Deep Interpretable Modelling

by

Housam Khalifa Bashier Babiker

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

The recent success of deep neural networks has exposed the problem of model transparency. The need for explainability is particularly critical in sensitive domains. In addition, regulatory frameworks for the "responsible" deployment of AI are emerging, creating legal requirements for transparent, explainable models.

There are many approaches to explainability, including the distinction between top-down methods. Such as adapting existing logical models of explainability to deep learning and bottom-up methods (e.g., augmenting the "semantics-free" networks with fragments of semantic components). However, there is the challenge of how a deep network can learn multi-level representations or create explanation support on demand when requested. Here we describe our development and experiments with building interpretable deep neural networks for Natural Language Processing (NLP). We focus on learning interpretable representations to generate reliable explanations that give users a deeper understanding of the model's behavior. These representations offer feature attribution, contrastive, and hierarchical explanations. We also show the effectiveness of our approach to model distillation and rationale extraction.

Preface

Chapters 3 to 6 have been published as conference papers. I was responsible for model development, conducting experiments, writing and editing the papers. My co-authors were responsible for providing relevant data and reviewing the manuscript drafts.

- Chapter 3 is published as Housam Babiker, Mi-Young Kim, and Randy Goebel. Locally Distributed Activation Vectors for Guided Feature Attribution. In the 29th International Conference on Computational Linguistics (COLING) 2022.

- Chapter 4 is published as Housam Babiker, Mi-young Kim, and Randy Goebel. RANCC: Rationalizing Neural Networks via Concept Clustering. The 28 International Conference on Computational Linguistics (COLING) 2020.

- Chapter 5 is published as Housam Babiker, Mi-Young Kim, and Randy Goebel. Neural Networks with Feature Attribution and Contrastive Explanations. In European Conference on Machine Learning and Data Mining (ECMLPKDD) 2022.

- Chapter 6 is published as Housam Babiker, Mi-Young Kim, and Randy Goebel. DISK-CSV: Distilling Interpretable Semantic Knowledge via Class Semantic Vector. The 16th conference of the European Chapter of the Association for Computational Linguistics (EACL) 2021.

- Chapter 7 is submitted to the conference of the European Chapter of the Association for Computational Linguistics (Under review).

Other publications

- Housam Babiker, Randy Goebel, and Irene Cheng. Facial expression using SVM classifier on salient mic-macro patterns. International Conference on Image Processing (ICIP) 2017.

- Mi-Young Kim, Shahin Atakishiyev, Housam Babiker, Nawshad Farruque, Randy Goebel, Osmar R. Zaiane, Mohammad-Hossein Motallebi, Juliano Rabelo, Talat Syed, Hengshuai Yao, and Peter Chun. A Multi-Component Framework for the Analysis and Design of Explainable Artificial Intelligence. Machine Learning and Knowledge Extraction, Special Issue on Advances in Explainable Artificial Intelligence - MDPI, 3, pp900-921, 2021.

Acknowledgments

I would like to thank my supervisor Randy Goebel for his support and guidance through my Ph.D. endeavor. I am lucky to have him as my supervisor. During the past five years, I have learned so many things from him, which helped me in different aspects of my life. Without his kind supervision and patience, I would not be able to complete this research dissertation, and for that, I am grateful. I would like to thank Mi-Young Kim for the time we spent working together. I would also like to thank my supervisory committee Nilanjan Ray, Levi Levas, and my examining committee for their constructive feedback.

I would like to express my honest appreciation to my family, my mother, father, brother, sister, and wife have always supported me with unconditional love, especially during difficult times.

Finally, to my grandmother and aunt (Nafisa and Fatima), who would have loved seeing me moving forward. May they rest in peace.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The lack of eXplainable AI (XAI) tools to examine complex machine learning models makes them less trustable, especially in high-stakes decisions. For example, a medical diagnosis model is responsible for human life. One might be skeptical of the predictions or not trust its results. Such predictive models are considered black-box because they cannot explain their predictions in terms "humans" can understand, nor can they explain the overall behavior of the decision-making process.

To demonstrate the importance of interpretations, researchers at Vanderbilt University developed a tool to identify cases of colon cancer from patients' electronic records [1]. Although the model performed well at first, the researchers eventually discovered it was making predictions based on the fact that patients with confirmed cancer cases were sent from a particular hospital rather than clues from their health records. The example shows the importance of XAI systems when deploying machine learning models. Another example is from [2], they observed that a neural network was focusing on the word "portable" within an x-ray image rather than the medical information, which showed that the model was not using the appropriate reasoning approach to analyze x-ray images.

Some nations are taking the lead to avoid such a potentially harmful outcome. One such attempt is with EU General Data Protection Regulation (GDPR) introduced in

2018, which requires justification for algorithmic decisions [3].

1.2 The Deep Learning Era

Recent advances in machine learning focus on building sophisticated neural network models known as Deep Learning. The primary appeal of deep learning is that a predictive model can be constructed automatically from a suitable volume of labeled inputs. In an increasing number of demonstration applications, the staging of a deep learning exercise needs only to outline the details of the supervised learning problem in terms of input data and leave the creation of the predictive classifier to the deep learning system. The fundamental improvement of current deep learning methods is that, unlike earlier, more shallow network layers, deep learning automatically identifies the appropriate stratification of a predictive model [4]. Therefore, finding appropriate multi-layer structures of a supervised classification problem has produced significant advances in AI systems.

Because many components of Artificial Intelligence systems include classification tasks, it is easy to imagine that the construction of an accurate predictive model is essential to overall intelligent systems. It is relatively easy to confirm whether a classifier performs well when classifiers are well-defined and straightforward categories, e.g., classifying hand-written digits. Nevertheless, there are more complex classification problems, e.g., classifying complex proteins and their potential docking targets into potentially active pairings. So it is difficult to determine how a deeply learned classifier reasons, mainly when predicting unexpected pairs. Deep networks represent their data input by transforming it into complex weighted networks that are difficult for humans to comprehend or debug. These methods tend to create black boxes because they represent data derived from annotated samples in high dimensional complex spaces [5].

Deep learning has achieved impressive performance in various classification tasks in vision and natural language processing (NLP) applications. However, this im-

proved performance comes at the cost of what has been called "explainability," which broadly indicates an inability to debug or explain classification outputs. There are many situations where domain "semantics-free" learning of predictive models fails or creates nonsensical predictions. So, despite tackling challenging tasks beyond human capabilities, deep networks come at the cost of model understanding.

1.3 Explaining Black-box Models

Explanations of black-box models not only help end-users understand the predictions of these sophisticated models. In addition, some nations impose legal requirements on machine learning algorithms (e.g., deep networks) to provide interpretations for each prediction (e.g., GDPR). Our work focuses on filling the gaps by building interpretable deep networks.

1.3.1 What is an Explanation?

An explanation is the interpretable description of the model prediction on a specific instance in vocabularies that end-users can understand. An explanation focuses on finding the evidence from the input to justify the black-box prediction. The terms "interpretation" and "explanation" are generally used interchangeably in the literature on NLP interpretability.

1.3.2 Generating Explanations from a Black Box

There are two approaches to generating explanations for a black box: 1) modifying or complementing an existing architecture to support interpretation and 2)using a post-hoc approach to generate explanations for a pre-trained model.

1.3.3 Types of Explanations

Existing deep networks enable a limited level of abstraction because of how they encode information from datasets. As a result, explanations can vary depending

on the task. For example, in computer vision research, saliency (e.g., a heat map) highlights the most discriminative pixels a network uses for a single prediction. In NLP, the task is more challenging; however, most work focuses on feature attribution. Feature attribution is defined as the scoring (or ranking) function that maps portions of the input to scores that communicate an aspect of importance about the prediction. Feature attributions aim to convey which parts of the input to a model decision are essential, responsible, or influential to the decision. Other techniques focus on extracting a rationale (a subset of text extracted from the input) to justify the model's prediction.

1.4 Post-hoc Explanations

Post-hoc means one can create a method to explain each prediction of a pre-trained deep model. Post-hoc relies on prior knowledge of a deep network behavior. For instance, Local Interpretable Model-agnostic Explanations (LIME) [6] claims that it can explain any model (including deep networks), regardless of its underlining architecture. LIME is one of the prominent explanation frameworks in current literature. An issue that has consistently appeared in the post-hoc explanation of deep networks is the faithfulness of interpretation, i.e., how to provide explanations that accurately represent the true reasoning behind the model's final decision. Interpretations that are untrustworthy or unfaithful are useless because they do not help humans make decisions. In this dissertation, the faithfulness of an explanation depends on how well the generated explanation approximates the prediction of the black-box model.

1.5 Inherently Interpretable Models

Another direction for solving the challenge of interpretation focuses on constructing inherently interpretable models. For example, building deep models that can learn interpretable representations and are also capable of providing faithful interpretations

[7].

1.6 Thesis Statement

In this dissertation, we focus on addressing the following problems (see Figure 1.1).

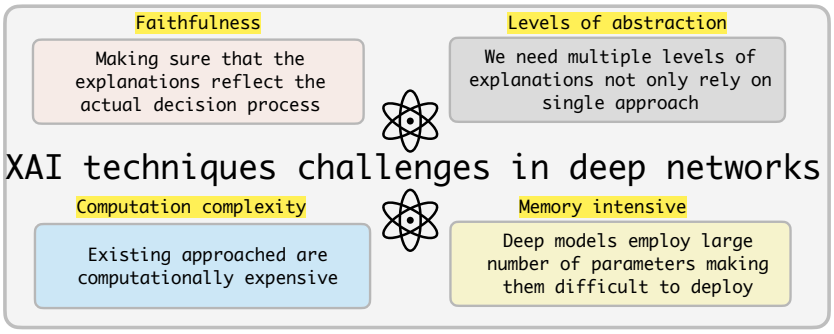


Figure 1.1: The four challenges we address in this manuscript.

1.6.1 Computational Complexity

Some of the existing post-hoc approaches are computationally inefficient, e.g., the method proposed by [8] takes up to one hour to produce a result (heat-map or feature attribution for a single instance). Similarly, LIME [6] is computationally expensive. It involves training an interpretable model to explain a single instance. Training such a model is resource-intensive. Computational cost is a serious problem, especially when making predictions in the high stakes domain, for instance, using clinical notes to predict hospital readmission with discharge summaries.

1.6.2 The Problem of Faithfulness

Post-hoc methods do not provide faithful explanations of the underlining model, as they attempt to mimic the behavior of a pre-trained model. If the explanation is wrong, then we cannot trust the explainer. If we cannot trust the explainer, we cannot trust the black-box model. The question remains, how faithful is the interpretation of the underlying decision-making model? Employing post-hoc methods to explain deep learning models might result in a misleading justification of the prediction due

to the prior assumption about the model's behavior. A helpful study on why post-hoc methods do not provide faithful explanations can be found in [5]. The faithfulness weakness suggests the need for building interpretable deep learning models. We need new representations to support the model's explainability. Therefore, we optimize a neural network classifier for meaningful explanations and high predictive accuracy (constrained optimization problem).

1.6.3 Limited Levels of Abstractions

Existing approaches to deep learning explainability are limited to providing feature importance only as an explanation. However, sometimes feature importance might fail. For example, consider a sentiment classifier that predicts the sentiment of a product review from a given text. The attribution of "bad" and "very bad" to classifying a review as (negative sentiment) must be completely different. In addition, "very bad" is semantically stronger than "bad." Most existing techniques cannot create such an explanation, nor can they improve abstraction. These weaknesses suggest the need for multiple levels of abstraction [9].

1.6.4 Memory Intensive

Neural networks tend to be deep, with millions of parameters. For example, GPT-2 [10] needs over 1.5 billion parameters. As a result, they are computationally intensive, making it difficult to deploy in real-world applications. Therefore, there is a need to learn interpretable models that are computationally efficient.

1.7 Contributions

Motivated by these points, we propose multiple methods to address each limitation and leverage strengths from a different perspective.

- First, we propose **L**ocally **D**istributed **A**ctivation vectors (`LDAV`). `LDAV` allows the building of interpretable deep networks without changing their underlining

architecture. The LDAV vectors are learned concurrently with neural networks. LDAV explains the form of feature attribution (e.g., provide feature importance to each token $w.r.t$ predicted class). In addition, LDAV vectors work with pre-trained Transformers to facilitate explainability.

- Second, we introduce **R**ationalizing **Neural** **N**etworks via **C**oncept **C**lustering (RANCC). RANCC explains its predictions by extracting a rationale from the input sentence. A rationale is the subset of the text's interpretable features (words). Unlike traditional post-hoc explanations, RANCC is interpretable. It provides explanations along with predictions. RANCC can work with many representation layers for predictive models.

- Third, we also extend the idea of LDAV and present a new approach for enabling deep networks to provide contrastive explanations.

- Fourth, we introduce a self-distillation approach (an extension of the LDAV). We present a new technique to concurrently learn a lightweight, interpretable version of the black box.

- Fifth and finally, we present an approach to generate hierarchical explanations from neural networks without introducing additional parameters.

1.8 Thesis Outline

In Chapter 2, we provide an overview of the fundamental concepts of XAI. For instance, we explore the importance of XAI and the difference between interpretability and explainability. We also discuss the trade-off between explainability and accuracy. At the end of Chapter 2, we review the popular related works for both post-hoc and interpretable models and the existing evaluation metrics. In Chapter 3, we provide motivation and discuss the limitation of existing related work on feature importance for deep learning models. Then, we introduce LDAV as an effective solution

for building interpretable models. We test the usefulness of LDAV on traditional deep learning models and pre-trained transformers. In Chapter 4, we propose RANCC for constructing interpretable neural networks. Chapter 5 extends LDAV to contrastive explanations. We discuss the proposed technique, present the new evaluation metrics and demonstrate the proposed method's effectiveness. In Chapter 6, we extend LDAV and introduce a new approach for self-distillation. We also present extensive experiments to demonstrate the effectiveness of using the idea of LDAV for distillation and model interpretability. In Chapter 7, we introduce SFFA, a new interpretable model based on the proprieties of the deep model, and show its effectiveness in generating hierarchical explanations. Chapter 8 summarizes the dissertation's contributions and discusses the current limitations and future works.

Chapter 2

From Black-boxes to Intrinsic Models

2.1 Background

Research on making deep learning models more interpretable and explainable is receiving much attention. One of the main reasons is the application of deep learning models to high-stake domains. In general, interpretability is an essential component for deploying deep learning models. For instance, XAI can be used to address a variety of problems: (i) the detection of biased views in a deep learning model, (ii) the evaluation of the fairness of a deep learning model, (iii) faithfully explaining the predictions of the classifier, i.e., the construction of accurate explanation that explains the underlying causal phenomena [11] and (iv) the use of explanations as a proxy for model debugging, which allows researchers/engineers to construct models better or debug existing models. On the other hand, the European Union introduced the General Data Protection Regulation (GDPR), focusing on the privacy of users' data. The importance of the GDPR arises from the surge of failures of these models. Similarly, the Defense Advanced Research Projects Agency (DARPA) announced a new initiative called Explainable Artificial Intelligence (DARPA XAI) in 2016 [12], focusing on approximating explanations to end-users. In this chapter, we first provide a sketch of the big picture of the field and then dive more into details as the central area of interest. We also touch on the existing evaluation metrics for XAI.

2.1.1 Definitions in XAI

Explainability and interpretability are sometimes used interchangeably in the literature [13]:

- Biran and Cotton [14]: Define interpretability as the degree to which end-users understand the underlining factors used by the model to make a prediction.
- Miller [9]: Assumes that interpretability and explainability are equal.
- Lipton [11]: Defines interpretable models within two groups: transparent models, which to some degree are comprehensible to the user, and post-hoc interpretable models, which are systems that try to explain a black-box model using alternative approaches.
- Doshi-velez and Kim [13]: Define interpretability as the ability to explain/present in understandable terms to humans.

On the other hand, Rudin [5] suggests that models that provide their own explanations can be classified as interpretable models. Hence, in this thesis, we follow Rudin's definition and build inherently interpretable deep models for NLP.

2.1.2 The Trade-off Between Interpretability and Accuracy

According to DARPA, there is a trade-off between accuracy and interpretability [12]. This assumption implies an inverse correlation exists between the accuracy and interpretability of classifiers—for instance, a high accuracy indicates less meaningful explanations and vice versa. Rudin [5] disagrees with the trade-off; there are no standard units to measure such a difference, and neither is there any standard in data science application. As a result, the claim is not universally valid and not always the case, as we will show in the following chapters.

2.2 Key Metrics

Evaluating the quality of an explanation is a challenging task. Unfortunately, we cannot rely on humans to assess the generated explanations from a pre-trained neural model. In general, interpretations are supposed to be faithful to what the model computes. We define a faithful explanation in the context of NLP as follows: an explanation method is *faithful* if it can identify the discriminative features used by the model. In the followings, we discuss some of the standard metrics used in our experiments:

Degradation test evaluates the faithfulness of the features used by the model for text classification. The degradation test measures 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 unique token `<pad>`) based on the attribution score. Then, we observe the change in the model's prediction compared with the original prediction when no words are removed. The degradation score can be defined as follows:

$$\texttt{Degradation-score}(u) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_u^{(i)} = \hat{y}^{(i)}), \tag{2.1}$$

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 [15].

Change in log-odds score calculates the change in the model's probability of the predicted class when the top u words are masked. Lower log odds indicate that masked features are more critical in model prediction. This metric is also used in

some previous models' interpretation [16]. The log-odds score is defined as follows:

$$\texttt{Log-odds}(u) = \frac{1}{m}\sum_{i=1}^{m}\log(\frac{p(\hat{y}|\boldsymbol{x}_u)_i}{p(\hat{y}|\boldsymbol{x})_i}), \tag{2.2}$$

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 features are deleted in the test sample i.

Switching point evaluates the sufficiency of salient tokens to conform with the model prediction. We mask features in the order of their importance, e.g., first x_1, second $x_2, ...$, and last x_n, where x_1 is the token with the highest importance for the predicted class and x_n is the word with the lowest score. Finally, for each test, we measure the number of words that need to be deleted before the prediction switches to another class, normalized by the number of words in the input [15].

ERASER is another approach to measure the faithfulness of the explanations and proposed by [17]. Deoung et al. [17] propose the following metrics:

Comprehensiveness measures whether all required model features to make a prediction are selected by the attribution method. For example, given $\boldsymbol{x}$, the new text input is defined as $\hat{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{z}$, where $\boldsymbol{z}$ is the set of relevant tokens identified as salient within the text. Let $f_\theta(\boldsymbol{x})_{\hat{y}}$ be the network's output for class $\hat{y}$, the comprehensiveness score is defined as follows:

$$\texttt{Comprehensiveness} = f_\theta(\boldsymbol{x})_{\hat{y}} - f_\theta(\hat{\boldsymbol{x}})_{\hat{y}} \tag{2.3}$$

A higher score implies that the identified feature tokens in $\boldsymbol{z}$ were more influential in the model's predictions, compared with other words.

Sufficiency evaluates whether the identified salient tokens have enough information to trigger the model to predict the same label as using the full text:

$$\texttt{Sufficiency} = f_\theta(\boldsymbol{x})_{\hat{y}} - f_\theta(\boldsymbol{z})_{\hat{y}} \tag{2.4}$$

A lower sufficiency score implies that the explanations are more adequate for a model's prediction. Please note that, comprehensiveness and sufficiency metrics are similar to ROAR [18], but do not require re-training.

Cohesion Score The cohesion-score [19] measures the synergy of words within a text span to the model prediction by shuffling the words to see the probability change on the predicted label. Given a salient span $\boldsymbol{x}_{(a,b]}$, we randomly select a position in the token sequence $\boldsymbol{x}_1, ..., \boldsymbol{x}_a, \boldsymbol{x}_{b+1}, ..., \boldsymbol{x}_m$ and re-insert a word. The process is repeated until a shuffled version of the original sentence $\boldsymbol{x}^{(q)}$ is constructed. Intuitively, the words in an important text span have strong interactions. By perturbing such interactions, we expect to observe the output probability decreasing. The cohesion score is defined as follows:

$$\texttt{cohesion} = \frac{1}{a}\sum_{i=1}^{a}\frac{1}{100}\sum_{q=1}^{100}(p(\boldsymbol{x}_i|\hat{y}) - p(\boldsymbol{x}_i^{(q)}|\hat{y})), \tag{2.5}$$

where $\boldsymbol{x}_i^{(q)}$ is the q^{th} perturbed version of $\boldsymbol{x}_i$. We repeat the experiment 100 times. Only salient spans are considered in this evaluation. Higher scores are better, which means the identified spans are more critical than others for predicting the label.

2.3 Related work

We dive more into model explainability as the main focus of our work. We divide existing interpretation methods into five categories: feature attribution, rationale-based, hierarchical explanations, contrastive explanations, and counterfactual explanations. Please note that many interpretable and transparent models exist in the literature, such as decision trees and rule-based systems. However, this thesis focuses on deep neural networks applied to NLP text classification tasks.

2.3.1 Feature Attribution

We define *feature attribution* as the scoring (or ranking) function that maps portions of the input to scores that communicate some aspect of their importance in making a

prediction. In other words, feature attributions aim to convey which parts of the input are important, responsible, or influential to the decision. Most related work has focused on identifying attribution using post-hoc approaches such as back-propagation, model-agnostic, and learning-based attribution techniques. In the following, we discuss each method in detail.

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 the input vector. The result is then used to construct a saliency map, which masks irrelevant features from the input [20, 21]. For example, Arras et al. [22] applied the layer-wise relevance propagation algorithm to explain the predictions of a complex non-linear deep model using feature attribution as shown in Figure 2.1.

Back-propagation methods generally rely on the partial derivatives of the input $w.r.t$ to the output. We find the attribution scores by multiplying the embedding with the gradient [23–27]. Another related approach uses partial derivatives to integrate over a linear interpolation path [28], where users define the path. One can also redistribute the prediction score from the output layer to the input layer, layer by layer, using relevance propagation [29]. Other approaches have relied on the computation of an L2 norm to obtain contribution scores [30].

Model Agnostic

Another method for feature attribution is the so-called model-agnostic approach. One of the popular methods for model interpretation is called LIME [6]. LIME approximates the information flow of a given deep network in the input neighborhood with an interpretable classifier (e.g., a linear classifier). It provides an interpretation in the form of feature attribution (see Figure 2.2).

LIME trains a classifier on randomly drawn samples with gold labels to provide

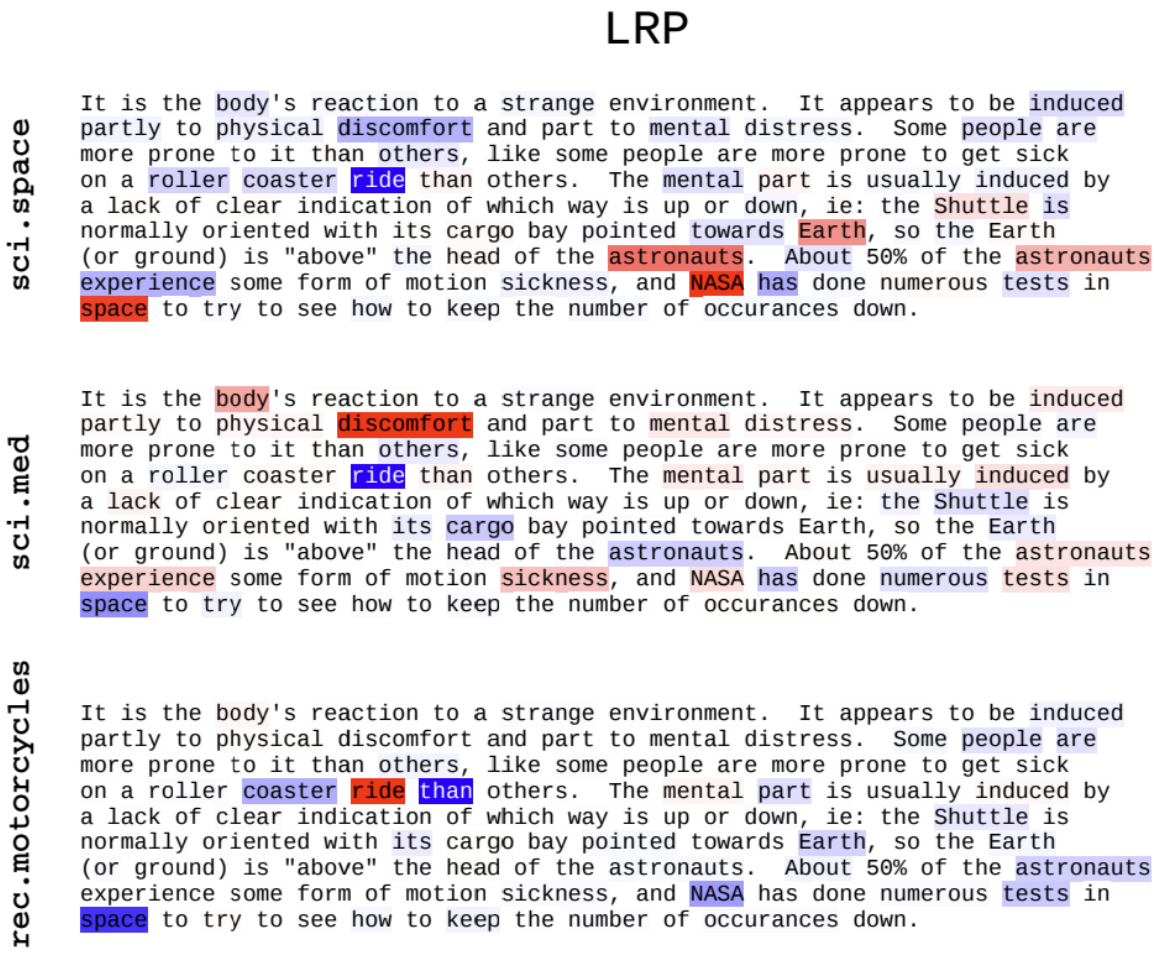


Figure 2.1: Heatmaps for input documents obtained from a deep neural network using layer-wise relevance propagation. Positive contribution are mapped to red and negative to blue. Figure from [22]

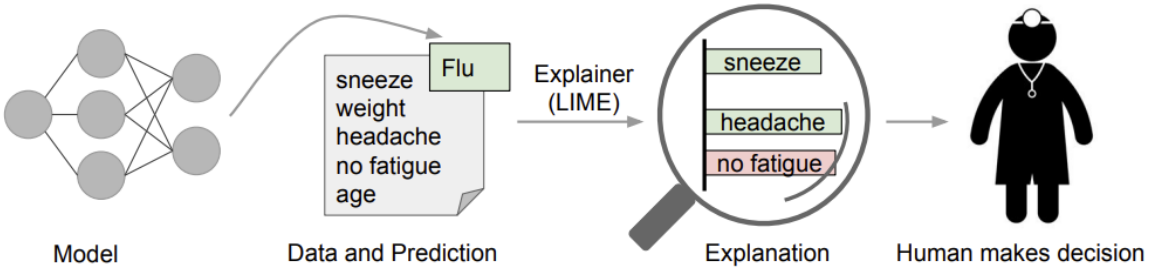


Figure 2.2: Feature attribution: Explaining the classifier's prediction using the LIME algorithm. From the figure, the model predicts that a patient has the flu and LIME highlights the most discriminative features (e.g., symptoms) from the patient's history. The highlighted features serve as the key factors used by the model to predict flu. Figure from [6].

feature attribution. However, one issue is that LIME relies on a Gaussian distribution for sampling and ignores the correlation between features. On the other hand,

Lundberg et al. [31] proposed to use Shapley values to quantify the importance of a given token. Lundberg et al. also presented faster approaches to Shapley values, "kernel SHAP, TreeSHAP" for approximating Shapley values. Both approaches provide feature attribution. Other methods focused on perturbation-based techniques to approximate feature attribution [32].

Learning-based Attribution Methods

Another line of work has focused on building models to learn feature attributions. For example, [33] employed mutual information to learn essential features from a classifier, as shown in Table 2.1. However, it assumes access to the output model. As a result, it learns the attribution score from a pre-trained model. In contrast, our model learns feature attribution concurrently when training a black-box model.

Truth	Predicted	Key sentence
positive	positive	There are few really hilarious films about science fiction but this one will knock your sox off. The lead Martians Jack Nicholson take-off is side-splitting. The plot has a very clever twist that has be seen to be enjoyed. This is a movie with heart and excellent acting by all. Make some popcorn and have a great evening.
negative	negative	You get 5 writers together, have each write a different story with a different genre, and then you try to make one movie out of it. Its action, its adventure, its sci-fi, its western, its a mess. Sorry, but this movie absolutely stinks. 4.5 is giving it an awefully high rating. That said, its movies like this that make me think I could write movies, and I can barely write.
negative	positive	This movie is not the same as the 1954 version with Judy garland and James mason, and that is a shame because the 1954 version is, in my opinion, much better. I am not denying Barbra Streisand's talent at all. She is a good actress and brilliant singer. I am not acquainted with Kris Kristofferson's other work and therefore I can't pass judgment on it. However, this movie leaves much to be desired. It is paced slowly, it has gratuitous nudity and foul language, and can be very difficult to sit through. However, I am not a big fan of rock music, so its only natural that I would like the judy garland version better. See the 1976 film with Barbra and Kris, and judge for yourself.
positive	negative	The first time you see the second renaissance it may look boring. Look at it at least twice and definitely watch part 2. it will change your view of the matrix. Are the human people the ones who started the war? Is ai a bad thing?

Table 2.1: Learning to explain using feature attribution: The scores are learned via a model in a post-hoc manner. Figure from [33].

The discussed approaches generate feature attribution in a post-hoc approach. They are only sometimes reliable in providing faithful explanations.

2.3.2 Inherently Interpretable Models

One popular approach for building itnerpretable models is based on rational extraction. The model explains its predictions by generating what is called a "Rationale" [7,

34–41]. The cited methods extract a subset of tokens as the rationale, then feed them to a deep network to make a prediction. In general, rationale extraction methods consist of two models: (1) a generator to extract the rationale, and (2) an encoder to make a prediction using the extracted rationale. However, existing rationale-based methods rely on using a complex function to extract rationales from the text. Other methods in literature for building inherently interpretable models focused on adding an interpretation layer on top of any existing NLP deep model. The interpretation layer identifies multiple spans and then aggregates the information using trainable weights [42]. The weights associated with each span provide attribution scores for words and phrases (see Figure 2.3). Other ideas focused on concurrently learning feature attributions and concepts while building the deep network for text classification [43]. Please note that interpretable models are sometimes quoted as self-explainable models.

2.3.3 Hierarchical Explanations

Previously discussed methods provide interpretation in the form of feature attribution. Recently, XAI research focused on identifying feature attribution using hierarchical explanations. There has been much recent work on developing hierarchical explanations in the context of NLP. Hierarchical explanations (see Figure 2.4) are essential because traditional feature attribution techniques do not capture the interaction between the tokens.

For instance, Chen et al. [19] proposed a post-hoc approach to generate a hierarchical explanation for text classification. The proposed method employs Shapley values for feature attribution and interaction using a bottom-up approach. Similarly, Singh et al. [45] use contextual decomposition scores to aggregate features based on their interactions. Note that hierarchical structure is intended to provide representation support for explanation at multiple levels of detail (cf. [46]). Unlike our technique, [45]'s work is post-hoc and suffers from finding faithful explanations.

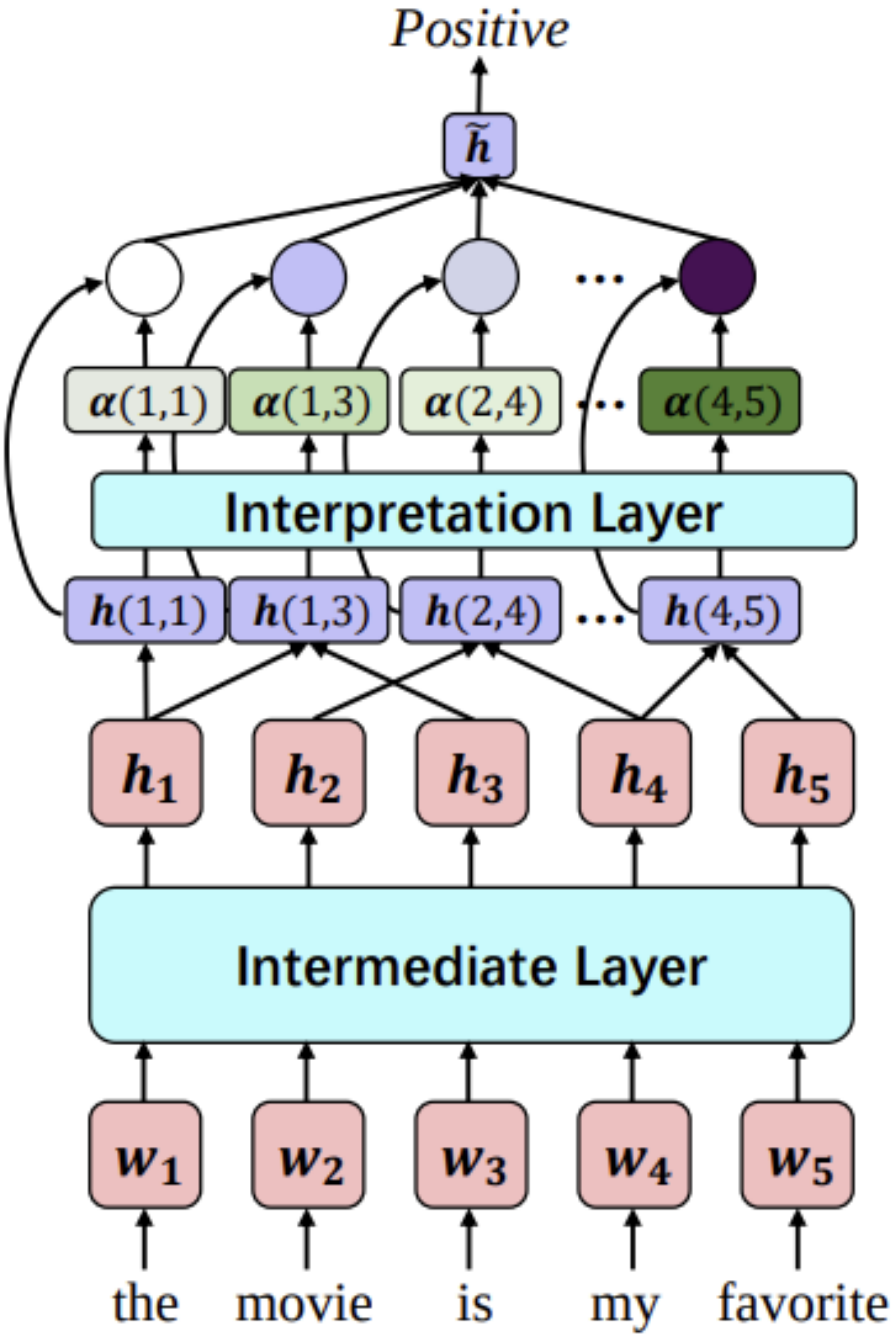


Figure 2.3: A self-explainable deep network which provides feature attribution. Figure from [42].

