures 9 - 12.

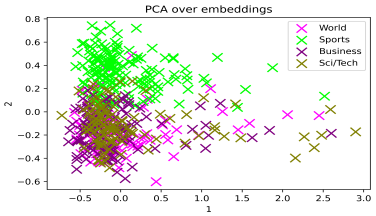


Figure 9: PCA to two dimensions using $\hat{\boldsymbol{x}}$ without employing LDAVs. X-axis and y-axis refer to the principal components (dataset: AG news, Model:BILSTM).

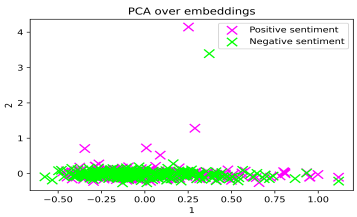


Figure 10: PCA to two dimensions using $\hat{\boldsymbol{x}}$ without employing LDAVs. X-axis and y-axis refer to the principal components (dataset: IMDB, Model:BILSTM).

LDAVs modify the representations of the embedding layer so that they can explain the classifier faithfully. Therefore, we expect the embedding vectors will be changed to better understand the semantic difference between classes while LDAVs are learned. As we can observe in Figures 11 and 12, the embedding vectors of the input texts after LDAVs are learned tend to be clustered collinearly depending on the predicted class. However, in Figures 9 and 10, the embedding vectors have not been clearly clustered when LDAVs are not learned.

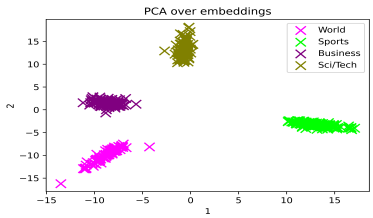


Figure 11: PCA to two dimensions using $\hat{\boldsymbol{x}}$ after employing LDAVs. x-axis and y-axis refer to the principal components (dataset: AG news, Model:BILSTM).

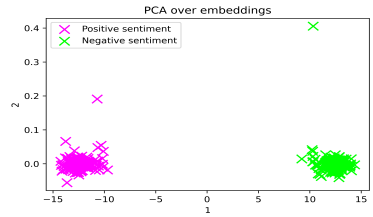


Figure 12: PCA to two dimensions using $\hat{\boldsymbol{x}}$ after employing LDAVs. x-axis and y-axis refer to the principal components (dataset: IMDB, Model:BILSTM).

The intuition here is that the optimization objective with LDAVs (ideally) forces the network to learn embedding representations where inputs of different classes are orthogonal, and inputs belonging to the same class are collinear. As a result, LDAVs can explain the classifier prediction well.

C Performance analysis with respect to distance metric

In this section, we evaluate the effectiveness of using cosine distance over Euclidean distance for

Term 1. We have found that cosine distance works relatively better and it does not sacrifice the performance of the baseline classifier (see Table 11). The performance of the baseline classifier is shown in Table 3 of the main paper. The increased accuracy of cosine distance likely results from inherent normalization during computation and the natural geometric structure it induces (orthogonality and collinearity) on the representations of the embeddings.

	Euclidean distance		**Cosine distance**	
Dataset	**Accuracy**	**F1 score**	**Accuracy**	**F1 score**
IMDB	0.73	0.73	**0.78**	**0.78**
Kaggle-CF	0.73	0.54	**0.79**	**0.66**
AG news	0.86	0.86	0.88	0.88
DBpedia	0.59	0.55	**0.94**	**0.88**

Table 11: Comparing distance metric for Term 1 on the Transformer model.

D Baseline details

Here we describe the baselines used in the evaluation.

Grad*Input is the gradient of the output w.r.t. the input, followed by multiplying the input with the gradient.

Integrated Gradient (IntGrad) calculates a path integral of the model gradient to the input from a non-informative reference point.

Layer-wise relevant propagation ($\epsilon-$LRP) is a layer-wise relevance method, which focuses on redistributing the relevance.

LIME focuses on creating an interpretable classifier by approximating it locally, with a linear model.

SHAP employs game theory to estimate feature attribution.

Saliency uses gradient of the output neuron with respect to the input.

Occlusion employs perturbation techniques to learn feature attribution in a post-hoc approach.

E LDAV score

We found that Euclidean distance in `LDAV` score (Equation 4 in the main paper) works relatively better than cosine distance in approximating feature attribution. The switching points on IMDB using Euclidean and cosine distances are 9% and 15% respectively.