Other methods for hierarchical explanation use the back-propagation approach; for example, [47] extends the integrated gradient method to feature interaction. However, back-propagation methods suffer from noisy gradients. Other ideas for hierarchical explanation employ a decision tree [48] to generate hierarchical explanations in a

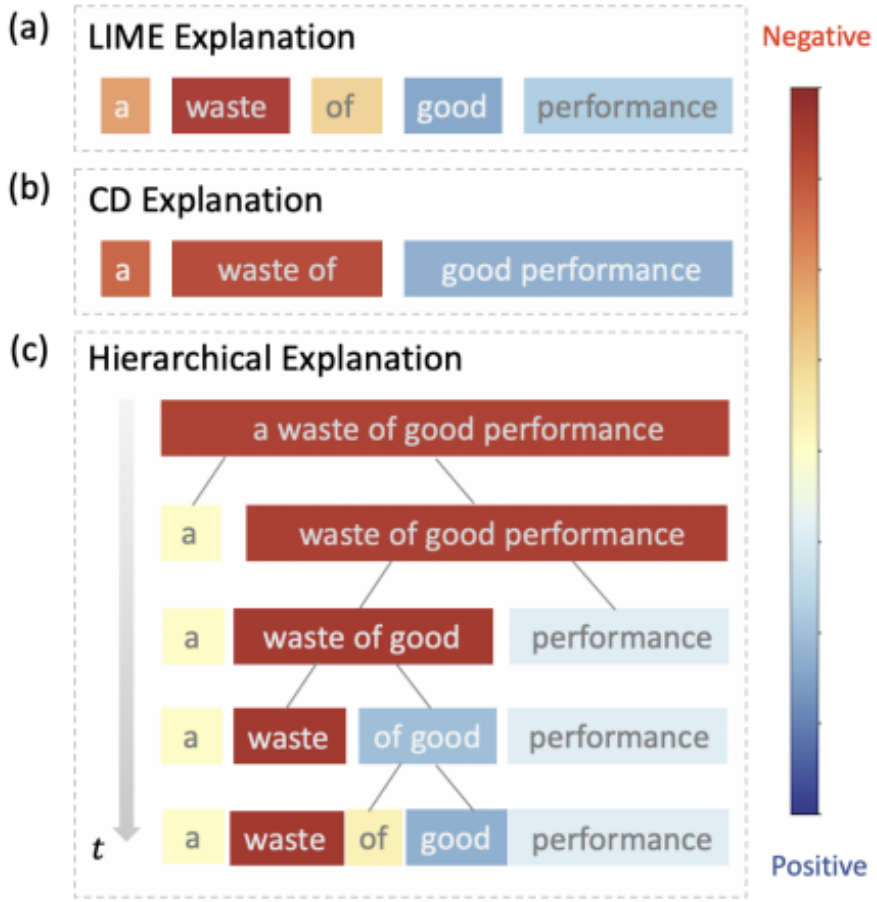


Figure 2.4: Comparing hierarchical explanations with feature attribution. As we can see from the Figure, LIME(a) and CD [44](B) cannot identify the interaction i.e., they only provide word-level and phrase-level explanations. In general, from hierarchical explanations, we obtain a set of features in each time-step. Figure from [19]

post-hoc approach. Some other methods focus on applying a post-hoc approach to learn about interactions inside Transformers [49], which is a different objective from ours. We focus on constructing self-explainable Transformers for text classification.

2.3.4 Contrastive Explanations

Studies in philosophy and social science show that humans, in general, prefer contrastive explanations, i.e., the explanation of an event is based on explaining the fact (p) in contrast to another event (q) [9, 50, 51]. Here "p" represents the model prediction, and "q" represents an alternative class we would use for a contrastive explanation. In general, contrastive explanations provide an answer to "why p and

not q?" questions (see Figure 2.5.) The majority of existing post-hoc techniques are only limited to providing answers to "*why p?*" and cannot provide answers to "*why p and not q?*"—for instance, gradient-based methods. Contrastive explanations are relatively new in NLP [52]. Our work focuses on building an inherently interpretable model that can support answers to both kinds of questions: "*why p?,*" and "*why p, not q?..*"

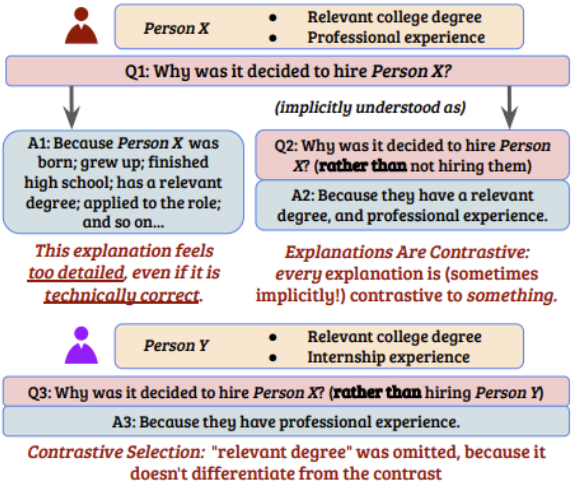


Figure 2.5: An example of a contrastive explanation obtained from [52].

Jacovi et al. [52] proposed a post-hoc approach that relies on a projection matrix to devise explanations. Similarly, [53] used SHAP to generate a contrastive explanation. Our approach is different; we propose an intrinsic neural model which supports answers to "*why p?*" and "*why p and not q?*" questions rather than relying on post-hoc approaches. In the context of contrastive explanations, we focus on finding the difference in the attributes that could distinguish the prediction "*p*" from the foil "*q.*" Other ideas for hierarchical explanation employ a decision tree [48] to generate hierarchical explanation in a post-hoc approach. Some other methods focus on applying a post-hoc approach to learn about interactions inside Transformers [49], which is a different objective from ours. We focus on constructing self-explainable Transformers for text classification.

2.3.5 Counterfactual explanations

Counterfactual explanations consist in generating text as a counterfactual example. In general, counterfactual explanations seek to identify a minimal change in model data that "flips" a predictive model's prediction, which is used for explanation. Watcher et al. [54] introduced the concept of unconditional counterfactual explanations. For text classification, [55] proposed a method to generate counterfactual text from a pre-trained model for the finance domain. In addition, Hendricks et al. [56] proposed a technique to find evidence for the target class but not present in the foil class to learn a model to generate counterfactual explanations for why a model predicts class "p" instead of "q." However, their approach was mainly designed for computer vision.

Chapter 3

Locally Distributed Activation Vectors for Guided Feature Attribution

3.1 Introduction

Deep neural network (DNN) models have become crucial tools in NLP and define state-of-the-art on various tasks. However, DNN 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 [57]. Our goal here is to uncover faithful feature attributions from deep networks, thus revealing, as accurately as possible, the most influential features used by the model to make a prediction. Moreover, unlike traditional XAI methods, we optimize a deep network model for faithful attribution and high prediction accuracy (see Figure 3.1). As a result, we design an inherently interpretable model for learning an attribution-specific representation called an "activation vector" for each label. Each activation vector focuses on identifying the salient features used by the model for a specific class. The main contributions are as follows: (1) We propose a method to identify feature attribution concurrently while training a black-box to ex-

plain the predictions faithfully; and (2) Our method outperforms traditional post-hoc techniques.

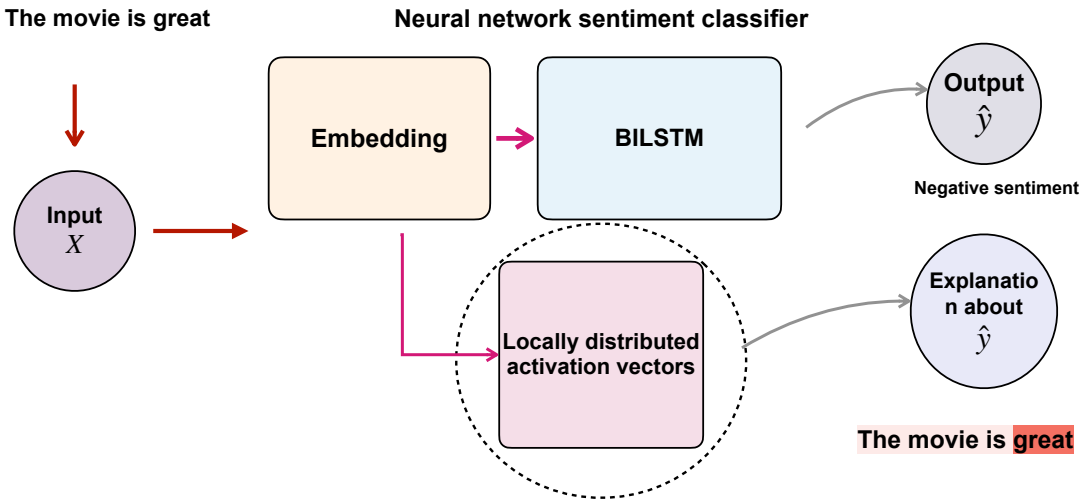


Figure 3.1: Example of our proposed model. We modify the network's architecture to support interpretation. Our approach allows deep neural networks to provide explanations in the form of feature attribution.

3.2 Proposed Method

We call our method **Locally Distributed Activation Vector**. Here, we explain the ideas and methods used to construct deep interpretable networks. A locally distributed activation vector (LDAV, activation vector) encodes the knowledge learned by a deep network for a text classification task, with a focus on interpreting the predictions (see Figure 7.1). 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.* Moreover, the activation vectors work with most of deep learning architecture, including Transformer, GRU, and LSTM methods. The activation vectors only require access to the embedding layer and the output layer. Also, the LDAV vectors require training the entire neural network and they do not work with pre-trained models. For notation, we denote scalars with

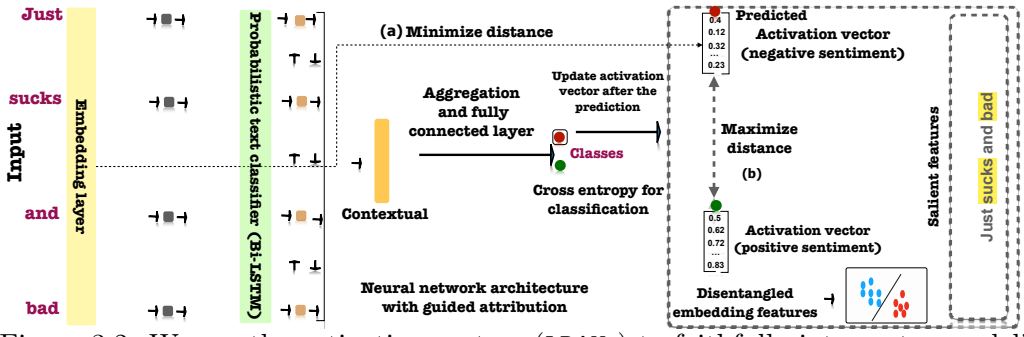


Figure 3.2: We use the activation vectors (LDAVs) to faithfully interpret a model's prediction. We edit the structure of the neural network to learn intermediate representations which are then used as a proxy to generate feature attribution. We obtain the embedding features and then feed the result to the 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)). Please note that, LDAV vectors are constructed simultaneously during the training of the entire neural network.

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 neural network (e.g., a BILSTM). In general, for text classification, we feed $\boldsymbol{x}_1, ..., \boldsymbol{x}_l$ to the network to obtain the context vector $\boldsymbol{h}$. The model predicts the label by feeding $\boldsymbol{h}$ to an output layer. The output $\boldsymbol{y}$ is a vector of class probabilities, and the predicted class $\hat{y}$ is a categorical outcome. To faithfully interpret the neural network's prediction using relative importance, we rely on information encoded by the LDAV. Moreover, the deep model learns k distributed activation vectors $\boldsymbol{z}_j$ $(j = 1, 2, ..., k)$, where the knowledge learned by the network to predict $\hat{y}$ is encoded using $\boldsymbol{z}_{\hat{y}} \in \mathbb{R}^d$ and k represents the number of classes. We concurrently update each $\boldsymbol{z}_j$ during neural network training. Our intuition is that the "locally distributed activation vector" for a given class is trained to emulate the average word embedding of all the instances predicted for that class while being maximally different from the LDAVs of the other classes.

3.2.1 Objective Function

Here, we discuss the steps for altering the optimization problem of the text classifier for faithful interpretation. Unlike traditional attribution methods for text classification, our optimization objective now includes new terms for interpretability. The loss function for the deep network is defined as follows:

Cross-entropy

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

$$\mathcal{L}_1 = -\frac{1}{k}\sum_{i=1}^{k} \bar{\boldsymbol{y}}_i \log(\boldsymbol{y}_i), \tag{3.1}$$

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

Towards Faithful Interpretations

We use back-propagation to learn the `LDAV` activation vector $\boldsymbol{z}_{\hat{y}}$. During training the network minimizes the distance between each feature $\boldsymbol{x}_i$ that triggers the class $\hat{y}$ and the activation vector $\boldsymbol{z}_{\hat{y}}$. So, the distance between semantically salient tokens and the activation vector is short. To faithfully model distance between $\boldsymbol{x}_i$ and its corresponding $\boldsymbol{z}_{\hat{y}}$, we propose the following hybrid distance approach:

Term 1. 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:

$$\mathcal{L}_2 = \rho_1 \left(1 - \frac{\hat{\boldsymbol{x}} \bullet \boldsymbol{z}_{\hat{y}}}{\|\hat{\boldsymbol{x}}\| \, \|\boldsymbol{z}_{\hat{y}}\|}\right) \tag{3.2}$$

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.

Term 2. Maximizes the distance between the activation vectors so that the distance between $\boldsymbol{z}_{\hat{y}}$ and words contributing to $\hat{y}$ is minimum. Also, the distance between

$\boldsymbol{z}_{\hat{y}}$ and features of other classes is maximum. Doing so ensures that words closer to their corresponding $\boldsymbol{z}_{\hat{y}}$ have a higher importance *w.r.t.* the predicted class and vice versa. One possible solution is to maximize the pairwise squared distance of $\boldsymbol{z}_1 ... \boldsymbol{z}_k$. We denote the loss as $\mathcal{L}_3$, which is the sum over distances. ρ_2 is a weight coefficient.

$$\mathcal{L}_3 = \rho_2 \left(\sum \left(\sum_i^k \sum_j^k \left(\boldsymbol{z}_i - \boldsymbol{z}_j \right)^2 \right) \right) \tag{3.3}$$

Overall, the proposed 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 :

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 - \mathcal{L}_3 \tag{3.4}$$

3.2.2 `LDAV` Score

The score mainly relies on the Euclidean distance, i.e., how far a given token is from the corresponding `LDAV` vector. Because the range of Euclidean distance is from to positive infinity, and 0 represents identical points, i.e., the smaller the distance between the word in the input and the corresponding `LDAV`, the higher the importance, and larger distances represent less importance. However, because we are interested in feature importance, we convert the distances into scores, i.e., higher scores indicate higher feature importance. Note that "standard score" is defined as the distance between a data point and the mean using standard deviation [58]. In general, Z-scores can be positive or negative and the signs indicates where the score is below or above the mean. Because, it is easy to interpret the values in the standard score, we converted all the distances into feature importance scores using the Z-score formula. Please note that we did not use the inverse of the Euclidean distance because of the range of the values. Given a deep interpretable network trained with `LDAV`s, the objective now is 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 $\boldsymbol{z}_{\hat{y}}$. To calculate the attribution score, we propose to use a Euclidean measure:

$$\boldsymbol{\alpha}\left(\boldsymbol{x}_i, \boldsymbol{z}_{\hat{y}}\right) = \sqrt{\sum_{j=1}^{d} \left(\left(\boldsymbol{z}_{\hat{y}}\right)_j - \left(\boldsymbol{x}_i\right)_j\right)^2} \tag{3.5}$$

The LDAV score is calculated as follows:

$$\texttt{LDAV\_score}(\boldsymbol{x}_i, \boldsymbol{z}_{\hat{y}}) = -\left(\frac{\boldsymbol{\alpha}(\boldsymbol{x}_i, \boldsymbol{z}_{\hat{y}}) - \mu}{\sigma}\right), \tag{3.6}$$

where μ and σ are the mean and standard deviation (std) of $\boldsymbol{\alpha}(\boldsymbol{x}_1, \boldsymbol{z}_{\hat{y}}), ..., \boldsymbol{\alpha}(\boldsymbol{x}_l, \boldsymbol{z}_{\hat{y}})$, respectively. In addition, the LDAV score is the normalized contribution score of $\boldsymbol{x}_i$ on the prediction of $\hat{y}$. A higher LDAV score indicates higher feature importance. We also tried the cosine distance for the attribution score and showed good attribution scores. However, using Euclidean distance showed a little better score. So, we used the Euclidean distance. The reason can be as follows: While the angle between vectors is important for Term 1, the magnitude of a vector is essential for feature attribution.

3.3 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.

3.3.1 Datasets

Here is the list of the dataset used in our experiments:

IMDB reviews were proposed by [59] for sentiment classification from movie reviews. It consists of two classes, i.e., positive and negative sentiments.

AG news was proposed by [60] for researchers to test machine learning models for news classification. It consists of four classes (sports, world, business, and sci/tech).

DBpedia ontology classification dataset proposed by [61] consists of 15 non-overlapping ontology classes.

Kaggle US Consumer Finance Complaints: dataset contains consumers' complaints about financial products and services to companies for response, obtained from consumer finance web-pages [62].

A summary of the datasets is shown in Table 3.1.

Dataset	Train	Test	Vocabulary size	Length	classes
IMDB ([59])	25000	25000	10000	50	2
Kaggle consumer finance ([62])	60125	6681	52943	60	11
DBpedia ([60])	63000	5600	50002	32	15
AG news ([60])	102080	25520	59706	20	4

Table 3.1: A summary of the datasets used in evaluation.

3.3.2 Implementation Specification

The `LDAV` is implemented on two popular architectures, namely Bi-directional Long Short Term Memory with attention mechanism (BILSTM) [63] and a Transformer architecture [64]. The dimension of the embedding vector, `LDAV`, and the context vector is 128, based on the cross-validation using $\{64, 128, 256\}$. We used the Adam optimizer with a learning rate of 0.0001 based on the cross-validation from $\{1e^{-4}, 1e^{-3}, 1e^{-2}\}$ and the batch size of 256 from $\{128, 256, 512\}$. We used 0.9 for both ρ_1 and ρ_2. We have tried different values for ρ_1 and ρ_2 with interval 0.1 between $[0, 1]$. 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 the `LDAV` method with seven baseline methods: IntGrad [65], SHAP [31], LIME [6], Occlusion [66], ϵ-LRP [67], Grad*Input [21] and Saliency [20].

3.3.3 Interpretability Does Not Affect Predictive Accuracy

The proposed constrained optimization problem to support a model's explainability does not sacrifice the performance of the deep neural networks, as shown in Tables 3.2 and 3.3. Moreover, our approach forces the 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 different 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 3.2: BILSTM performance on four datasets. The BILSTM is from [63]

	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 3.3: Transformer's performance on four datasets. The Transformer baseline is from [64]

3.3.4 Quantitative Evaluation

We evaluate the faithfulness of the feature attribution obtained by post-hoc approaches and `LDAV` and compare performance. We followed the current practice standard evaluation techniques to evaluate faithfulness and adopt the metrics introduced in Chapter 2. We note that human evaluation might not be the best metric for evaluating the faithfulness *w.r.t.* the black-box [68]. For example, a human annotation may not correlate with the salient features used by the neural network.

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.

Degradation Test

We measure the local fidelity by incrementally deleting words according to their attribution score for the predicted class. A higher drop indicates the capture of more informative tokens, which leads to a better explanation for the model's prediction.

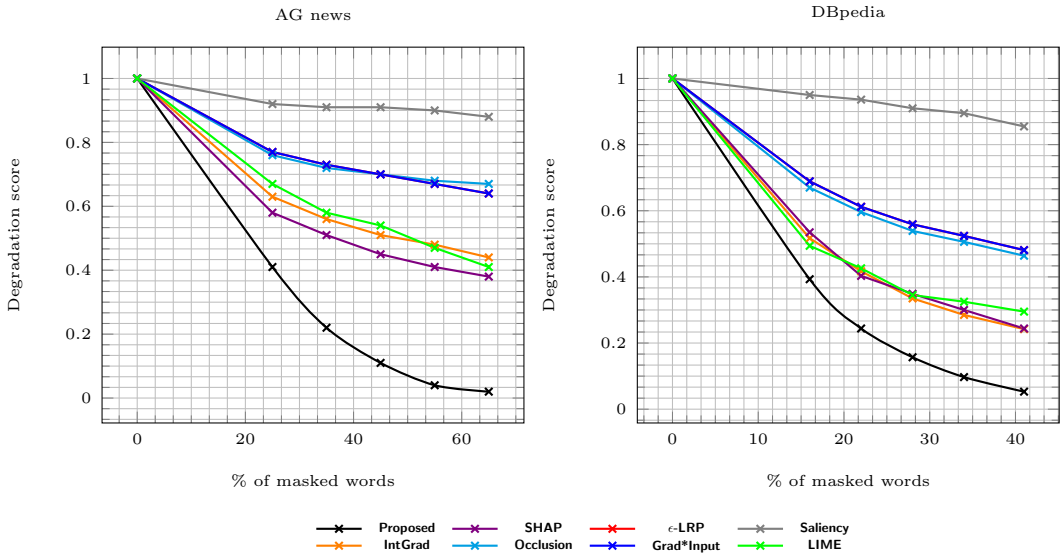


Figure 3.3: Change of degradation score when words are masked on the BILSTM. Lower scores are better.

Figures 3.3 and 3.4 show the results of degradation scores in different explanation methods as a function of masked words. Interestingly, the `LDAV` captures informative

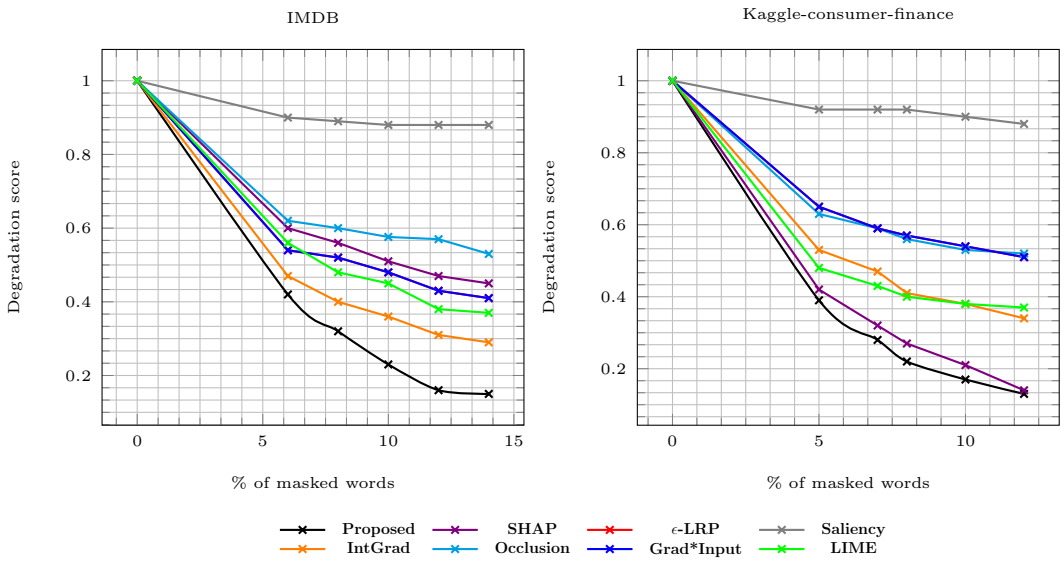


Figure 3.4: Change of degradation score when words are masked on the Transformer model. Lower scores are better.

words for the model's prediction better than traditional attribution methods. For example, IMDB has a steep decline in the curve when removing the top 6% of essential words, meaning that the classifier uses a smaller context. Similarly, the AG news and DBpedia classifiers employ a subset of the features to predict the label. We arrived at the same conclusion for the Transformer tested on DBpedia and AG news.

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 tokens are masked. As discussed in Chapter 2, lower log odds indicate that masked features are more critical in the model prediction. This metric is also used in some previous models' interpretation [16]. Our approach achieves the lowest scores in Figure 3.5 and 3.6.

Switching Point

The switching point test evaluates the sufficiency of salient words to conform with the model prediction. We mask tokens in the order of 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

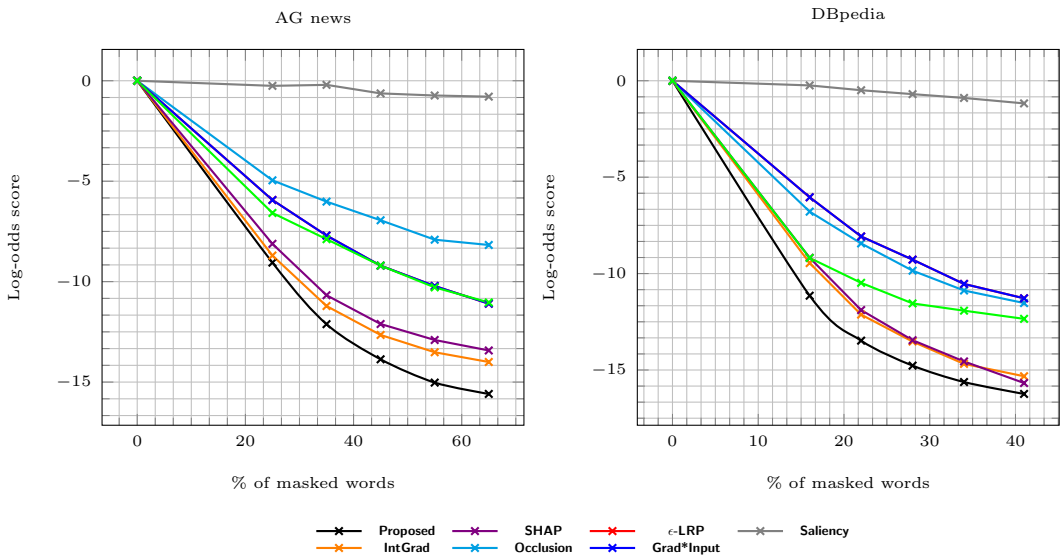


Figure 3.5: Change of log-odds score when words are masked on the BILSTM.

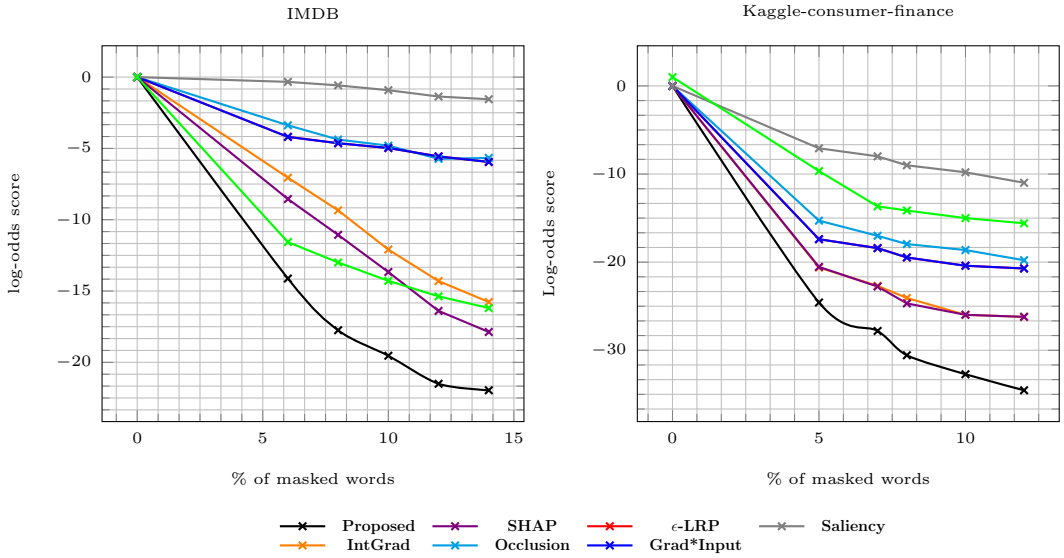


Figure 3.6: Change of log-odds score when words are masked on the Transformer. Lower scores are better.

predicted class based on the LDAV score and x_n is the word with the lowest score. 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 [15]. Our model (LDAV) employs fewer words for classification on DBpedia and AG news with the BILSTM architecture and IMDB and Kaggle data with the Transformer (see Table 3.4). As a result, LDAV identifies more accurately the

ordering of essential words.

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

Eraser

Here we use another alternative metric to evaluate our approach. Moreover, the eraser provides two different terms for faithfulness: comprehensiveness and sufficiency. Here, we use comprehensiveness to evaluate if all tokens needed to make a prediction are selected (see Chapter 2). A higher score implies that the removed words are more influential in the prediction.

Table 3.5 shows the comprehensiveness score in terms of Area Over the Perturbation Curve (AOPC) of different attribution techniques. The comprehensiveness was calculated at various percentages, 10%, 13%, 16%, 20%, 23% (IMDB, Kaggle consumer finance) and 15%, 21%, 28%, 34%, 40% for (DBpedia, AG news), and the AOPC is reported.

3.3.5 LDAV for Pre-trained Transformers

We also show the integration of LDAVs with pre-trained language transformer models. We evaluate the effectiveness of LDAVs on two datasets: IMDB and AG new. We use

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

the RoBERTa encoder [69], a robustly optimized version of BERT. We incorporate `LDAV`s into the RoBERTa encoder and make the optimization trainable in an end-to-end fashion by modifying the objective function to learn `LDAV`s along with the classification task. The model was trained on an NVIDIA GeForce RTX 3070 8 GB GDDR6. We used two metrics, degradation score and comprehensiveness (using different percentages $1\%, 5\%, 10\%, 20\%, 50\%$). The results in Figure 3.7 and Table 3.6 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 model's predictive power.

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

Table 3.6: Comprehensiveness in terms of AOPC on RoBERTa.

3.3.6 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 $\boldsymbol{x}^{(h)}$,

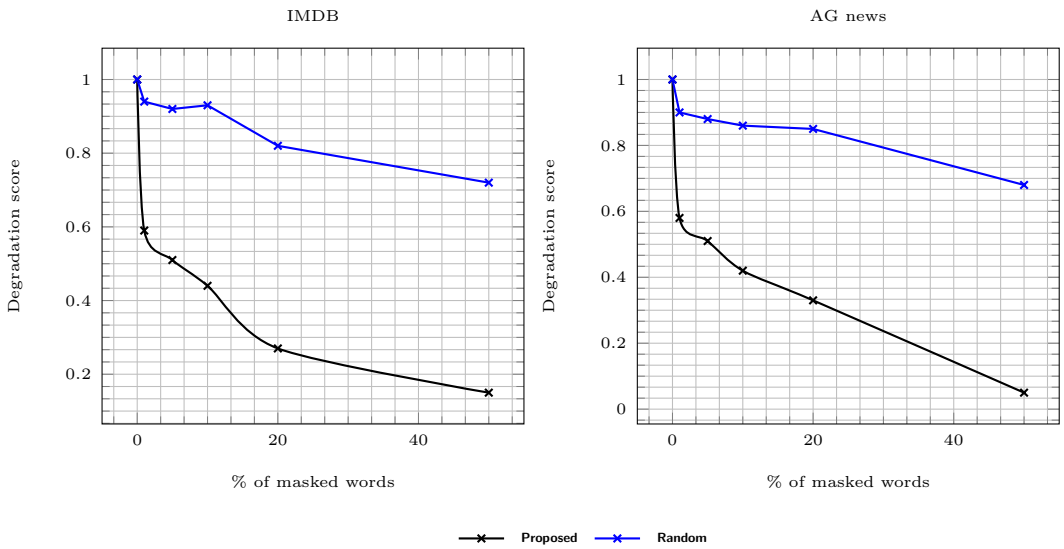


Figure 3.7: Degradation score on the RoBERTa model.Lower scores are better.

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 [70] for model training. The dataset consists of $408,579$ samples for training and $9,824$ for testing. We build the model using the Decomposable Attention network (DA) [71].

Decomposable Attention network with `LDAV` Like other tasks, we create an `LDAV` for each of the three classes. During the training, we updated the neural network following our proposed method. Because we have two inputs (premise, hypothesis), Equation 3.1 will be modified to consider information from both sentences when learning the `LDAV`s. To encode information: 1)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 $\boldsymbol{x}^{(h)}$; 2) inspired by the idea of [72], to extract relations between $\hat{\boldsymbol{x}}^{(p)}$ and $\hat{\boldsymbol{x}}^{(h)}$, we use the element-wise product $\bar{\boldsymbol{x}}^{(p,h)} = \hat{\boldsymbol{x}}^{(p)} * \hat{\boldsymbol{x}}^{(h)}$; 3) minimize the cosine distance between $\bar{\boldsymbol{x}}^{(p,h)}$ and the corresponding `LDAV` vector.

Moreover, the element-wise product encodes the interaction between the premise and hypothesis sentences. In general, multiplication 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%. We first predict the relation to calculate the attribution score of each word in the premise and hypothesis. Then we use the corresponding LDAV of the predicted class. For instance, to compute the attribution score for the token $\boldsymbol{x}_0^{(p)}$ using Equation 3.4: (1) We calculate the token vector representation as $\bar{\boldsymbol{x}}_0^{(p)} = \boldsymbol{x}_0^{(p)} * \hat{\boldsymbol{x}}^{(h)}$ given the hypothesis sentence; (2) calculate the attribution score $\bar{\boldsymbol{x}}_0^{(p)}$. We use the same approach for the hypothesis.

Result . Initial results are shown in Figure 3.8 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 3.7 and showed that our proposed method is better than post-hoc approaches.

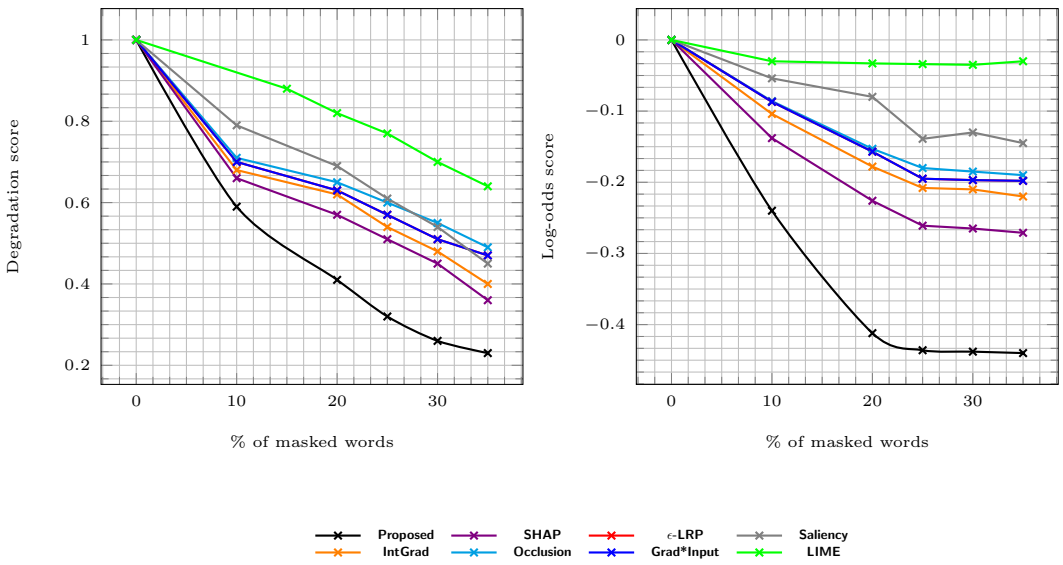


Figure 3.8: Change of degradation score and log-odds when words are masked on the DA network. (SNLI dataset). Lower values are better.

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 3.7: Comprehensiveness in terms of AOPC.

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 3.8: LDAV scores on AG news for hypothesis testing.

3.3.7 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. End-users might 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. Our approach allows evaluating any combination of features without feeding it to the classifier. Note that a feature can be a single word or a phrase. We use mean-pooling to calculate the sentence vector of a phrase. In Table 3.8, we analyze the BILST trained on AG news. From the table, `sentiment analysis` is correlated with the "business news" class based on the high LDAV score. However, *sentiment classification* is associated 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`.

3.3.8 Analyzing Features in Sentiment Classification

The LDAVs can also be used to measure whether the sentiment classifier uses specific phrases that indicate the author's sentiment towards the movie (e.g., "great perfor-

mance", "wonderful story") or not. We used the dataset proposed by [73], which contains a list of salient phrases for each class (e.g., positive and negative).

Experiment we compute a sentence vector for each list. Given the two sentences, we calculate the `LDAV` score using the `LDAV` vectors constructed when training the model on the IMDB dataset. The `LDAV` vector representing the positive sentiment has a score of -1 *w.r.t.* the sentence vector from negative phrases list while it has the score of 1 *w.r.t.* the positive sentiment list. Similarly, the `LDAV` of the negative class has the score of 1 and -1 for the negative sentiment list and the positive list, respectively. The result shows that each constructed `LDAV` from IMDB captures the positive and negative concepts, respectively.

3.3.9 Runtime

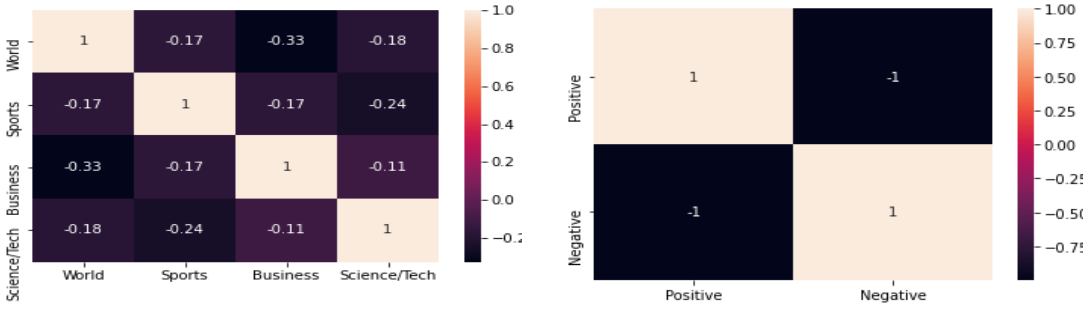
We evaluate the computation time of each explanation method on two architectures (BILSTM and Transformer). In Table 3.9, we compare the average runtime of 500 samples in seconds. SHAP and Occlusion remain expensive compared to other techniques. `LDAV` achieves the lowest time $\sim$ $1e^{-4}$ as it only requires feeding the input to the model followed by calculating the `LDAV` scores. We used TensorFlow running on an Ubuntu machine with an Intel Core i7 CPU at 3.60 GHz and Nvidia GPU with 6GB memory.

3.3.10 How Correlated are `LDAV` Vectors

We found that the `LDAV`s are not correlated. We calculated the correlation coefficient between the `LDAV`s for a deep network trained on IMDB to validate our assumption. In Figure 3.9 we show the correlation coefficient between `LDAV`s of classes. All the negative values imply that each learned vector negatively correlates with others. In conclusion, the model is learning discriminative features that do not relate or overlap with features from other classes.

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



(a) AG news. (b) IMDB.

Figure 3.9: Correlation analysis between LDAVs trained on a BILSTM.

3.3.11 Ablation Study

Loss terms. We conducted an ablation study to understand the impact of each loss term on interpretation. Here, we remove one term from the loss function and evaluate using a switching point metric to assess the quality of the explanations. We found that the proposed terms effectively capture the salient tokens used by the classifier 3.10. Moreover, in Table 3.10, we show the minimum percentage of tokens required to be removed from the input so that the prediction changes to another class. For instance, 0.63 means we must remove about 63% of the tokens to switch the prediction.

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

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

Table 3.10: Ablation study for the proposed loss terms.

words, we compare 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 3.10 - 3.13.

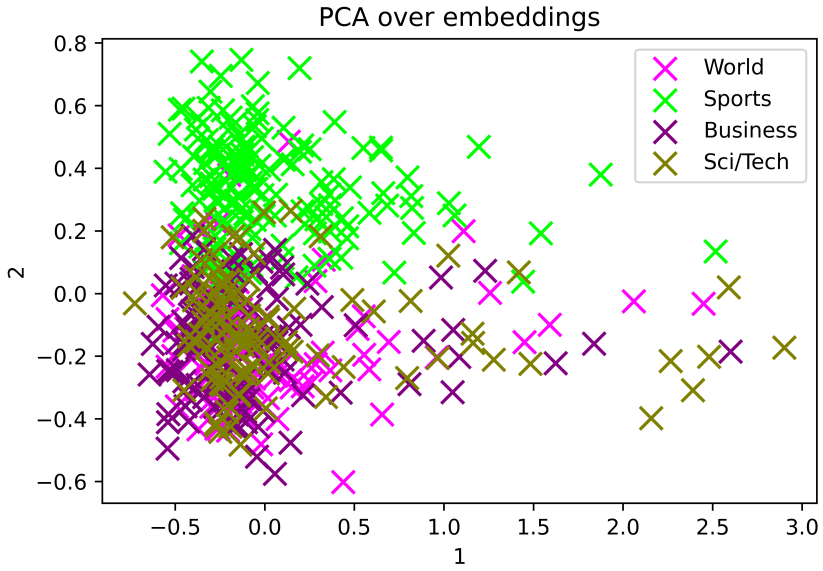


Figure 3.10: PCA to two dimensions using $\hat{\boldsymbol{x}}$ without employing LDAVs. X-axis and y-axis refer to the principal components (dataset: AG news, Model:BILSTM).

Figure 3.11: 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).

As we can see in Figures 3.12 and 3.13, the embedding vectors of the input texts after LDAVs are training tend to be clustered collinearly depending on the predicted class. However, in Figures 3.10 and 3.11, the embedding vectors without using LDAVs.

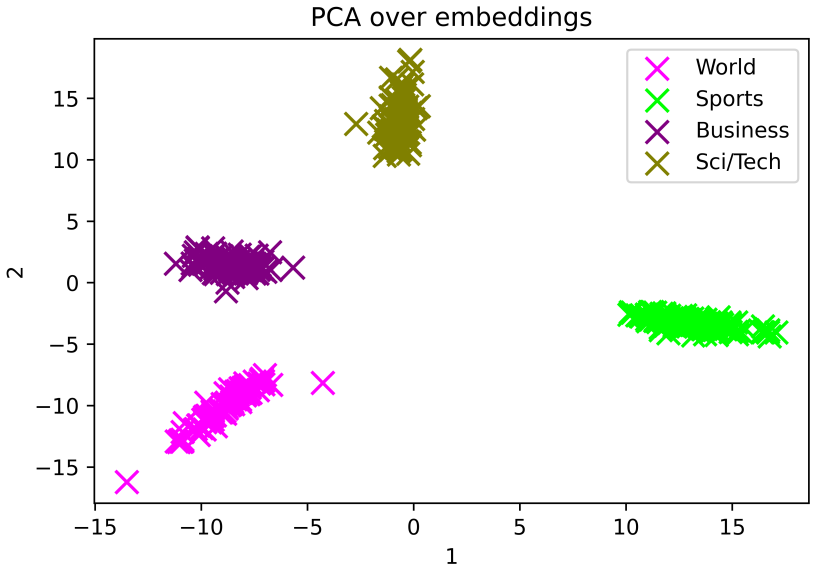


Figure 3.12: PCA to two dimensions using $\hat{\boldsymbol{x}}$ after employing LDAVs. x-axis and y-axis refer to the principal components (dataset: AG news, Model:BILSTM).

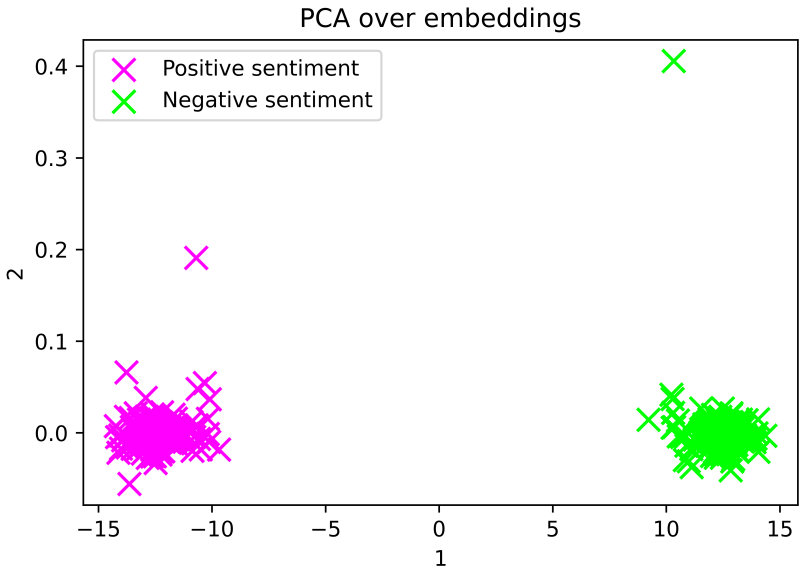


Figure 3.13: PCA to two dimensions using $\hat{\boldsymbol{x}}$ after employing LDAVs. x-axis and y-axis refer to the principal components (dataset: IMDB, Model:BILSTM).

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.

3.4 Conclusion

We have presented a method to learn locally distributed activation vectors (LDAVs) that can be adapted to interpret neural network predictions faithfully. The LDAV vectors are jointly trained with the deep model to provide feature attribution along with the model's prediction. We found that learning intermediate representation as a proxy for feature attribution is computationally efficient and also provides faithful explanations. 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.

Chapter 4

Rationalizing Neural Networks via Concept Clustering

4.1 Introduction

There has, alternatively, been attention to models' attempts to learn rationales concurrently with the classification task. For example, Lei et al. [7] proposed a neural network architecture for text classification which "justifies" its predictions by selecting relevant tokens in the input text known as "rationales". However, the proposed method is computationally expensive. This section presents a simple and efficient approach to extracting a rationale while learning the classifier. Moreover, the proposed method groups similar rationales into distinct groups.

4.2 Proposed Method

We call our model **RANCC** - **R**ationalizing **N**eural **N**etworks via **C**oncept **C**lustering. This section explains the ideas and methods in RANCC: (a) how to build an inherently deep network model for text classification (e.g., a model that provides a rationale concurrently with the prediction), (b) how to learn concepts of interest from the training data simultaneously. The overall model is shown in Figure 4.1.

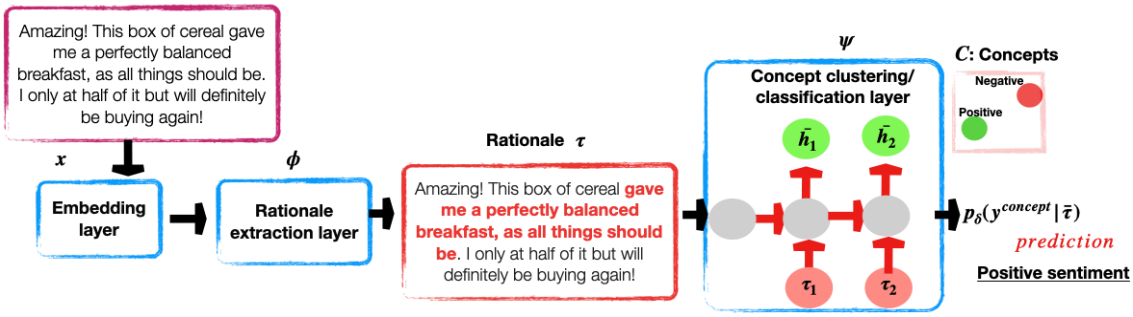


Figure 4.1: Block diagram of our method. A text instance is fed into the embedding layer. The rationale is extracted from the text instance and forwarded to the next layer to predict the target class. The loss is computed and used to update the entire network in an end-to-end approach. Note that the black arrows indicate the steps of our approach, and the red arrows indicate the process inside any recurrent network.

4.2.1 Steps for Building RANCC

In a text classification task, an input ordered sequence $x = x_1, ..., x_l$ is mapped to a distribution over class labels using a parameterized θ neural network architecture (such as a Long Short Term Memory network or LSTM), i.e., $\mathcal{F}(x; \theta)$. Usually, the input to $\mathcal{F}$ is in sentences or short paragraphs. The output y is a vector of class probabilities. The target class $y_i \in y$ is a categorical outcome, such as a sentiment class like "positive review." The distribution over the labels is defined as $y|x \sim Cat(\mathcal{F}(x; \theta))$.

Unsupervised rationale extraction

We define a *rationale* as a subset of text extracted from the source document of the task, which provides sufficient evidence for predicting the correct output. Our approach assumes that an explanation of a black-box's prediction is useful if it relies on a small number of tokens (a rationale), where each rationale relates to parts of the text that are semantically consistent across multiple texts. Given an input sequence x where x_i, a word in the sentence, is represented as a fixed size vector where $x_i \in \mathbb{R}^d$, and d is the dimension of embedding vectors. For each sequence, we first extract a rationale to be used by the downstream classifier. We feed x to a $\phi(x)$ (a convolution operation over the embedding matrix). The convolution layer learns v feature maps

$\boldsymbol{Z} = \{\boldsymbol{Z}_i\}_{i=1}^{v}$, whose shape is $l \times d \times v$. For instance, in the case of a movie review input of length 50 tokens and feature dimension of size 100, the convolution produces $50 \times 100 \times v$ feature maps. The stride of the convolution filter is set to one, padding is set to 'same' and the number of filters is set to 128. This choice of filter has the intuitive impact of learning more meaningful representations which could then be used as a proxy to extract the rationales. We then aggregate all feature maps to obtain a single matrix $\bar{\boldsymbol{Z}} \in \mathbb{R}^{l \times d}$. The aggregated feature map matrix has the following properties: 1) each row represents a word from the input sentence, and 2) if the feature values on a row are larger than average, then the corresponding word has a better chance of selection. Finally, we compute the score for each word to be selected in the rationale using Softmax as follows :

$$\sigma(w)_i = \frac{e^{w_i}}{\sum_{j=1}^{l} e^{w_j}}, \tag{4.1}$$

where $w_i = (\bar{\boldsymbol{Z}}_i)_{i=1}^{l}$ is the sum of the row i in $\bar{\boldsymbol{Z}}$. We uncover the rationale $\boldsymbol{\tau}$ by selecting $\hat{l}$ tokens $\boldsymbol{\tau} \sim p(\sigma(w)_1 ..., \sigma(w)_l | x)$ which produces the rationale $\boldsymbol{\tau} \in \mathbb{R}^{\hat{l} \times d}$. Note that the user defines the length of the rationale $\hat{l}$. During the test phase, we make predictions based on the most likely assignment for each $\boldsymbol{\tau}_i$ using $argmax$. Note that $p(\sigma(w)_1 ..., \sigma(w)_l | x)$ also provides a measure for feature importance.

Uncovering Concept Vectors

We aim to group $\boldsymbol{\tau}$ into concepts of interest (i.e., to transform rationales into meaningful concept groups) concurrently with rationale extraction. Let $\beta \in \mathbb{R}^{\hat{l}}$ denote the probabilities of the selected tokens. Given $\boldsymbol{\tau}$, let us suppose that an analyst is interested in a concept representing negative sentiments in movie reviews and wants to know whether the rationales used by the classifier are discriminative. Grouping rationales give the analyst a better understanding of how the model encodes discriminative features from the raw embedding. To learn concept vectors, we first initialize a matrix of weights $\boldsymbol{C} \in \mathbb{R}^{m \times d}$, where m is the number of target concepts and $c_i \in \boldsymbol{C}$

represents the concept vector for the target $y_i \in y$. We aim to find a concept of interest (i.e., a row in the matrix). To find the concept vector c_s, we obtain $\bar{\boldsymbol{\tau}}$ through multiplying the values in the i-th row of $\boldsymbol{\tau}$ by β_i. We feed the rationale to an LSTM ψ i.e., $\psi(\bar{\boldsymbol{\tau}})$ to obtain a latent representation $\hat{\boldsymbol{H}} \in \mathbb{R}^{\hat{l} \times d}, \hat{h}_i \in \hat{\boldsymbol{H}}$. Finally, the last state $\hat{h}_{\hat{l}}$ is fed into a non-linear layer with parameters $\boldsymbol{\delta}_{concept} \in \mathbb{R}^{d \times m}$ which produces a score for every concept. The output is a vector $y^{concept}$ of probabilities and we extract the corresponding concept vector c_s using $argmax$.

4.2.2 Learning Rationales

The objective function aims to learn the following tasks: a) learning a rationale from text input; b) grouping rationales based on concepts.

Learning Rationales

The loss function for the rational extraction aims to maximize the scores of salient tokens. In general, the loss maximizes the log probability of the selected tokens that lead to a correct prediction:

$$\mathcal{L}^{rationale} = \lambda \Bigg(- \sum_{i}^{s} A_i \log p(\beta | x) \Bigg), \tag{4.2}$$

where β is the probabilities of the selected tokens, s is the batch size, λ is used to weigh the importance of this loss, and A_i is a scalar. For example, a scalar A_i could be 1 if the model predicts the correct class label for x using the rationale τ and 0 otherwise. We used a custom gradient to pass the updates through the selection step in the rationale extraction layer. The custom gradient function works as follows: first, assign a gradient of 1 to each selected $\boldsymbol{x}_i$, and 0 otherwise. Please note that we also use cross-entropy to learn the classification problem.

Grouping Rationales

Each concept vector should correspond to semantically consistent rationales. We assume that every class and rationale has only one concept of interest. We use a loss

to minimize the distance between each rationale and the concept vector:

$$\mathcal{L}^{groups} = \begin{cases} 1 - \frac{\hat{\tau} \bullet \mathbf{c}_s}{\|\hat{\boldsymbol{\sigma}}\| \|\mathbf{c}_s\|}, & \text{if } s = y_i \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $\bullet$ is the dot product operation, $\hat{\boldsymbol{\tau}}$ is obtained by taking the average over the columns of $\bar{\boldsymbol{\tau}}$ and s is the index of the predicted concept vector c_s. This loss is only applied if the prediction at the output layer is correct given the rationale $\boldsymbol{\tau}$ and the concept vector c_s; otherwise, the loss is zero. Every rationale $\boldsymbol{\tau}$ is grouped around its concept by computing the mean (x-axis) and the standard deviation (y-axis) of $(c_s + \hat{\boldsymbol{\tau}})$ from every rationale.

4.3 Experiments

Our primary intent is *not* predictive accuracy but instead building a deep interpretable network, and the experiments can be summarized as follows: 1) we show how RANCC outperforms the baseline for rationale extraction; 2) the effectiveness of using RANCC for feature importance, and 3) we show how RANCC can be used to group rationales; The hyper-parameters used for the experiments are shown in Table 4.1.

Optimizer	Adam
Text length	50 tokens for IMDB, 20 tokens for AG news
Learning rate	$1e-3$
Embedding dimension	300
Concept vector dimension	300
Cell	LSTM
LSTM Hidden dimensions	300
Scalar A	1.0 if correct class prediction, 0.0 otherwise
Batch-size	256

Table 4.1: Hyperparameters used in the experiments.

4.3.1 Evaluation

When rationalizing neural networks' text classification prediction, our goal is to perform as well as systems using the entire input text while using only a subset of the text, leaving unnecessary tokens out for explainability.

Rationalizing text prediction in sentiment analysis. We use the IMDB dataset for evaluation. We compare our approach against [34]. For evaluation, we calculate the accuracy as a function of the length of the extracted rationale [34]. Figure 4.2 shows the performance for various percentages of selected text. Our approach outperforms the work of [34], achieving a similar accuracy as the baseline system by using only the top 10% tokens. The results show that RANCC captures better discriminating features than the baseline.

Rationalizing text prediction in news classification. We use the AG news dataset [60] to test the performance on topic classification. The dataset consists of 127600 samples divided into 4 classes. We split it into training set 80% and testing set 20%. Figure 4.2 shows the results on AG news, and RANCC outperforms the baseline and [34] by identifying better discriminative features.

4.3.2 Faithfulness: Are "Relevant" Features Truly Faithful to What The Model Computes?

Goal. Verify whether the identified salient features are "faithful" to what the model computes. Standard techniques for evaluating the importance rely on observing the effect on the model's prediction after removing a salient token. In this subsection, we evaluate the faithfulness against post-hoc explanations by comparing the feature importance approximated by our approach and that of the post-hoc methods. We use the AG news dataset and IMDB and split the data into training and testing sets. We compare our method with several competitive algorithms for feature importance scoring on black-box models, including gradient-based methods such as **ϵ-LRP** [67], **Grad*Input** [21], and **Intgrad** [65]. Model-agnostic such as **LIME** [6].

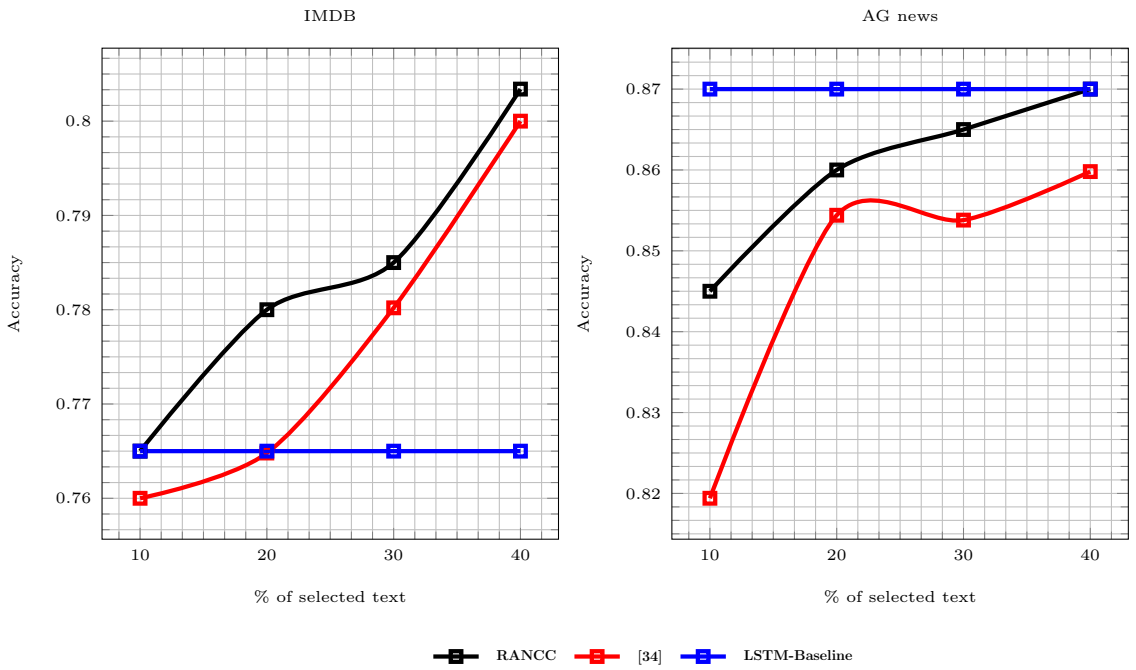


Figure 4.2: IMDB test accuracy (left) and AG news test accuracy (right) for various percentages of extracted text. Baseline refers to the LSTM network trained on the full text.

Change in log-odds ratio. Similar to the metric used in the previous chapter. Figure 4.3 shows the results of the change in the log-odds ratio experiment on the AG news dataset. Note that **Grad*Input** has the same performance as **ϵ-LRP**. Our method achieves the lowest log-odds ratio (the biggest change in log-odds ratio) when removing salient features from the text input. For example in Figure 4.3, the log-odds score of RANCC when removing the top 20% features from IMDB is -3.185 and the log-odds score from the best performing baseline (IntGrad) is -2.069. Our approach considers maintaining accuracy and explainability; therefore, RANCC could correctly capture the important features affecting the prediction output compared to the baselines.

4.3.3 Visualizing Concepts

Results on IMDB are shown in Figure 4.4. As we can observe, our approach provides better results than t-SNE [74] and PCA. The grouping results on AG news are also

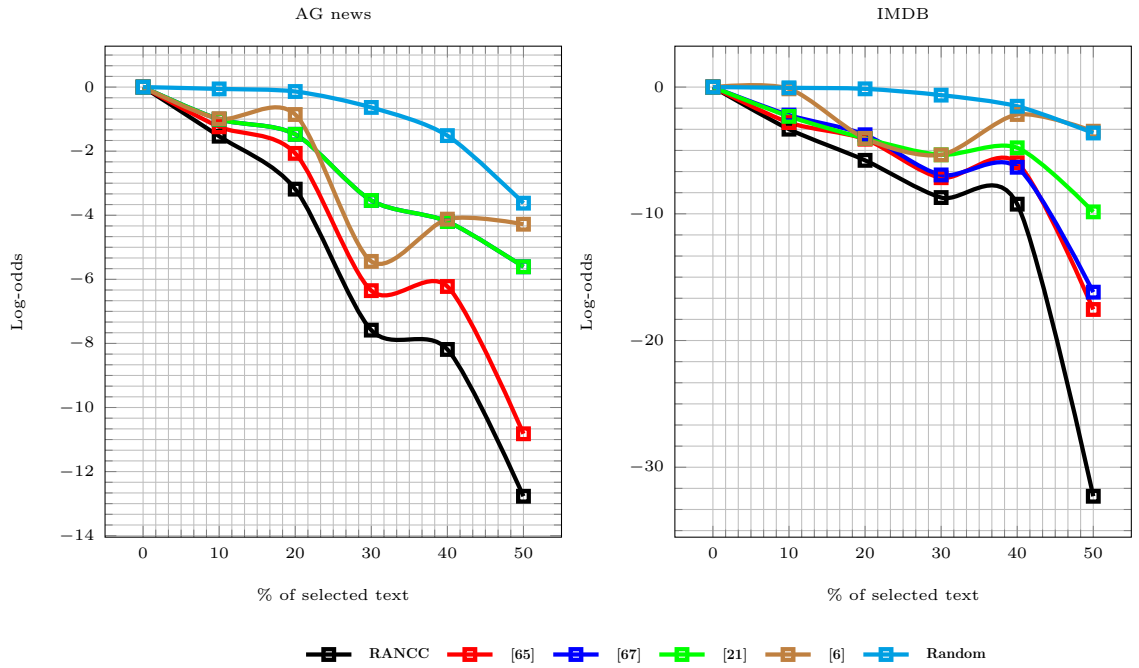
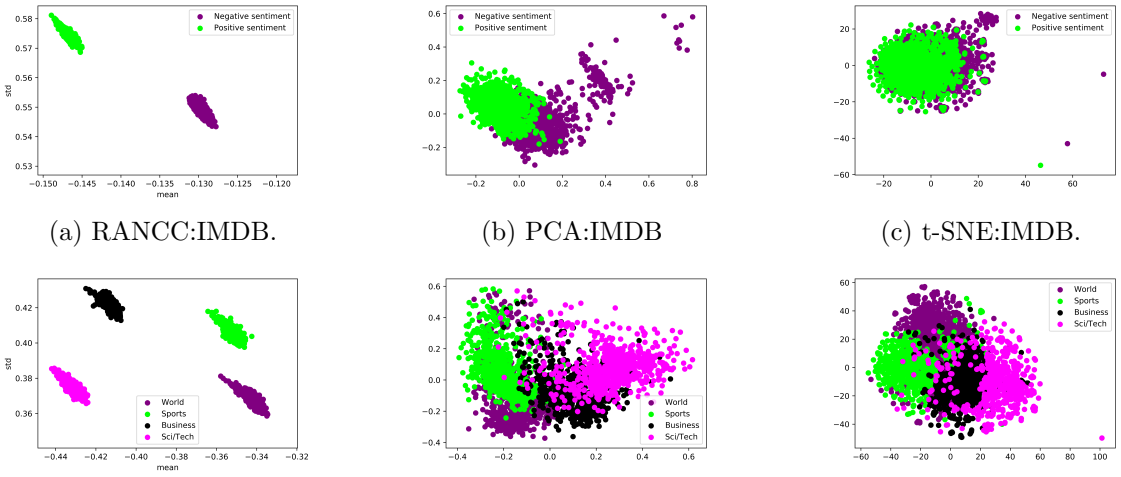


Figure 4.3: Change of log-odds ratio for various percentages of extracted rationale. Lower log-odds scores are better.

shown in Figure 4.4(the four classes in the dataset representing four concepts). Our work does a better job of revealing the natural classes in the data than t-SNE and PCA. Thus RANCC is better at accurately generating distribution and partitioning the data.



(a) RANCC:IMDB. (b) PCA:IMDB (c) t-SNE:IMDB.

(d) RANCC: AG news. (e) PCA: AG news (f) t-SNE:AG news.

Figure 4.4: Groups of correctly classified rationales using concept vectors for both IMDB and AG news using RANCC, PCA and t-SNE.

.

4.4 Conclusion

We have presented a new approach for building interpretable deep models. The proposed neural framework automatically extract sufficient text fragment from the input as the model's justification to each predictions. We demonstrated the effectiveness of the proposed method in identifying the rationales without any explicit rationale annotations. We found that our proposed approach provides faithful explanations compared to existing methods.

Chapter 5

Contrastive Explanations

5.1 Introduction

Most existing related research has focused on identifying feature attribution (e.g., possible causal attributes) to explain the prediction of a black-box neural network. This type of explanation is defined as answers to "*why-questions*", which are generally thought of as causal-like explanations [75]. Existing techniques to *why-questions* rely on using a post-hoc approach to identify the causal attributes for a single black-box prediction. Post-hoc methods generally do not always provide accurate explanations [5]. There are many possible reasons for this limitation; for instance, feature attributions typically suffer from noisy gradients in back-propagation techniques [76].

On the other hand, a contrastive explanation provides an explanation for why an instance had the current output (fact) rather than a targeted outcome of interest (foil) [77] e.g., "*why p and not q?*". In this section, we extend the `LDAV` and propose a deep interpretable network that provides feature attribution and contrastive explanations (see Figure 5.1). We also propose three metrics to evaluate the faithfulness of the contrastive explanations.

5.1.1 Contrastive vs. Counterfactual

The evolving discussions of explainable AI (XAI) have articulated several distinguishing aspects of explanation (e.g., [9]), including a difference between contrastive (e.g.,

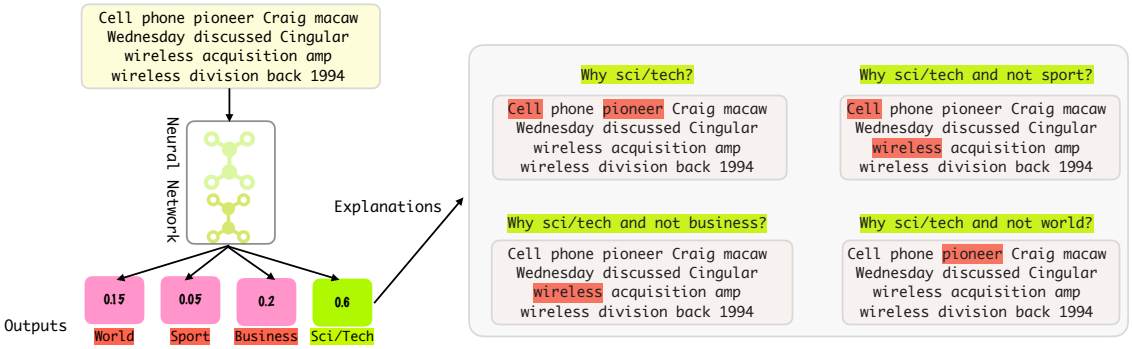


Figure 5.1: An example of the proposed interpretable deep neural network model with answers to "*why p?*" and "*why p and not q?*" questions. Here we visualize the top salient attributes.

what made the *difference* between the students who failed the exam and those who did not fail?) and counterfactual (e.g., will we reduce climate change *if we reduce fuel consumption?*) explanations. Contrastive explanations are different from counterfactual explanations [78]. Contrastive and counterfactual reasoning generally emphasize different aspects of causation[79]. In counterfactual reasoning, we focus on instances where the salient causal attributes are absent (missing from the text). In contrast, in a contrastive explanation (our focus here), one considers the difference in attributes between two predictions. The difference between the two approaches is in the knowledge support required for the explanation. For instance, a counterfactual explanation focuses on the question of "*What if?*," while a contrastive explanation focuses on "the difference."

5.2 Proposed Method

We propose a neural network architecture that provides a classification task and an explanation. We jointly optimize the network for both classifications and faithful explanations. In classification, an input sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_l \in \mathbb{R}^d$, where l is the length of the text input and d is the vector dimension, is mapped to a distribution over class labels using a parameterized neural network (e.g., Multi-head attention). In general, the contextual vector $\acute{\boldsymbol{h}} \in \mathbb{R}^d$ is passed to a linear layer with parameters

$\boldsymbol{W} \in \mathbb{R}^{d \times n}$ which provides a probability distribution over n classes. The output $\boldsymbol{y}$ is a vector of class probabilities of dimension $\mathbb{R}^n$, where n is the number of classes. The predicted label p of the text input is the index of the maximum element in $\boldsymbol{y}$, i.e., $p = argmax f(\boldsymbol{x}), \forall k \in [1, n]$. Here, k iterates over the probabilities, and $f(\boldsymbol{x})$ denotes a neural network. During training, an empirical loss (e.g., cross-entropy) $\mathcal{J}(p, y^{'}, \theta)$ is minimized using gradient descent, where $y^{'}$ is the ground truth label and θ represents the network's parameters. We propose to augment the network to provide two types of explanations "*Why p?*" and "*Why p and not q?*." To do so, we first define a randomly initialized centroid vector for each class and then use the centroid vector as a proxy to explain the black-box prediction. For instance, if the neural network's prediction is class 1, we use the centroid vector representing that class to calculate the scores for *why p?*. For contrastive explanation, we find the difference between the scores of the centroid vector representing the predicted class and the centroid vector representing the contrast class (e.g., the centroid vector for class 2). The centroid vector of label p pulls the weighted sentence vector of the text input $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_l$ closer. In the following, we discuss the steps for augmenting a neural network with the centroid vectors.

Let $\boldsymbol{c}_j (j = 1, 2, ..., n)$ be a collection of randomly initialized centroid vectors, where $\boldsymbol{c}_j \in \mathbb{R}^d$ is a vector representing label $\boldsymbol{y}_j$. We propose a new objective function, centroid-loss, to effectively explain the neural network predictions. Our solution enhances the discriminative power of the deeply learned features in neural networks. Specifically, the deep network learns a centroid $\boldsymbol{c}_j$ (a vector with the same dimension as an embedding feature) of each class. In the course of training, we simultaneously update the centroid vector and minimize the distances between the embedding features and their corresponding centroid vector.

5.2.1 Joint objective

Recall that a supervised learning algorithm input is a set of training instances and the corresponding label. The goal is to learn a function that accurately maps input examples to their desired labels using cross-entropy. Here we extend the idea of the `LDAV` to provide contrastive explanations. Given the prediction p, the model learns $\boldsymbol{c}_p \in \mathbb{R}^d$ to pull the sentence vectors representing class p closer. Intuitively, we minimize the intra-class variations while keeping the features of different classes separable. In the following, we discuss the optimization objective of our proposed network.

Cross-entropy: term one in the optimization objective function is the standard loss function for classification. We denote this loss as $\mathcal{L}^{\text{cls}}$.

Attractive term: term two focuses on minimizing the cosine distance between the sentence vector and the corresponding $\boldsymbol{c}_p$. Let $\boldsymbol{X}$ be a matrix consisting of embedding vectors $[\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_l]$ and the sentence vector of $\boldsymbol{X}$ is $\hat{\boldsymbol{x}} \in \mathbb{R}^d$. Let, $\bar{\boldsymbol{w}} \in \mathbb{R}^l$ be the importance scores, where each:

$$\bar{\boldsymbol{w}}_i = \frac{\boldsymbol{x}_i \cdot \hat{\boldsymbol{x}}}{\|\boldsymbol{x}_i\| \, \|\hat{\boldsymbol{x}}_i\|}, \tag{5.1}$$

where $\bar{\boldsymbol{w}}_i$ is the importance score of word i, and $\hat{\boldsymbol{x}}$ is the sentence vector of the input $\boldsymbol{X}$. Term two minimizes the cosine distance between the weighted sentence vector $\bar{\boldsymbol{x}}$ of each input with the corresponding centroid vector $\boldsymbol{c}_p$. The sentence vector is defined as follows:

$$\bar{\boldsymbol{x}} = \boldsymbol{X}\left(\frac{e^{(\bar{\boldsymbol{w}})}}{\sum_{i=1}^{l} e^{(\bar{\boldsymbol{w}}_i)}}\right) \tag{5.2}$$

From equation 5.2, we obtain the weighted sentence vector by multiplying the values in the i-th row of $\boldsymbol{X}$ by $\bar{\boldsymbol{w}}_i$ followed by calculating the sentence vector $\bar{\boldsymbol{x}} \in \mathbb{R}^d$. We define the loss of term two as follows:

$$\mathcal{L}^{\text{attr}} = 1 - \frac{\bar{\boldsymbol{x}} \cdot \boldsymbol{c}_p}{\|\bar{\boldsymbol{x}}\| \, \|\boldsymbol{c}_p\|} \tag{5.3}$$

Term two is the second loss of our proposed optimization objective. We call this term "attractive loss" similar to [80].

Repulsive term: term three (the third term in the overall loss function) focuses on maximizing cosine distance of $\bar{\boldsymbol{x}}$ from other centroid vectors, i.e., $\boldsymbol{c}_j$, where $j \neq p$, so that cosine distance between them is maximum. We call this term "repulsive loss" similar to [80]; we denote the loss as $\mathcal{L}_{\text{rep}}$.

Pairwise term: term four in our objective maximizes the pairwise distance matrix of the centroid vectors. For the distance, we proposed to use the squared euclidean distance, and we denote the loss as $\mathcal{L}_{\text{pair}}$.

Overall loss is defined as

$$\mathcal{L} = \mathcal{L}^{\text{cls}} + (\lambda_1 \mathcal{L}^{\text{attr}}) - (\lambda_2 \mathcal{L}^{\text{rep}}) - (\lambda_3 \mathcal{L}^{\text{pair}}) \tag{5.4}$$

where $(\lambda_1, \lambda_2, \lambda_3)$ are the coefficients. The hyperparameters $(\lambda_1 : 1000, \lambda_2 : 10, \lambda_3 : 1000)$ are essential for minimizing the intra-class variation (to reduce the variance within the same class). Specifically, terms 3 and 4 focus on keeping the features of different classes separable, and term 2 focuses on minimizing the intra-class distances. All of them are essential to our model. We refer to the combination of the newly added terms as the centroid loss, i.e., term 2, term 3, and term 4.

5.2.2 Explanations

We seek to identify a feature with a causal impact on the model prediction decision process. We follow [81]'s definition of intervention: an intervention is an idealized experimental manipulation carried out on some variable $\boldsymbol{x}$ which is hypothesized to be causally related to changes in some other variable p. Any intervention on the text input using attributions on the prediction p is a causal process that changes the model prediction. Therefore, if the intervention changes the model prediction, it is probably due to the adjustment in the causal space of the text input. We will use the idea of "intervention" to understand the effectiveness of our approach for both *why*

p? (discussed in the previous chapter) and *why p and not q?*.

Why p and not q?: Given any text instance, a classifier predicts p and a the LDAV vector $\boldsymbol{c}_p$. A p-contrast question is of the format "Why [predicted-class (p)] not [desired class (q)]?". We limit our search space by specifying the desired class to a single alternative. Given the text input, we estimate attribution scores for "p" using $\boldsymbol{c}_p$. For the desired class q, we calculate the attribution scores of the text input using $\boldsymbol{c}_q$. Please note that here we also use cosine similarity. We find the attribution scores for contrastive explanations as $\boldsymbol{c}_c = \boldsymbol{c}_p - \boldsymbol{c}_q$, where $\boldsymbol{c}_p$ is the attribution score for the predicted class p obtained using $\boldsymbol{c}_p$ and $\boldsymbol{c}_q$ is the attribution scores for the foil class q obtained using $\boldsymbol{c}_q$.We follow the intervention approach in "*Why p?*," to find the candidate attributes for the contrastive explanation.

5.3 Experiments

To effectively evaluate our approach, for contrastive explanations, we rank each attribute by how contrastively applicable it is to the model for choosing "p" against "q."

5.3.1 Setup

Datasets. We adopt the IMDB datasets [59] (train:25000, test:25000 samples) with binary labels, AG news [60](train:102080, test:25520 samples) with four classes to evaluate the quality of the contrastive explanation. We hold out 10% of the training examples as the development set. We limit the input length to 50 for IMDB and 20 for AG news.

5.3.2 Evaluating Why p and not q?

We skipped the evaluation of feature attribution because we discussed it in Chapter 3. To evaluate the faithfulness of contrastive explanations, we propose the following metrics:

Contrastive overlap score (COS) (%): We calculate the overlap (%) between the sets of causal attributes of "*Why p?*" and "*Why p and not q?*. Lower % indicates more difference between the explanations of "*Why p?*" and "*Why p and not q?*.

Contrastive confidence score (CCS): For CCS, we analyze the change in the probability of the contrastive class "*q*." We remove the attributes that distinguish "*p*" from "*q*" in order of their importance until the model's prediction is flipped to another class. Please note that we estimate the scores using "*why p and not q?*" We calculate the difference in the probability of "*q*" before and after the intervention. An increase in the probability indicates an informative contrastive explanation.

Contrastive gain (CAG): Measures the quality of contrastive explanations compared to non-contrastive explanations. Here, our explanations for the question "*why p?*" will be called non-contrastive explanations. Given a prediction "*p*" and foil "q," we measure the change in the probability score of "*q*" after removing salient features using attribution-scores obtained from "*why p?*" and also from "*why p and not q?*" explanation. We use our approach as the baseline for "*why p?*" because our method outperformed [16]. For the "*why p and not q?*" explanation, we used the method described in Section (3). A higher contrastive gain indicates that the method is better in answering "*why p and not q?*" questions. In summary, the contrastive gain measures the change in the probability of the foil class after removing some features.

Results

We use the AG news dataset to evaluate the contrastive explanation method. For contrastive overlap (COS), the results in Figure 5.2 show that most contrastive explanations do have fine-grained differences from "*Why p?*" questions. The result suggests that the model is not using the same reasoning for "*Why p?*" when answering the contrastive questions. We observed that, for multi-class problems, there are fine-grained differences between "*Why p?*" and "*Why p and not q?*" compared to a binary problem such as sentiment classification where there might be a higher

similarity between the two explanations.

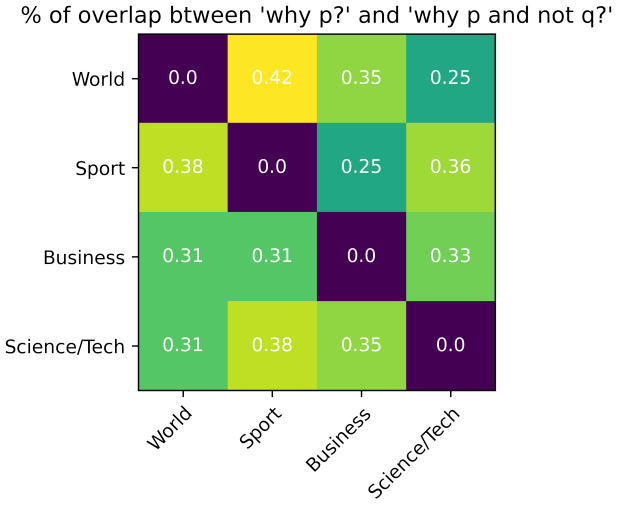


Figure 5.2: Overlap score between "*why p?*" and "*why p and not q?*" questions. "0.0" means that we did not consider the contrastive explanations when "p" and "q" are the same (X-axis: refers to why p? questions and Y-axis: refers to why p and not q? questions

For (CCS), results shown in Table 5.1 indicate the effectiveness of the proposed losses in finding contrastive information, i.e., there is an increase in the score of the foil "q" when removing the features that distinguish "p" from "q." Similarly, we

	World(q)		Sport(q)		Business(q)		Sci/Tech(q)	
Class(p)	Before	After	Before	After	Before	After	Before	After
World			0.05	0.22	0.05	0.41	0.01	0.33
Sport	0.07	0.45			0.01	0.35	0.04	0.21
Business	0.06	0.38	0.003	0.16			0.13	0.37
Sci/Tech	0.02	0.32	0.04	0.12	0.13	0.52		

Table 5.1: CCS: Empty cells mean we cannot find a contrastive explanation for the same class, i.e., the foil should be different from the predicted class. The highlighted cells show the scores of the foil after removing the salient features.

show the scores of other classes when using the (CCS) metric. We re-trained the same model again on AG news and re-calculated the CCS. The results are shown in Table 5.2. We can see that when the foil q is set to "business" and evaluated with different classes (p) such as "world, sport, sci/tech ."The probability score for

the "business" is > than other classes when using *why p and not business.?* Figure

	World	Sport	Business(q)	Sci/Tech
World(p)	0.2	-0.8	0.4	0.1
Sport(p)	0.3	0.1	0.5	-0.8
Sci/Tech(p)	-0.7	0.1	0.4	0.1

Table 5.2: We compare the scores of other classes when evaluating the CCS for the foil. Here the foil is the business class.

5.3 compares our non-contrastive explanations and contrastive explanation methods (CAG). We use the AG news data and plot the results for different "*why p and not q?*" questions. The results in Figure 5.3 indicate that the contrastive explanations better capture the features that contribute prediction of ", not q" than non-contrastive explanations, especially when there are more fine-grained differences. The results show that non-contrastive explanation is not consistently achieving high contrastive scores when top features are masked. Instead of tracking the change in probability

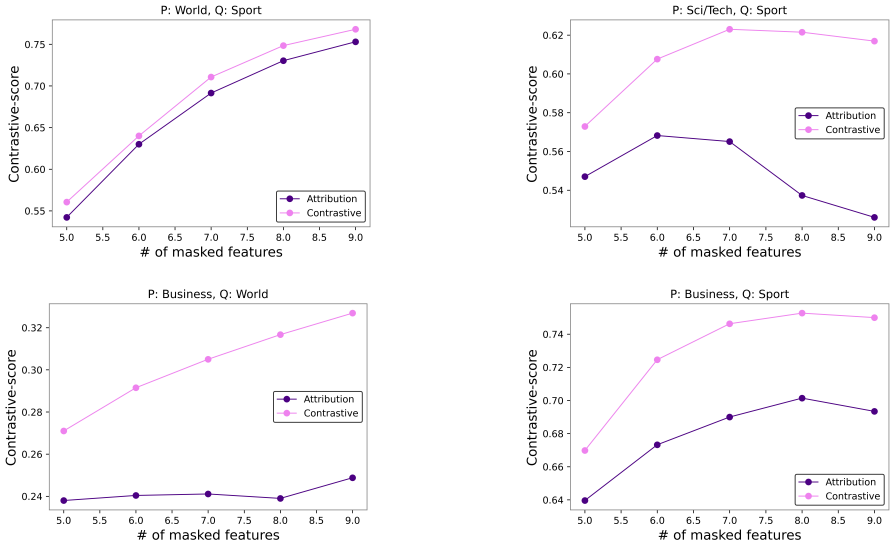


Figure 5.3: Contrastive gain as a function of removed tokens. A higher gain indicates that the method was better in capturing contrastive information. Attribution refers to our non-contrastive method.

score of "q" after removing salient as in contrastive gain, we calculate the AOPC using different percentages $(25\%, 30\%, 35\%, 40\%, 45\%)$. The results are summarized

in Table 5.3. The contrastive explanation has the highest AOPC compared to our non-contrastive explanation method.

P	**Q**	AOPC(non-contrastive)	AOPC(contrastive)
World	Business	0.04	0.065
Business	Sci/tech	-0.001	0.002
Sci/tech	World	0.304	0.341
Sci/tech	Business	0.056	0.058
Sport	Business	0.05	0.052
Sport	Sci/tech	0.006	0.009

Table 5.3: Contrastive gain (CAG): Evaluating the effectiveness of using contrastive explanation when there are fined grained differences. We use different percentages $(25\%, 30\%, 35\%, 40\%, 45\%)$ to calculate the AOPC.

Highlighting *why p and not q?* questions: We show qualitative results for interpreting the model predictions using our proposed approach; for example, answers to the "*Why p?*" and "*Why p and not q?*" questions are shown in Table 5.4. These results show that the model implicitly learns the contrastive information when making the prediction.

Contrastive explanations applied to sentiment classification. For a contrastive explanation, if there are no fine-grained differences between "p" and "q", then the same reasoning used for "*why p?*" will also be used to answer "*why p and not q?*" questions. We observed this behavior in binary text classification. For instance, we found that the model uses the same reasoning for both questions (see Table 5.5). We attribute this observation to the fact that "*why p and not q?*" cites the causal difference between p and *not-q*, i.e., consisting of a cause of p and the absence of a corresponding event in the history of q. We also found that explaining "*Why p and not q?*" is not the same as explaining "*Why q and not p?*." In the case of sentiment classification, we found that these two questions provide different answers, consistent with the work of [82]. To validate our sentiment classification observations, we now focus on the overlap between "*why p and not q?*" and "*why q and not p?*." We

Text	World	Sport	Business	Sci/tech
record shown mutilated body found iraq kidnapped aid worker margaret hassan british official say still believe british irish citizen dead(P:World,Q:others)	iraq	dead	hassan	kidnapped
search war begin today software giant microsoft unveils test version new search engine looking remarkably like one chief rival google. (P:Sci/tech,Q:others)	engine	microsoft	engine	version
version desktop search tool computer run apple computer mac operating system google chief executive eric schmidt said friday(P:Sci/tech,Q:others)	desktop	apple	schmidt	mac
inflation dozen nation sharing euro slowed initially estimated september company reduced price lure customer store offsetting record energy cost.(P:World,Q:others)	store	inflation	price	lure

Table 5.4: Contrastive explanations on AG news.

use the IMDB dataset and calculate the overlap between the attributes (a minimum subset of the attributes required to flip the prediction) of "*why p and not q?*" and "*why q and not p?*." The similarity ratio was zero, meaning the explanations are entirely different.

5.4 Conclusion

We have shown that `LDAV`s can also be used to provide contrastive explanations. In general, contrastive explanations are different from feature attribution, they mainly focus on the difference in attributes between two predictions. Contrastive explanations can provide additional insights into non-contrastive explanation, resulting in a better understanding of the neural model predictions. We have proposed a neural

Text	Highlight
the story is enjoyable and easy to follow this could have been easily messed up but the actors and director do a great job of keeping it together the actors themselves are fantastic displaying wonderful character and doing a terrific job gotta find a copy somewhere (P:positive,Q:negative)	fantastic
this performance that should elevate the film to a platform where it a place on the best ever lists of courtroom dramas however despite its apparent obscurity sergeant still remains a taut and compelling examination like a book that you just can't put down highly recommended (P:positive,Q:negative)	recommended
imagined in my mind what i saw on screen was slightly different however it wasn't enough to make me dislike the mini series i recommend this for anyone who has read the novel you will not be disappointed if you have 8 out of 10 stars (P:positive,Q:negative)	8
provide someone to at well one must do something beside during this film the movie is being sold on vhs now by people on e bay spare yourself the expense and the waste of time a comedy without a laugh a musical without a memorable song or dance (P:negative,Q:positive)	waste

Table 5.5: *why p and not q?* Contrastive explanation is the same as *why p?* explanations in binary sentiment classification.

network architecture capable of provide contrastive explanations along with feature attribution. We have also proposed three metrics to evaluate contrastive explanations.

Chapter 6

Self Knowledge Distillation

6.1 Introduction

Pre-trained Language Models (LM) are being used in many NLP tasks (e.g., [83, 84]). The BERT system is one of the recent advances in NLP [85], which learns contextualized representations from a large-scale text corpus. BERT models can be fine-tuned with a task-specific layer to tackle application-specific problems such as text classification. Nevertheless, these LMs rely on hundreds of millions of parameters, making them difficult to train and deploy. *Knowledge distillation* is a training pipeline that poses loss in training a small model (student) from the knowledge of a pre-trained large model (teacher) [86]. The goal is to compress a teacher into a smaller model (student) while employing fewer parameters than the teacher. In general, a knowledge distillation technique minimizes the errors between the soft targets of the teacher and the student. Various research has devoted effort to compressing large networks to accelerate inference, transfer, and storage. One of the earliest attempts focused on pruning "unimportant" weights [87]. Other methods focused on modifying devices to improve floating point operations [88]. In contrast, some works have focused on quantizing neural networks [89]. However, the main drawbacks of the methods mentioned are: (1) they only work with pre-trained networks; (2) the compressed models are in general black-box; and (3) they require additional computation, such as training student models. Inspired by the idea of `LDAV`s, we build an interpretable

lightweight **V**ector **S**pace **M**odel (VSM) concurrently while training the black box.

6.2 Self Knowledge Distillation

In a text classification task, an input sequence $x = x_1, ..., x_l, x_i \in \mathbb{R}^d$, where l is the length of the input text, and d is the embedding vector dimension, is mapped to a distribution over k class labels using a parameterized θ deep network. The output y is a vector of class probabilities, and the predicted class y_j is a categorical outcome. We focus on learning a VSM from a black box. We refer to the deep network as $\mathcal{T}$ and the VSM as $\mathcal{S}$. As shown in Figure 7.1, the deep model learns a Class Semantic Vector (CSV) $v_j \in \mathbb{R}^d$ for each target j, e.g., a vector with the same shape (same length) as the embedding vector. The class semantic vector and the LDAV are similar. However, the objective is different; we use the semantic vectors as the core representation for the vector space model. Here, we show how we can use the CSV to build an effective vector space model.

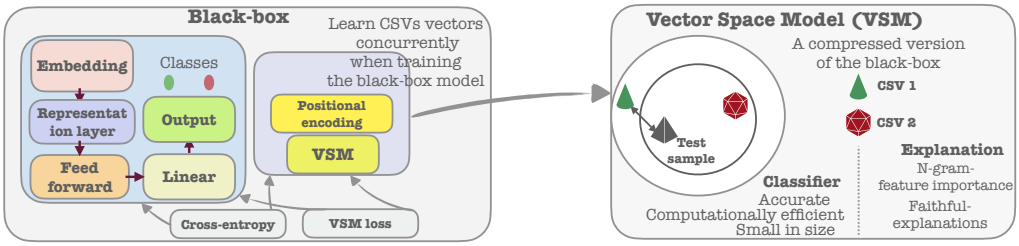


Figure 6.1: Learning a VSM concurrently when training the black-box using class semantic vectors. The model training is similar to the LDAV method presented in Chapter 3. After training the model we use the CSV vectors to construct a vector space model.

We provide the following details for learning a vector space model: (a) how to transfer the knowledge from $\mathcal{T}$ into $\mathcal{S}$ concurrently, and (b) how to explain the predictions of $\mathcal{S}$. Neural networks learn by optimizing a loss function to reflect the true objective of the end-user. For $\mathcal{S}$, our objective is to generalize in the same way as $\mathcal{T}$. We will show how we learn $\mathcal{S}$ concurrently with an LSTM and then discuss how it can be generalized to different types of architectures. The last state $h_l \in \mathbb{R}^d$ is fed into an output layer with parameters $\boldsymbol{W} \in \mathbb{R}^{d \times k}$ which provides a probability distribution

over k classes.

Neural network classifiers in general use the cross-entropy loss to penalize misclassification as $\mathcal{L}^{classifier} = -\frac{1}{k}\sum_{j=1}^{k} r_j \log(y_j)$, where $r \in \mathbb{R}^k$ is the one-hot represented ground truth and r_j is the probability (0 or 1) for class j. We use the embedding space as the base for encoding the semantics. We encode the knowledge from $\mathcal{T}$ into a k-CSVs $v_j \in v$. The dimension of each v_j is equal to the dimension of the embedding vector x_i.

These semantic vectors have the following properties: (1) Each v_j encodes the knowledge of class j from $\mathcal{T}$; (2) These vectors serve as the core representation for building the vector space model; (3) By using cosine similarity, we compute the contribution of each word in x with the corresponding v_j to the class j; (4) These vectors add another level of abstraction by explaining the feature importance of a phrase; (5) The weights of the CSVs are randomly initialized, (6) The semantic vectors can be viewed as centroids to define the boundaries for the VSM and (7) The CSVs can enable end users interact with the model.

Given the text sequence, we use the LSTM network to obtain a latent representation from the sentence embedding. We reformulate the optimization of $\mathcal{T}$ to learn the CSVs concurrently as follows:

$$\mathcal{L} = \mathcal{L}^{classifier} + \lambda_1 \overbrace{\left(1 - \frac{\bar{x} \cdot tanh(v_j)}{\|\bar{x}\| \, \|tanh(v_j)\|}\right)}^{Representation} - \lambda_2 \left(\overbrace{D}^{Contiguity} \right) \tag{6.1}$$

where D is the pairwise distance, j in term 2 is the index of the predicted class with the highest probability, $\bar{x}$ is the sentence vector, and $\{\lambda_1 : 0.9, \lambda_2 : 0.9\}$ are used to weigh the importance of the proposed terms. In what follows, we discuss the new terms added to the optimization problem.

Document representations: The second term of Equation 6.1 is the second loss function we use, which encodes the information of semantically consistent sentences in a single vector v_j. Intuitively, the desired property forces the distance of sentences belonging to the same class j to be close to each other and further away from sen-

tences belonging to other classes. An obvious way to learn semantic information is to minimize the cosine angle between the sentence vector $\bar{x}$ of the input sentence x and the corresponding class semantic vector v_j. The sentence vector is average of the embedding vectors of the input. The vector v_j captures the semantics of consistent inputs to encourage semantic consistency. This step is essential for the `VSM` as it encodes the knowledge from the black box.

Contiguity hypothesis: Sentences in the same class form a contiguous region and regions of different classes do not overlap. To learn a `VSM`, we enforce that the `CSV`s do not overlap with each other. We maximize the pairwise Euclidean distance between these vectors using the third term in Equation 6.1.

6.2.1 Vector Space Model

The Vector Space Model (VSM) is a model for representing text document as vectors of identifiers. Documents and queries are represented as vectors. Here the `CSV`s are the documents of the `VSM` and queries are the test sentences (e.g, x.) The classification of a query is based on similarities determined by the deviation of angles between each document and the query vector. The `VSM` is based on the semantic vectors v learned via back-propagation when training $\mathcal{T}$. The `VSM` has k vectors, where k is the number of classes in the corpora. A query is computed as the sentence vector $\bar{x}$ of x. In general, decisions of many vector space models are based on a notion of distance. For an unclassified query $\bar{x}$, we determine distance with a `CSV` v_j by computing cosine distance between $\bar{x}$ and v_j. The predicted class is the index j of the v_j with the shortest cosine distance. The proposed classifier is easy to understand as it relies only on semantic vectors and cosine distance.

6.2.2 `VSM` Explanations

The vector space model provides two levels of explanations for text classification: (1) Word feature attribution, and (2) End-user interaction.

- **Feature attribution**: We define feature attribution as the scoring (or ranking) function that maps portions of the input to scores that communicate some aspect of their importance in making a prediction. In other words, feature attributions aim to convey which parts of the input are important, responsible or influential to the decision. To find the attribution of each word w.r.t. a predicted class y_j, we calculate the cosine similarity between each x_i and the nearest class vector v_j. A word with high semantic similarity with v_j means that it is important for the prediction.

- **Phrase importance score**: Word feature importance is sometimes insufficient to explain a model's prediction. The end-user might also be interested in querying the classifier to answer different types of questions. For example, in the situation where the model shows the feature importance (in sentiment classification) of each individual word "not," "too," and "bad," an end-user might also be interested in the importance of the phrase "not too bad," which cannot be calculated just by merging the three different feature importance values. To obtain the feature importance for a phrase, we find the phrase vector (the average of the word embedding vectors) and then compute the cosine similarity w.r.t. the nearest `CSV` v_j.

The proposed technique can be adapted to a variety of architectures such as Bi-LSTM, GRU, and RNNs, as it requires access to only the embedding layer from the network and the Softmax layer.

6.3 Experiments

Here we focus on evaluating the effectiveness of our approach in learning a lightweight interpretable classifier.

6.3.1 Datasets

We train the model using IMDB, AGnews, and DBpedia datasets (similar to the previous chapters). We also include the HealthLink dataset in the evaluation. **HealthLink** constructed by Alberta Health Services, Canada. It contains a set of text transcripts written by registered nurses while talking with callers to the Tele-Health service in real-time. It consists of 2 classes ("go to hospital" and "home care"), and each class can be sub-categorized into sub-classes. This dataset will be available based on request. The summary of the datasets is shown in Table 7.1.

Data set	Classes	Max length	Train size	Test size	Vocabulary size
IMDB	2	50	25000	25000	10000
HealthLink	2	20	60475	15119	23174
DBpedia	15	32	5600	63000	50002
AGnews	4	20	102080	25520	59706

Table 6.1: Summary of the datasets used in our experiments

6.3.2 Baselines

We compare our approach with several models for text classification including Transformers [64], IndRNN [90], BLSTM [63], hierarchical attention [91], LSTM [92] and GRU [93].

Transformer employs a multi-head self-attention mechanism based on scaled dot-product attention. We use only the encoder layer, and average the new representations before arriving in the classification's output layer.

IndRNN is an improvement over RNNs, where neurons in the same layer are independent of each other and connected across layers. We use the last hidden state as the feature vector.

Bi-LSTM employs an attention-based bidirectional mechanism on the LSTM network, which captures the salient semantic information (word-attention) in a sentence. These attentions enable the network to attend differently to more and less critical con-

tent when learning the representation. The last hidden state is used for classification.

Hierarchical attention provides two levels of attention mechanisms applied to the word and sentence level. In this work, we use a sentence level-attention mechanism applied on a Bi-LSTM. The feature vector for classification is based on aggregating the hidden representation values.

LSTM and GRU process the input word by word, and the last hidden state is used as the feature vector for classification.

6.3.3 Network Configuration and Training

We tokenize sentences and use the top N words that appeared in every instance for the vocabulary size. We did not use any pre-trained embeddings, and thus we randomly initialized the embedding layer. We also randomly initialized the **CSV**s. We did not use hyper-parameter tuning on the validation as we are not focusing on achieving state-of-the-art predictive accuracy. Instead, we aim to show that our method can achieve similar/better performance to the black-box and provides a better explanation than existing approaches. The dimensions of word embedding, semantic vector, and feature vector (at the output layer) are 128. For training each network, we use the Adam optimizer with a batch size of 256 and a learning rate of 0.0001.

Performance

We trained six different models (architectures) on four datasets. We have tried different values as the weight of each proposed loss term. The results in Table 6.2 show that our semantic distillation approach captures more helpful information from the training data than the baseline black box. Our `VSM` outperforms the black-boxes on all datasets, achieving better performance than the black box. The new optimization problem does not affect the performance of the black-box model (see BBO (Black-Box with our new Objective function) in Table 6.2).

Parameters. We compare the number of parameters used by our nearest neighbor

	IMDB				AGnews				Dpedia				HealthLink			
Transformer [64]																
	F1	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**	**Accuracy**	**F1**	**Precision**	**Recall**	**Accuracy**
Black-box	0.7703	0.7703	0.7703	0.7703	0.8794	0.8798	0.8796	0.8796	0.8653	0.8655	0.8655	0.927	0.6642	0.6645	0.6641	0.6647
BBO	0.7785	0.7787	0.7786	0.7786	0.8831	0.8836	0.8835	0.8835	0.8799	0.8802	0.8799	0.943	0.6887	0.6896	0.6887	0.6894
VSM	**0.8117**	**0.8187**	**0.8187**	**0.8187**	**0.9038**	**0.9039**	**0.9041**	**0.9041**	**0.8806**	**0.8811**	**0.8809**	**0.9438**	**0.7216**	**0.722**	**0.722**	**0.7216**
Attention-based Bi-LSTM [63]																
Black-box	0.7961	0.7993	0.7966	0.7966	0.8887	0.8888	0.887	0.8888	0.843	0.8443	0.8434	0.9037	0.6706	0.6706	0.6705	0.6708
BBO	0.798	0.7999	0.7983	0.7983	0.8929	0.8941	0.8928	0.8927	0.8772	0.8774	0.8772	0.9399	0.6704	0.6705	0.6706	0.6705
VSM	**0.8025**	**0.8025**	**0.8025**	**0.8025**	**0.8956**	**0.8955**	**0.896**	**0.896**	**0.8812**	**0.8816**	**0.8815**	**0.9445**	**0.7207**	**0.7207**	**0.7209**	**0.7208**
IndRNN [90]																
Black-box	0.776	0.7761	0.776	0.776	0.8773	0.878	0.8769	0.8769	0.8763	0.8765	0.8765	0.9391	0.6808	0.6814	0.6813	0.6808
BBO	0.7805	0.7858	0.7814	0.7814	0.8845	0.8847	0.8847	0.8848	0.8845	0.8889	0.888	0.9515	0.6808	0.6814	0.686	0.6869
VSM	**0.8018**	**0.8022**	**0.802**	**0.802**	**0.9025**	**0.9026**	**0.9028**	**0.9028**	**0.8887**	**0.889**	**0.8889**	**0.9524**	**0.7162**	**0.7184**	**0.7174**	**0.7164**
Hierarchical recurrent net [91]																
Black-box	0.7917	0.7919	0.7917	0.7917	0.8845	0.8855	0.8846	0.8846	0.847	0.8475	0.8467	0.9073	0.6708	0.6708	0.609	0.671
BBO	0.7808	0.7844	0.7813	0.7814	0.8874	0.8876	0.8874	0.8876	0.8709	0.871	0.8709	0.933	0.6829	0.6833	0.6833	0.6829
VSM	**0.8146**	**0.8146**	**0.8146**	**0.8146**	**0.9013**	**0.9013**	**0.9016**	**0.9016**	**0.8794**	**0.8796**	**0.8797**	**0.9425**	**0.7156**	**0.7158**	**0.7159**	**0.7157**
LSTM [92]																
Black-box	0.745	0.7456	0.7452	0.7452	0.8711	0.8712	0.8714	0.8715	0.6597	0.7187	0.6445	0.6905	0.5922	0.6155	0.6044	0.6006
BBO	0.7461	0.7488	0.7466	0.7466	0.8745	0.875	0.8745	0.875	0.7993	0.8129	0.797	0.8593	0.6127	0.6718	0.6717	0.6712
VSM	**0.7912**	**0.7913**	**0.7912**	**0.7912**	**0.9005**	**0.9005**	**0.9009**	**0.9009**	**0.8657**	**0.8667**	**0.8662**	**0.928**	**0.7171**	**0.7178**	**0.7177**	**0.7171**
GRU [93]																
Black-box	0.748	0.7493	0.7483	0.7483	0.8709	0.8708	0.8711	0.8712	0.6537	0.7006	0.6442	0.6902	0.6106	0.6266	0.6187	0.6165
BBO	0.74483	0.753	0.7493	0.75	0.8847	0.885	0.8851	0.8906	0.8193	0.8123	0.812	0.875	0.6478	0.6572	0.6522	0.6562
VSM	**0.8069**	**0.8069**	**0.8069**	**0.8069**	**0.9046**	**0.9047**	**0.9049**	**0.9049**	**0.8831**	**0.8834**	**0.8833**	**0.9041**	**0.7154**	**0.7159**	**0.7158**	**0.7154**

Table 6.2: Comparison of our test performances with the baseline neural architectures on four datasets. The VSM classifier achieves better performance than the black-box models. For the black-box models, we followed the implementation proposed by the authors of each baseline.

classifier and that of the black-box approach using the HealthLink data in Table 6.3. The number of parameters used by the VSM is less than that of each black box. Our model relies only on the embeddings and the **CSV**s. The number of parameters of the proposed classifier is the same for all architectures. Our model also reduced the inference time from $0.037 - 0.085$ seconds to 0.007 seconds, as shown in Table 6.3.

6.3.4 Interactive Explanations

Because we have evaluated the faithfulness of feature attribution in Chapter 3, we only focus on interactive explanations in this experiment. In some cases, end-users are interested in understanding the contribution of phrases instead of words. In addition, an end-user might be interested in understanding the contribution of a word/phrase w.r.t. other classes, not only to the predicted class. Results are shown in Table 6.4. Our method can identify the contributions of phrases instead of words and provide evidence w.r.t. other classes. For example, using the feature attribution,

Method	# of parameters	# of dropped parameters	Inference time
Transformer			
Black-box	3049602	83074	0.085
VSM	**2966528**		**0.007**
IndRNN			
Black-box	2991490	24962	0.037
VSM	**2966528**		**0.007**
Attention-based bi-LSTM			
Black-box	3229826	263298	0.056
VSM	**2966528**		**0.007**
Hierarchical recurrent network			
Black-box	3114754	148226	0.039
VSM	**2966528**		**0.007**

Table 6.3: Number of parameters used for black-box and our proposed model and the inference time.

"bad cough" has a stronger semantic contribution than "cough" w.r.t. the label "going to the hospital." Similarly, the difference between "mild chest pain" and "chest pain." In sentiment analysis, "good" contributes to positive sentiment, while "not good" contribute to negative sentiment. Also, note that "very good" contributes more importantly to a positive sentiment than "good."

Example	Pos	Neg	Example	Home care	Go to hospital	Example	Home care	Go to hospital
Good	0.756	-0.752	Cough	-0.984	0.984	Fever	0.933	-0.932
Not good	-0.151	0.144	Bad cough	-0.993	0.993	Bad fever	-0.962	0.952
Very good	0.877	-0.878	Cough+sore throat	-0.995	0.995	fever+headache	0.929	-0.929
Sucks	-0.607	0.607	Chest pain	-0.959	0.958	Cold	0.168	-0.170
Not sucks	0.688	-0.681	Mild chest pain	-0.861	0.861	Cold+chest pain	-0.934	0.934
Just sucks	-0.825	0.828	Chest pain+high blood pressure	-0.991	0.991	Cold+fever	-0.532	0.961
Sucks but very good	0.255	-0.262	Breathing	-0.968	0.968	Blood pressure	-0.980	0.980
Heart-warming	0.335	-0.3444	Breathing difficulty	-0.992	0.991	Bad blood pressure	-0.990	0.990
Heart-warming+entertaining	0.538	-0.54	vomiting+breathing	-0.883	0.883	High blood pressure	-0.981	0.981

Table 6.4: Explaining word/phrase contributions. The models are trained on IMDB and HealthLinl datasets. For phrase importance, we claculate the avergae of the embedding tokens and then using the class semantic vectors to calcualte the score.

6.4 Conclusion

We have explored an approach to knowledge distillation concurrently from a black-box model to produce a simple, interpretable classifier. The distilled model relies on the idea of the locally distributed vectors. We found that for simple classification problems, the model outperforms the black-box model. We have also shown that the proposed method is computationally efficient and employs less number of parameters as compared to the black box.

Chapter 7

Hierarchical Explanations

7.1 Introduction

In the past two years, several studies have focused on tackling hierarchical explanation rather than only individual feature attribution [19, 94–96]. One issue that remains, however, is that existing feature interaction techniques are based on post-hoc approaches. For example, a post-hoc approach such as in [19] used cooperative game theory ideas to generate interpretable explanations for higher-order interactions from a pre-trained black box. However, dissociating the model's prediction from its feature attribution can lead to explanations of poor quality. On the other hand, interpretable models typically incorporate an explanation representation layer into their architecture to support the provision of explanations for their predictions as an alternative to the post-hoc approach. Unlike previous studies on feature interactions, we propose learning interpretable representations to generate hierarchical explanations. Hierarchical explanations are essential to understand a deep model's decision-making process. Consider the following negative review for sentiment classification "a waste of excellent concert." Looking only at feature attribution might not tell us how words and phrases interact with each other and are composed together for the final prediction. However, using hierarchical explanations, we can see how the features interact with each other and thus give users a better understanding of the model prediction. The main contributions is three-fold: 1) We introduce a deep model capable

of learning interpretable representations, 2) The learned representations can be used to generate feature and phrase attributions to each prediction, and 3) Can also be used for hierarchical explanations. Our approach provides comprehensive picture of how different granularity of tokens interacting with each other within the model (see Figure 7.1.) Moreover, we address the limitations of the LDAV method: 1) We first address the problem of learning context-aware sentence vector, 2) We do not introduce additional representations such as the LDAV for each class; but instead rely on the proprieties of the network, 3) We focus on a challenging problem, i.e., feature interaction and hierarchical explanations.

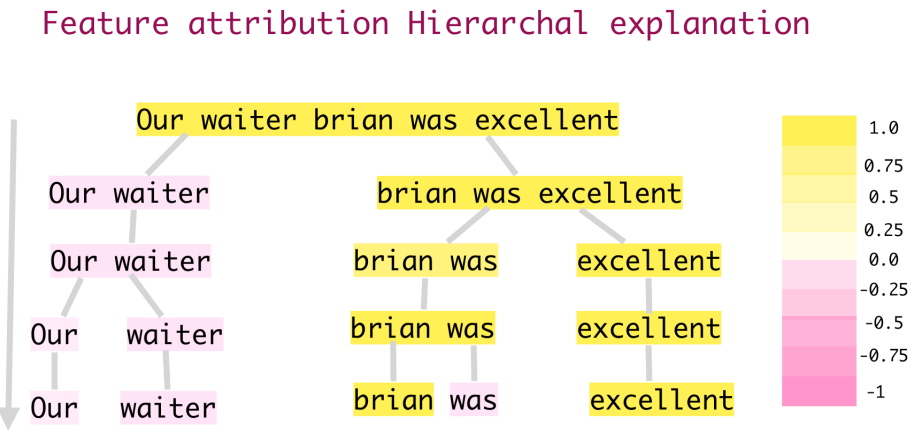


Figure 7.1: An example of the proposed deep model . We train a deep network for predictive accuracy and faithful explanations. The color of each token/phrase corresponds to the importance score.

7.2 Soft Faithful Feature Attribution

Here we present SFFA: Soft Faithful Feature Attribution, an effective approach to learn interpretable representations from deep nets, which can be used as a proxy to generate hierarchical explanations. We consider the example problem of predicting the sentiment of a textual movie review. Note that our approach does not require model re-training. In the following, we discuss how SFFA is integrated with deep networks to provide hierarchical explanations.

7.2.1 Background

Let $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_m)$ be a sequence of tokens of length m, for an input text. We use $\boldsymbol{x}_i \in \mathbb{R}^d$ to denote the feature embedding that represents the i^{th} token of the sentence, where d is the feature vector dimension. We consider probabilistic predictions and consider the prediction as a probability vector. The final layer $\boldsymbol{W}^{k \times d}$ of a typical deep model takes the context vector (obtained from Step 4 in Figure 7.2) to yield a vector of probabilities, where k is the total number of labels. This probability interpretation is implicitly informed by the statistical distribution that is approximated by the deep learning method. The output $\boldsymbol{y}$ from the model is a vector of class probabilities, and the predicted class $\hat{y}$ is a categorical outcome. The output of the network is defined as the inner product between the latent representation (e.g., context vector) and $\boldsymbol{W}$. Our problem is to learn an accurate predictor and to generate hierarchical explanations. The `SFFA` model consists of a predictor $\{(\cdot)$, and an explanation interaction generator, which are jointly trained. In the inference phase, each part can be invoked on-demand: to either predict the label $\{(\boldsymbol{x})$ for a new instance $\boldsymbol{x}$ or alternatively, to construct a prediction's hierarchical explanation.

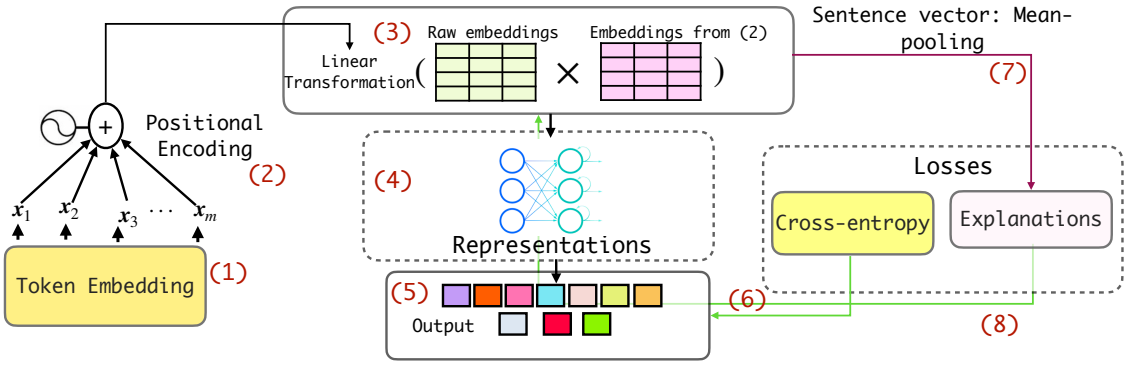


Figure 7.2: An example of the proposed intrinsic deep network model. The steps are summarized as follows: (1) Obtain the token embedding of each token, (2) Add the positional encoding information to each token, (3) Further modify the embedding using a linear transformation, (4) The representation layer over the new embedding features (e.g., CNNs, Multi-head attentions, LSTMs), (5) Output layer over the context vector to predict the class label, (6) Minimize cross-entropy to penalize incorrect predictions and update the network's weights, (7) Mean-pooling to obtain the sentence vector and, finally, (8) Apply the additional loss function to learn discriminative embedding features and use only the gradient to update through steps 1, 2 and 3.

7.2.2 Softmax Layer

The features extracted by a deep network are provided to a Softmax layer. The output of the fully connected layer is the multiplicative product of the weights (inner product) and the previous layer's output, plus bias. This is followed by using a Softmax activation which produces the probability for each class (these probabilities will sum to 1.) Cross-entropy (aka Softmax loss) is used to penalize incorrect predictions based on available ground-truth data. The cross-entropy is the sum of the negative logarithm of the probabilities. Consider the binary classification and we have a sample $\boldsymbol{x}$ from class 0. The Softmax loss goal is to force $\boldsymbol{W}_0^T\boldsymbol{x} > \boldsymbol{W}_1^T\boldsymbol{x}$ (i.e., $\|\|\boldsymbol{W}_0\|\|\,\|\|\,\boldsymbol{x}\|\cos(\theta_0) > \|\|\boldsymbol{W}_1\|\|\,\|\boldsymbol{x}\|\cos(\theta_1)$) in order to correctly classify $\boldsymbol{x}$ [97].

In other words, the objective of determining the Softmax loss is to push features in the same class to be closer, and to further separate samples from different classes. This makes the inter-class variances larger and intra-class variances smaller. We aim to use $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ to learn discriminative token embedding features. We also use $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ as a proxy for feature attribution and hierarchical explanation. One can think of $\boldsymbol{W}_i$ as a centroid vector of class i. Intuitively, during training, we use $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ to minimize the intra-class variations.

7.2.3 `SFFA`: Soft Faithful Feature Attribution

In this subsection we discuss the proposed deep model. We aim to learn effective embedding representations, i.e., identifying methods to optimize the inter-class difference (separating features of different classes) and reducing the intra-class variation (making features of the same class compact). The network's structure is summarized in Figure 7.2. `SFFA` makes two important changes to the network's architecture: 1) order-aware sentence representation, and 2) addition of a new term to the loss function. In addition, `SFFA` generates hierarchical explanations based on the embedding features and thus our goal is to learn interpretation-specific representations at the embedding layer.

Order-aware Attribution

One approach to learning interpretation-specific representations is to minimize the distance between the samples belonging to the same distribution in the latent space. For example, one could compute each sentence's mean-pooling and then minimize the distance between the sentence vector and the corresponding centroid vector of the positive class. However, the problem with using mean-pooling as a proxy to represent the sentence vector is that it does not consider word order when computing the mean-pooling. We propose injecting information about each token's absolute position. We add the positional encoding signal and then use element-wise multiplication as follows:

$$\tilde{\boldsymbol{x}}_i = (PE(i) + \boldsymbol{x}_i) \odot \boldsymbol{x}_i \tag{7.1}$$

where $PE(i)$ is the positional encoding vector [64] of token $\boldsymbol{x}_i$ at index i. Let, $\tilde{\boldsymbol{x}} = (\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_m)$ be the new sequence embedding features. The sentence vector $\bar{\boldsymbol{x}} \in \mathbb{R}^d$ of $\tilde{\boldsymbol{x}}$ is defined as the mean-pooling of the embedding features. The sentence vector is essential in learning the discriminative embedding features. As we can see, the mean-pooling of the embedding features will change, based on the position of the tokens. Consider the following examples for sentiment classification: *'I don't like the actor, but I really like the movie'* and *'I like the actor, but I really don't like the movie'*. The proposed approach will reflect the different meaning of these two sentences according to the position of their tokens.

Proposed Losses

Recall that the Softmax loss will push training data in the same class to be closer, and increase separation of samples from different classes. We exploit this property and apply the cosine distance over the sentence vector $\bar{\boldsymbol{x}}$. The loss is defined as follows:

$$\mathcal{L}^{embed} = 1 - \frac{\bar{\boldsymbol{x}} \cdot \boldsymbol{W}_{\hat{y}}}{\|\bar{\boldsymbol{x}}\| \, \|\boldsymbol{W}_{\hat{y}}\|} \tag{7.2}$$

where $\boldsymbol{W}_{\hat{y}}$ is the $\hat{y}$-th column of $\boldsymbol{W}$. During training, we use the stop-gradient operation which stops the accumulated gradient from flowing so that we do not compute the derivative of the above loss for the given weights $\boldsymbol{W}$.

The overall objective function of the network is based on combination of two losses:

$$\mathcal{L} = \frac{\mathcal{L}^{embed}}{\lambda} - \sum_{j=1}^{k} \boldsymbol{r}_j \log(\boldsymbol{y}_j) \tag{7.3}$$

where $\boldsymbol{r} \in \mathbb{R}^k$ is the one-hot represented ground truth, $\boldsymbol{r}_j$ is the target probability (0 or 1) for class j, and λ is a scaling factor. The value of λ is between $\{0.000001, 0.8\}$. We have experimented with different values for λ and we found the optimal value is 0.0006. 0.0006 is used as λ in the experiments. Let $\mathcal{L}^c$ be the cross-entropy loss. The back-propagation for the embedding $\frac{\delta \mathcal{L}}{\delta \boldsymbol{x}_i}$ for each token is given by

$$\frac{\delta \mathcal{L}}{\delta \boldsymbol{x}_i} = \frac{\delta \mathcal{L}^c}{\delta \boldsymbol{x}_i} + \frac{\frac{\delta \mathcal{L}^{embed}}{\lambda}}{\delta \boldsymbol{x}_i} \tag{7.4}$$

7.2.4 Feature Attribution and Interaction

In general, feature attribution focuses on estimating a contribution score for each token, to the model prediction. Here we describe the steps to find feature attribution and phrase attribution scores. Recall that the sentence vector provides a contextualized mean-pooling measure. It heuristically captures the intended semantics of the sentence. Equation 2 pushes the sentence towards the class vector $\boldsymbol{W}_{\hat{y}}$, which indirectly forces the salient tokens to be close to $\boldsymbol{W}_{\hat{y}}$. For a given token $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^d$, the attribution score is:

$$\Phi(\tilde{\boldsymbol{x}}_i, \boldsymbol{W}_{\hat{y}} | \hat{y}) = \frac{\tilde{\boldsymbol{x}}_i \cdot \boldsymbol{W}_{\hat{y}}}{\|\tilde{\boldsymbol{x}}_i\| \|\boldsymbol{W}_{\hat{y}}\|} \tag{7.5}$$

Equation 7.5 is the cosine similarity between $\tilde{\boldsymbol{x}}_i$ and $\boldsymbol{W}_{\hat{y}}$. The cosine angle is used to validate whether the token $\tilde{\boldsymbol{x}}_i$ is pointing roughly in the same direction as $\boldsymbol{W}_{\hat{y}}$. A higher score indicates that the angle between $\boldsymbol{W}_{\hat{y}}$ and $\tilde{\boldsymbol{x}}_i$ is smaller. Similarly the

phrase attribution score is defined as:

$$\Phi(\boldsymbol{z}, \boldsymbol{W}_{\hat{y}}|\hat{y}) = \frac{\boldsymbol{z} \cdot \boldsymbol{W}_{\hat{y}}}{\|\boldsymbol{z}\|\|\boldsymbol{W}_{\hat{y}}\|} \tag{7.6}$$

where $\boldsymbol{z} \in \mathbb{R}^d$ is the pooled-mean of the phrase/span $(\tilde{\boldsymbol{x}}_1, ..., \tilde{\boldsymbol{x}}_n)$ and n is the total number of tokens in the span.

7.2.5 Hierarchical Explanation

In addition to feature and phrase attribution, `SFFA`'s network can also be used to generate hierarchical explanations, i.e., it can segment a text recursively into phrases and then words for explanation. Similar to the work of [19], we use a top-down approach to partition the tokens into subsets. The hierarchical explanation exploits these multiple levels, where each level consists of multiple subsets (except level 0). We divide each subset into smaller text subsets according to the positions of the low interaction. The interaction score between two subsets (left,right) is defined as follows:

$$\phi(\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2, \boldsymbol{W}_{\hat{y}}|\hat{y}) = \left| \frac{\bar{\boldsymbol{s}}_1 \cdot \boldsymbol{W}_{\hat{y}}}{\|\bar{\boldsymbol{s}}_1\|\|\boldsymbol{W}_{\hat{y}}\|} - \frac{\bar{\boldsymbol{s}}_2 \cdot \boldsymbol{W}_{\hat{y}}}{\|\bar{\boldsymbol{s}}_2\|\|\boldsymbol{W}_{\hat{y}}\|} \right|, \tag{7.7}$$

where $\bar{s}_1$, and $\bar{s}_2$ represent the mean-pooling of subsets 1 and 2, respectively.

Algorithm 1 Top-down approach for hierarchical explanation

input : Token embedding $\tilde{x} \in \mathbb{R}^{m \times d}$, $\boldsymbol{W}_{\hat{y}} \in \mathbb{R}^{d}$
Initialize the first subset $\mathcal{S} \leftarrow []\{\tilde{x}_{(0,m]}\}$
Initialize the hierarchy $\mathcal{H} = [\mathcal{S}]$
for $level \leftarrow 1$ **to** $m - 1$ **do**
  Initialize minimum score $sc = 2$
  foreach $s \in \mathcal{S}$ **do**
    if $|s|$ *=1* **then**
      Continue
    end
    $l \leftarrow |s|$
    Initialize scores $\rho = \emptyset$
    for $st \leftarrow 1$ **to** l **do**
      $\alpha \leftarrow \phi(sub_{(0,st]}, sub_{(st,l]}, \boldsymbol{W}_{\hat{y}}|\hat{y})$
      $\rho.add(\alpha)$ Add interaction score
    end
    if $\min(\rho) < sc$ **then**
      $\xi \leftarrow \arg\min(\rho)$ index
      $sc \leftarrow \min(\rho)$ score
      $s^{(t)} \leftarrow s$
    end
  end
  $s^{(l)} \leftarrow s^{(t)}_{(0,\xi]}$ Get left subset
  $s^{(r)} \leftarrow s^{(t)}_{(\xi,:]}$ Get right subset
  $\Gamma \leftarrow \forall e \in \mathcal{S}, e \neq s^{(t)}$ Find other subsets except picked one
  $\Gamma \leftarrow \Gamma \cup \{s^{(l)}\}$ Add left subset
  $\Gamma \leftarrow \Gamma \cup \{s^{(r)}\}$ Add right subset
  $\mathcal{S} \leftarrow \Gamma$ Start from this subset
  $\mathcal{H}.add(\mathcal{S})$
end
output: $\mathcal{H}$

The main algorithm for partitioning the tokens into different text spans is summarized in Algorithm 1. The objective is to segment any subset which contains more than two words. The split is based on finding the minimum interaction between two subsets using Equation 7.7. We search all possible subsets, then pick the partitions to split which result in the minimum interaction score between the two subsets.

7.3 Experiments

`SFFA` is evaluated on text classification problems with three different deep learning architectures: an attention bi-directional LSTM (Attbilstm) [63], a convolution neural network (CNN) [98], and RoBERTa [69]. Each deep learning method is applied to three benchmarks: IMDB [59], YELP [99], and AG news [60]. Similar to [19], we focus on sentiment classification (IMDB, YELP), and in addition, we focus on document classification (AG news). For evaluation, we rely on proxy metrics and human evaluation to demonstrate the effectiveness of `SFFA` in identifying faithful explanations.

7.4 Summary of Datasets and Implementations

The summary of the benchmark datasets is shown in Table 7.1. We use 10% of the training data as the validation set. The networks were trained on an NVIDIA GeForce RTX 3070 8 GB GDDR6. The reported results were based on the average of 2 runs. Links to the public datasets: 1) IMDB, 2)YELP, 3) AG news and 4) SNLI. Number of parameters for Attbilstm is: 1) IMDB:3, 624, 706 , 2) YELP:3, 624, 706 , 3) AG news: 16, 342, 788, for CNN: 1) IMDB:3, 361, 282 , 2) YELP:3, 361, 282 , 3) AG news: 16, 079, 364, for NLI: 1)8054020 and RoBERTa: 1)IMDB:82, 710, 530 , 2) YELP:82, 710, 530 , 3) AG news: 82, 712, 068. The average inference time (prediction and generating attribution scores) for Attbilstm on YELP(input length: 50, number of samples: 1024) is 0.000868 second. For a CNN trained on AG news, the average inference time is 0.0008495 seconds. We used Tensorflow version 2 for implementing the proposed approach. All datasets used in the experiments are publicly available. We manually checked a few samples from the datasets for offensive contents.

Training details The embedding vector and hidden layer feature vectors were set to 256. We use Adam optimizer with a learning rate of 0.0001 and a batch size of 512. W was experimented with different values of λ (0.001 for sentiment classification

Datasets	Labels	Average length	Train	Test
IMDB	2	50	25000	25000
YELP	2	50	110400	27600
AG news	4	20	102080	25520

Table 7.1: Summary statistics for the benchmarks. Dataset language: English.

and 0.00001 for AG news and SNLI). We train for a maximum of 800 epochs with early stopping if the validation-set score has not improved in 20 consecutive epochs. Moreover, performance evaluation between the alternative deep networks, with and without `SFFA`, is shown in Table 7.2. We found that performances are almost similar with no significant degradation.

	Attbilstm			CNN		
	IMDB	YELP	AG news	IMDB	YELP	AG news
Baseline (without `SFFA`)	0.792	0.89	0.88	0.797	0.856	0.876
`SFFA`	0.79	0.89	0.88	0.79	0.853	0.886

Table 7.2: Model's accuracy on three benchmarks

7.4.1 Token-level Evaluation

We focus on local fidelity, and use proxy metrics for evaluation. We adopt two metrics from related work: log-odds scores [16] and ERASER [100].

We calculate the AOPC for both comprehensiveness and sufficiency using a variety of token percentages (u values): $5\%, 10\%, 15\%, 20\%$, and 25% for IMDB and YELP, and $25\%, 35\%, 45\%, 55\%$, and 65% for AG news.

Results we compare our method with several baselines, such as LIME [6], IntGrad [65], L-Shapley and C-Shapley [33], and HEDGE [19].

The log-odds experiments for Attbilstm are shown in Figure 7.3. The results show that our approach better captures faithful features essential for the final prediction compared with traditional post-hoc techniques.

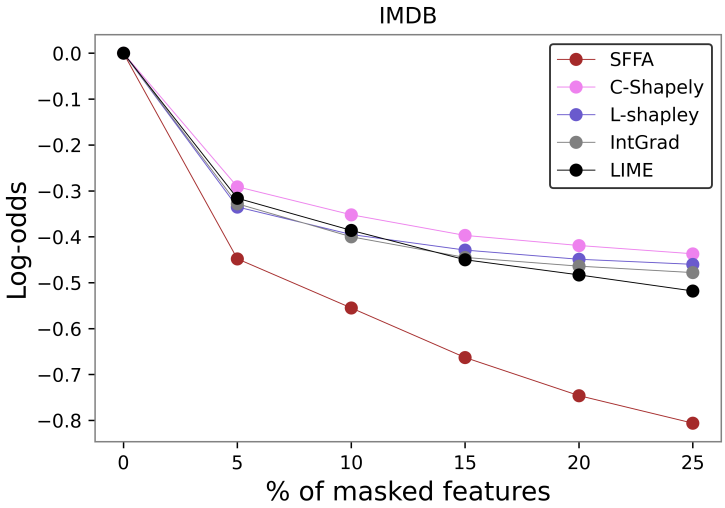


Figure 7.3: Log-odds as a function of masked tokens trained on Attbilstm.

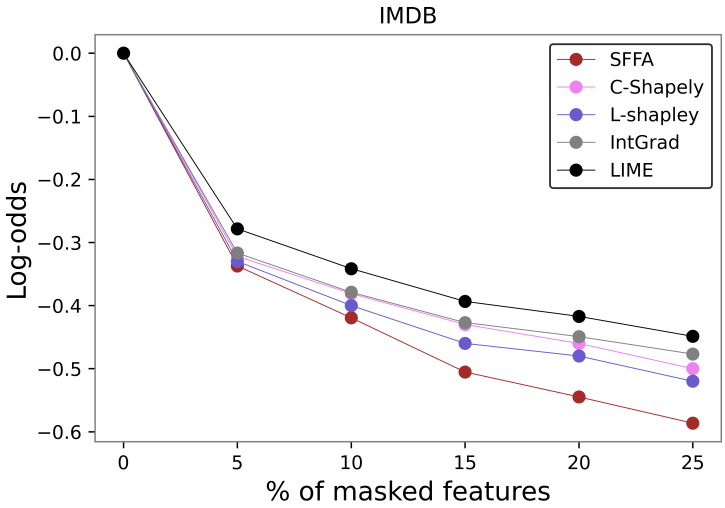


Figure 7.4: Log-odds as a function of masked feature trained on CNN.

Similarly, the CNN's results for log-odds are shown in Figure 7.4. `SFFA` outperformed post-hoc methods in both metrics. Table 7.3 compares the scores by different explanation methods for ERASER. We found that `SFFA` results in a significant improvement in both comprehensiveness and sufficiency for sentiment and news classification.

		SFFA	L-Shapley	C-Shapley	IntGrad	LIME			SFFA	L-Shapley	C-Shapley	IntGrad	LIME
Attbilstm	**IMDB**						**CNN**	**IMDB**					
	Comprehensiveness	0.643	0.4136	0.127	0.423	0.459		Comprehensiveness	0.476	0.438	0.418	0.408	0.375
	Sufficiency	0.020	0.083	0.101	0.061	0.185		Sufficiency	−0.134	-0.125	−0.118	−0.115	0.014
	YELP							**YELP**					
	Comprehensiveness	0.631	0.406	0.394	0.402	0.439		Comprehensiveness	0.513	0.468	0.466	0.472	0.207
	Sufficiency	0.110	0.266	0.268	0.150	0.234		Sufficiency	−0.138	−0.133	−0.32	−0.141	0.011
	AG news							**AG news**					
	Comprehensiveness	0.721	0.295	0.259	0.483	0.291		Comprehensiveness	0.684	0.212	0.167	0.351 higher	0.275
	Sufficiency	0.003	0.07	0.089	0.031	0.103		Sufficiency	−0.021	0.134	0.162	0.044	0.111

Table 7.3: Eraser benchmark scores: Sufficiency and comprehensiveness are in terms of AOPC. Lower scores are better for sufficiency and higher scores are better for comprehensiveness.

7.4.2 Hierarchical Explanations

Here we focus on evaluating the quality of hierarchical explanations. We adopt the cohesion-score metric proposed by [19] to evaluate the salient spans identified by each method. Given a salient span $\boldsymbol{x}_{(a,b]}$, we randomly select a position in the token sequence $\boldsymbol{x}_1, ..., \boldsymbol{x}_a, \boldsymbol{x}_{b+1}, ..., \boldsymbol{x}_m$ and re-insert a word. The process is repeated until a shuffled version of the original sentence$\boldsymbol{x}^{(q)}$ is constructed. Intuitively, the words in an important text span have strong interactions. By perturbing such interactions, we expect to observe the output probability decreasing. The cohesion score is defined as follows:

$$\texttt{cohesion} = \frac{1}{a}\sum_{i=1}^{a}\frac{1}{100}\sum_{q=1}^{100}(p(\boldsymbol{x}_i|\hat{y}) - p(\boldsymbol{x}_i^{(q)}|\hat{y})), \tag{7.8}$$

where $\boldsymbol{x}_i^{(q)}$ is the q^{th} perturbed version of $\boldsymbol{x}_i$. We repeat the experiment 100 times. Only salient spans are considered in this evaluation. Higher scores are better, which means the identified spans are more critical than others for predicting the label.

Results Table 7.4 compares the cohesion score between SFFA and HEDGE on three benchmarks using a CNN and an Attbilstm. The results indicate that SFFA is better at capturing the interaction and identifying salient subsets from the sentence than using a post-hoc approach. The advantage of this approach over post-hoc is that explanation-specific representations are learned directly by the deep model and without an intermediate model.

Methods	Models	IMDB	YELP	AG news
		Cohesion-score		
HEDGE	CNN	0.092	0.079	0.052
	Attbilstm	0.071	0.055	0.023
SFFA	CNN	0.129	0.113	0.094
	Attbilstm	0.099	0.191	0.052

Table 7.4: Cohesion scores between SFFA and HEDGE. Higher scores are better.

7.4.3 Human Evaluation

Our human evaluators were undergraduate and graduate students from computing science (a total of 10). For both SFFA and HEDGE, we consider only the most important span/phrase from each sentence as a proxy for each model's explanation (please note that this is not the full sentence). Similar to [19], we focus on sentiment classification and provide sentences from IMDB and YELP.

We ask evaluators to predict the type of the sentiment from the provided explanation [15] from "Positive," "Negative," "N/A", where "N/A" means that the evaluator cannot predict the sentiment from the provided explanations. An example of the form used for human evaluation is shown in Figure 7.5.

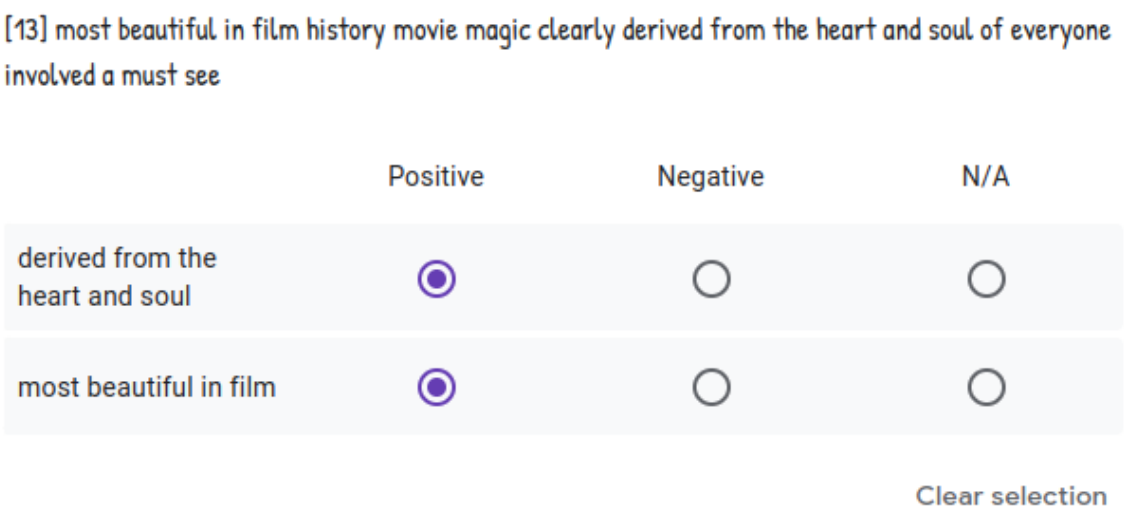


Figure 7.5: The GUI used for human evaluation experiment. The review and the subset with the highest interaction is provided to the user to predict the sentiement.

The model used in this study was trained using an Attbilstm. We randomly picked 100 reviews from the two benchmarks. We measure the number of human annotations coherent with the model's prediction and then define the coherence score as the ratio between the coherent annotations and the total number of examples [19].

Results Table 7.5 compares the coherence score between each of SFFA and HEDGE explanations and the human annotation. SFFA outperformed HEDGE achieving relatively better scores which suggests that it is better aligned with human annotation at identifying important spans from the reviews.

Methods	Coherence score
HEDGE	0.51
SFFA	0.8136

Table 7.5: Human evaluation of SFFA and HEDGE with Attbilstm on IMDB and YELP benchmarks.

7.4.4 SFFA for Pre-trained Transformers

We also applied SFFA to language models to generate faithful explanations. We evaluate the method on three benchmarks. We use RoBERTa [69]. We incorporate SFFA for Transformers by allowing gradients from the explanation-related loss function to pass through the token embedding. The output layer is fine-tuned for the downstream classification task. Due to page limits, we only report the results on ERASER for token-level evaluation. We limit our experiments to Shapley-based methods because of challenges with baseline implementation. For hierarchical explanations, we again use the cohesion-score metric to evaluate the quality of the generated spans.

Results The performance comparison is summarized in Table 7.6. Table 7.7 compares ERASER's scores on three benchmarks against Shapley-based methods. SFFA outperforms post-hoc methods in both metrics achieving better scores. Moreover, the cohesion scores for RoBERTa are shown in Table 7.8.

	IMDB	YELP	AG news
Baseline without SFFA	0.838	0.916	0.9074
SFFA	0.8401	0.915	0.8924

Table 7.6: RoBERTA: Model's accuracy.

	SFFA	L-Shapley	C-Shapley
IMDB			
Comprehensiveness	0.506	0.192	0.146
Sufficency	0.085	0.166	0.159
YELP			
Comprehensiveness	0.497	0.204	0.187
Sufficency	0.073	0.172	0.166
AG news			
Comprehensiveness	0.535	0.128	0.127
Sufficency	0.035	0.125	0.125

Table 7.7: ERASER benchmark score: Comprehensiveness and sufficiency are in terms of AOPC. Results are based on RoBERTA's model.

Methods	Models	IMDB	YELP	AG news
		Cohesion-score		
HEDGE	RoBERTa	0.117	0.096	0.006
SFFA	RoBERTa	0.151	0.131	0.032

Table 7.8: Comparing the cohesion scores between SFFA and HEDGE for RoBERTA.

7.4.5 Ablation Study

To further understand λ in the SFFA objective function, we plot the log-odds score as a function of λ. We use the AG news and IMDB trained using a CNN, mask the top five tokens from each sample, and then analyze the change in the log odds. The results in Figure 7.6 show that the smaller value of λ means a lower log-odds score.

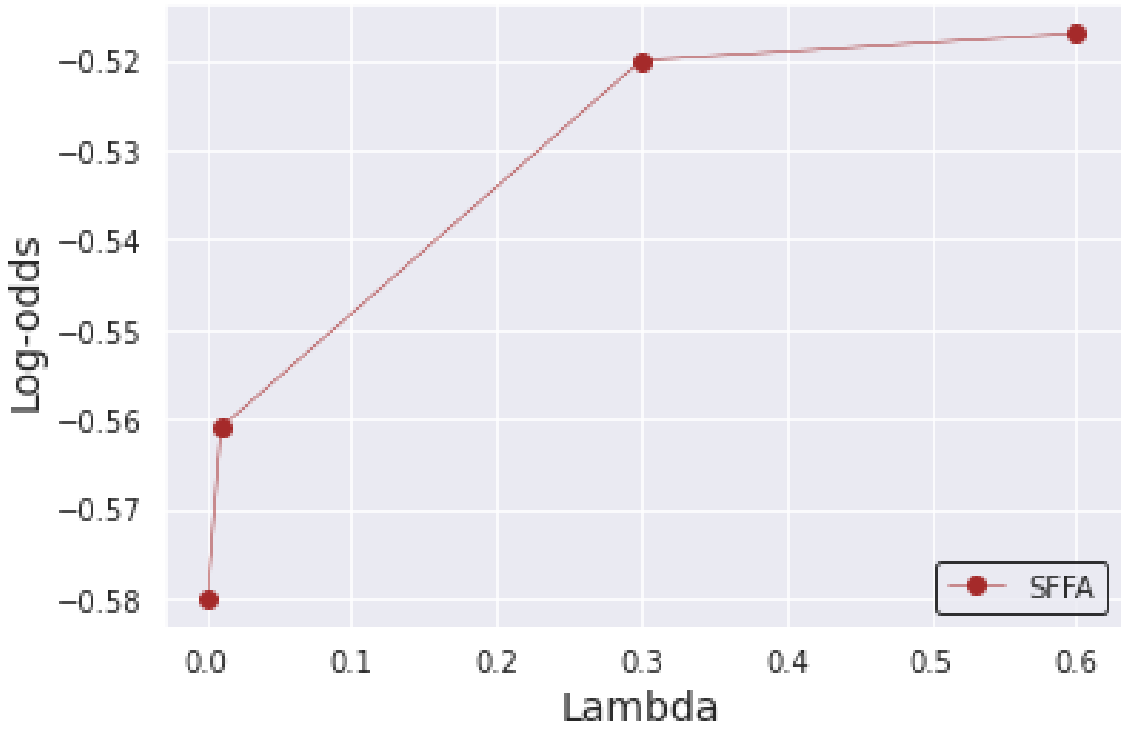


Figure 7.6: Log-odds as a function of λ on AG news. We found that the smaller the value of λ, the better the faithfulness of the explanation.

7.4.6 Qualitative Results

Here we provide additional examples of hierarchical explanations from a sentiment classifier (see Figures 7.7 - 7.11). They show that `SFFA` is capturing meaningful spans (each color in the subset represents the importance score for the token/phrase to the final prediction.) The hierarchical explanation illustrates that the model is capturing the most salient span with the minimum number of tokens.

7.4.7 Discussion

The importance of concurrent supervision If we rely on the `SFFA` as the primary signal, the resulting learned token embedding features would have meaningful representations that can be used to faithfully explain the network's prediction. The visualization of the learned embedding features in Figure 7.12 shows good discrimination of the embeddings with a clear geometric interpretation. The `SFFA` approach

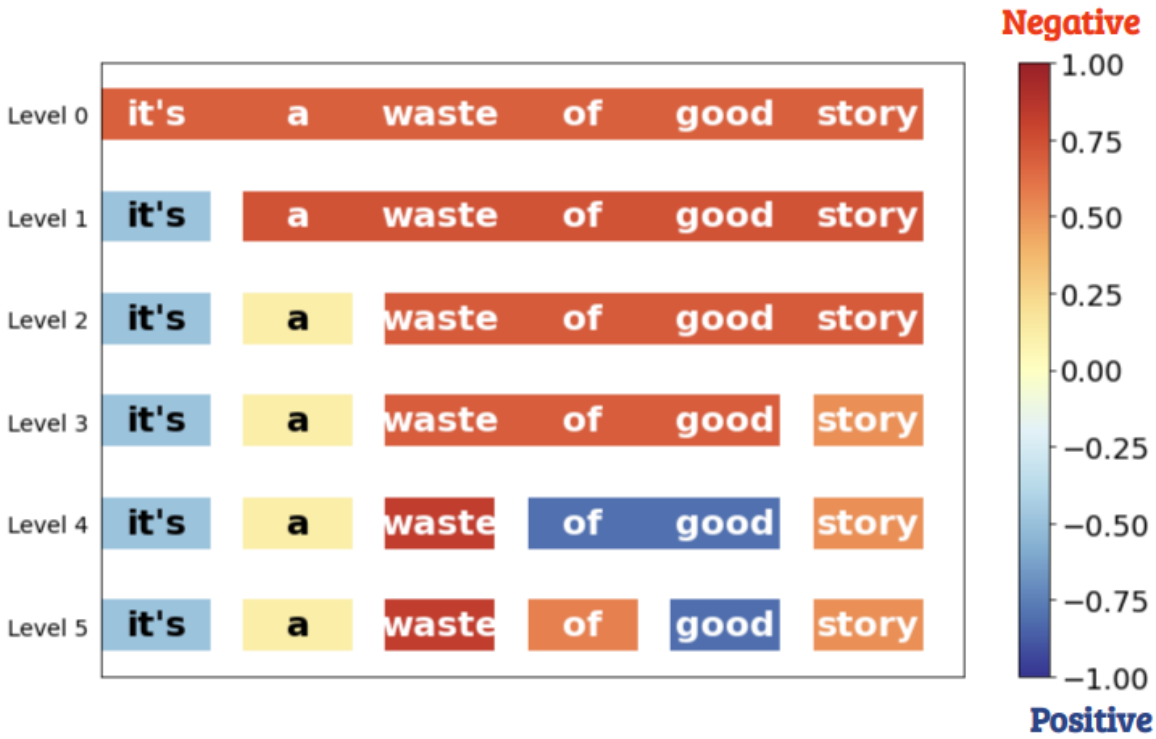


Figure 7.7: An example for negative sentiment classification. Numbers on the right bar represent the range of the scores based on the color.

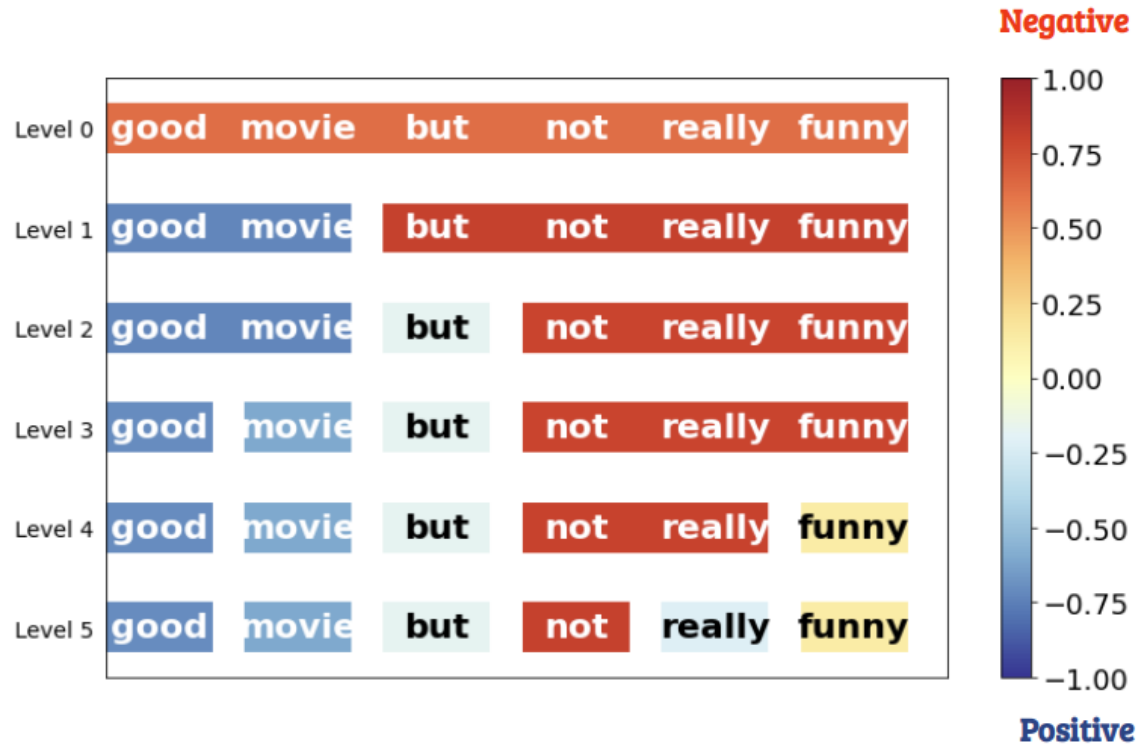


Figure 7.8: `SFFA` for Attbilstm on a negative review. The model correctly captures the salient interaction `not really funny`.

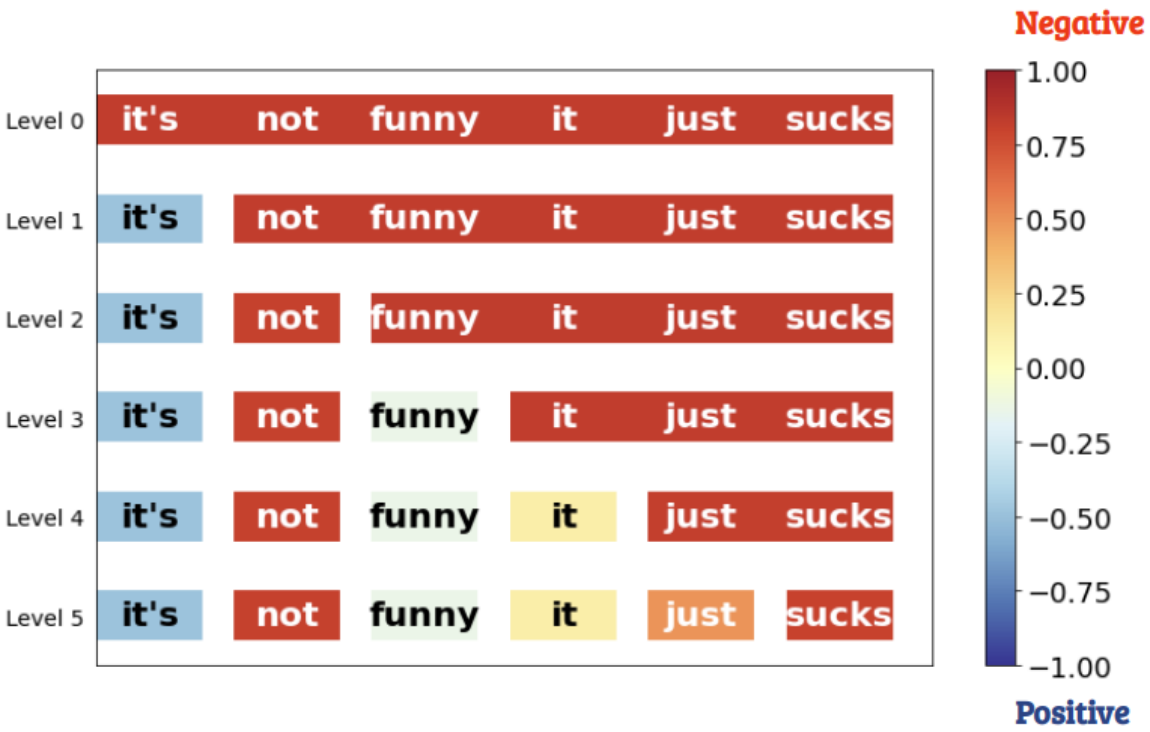


Figure 7.9: `SFFA` for Attbilstm on a negative review. The model correctly captures the salient interaction `just sucks`.

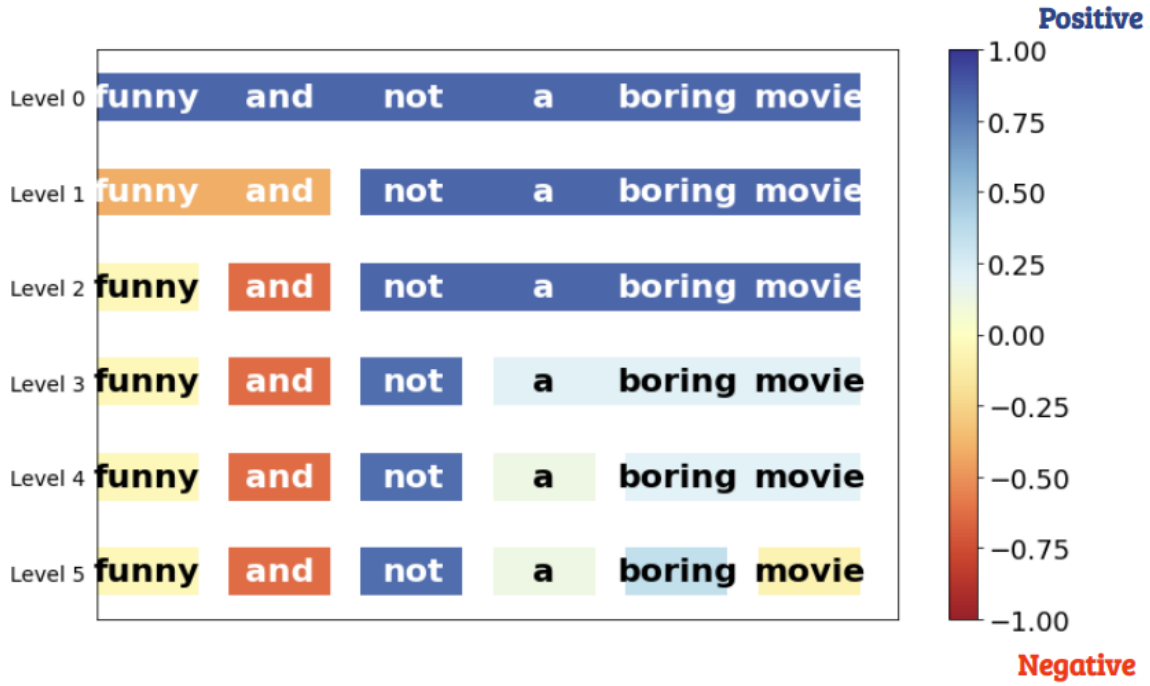


Figure 7.10: `SFFA` for Attbilstm on a positive review. Then negation token `not` was important for the model to predict the positive sentiment.

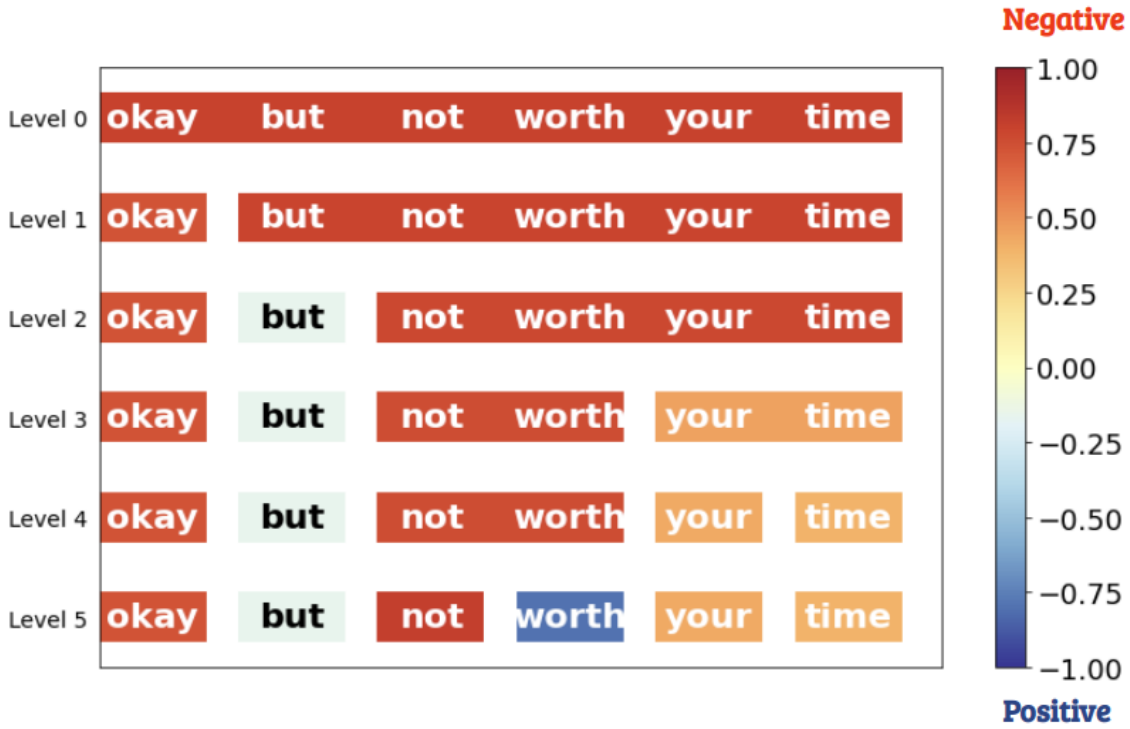


Figure 7.11: `SFFA` for Attbilstm on a negative review. Most salient span is `not worth`.

forces the features of the same class to form a unique cluster, enabling it to be used along with cosine similarity to obtain attribution scores. The results show that there was no significant difference between deep net training with and without the proposed loss; it implies that the models were neither shallow nor linear.

Compared to existing methods Our method does not use complex models to learn explanations nor does it use post-hoc techniques to learn feature attribution. The proposed method is built on the main properties of the deep network, more specifically the Softmax loss. SFFA improves the representations of the embedding layer and thus enables the deep net to generate faithful explanations compared to post-hoc approaches.

Rationalization methods This work is different from existing methods on rationalization methods. It forces deep networks to learn representations that can be used as a proxy for hierarchical explanations. Thus, comparing with rationalization methods is challenging because of the generated output and the objective, plus the

need for more relevant metrics. We also believe that such evaluation is not fair.

Order-aware Attribution Although positional encoding enables capturing the position of the tokens, calculating the mean pooling for the sentence vector does not really consider the order of the tokens. Hence, the proposed remedy tackles the problem without sacrificing performance.

Feature interaction We use the idea of interactions to generate hierarchical explanations. Note that the hierarchical structure emerging from the rationale identification provides the basis for guiding the explanatory interaction at several levels of detail. So the feature interaction can provide a subset's interaction score (e.g., what is the interaction score between two tokens), but the hierarchical explanation automatically discovers the highest interaction between tokens in a given sentence. Consider the example of predicting the sentiment, "a waste of a good story" (prediction: negative). With feature interaction, one could randomly pick a subset and find the interaction. However, with hierarchical explanation one can automatically identify the highest interaction in the sentence

7.5 Conclusion

We have introduced a new intrinsic neural model which does not require additional parameters to generate an explanation. The network learns an appropriate interpretation-specific representation. We have demonstrated the effectiveness of `SFFA` in learning faithful interpretations compared with traditional post-hoc approaches. Our approach does not use existing predefined properties in the literature but instead relies on the structure of the deep model. Additionally, we extend our idea to generate hierarchical explanations using a top-down approach.

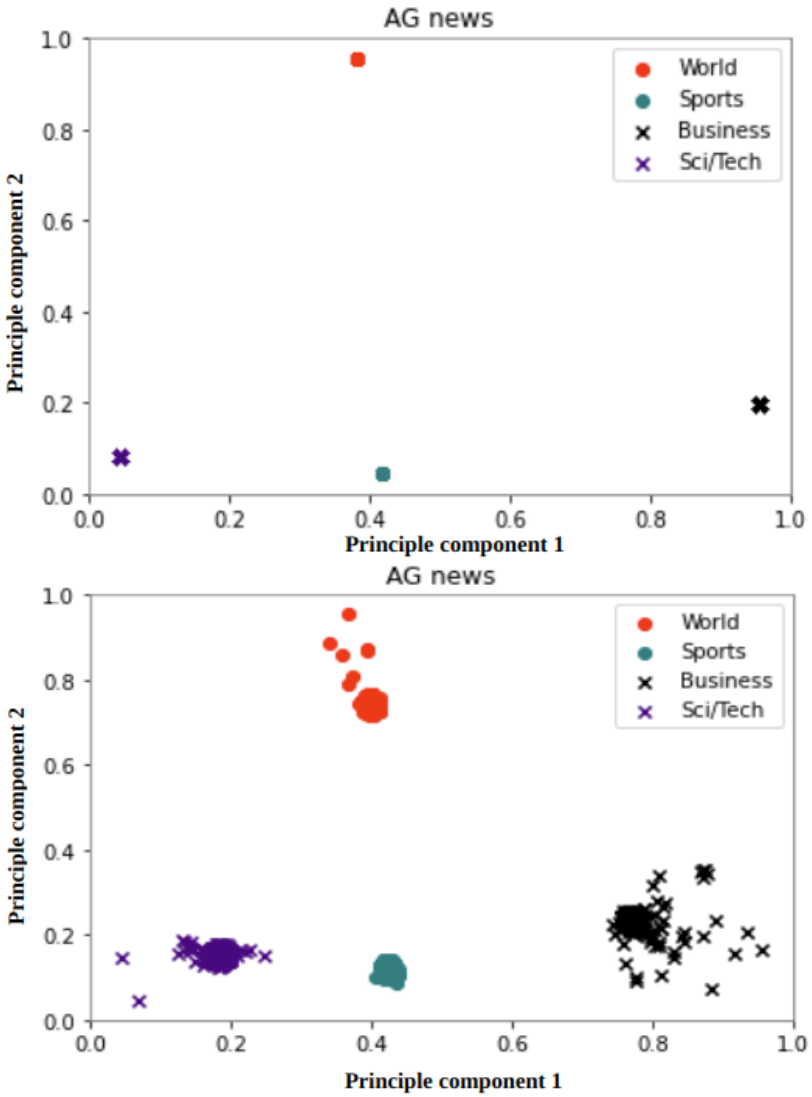


Figure 7.12: Distributions of the learned token embeddings (right) and their corresponding weight vector (left) from the output layer after using our SFFA.

Chapter 8

Conclusion and Future Work

In this chapter, we summarize our contributions, limitations, and future directions that could be addressed moving forward with our conclusions. Throughout this dissertation, we have investigated eXplainable AI related work as a remedy to model understanding using explainability.

8.1 Summary of Contributions

To summarize the thesis, we contribute the following:

- **Interpretable neural networks**: Chapter 3 introduced an approach to constructing a deep interpretable network. Our approach is based on theories from contrastive learning methods. We showed that learning interpretable models provides more trustworthy interpretations than post-hoc approaches. In doing so, we did not observe any significant degradation in the model's performance when constructing interpretable models. In Chapter 4, we proposed an approach to learning a rationale as the model's explanation of its predictions. In all chapters, we found that building inherently interpretable models always provides a more faithful explanation than those based on mimicking a pre-trained model's behavior.

- **Contrastive explanations**: In Chapter 5, our results showed that we could use `LDAV` to provide contrastive explanations, i.e., answers to "why p and not

q?" questions. This type of explanation is also essential when non-contrastive explanations are not helpful. We proposed proxy metrics for faithfulness evaluation to evaluate the quality of contrastive explanations.

- **Self-distillation**: In Chapter 6, we introduced an approach for self-distillation, i.e., learning an interpretable model while training the black box. Our results showed that we could learn a vector space model (`VSM`) concurrently with the black box. In addition, the `VSM` employs fewer parameters than the black-box; it only relies on k vectors. Hence, the `VSM` is a lightweight version of the black box. Our results suggest that a deeply learned model can be compressed into a set of vectors (only using $\sim<$ 50% of the original parameters) without sacrificing performance for document classification.

- **Hierarchical explanations**: As discussed in Chapter 7, deep networks learn from higher-order interaction between the input features. Thus, feature interaction is as important as feature attribution. Unlike existing post-hoc methods, we proposed an interpretable model to learn representations from the network's property to support hierarchical explanations.

8.2 Discussion

Here we discuss the challenges with visualization, interaction, and the importance of multiple explanations.

8.2.1 Visualization

We found that visualization is critical to the success of an XAI technique. Here we discuss a few points about visualizations:

- There is no single solution to the best visualization result; it mainly depends on the explainee (end-user) and the task. As a result, providing multiple visualizations overcome the belief bias in the explainer.

- The explainee tries to understand how the model reasons by analogy to their knowledge. Some of the leading causes why end-users might not fully understand the predictions of a model are as follows: 1) the complexity of the sentence structure affects how the explainee perceives the information, 2) the complexity of the task, 3) the quality of the explanation generated by the model, and 4) overall understating of the model's objective.

- We must also investigate how explainees respond to different visualizations for feature attribution. We must recognize the importance of visualization. A robust explanation method can only be fully utilized with proper visualization.

8.2.2 Interaction

We showed that end-user interaction is helpful for an XAI method. End-users interaction enables them to study how a model analyzes the attribution and how the attributions can influence the prediction.

8.2.3 The Need for Multiple Explanations

We should focus on building multiple levels of explanations to meet end-users expectations. Doing so allows end-users to switch between different types of explanations to gain meaningful insights.

8.3 Limitations

Here we discuss the existing limitations of the proposed methods.

- **Limited abstractions**. One of the main limitations of an XAI approach is the limited levels of abstraction. This limitation is not due to the explanation method but rather to the model being explained (e.g., a deep neural network). The quality of the expression language of the proposed methods is subject to the vocabulary used to learn the downstream task. The dataset (a proxy for

representing the world) is the primary source of knowledge for training a supervised model. Suppose the model does not access external resources about the world for training. In that case, the explanation is limited to the input only. Humans generally provide better explanations as they can access external knowledge about the world compared to a learned model.

- **Computational cost**. We observed that it took more learning time (5x slower than the model with proposed solutions) because of the constraint added to the loss function.

- **XAI**. Our current contributions are limited to feature attribution, contrastive and hierarchical explanations. However, many other possible dimensions exist for a model's explanation, including counterfactual explanations, rules extraction, and beyond.

8.4 Future Work

For our future work we leave two research questions:

- **Proxy Metrics:** Majority of existing works mainly focus on evaluating the plausibility of the explanations and ignore the faithfulness. We think that it is important to distinguish between plausibility of an explanation and the faithfulness of an explanation.

- **Improving the quality of visualizations:** We need tools that can translate the requirements from regulators into usable solutions. These systems should focus on converting the results of XAI methods into useful tools. The current visualizations are not ready where society can use and are far from the certification of AI systems.

8.5 Closing Remarks

Deep neural models generally identify patterns found in variables and the relations among those variables. Unfortunately, the learned representations from these variables can be so complex that they can be difficult for humans to understand or interpret. The complexity of learned representations creates a severe problem, especially when the variables we input into the deep model provide critical decisions. As a result, the deep models are generally referred to as a "black box," making it challenging for humans to answer crucial questions about the learned representation and the prediction explanation. The need to answer these questions leads to the demand for explainable AI: The ability to generate explanations for humans.

8.5.1 What Makes a Good Explanation

Explainable AI techniques should provide meaningful answers to their intended audience and be easy to comprehend so that users can confidently come to a conclusion or make a recommendation. However, the explanation might not meet the end user's expectations due to the nature of the learning models. However, plausibility is a desired property. What matters is that the explanations need to reflect the actual decision-making process used by the model to make a single prediction. Thus, the vocabulary of the explanation must meet end-users expectations. For instance, an explanation to a doctor might be completely different from the explanation of the model's developer. However, providing multiple explanation levels is challenging, as discussed in the limitation section. This line of research requires the community to investigate different visualization techniques.

8.5.2 When Explanations are Needed

Generating explanations is an expensive process that requires time and resources. Nevertheless, it is essential to asses when the outputs of the models need to be explained. Understanding the model's prediction requires developers, stakeholders,

and regulators to identify the risk associated with the model's predictions.

8.5.3 Explanations as a Proxy for Model's Debugging

In some cases, explainability might help significantly improve the model's accuracy against relevant benchmarks by tweaking the features without a deep understanding of how the black box operates. Debugging deep networks using explanations will help end-users identify weaknesses and enable researchers addresses biases in the predictions.

8.6 Explaining pre-trained models

Explainable AI tackles complex problems, such as the faithfulness and plausibility of the explanation. Therefore, the complexity of the problems results in a cloudy big-picture regarding practical gain and different sub-problems' expectations. We draw the final closing remarks: 1) If we can build an interpretable model, this would be the ideal solution, so we will not need a post-hoc approach. 2) To generate faithful explanations for a pre-trained model, we provide a simple recipe to select the best approach in Figure 8.1. We include only the most effective techniques from a personal experience in terms of faithfulness.

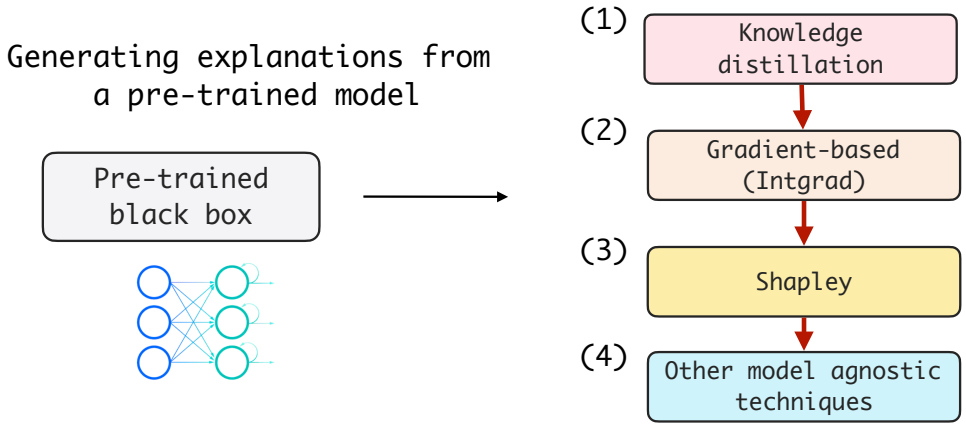


Figure 8.1: Steps required to select an interpretation method for a pre-trained model.

Bibliography

[1] D. Robinson, "You better explain yourself, mister: Darpa's mission to make an accountable ai," *visited on August*, 2022.

[2] J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann, "Confounding variables can degrade generalization performance of radiological deep learning models," *arXiv preprint arXiv:1807.00431*, 2018.

[3] "General data protection regulation," 2020, April. 2020. [Online].

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] C. Rudin, "Please stop explaining black box models for high stakes decisions," *32nd Conference on Neural Information Processing Systems (NIPS 2018), Workshop on Critiquing and Correct- ing Trends in Machine Learning.*, 2018.

[6] M. T. Ribeiro, S. Singh, and C. 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*, ACM, 2016, pp. 1135–1144.

[7] T. Lei, R. Barzilay, and T. Jaakkola, "Rationalizing neural predictions," in *Proceedings of EMNLP*, 2016, pp. 107–117.

[8] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," in *Proceedings of ICLR*, 2017.

[9] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[11] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[12] J. Doe, "Darpa xai," *visited on August*, 2022.

[13] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, IEEE, 2018, pp. 0210–0215.

[14] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, 2017, pp. 8–13.

[15] D. Nguyen, "Comparing automatic and human evaluation of local explanations for text classification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1069–1078.

[16] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, "L-shapley and c-shapley: Efficient model interpretation for structured data," *ICLR 2019*, 2018.

[17] J. DeYoung *et al.*, "Eraser: A benchmark to evaluate rationalized nlp models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4443–4458.

[18] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, "A benchmark for interpretability methods in deep neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[19] H. Chen, G. Zheng, and Y. Ji, "Generating hierarchical explanations on text classification via feature interaction detection," in *ACL*, 2020.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[21] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the International Conference on Machine Learning*, 2017, pp. 3145–3153.

[22] L. Arras, G. Montavon, K.-R. Müller, and W. Samek, "Explaining recurrent neural network predictions in sentiment analysis," in *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis in ACL*, 2017, pp. 159–168.

[23] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, 2017, pp. 3145–3153.

[24] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3145–3153.

[25] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[26] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, 2017.

[27] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[28] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *Proceedings of International Conference on Machine Learning (ICML)*, 2017.

[29] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, e0130140, 2015.

[30] Y. Hechtlinger, "Interpretation of prediction models using the input gradient," *arXiv preprint arXiv:1611.07634*, 2016.

[31] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in Neural Information Processing Systems*, vol. 30, pp. 4765–4774, 2017.

[32] M. Zeiler and F. R., "Visualizing and understanding convolutional networks," in *European conference on computer vision*, 2014, pp. 818–833.

[33] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *International Conference on Machine Learning*, PMLR, 2018, pp. 883–892.

[34] J. Bastings, W. Aziz, and I. Titov, "Interpretable neural predictions with differentiable binary variables," in *Proceedings of ACL*, 2019, pp. 2963–2977.

[35] D. Pruthi, B. Dhingra, G. Neubig, and Z. C. Lipton, "Weakly-and semi-supervised evidence extraction," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 3965–3970.

[36] S. Chang, Y. Zhang, M. Yu, and T. Jaakkola, "Invariant rationalization," in *International Conference on Machine Learning*, PMLR, 2020, pp. 1448–1458.

[37] M. Yu, S. Chang, Y. Zhang, and T. Jaakkola, "Rethinking cooperative rationalization: Introspective extraction and complement control," 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)*, 2019, pp. 4094–4103.

[38] D. Antognini and B. Faltings, "Rationalization through concepts," in *ACL/IJCNLP (Findings)*, 2021.

[39] M. Yu, Y. Zhang, S. Chang, and T. Jaakkola, "Understanding interlocking dynamics of cooperative rationalization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 822–12 835, 2021.

[40] B. Paranjape, M. Joshi, J. Thickstun, H. Hajishirzi, and L. Zettlemoyer, "An information bottleneck approach for controlling conciseness in rationale extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 1938–1952.

[41] M. Lamm *et al.*, "Qed: A framework and dataset for explanations in question answering," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 790–806, 2021.

[42] Z. Sun *et al.*, "Self-explaining structures improve nlp models," *arXiv preprint arXiv:2012.01786*, 2020.

[43] D. Rajagopal, V. Balachandran, E. H. Hovy, and Y. Tsvetkov, "SELFEXPLAIN: A self-explaining architecture for neural text classifiers," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 836–850.

[44] W. J. Murdoch and A. Szlam, "Automatic rule extraction from long short term memory networks," in *Proceedings of International Conference on Learning Representation (ICLR)*, 2017.

[45] C. Singh, W. J. Murdoch, and B. Yu, "Hierarchical interpretations for neural network predictions," in *International Conference on Learning Representations*, 2018.

[46] M.-Y. Kim *et al.*, "A multi-component framework for the analysis and design of explainable artificial intelligence," *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 900–921, 2021.

[47] S. Sikdar, P. Bhattacharya, and K. Heese, "Integrated directional gradients: Feature interaction attribution for neural nlp models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 865–878.

[48] D. Zhang *et al.*, "Building interpretable interaction trees for deep nlp models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 14 328–14 337.

[49] Y. Hao, L. Dong, F. Wei, and K. Xu, "Self-attention attribution: Interpreting information interactions inside transformer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 12 963–12 971.

[50] P. Lipton, "Contrastive explanation," *Royal Institute of Philosophy Supplements*, vol. 27, pp. 247–266, 1990.

[51] D. J. Hilton, "Conversational processes and causal explanation.," *Psychological Bulletin*, vol. 107, no. 1, p. 65, 1990.

[52] A. Jacovi, S. Swayamdipta, S. Ravfogel, Y. Elazar, Y. Choi, and Y. Goldberg, "Contrastive explanations for model interpretability," *arXiv preprint arXiv:2103.01378*, 2021.

[53] S. Rathi, "Generating counterfactual and contrastive explanations using shap," *arXiv preprint arXiv:1906.09293*, 2019.

[54] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harv. JL & Tech.*, vol. 31, p. 841, 2017.

[55] L. Yang, E. Kenny, T. L. J. Ng, Y. Yang, B. Smyth, and R. Dong, "Generating plausible counterfactual explanations for deep transformers in financial text classification," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 6150–6160.

[56] L. A. Hendricks, R. Hu, T. Darrell, and Z. Akata, "Generating counterfactual explanations with natural language," *In ICML Workshop on Human Interpretability in Machine Learning, pages 95–98*, 2018.

[57] J. Bastings and K. Filippova, "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*, 2020, pp. 149–155.

[58] D. S. Moore and S. Kirkland, *The basic practice of statistics*. WH Freeman New York, 2007, vol. 2.

[59] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of ACL*, Association for Computational Linguistics, 2011, pp. 142–150.

[60] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[61] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[62] Kaggle, "Us consumer finance complaints," *Kaggle*, 2016.

[63] P. Zhou *et al.*, "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)*, 2016, pp. 207–212.

[64] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[65] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of International Conference on Machine Learning (ICML)*, 2017, 3319–3328.

[66] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision*, Springer, 2014, pp. 818–833.

[67] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS One*, vol. 10, no. 7, e0130140, 2015.

[68] A. Jacovi and Y. Goldberg, "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*, 2020, pp. 4198–4205.

[69] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[70] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 632–642.

[71] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2249–2255.

[72] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *EMNLP*, 2017.

[73] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 168–177.

[74] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[75] A. Koura, "An approach to why-questions," *Synthese*, vol. 74, no. 2, pp. 191–206, 1988.

[76] A. A. Ismail, H. Corrada Bravo, and S. Feizi, "Improving deep learning interpretability by saliency guided training," *Advances in Neural Information Processing Systems*, vol. 34, pp. 26 726–26 739, 2021.

[77] J. v. d. Waa, M. Robeer, J. v. Diggelen, M. Brinkhuis, and M. Neerincx, "Contrastive explanations with local foil trees," 2018.

[78] A. L. McGill and J. G. Klein, "Contrastive and counterfactual reasoning in causal judgment.," *Journal of Personality and Social Psychology*, vol. 64, no. 6, p. 897, 1993.

[79] H. J. Einhorn and R. M. Hogarth, "Judging probable cause.," *Psychological Bulletin*, vol. 99, no. 1, p. 3, 1986.

[80] Y. Wang, H. Huang, C. Rudin, and Y. Shaposhnik, "Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization.," *J. Mach. Learn. Res.*, vol. 22, no. 201, pp. 1–73, 2021.

[81] J. Woodward, *Making things happen: A theory of causal explanation*. Oxford university press, 2005.

[82] P. Lipton, "Contrastive explanation," *Royal Institute of Philosophy Supplements*, vol. 27, pp. 247–266, 1990.

[83] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020.

[84] M. Lewis *et al.*, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[85] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[86] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[87] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[88] R. Tang, A. Adhikari, and J. Lin, "Flops as a direct optimization objective for learning sparse neural networks," *arXiv preprint arXiv:1811.03060*, 2018.

[89] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[90] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5457–5466.

[91] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of NAACL-HLT*, 2016, pp. 1480–1489.

[92] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[93] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[94] J. Chen and M. Jordan, "Ls-tree: Model interpretation when the data are linguistic," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3454–3461.

[95] X. Jin, Z. Wei, J. Du, X. Xue, and X. Ren, "Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models," in *International Conference on Learning Representations*, 2019.

[96] M. Tsang, S. Rambhatla, and Y. Liu, "How does this interaction affect me? interpretable attribution for feature interactions," *Advances in neural information processing systems*, vol. 33, pp. 6147–6159, 2020.

[97] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *ICML*, 2016.

[98] Y. Kim, "Convolutional neural networks for sentence classification," *In Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[99] "Yelp dataset challenge 2018," 2018.

[100] J. DeYoung *et al.*, "Eraser: A benchmark to evaluate rationalized nlp models," *arXiv preprint arXiv:1911.03429*, 2019.

Appendix A: Evaluating feature attribution for the vector space model

A.1 Faithfulness evaluation

We evaluate the effectiveness of the vectors space model in providing faithful explanation and we use the following baselines:

- **Random**. A random selection of words from the input sentence.
- **LIME** is a model-agnostic approach which involves training an interpretable model such as a linear model on instances created around the specific data point by perturbing the data. We evaluated by training the linear classifier using $\sim$ 5000 samples.

We show the effectiveness of our method in explaining the prediction on three architectures (Transformer, IndRNN and hierarchical attention network) in Figures A.1-A.6.

A.1.1 Automatic evaluation

We measure the local fidelity by deleting words in the order of their estimated importance for the prediction, then evaluate the change in F1 score w.r.t. the predicted class when no word is deleted. Results are shown in Figures A.1-A.2. A larger drop in F1 indicates that the method could identify the words contributing most towards the predicted class by our classifier. Through Figures A.1, A.2 and A.3, we can clearly see that our approach is capable of identifying the most salient features better than LIME.

A.1.2 Change in log-odds

This metric requires no knowledge of the underlying feature representation, and it requires access to only the instances. Like the previous experiment, instead of tracking the change in F1, we observe the change in the scores. We mask the top k features ranked by semantic similarity, and zero paddings replace those masked words. We then feed the input and measure the drop of the value between the target class's probability when no word is deleted and when k words are removed. Results are shown in Figures A.4-A.6 reveal the effectiveness of our approach in capturing the words that affect the classifier's prediction. The experimental results show that our method delivers more insightful explanations than LIME.

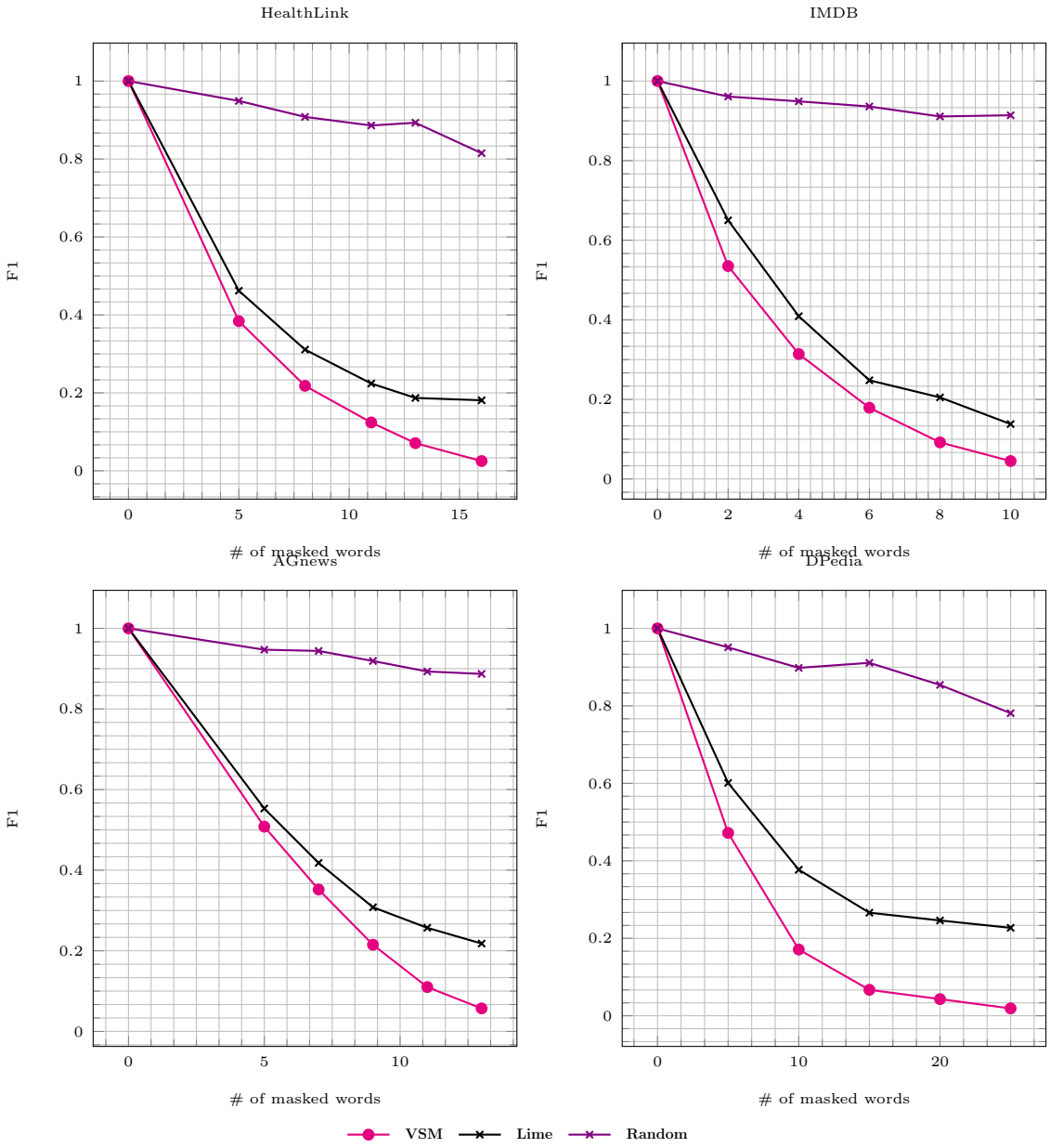


Figure A.1: Change of F1 according to the number of masked important words. (Teacher model: Transformer)

A.1.3 The effectiveness of using cosine distance in learning discriminative representations for the VSM

We compare our proposed method's performance with and without capturing the semantic information second term in the loss function. Results depicted in Table A.1 show the effectiveness of using cosine distance.

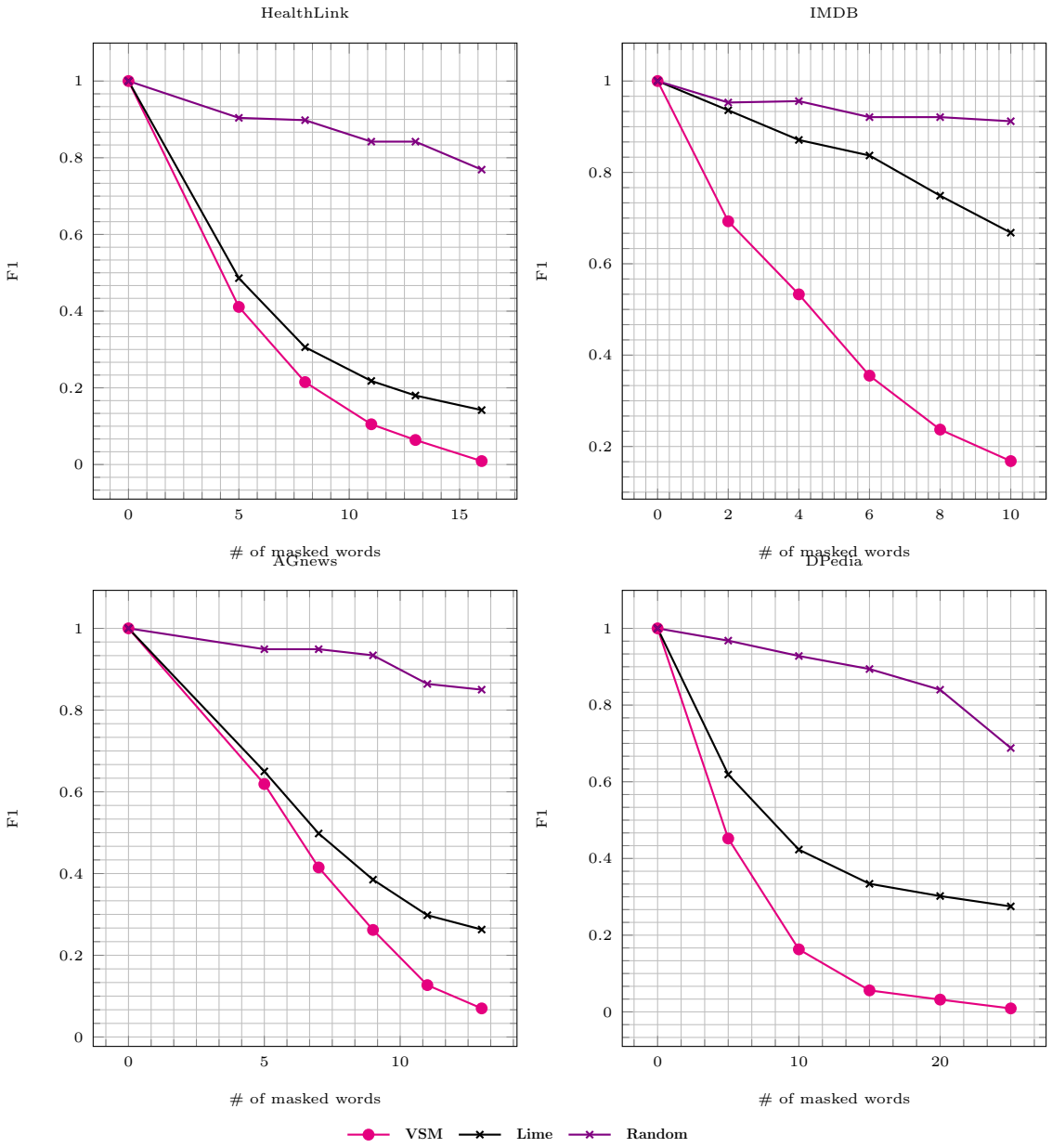


Figure A.2: Change of F1 according to the number of masked words. (Teacher model: INDRNN)

	Proposed				Without semantic			
	F1	**Accuracy**	**Precision**	**Recall**	**F1**	**Accuracy**	**Precision**	**Recall**
Dpedia	0.8806	0.9438	0.8811	0.8809	0.0425	0.624	0.0693	0.0582

Table A.1: The impact of the cosine distance on the classifier's performance

A.1.4 Analyzing the features used by the VSM

We are interested in what kind of words contribute most to the class prediction. For this analysis, we exploit the word-level sentiment annotation (Opinion Lexicon)

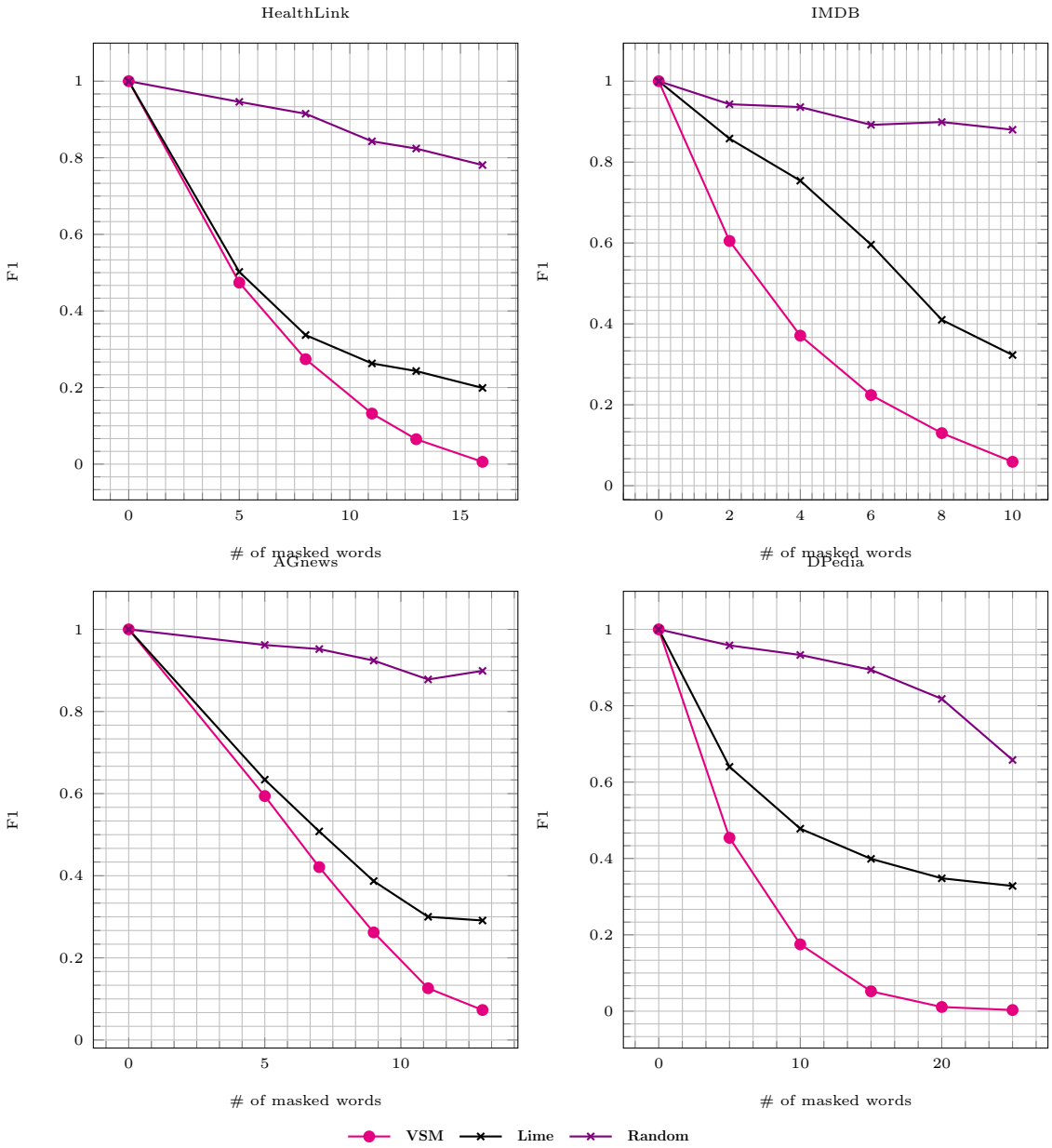


Figure A.3: Change of F1 according to the number of masked words. (Teacher model: Hierarchical attention network)

provided by Liu [73] to track the top 10 words whose importance was the highest when predicting the sentiment class in the IMDB dataset. We evaluated the number of words contributing to each of the negative and positive sentiments on 1000 movie reviews. Table A.2 shows that our approach can identify more salient words that lead to correct sentiment classification, i.e., our method can pick better sentiment lexicons than LIME.

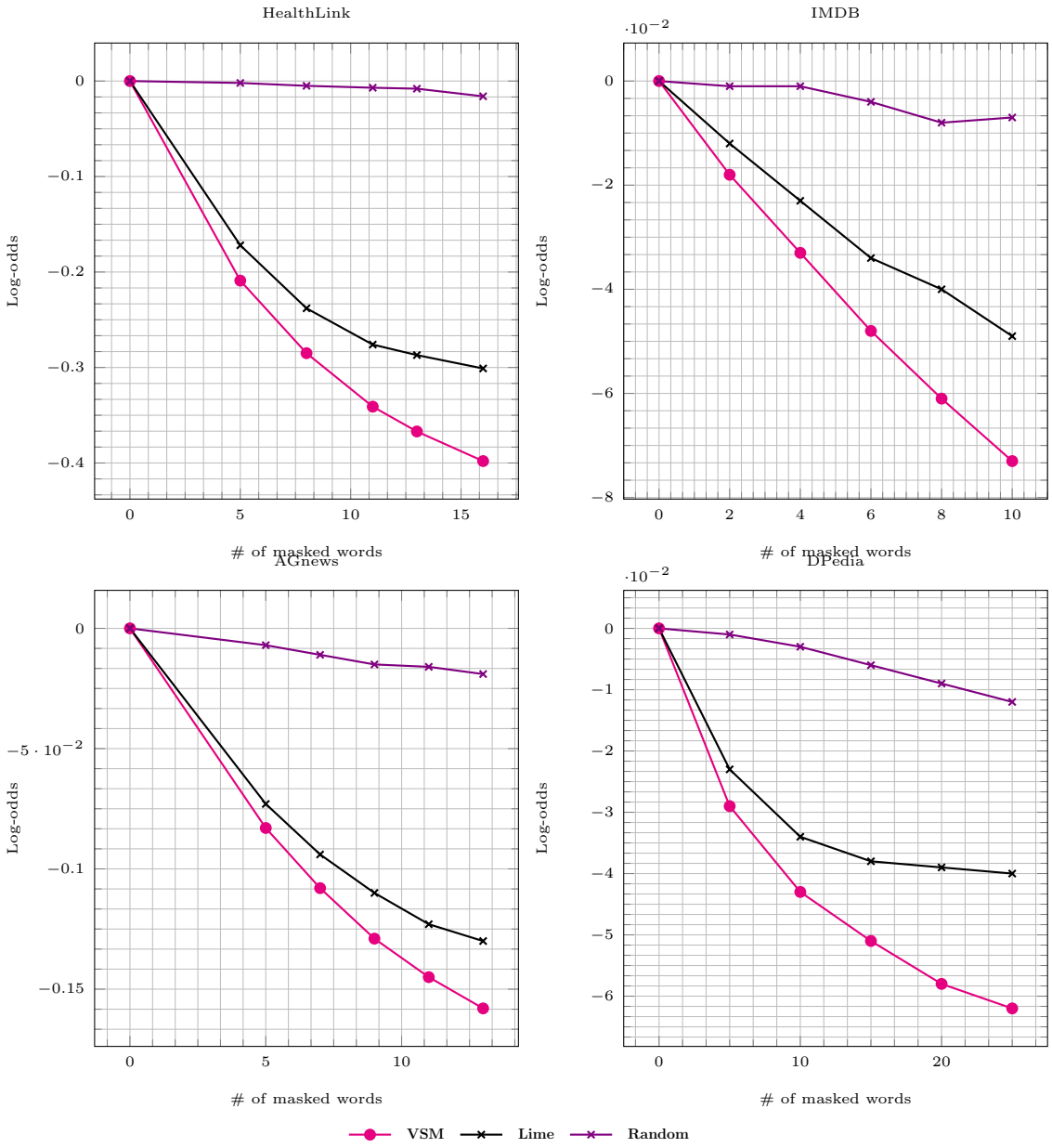


Figure A.4: Change of log-odds according to the number of masked words. Lower log-odds scores are better. (Teacher model:Transformer)

	Proposed	**Lime**	**Random**
Positive sentiment	**597**	423	286
Negative sentiment	**382**	353	236

Table A.2: The number of words in each sentiment class for 1000 samples from the test set.

A.2 Additional experiments

The mean and standard deviation of the proposed VSM model trained on three datasets.

Dataset	IMDB	AGnews	HealthLink
F1	0.8192 ± 0.0011	0.9033 ± 0.0004	0.7196 ± 0.0007

Table A.3: F1 score of the VSM trained using the Multi-head architecture.

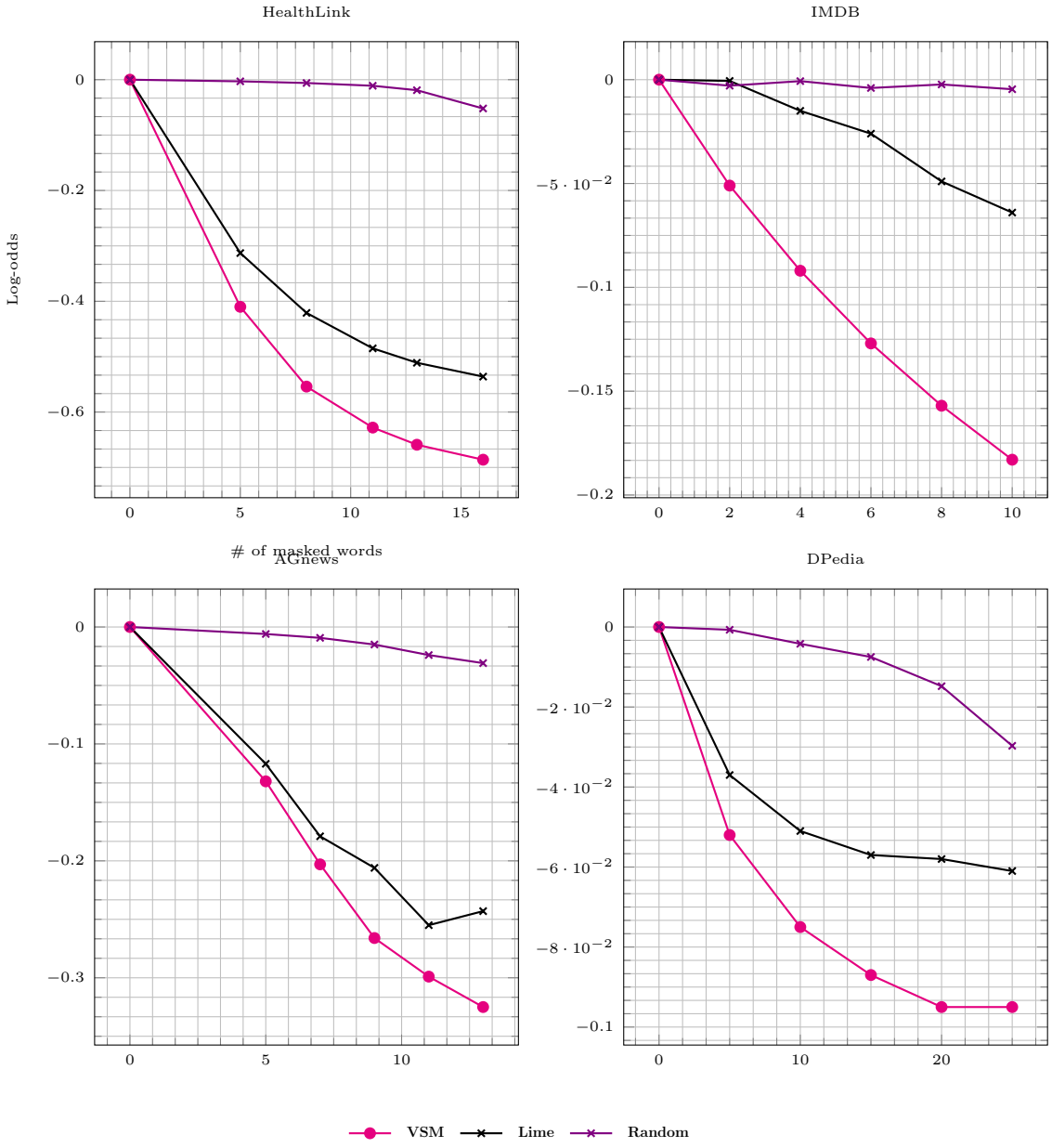


Figure A.5: Change of log-odds according to the number of masked words. (Teacher model: INDRNN)

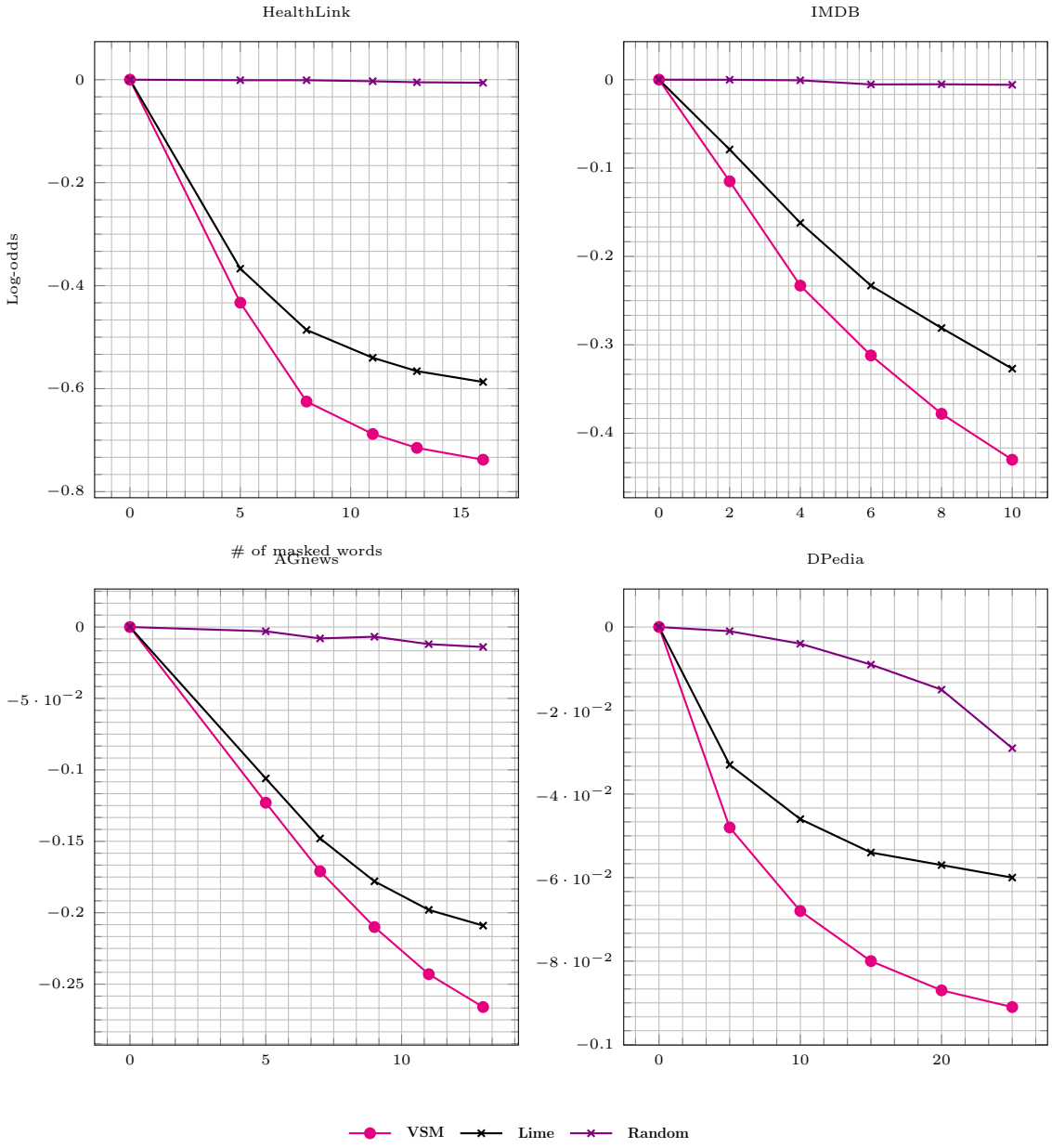


Figure A.6: Change of log-odds according to the number of masked words. Lower log-odds scores are better. (Teacher model: Hierarchical attention network)

Appendix B: Additional Experiments for SFFA

B.1 Degradation Score

Results are shown in Figures B.1, B.2, B.3, B.4, B.5 and B.6. In all the figures, our SFFA shows the steepest decline, which means the best explanation for each model's prediction.

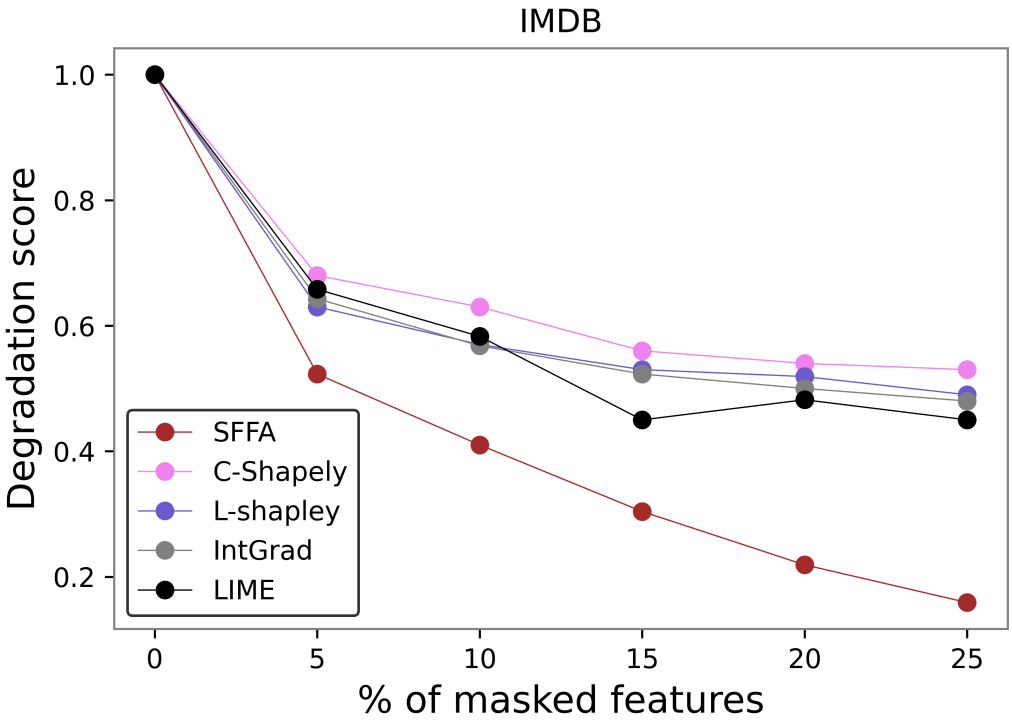


Figure B.1: Degradation score as a function of masked tokens on Attbilstm (IMDB).

B.2 Log-odds Score

Additional results on AG news and YELP for both Attbilstm and CNN are shown in Figures B.7-B.10. In all the figures, our *SFFA* shows the steepest decline, which means the best explanation for each model's prediction.

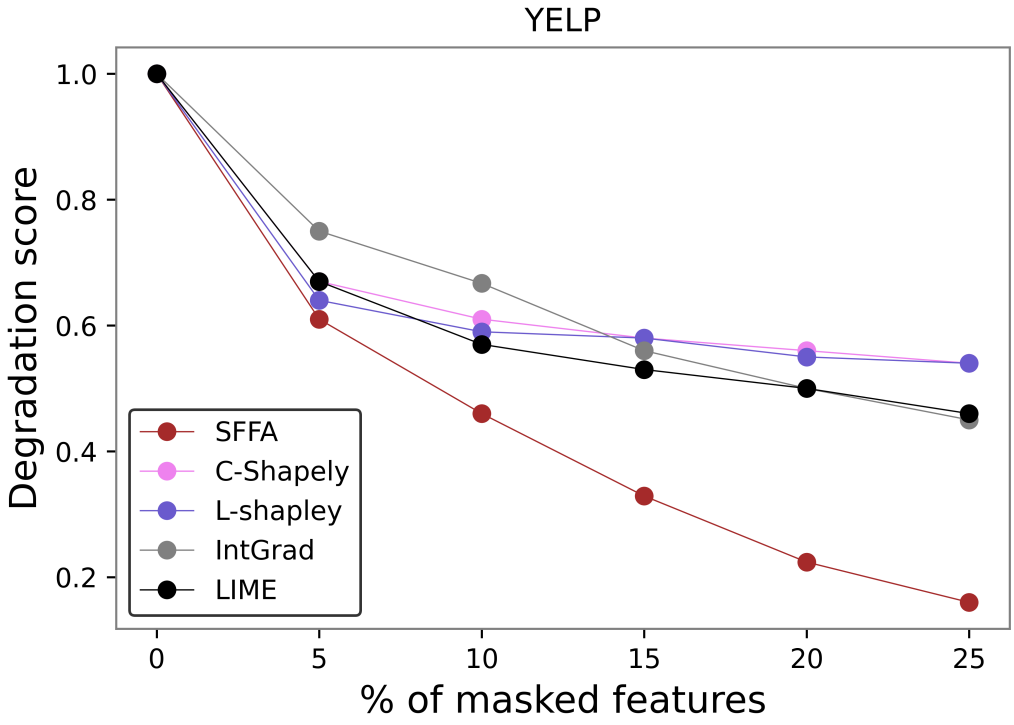


Figure B.2: Degradation score as a function of masked tokens on Attbilstm (YELP).

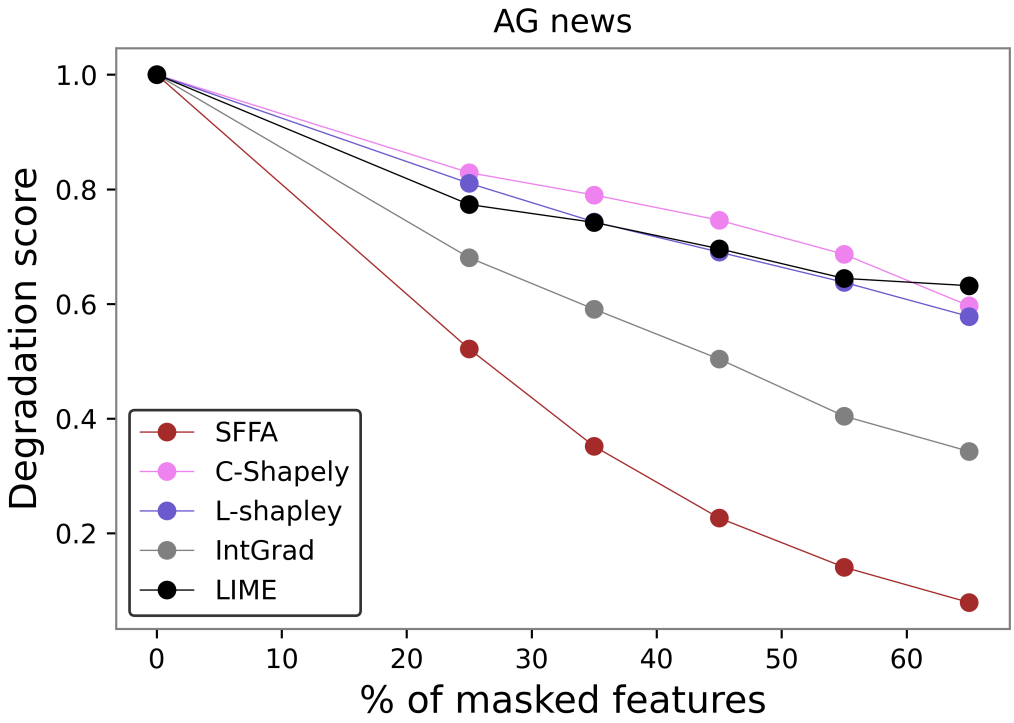


Figure B.3: Degradation score as a function of masked tokens on Attbilstm (AG news).

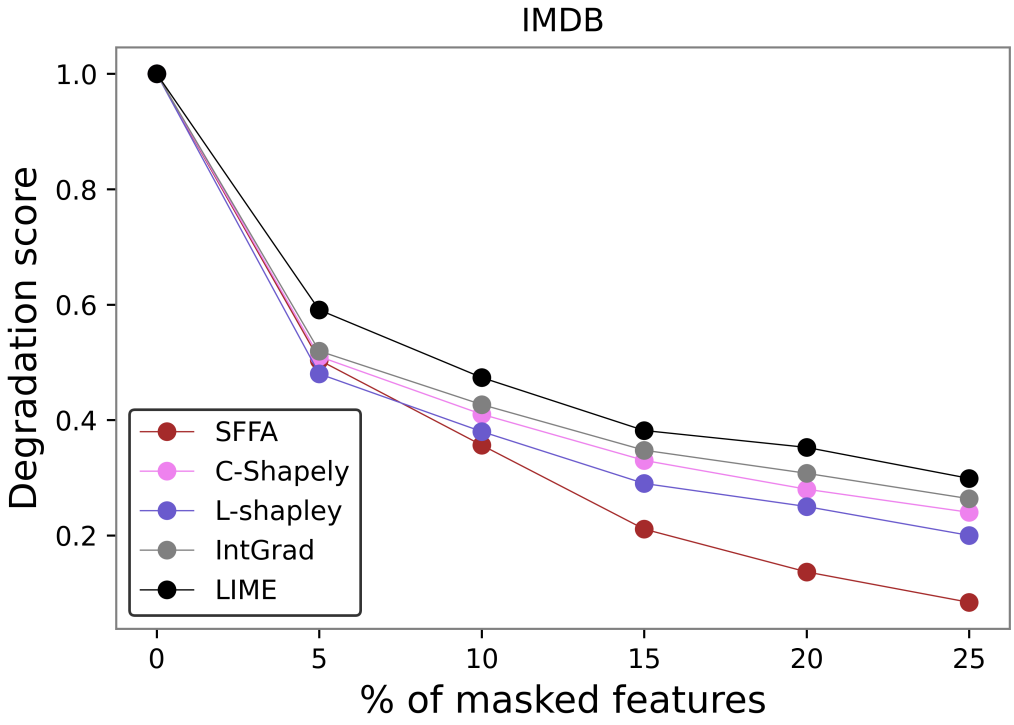


Figure B.4: Degradation score as a function of masked tokens on CNN (IMDB).

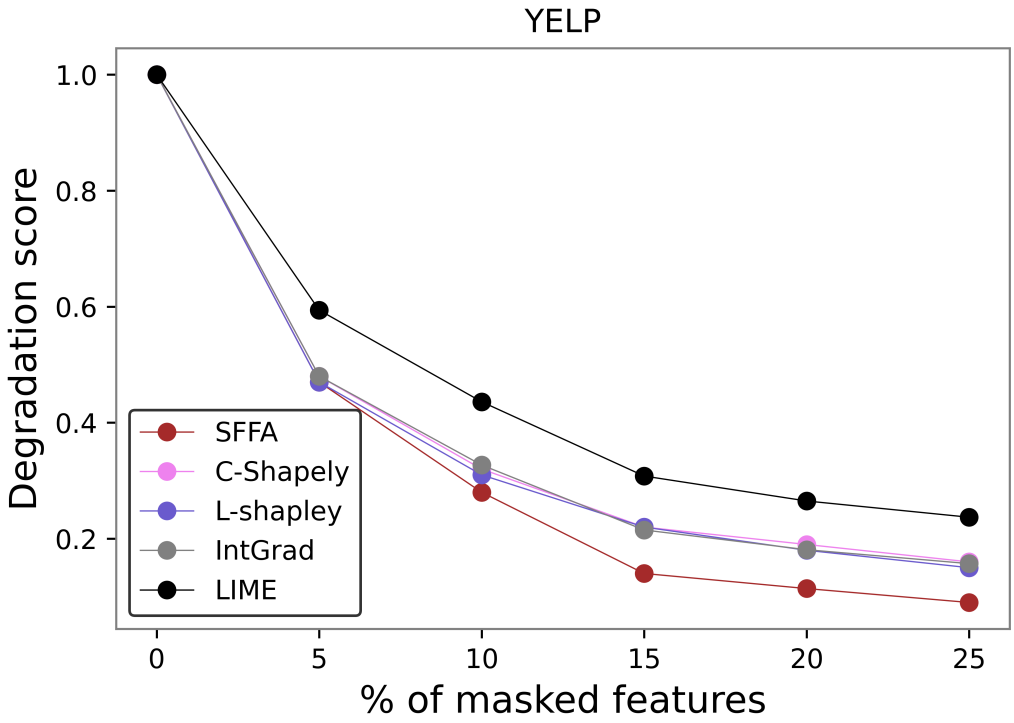


Figure B.5: Degradation score as a function of masked tokens on CNN (YELP).

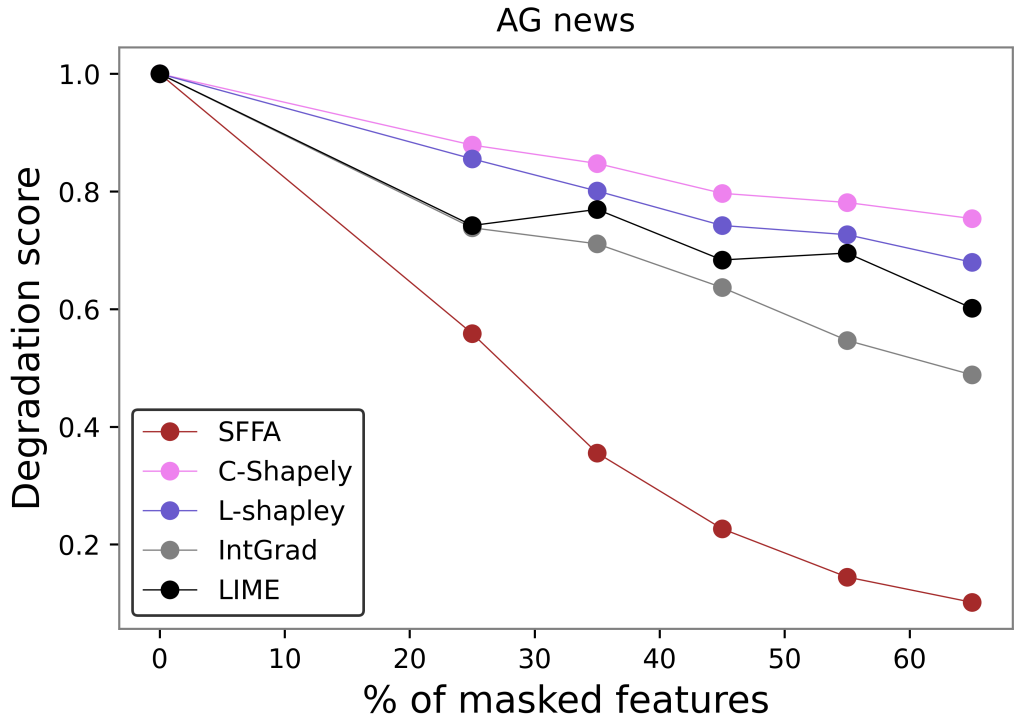


Figure B.6: Degradation score as a function of masked tokens on CNN (AG news).

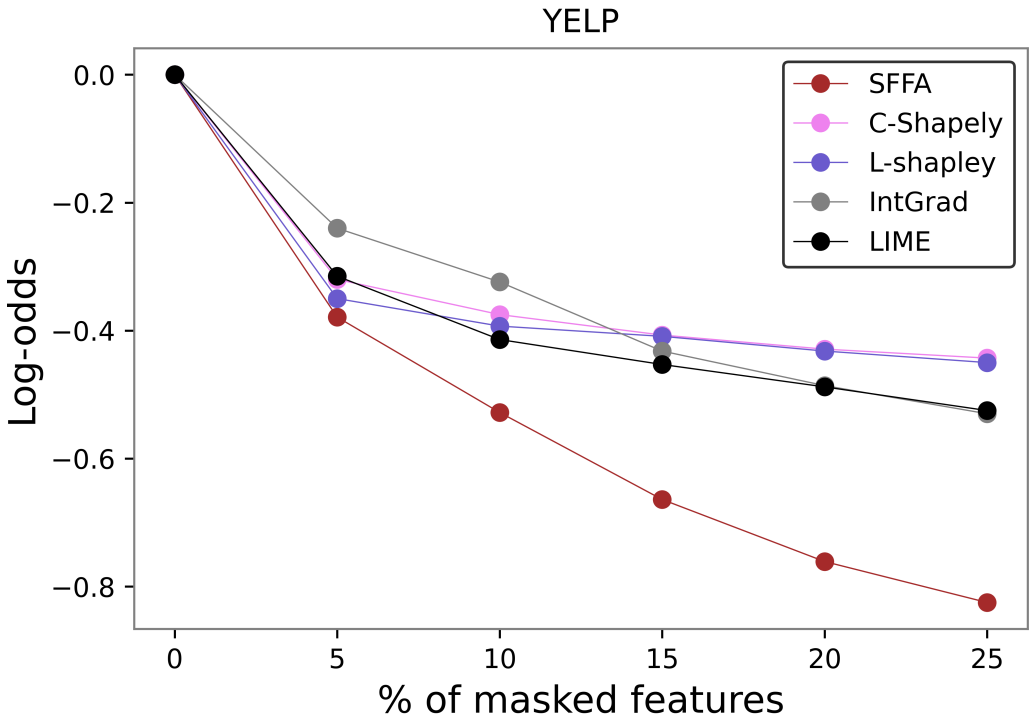


Figure B.7: Log-odds as a function of masked tokens on Attbilstm (YELP).

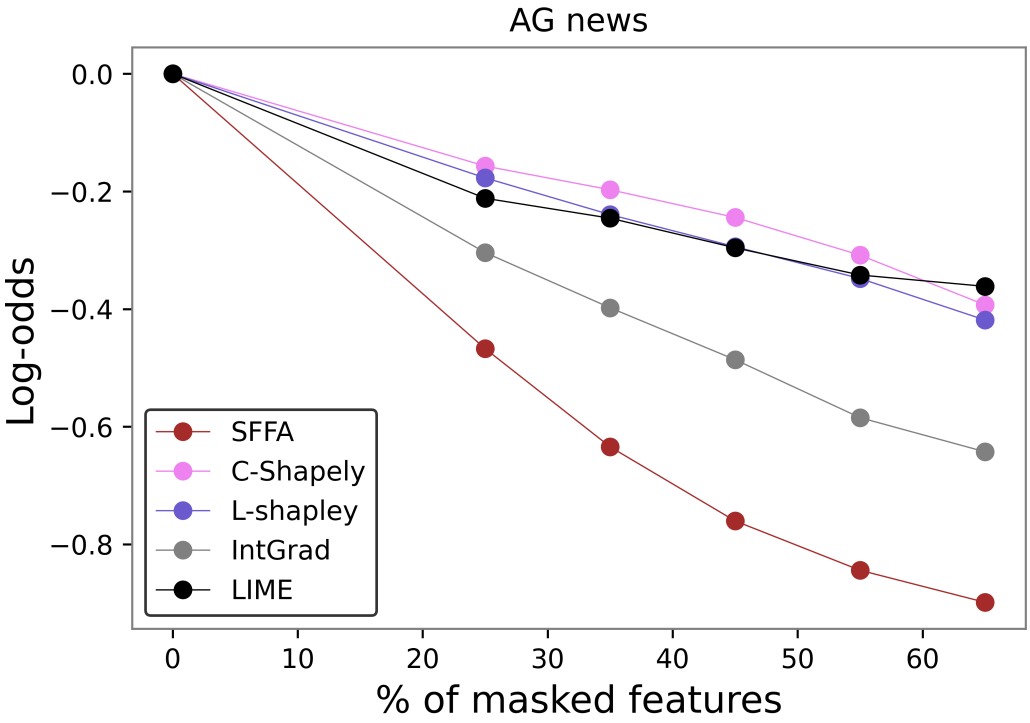


Figure B.8: Log-odds as a function of masked tokens on Attbilstm (AG news).

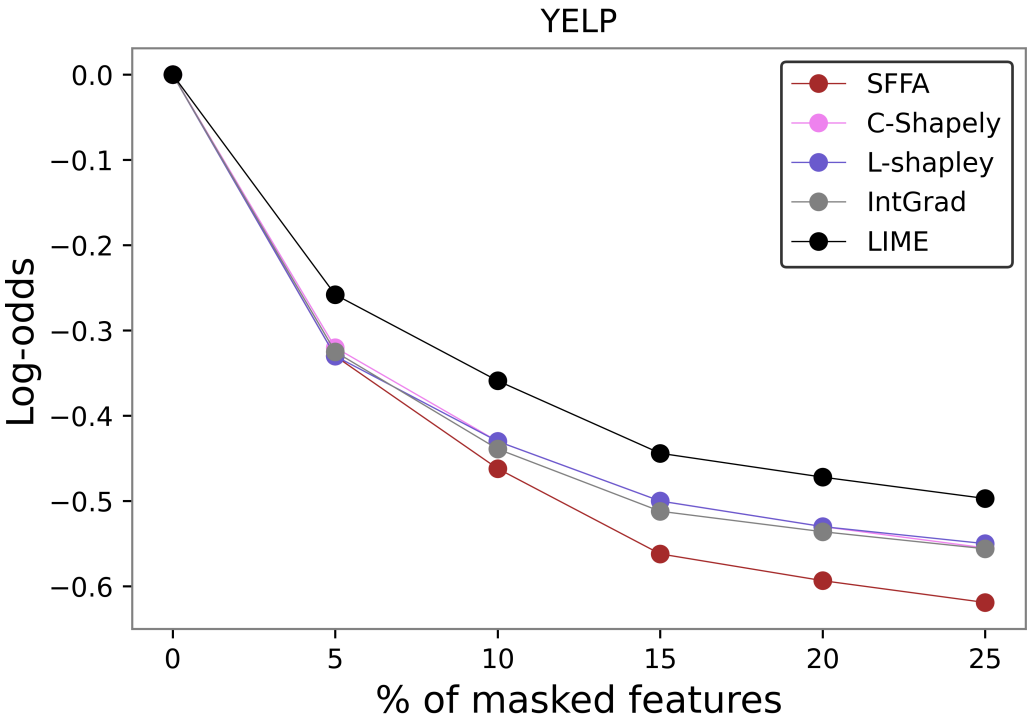


Figure B.9: Log-odds as a function of masked tokens on CNN (YELP).

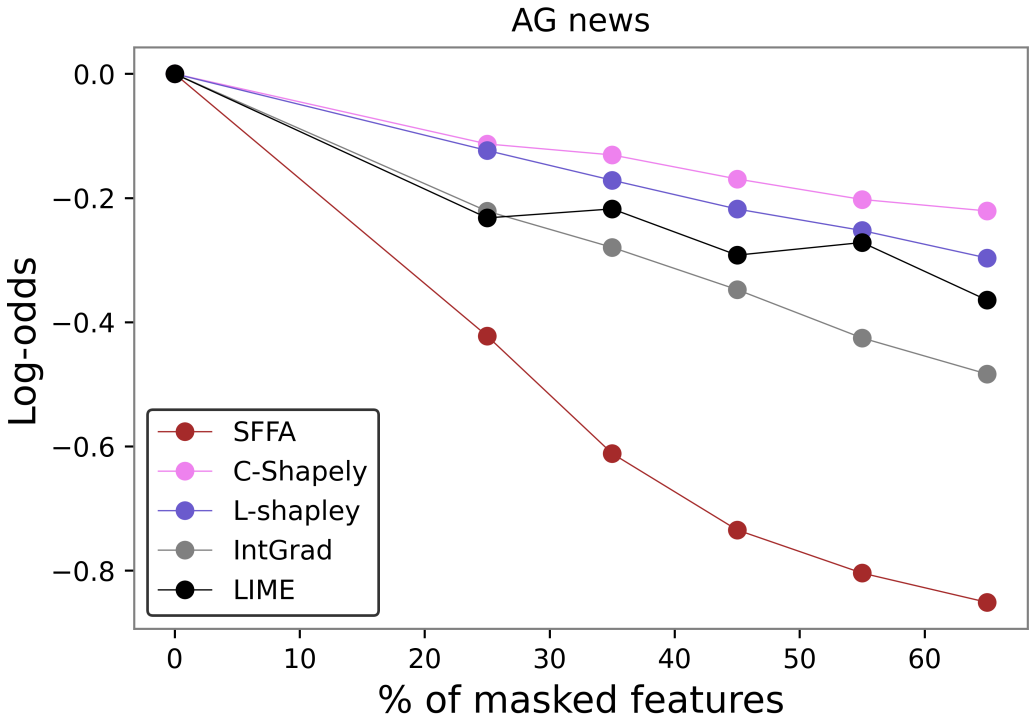


Figure B.10: Log-odds as a function of masked tokens on CNN (AG news).

Results on RoBERTa are shown in Figures B.11, B.12 and B.13. SFFA shows the steepest decline in all the figures. We have found the L-Shapley and C-Shapley achieve almost similar log-odds scores for AG news dataset.

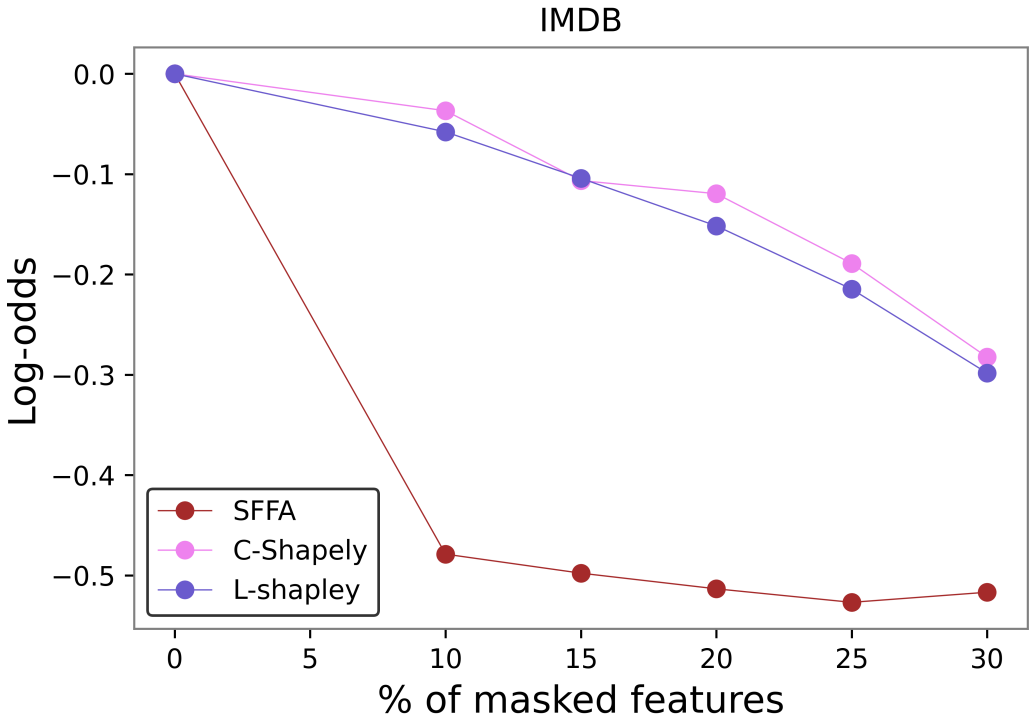


Figure B.11: Log-odds as a function of masked tokens on RoBERTa (IMDB).

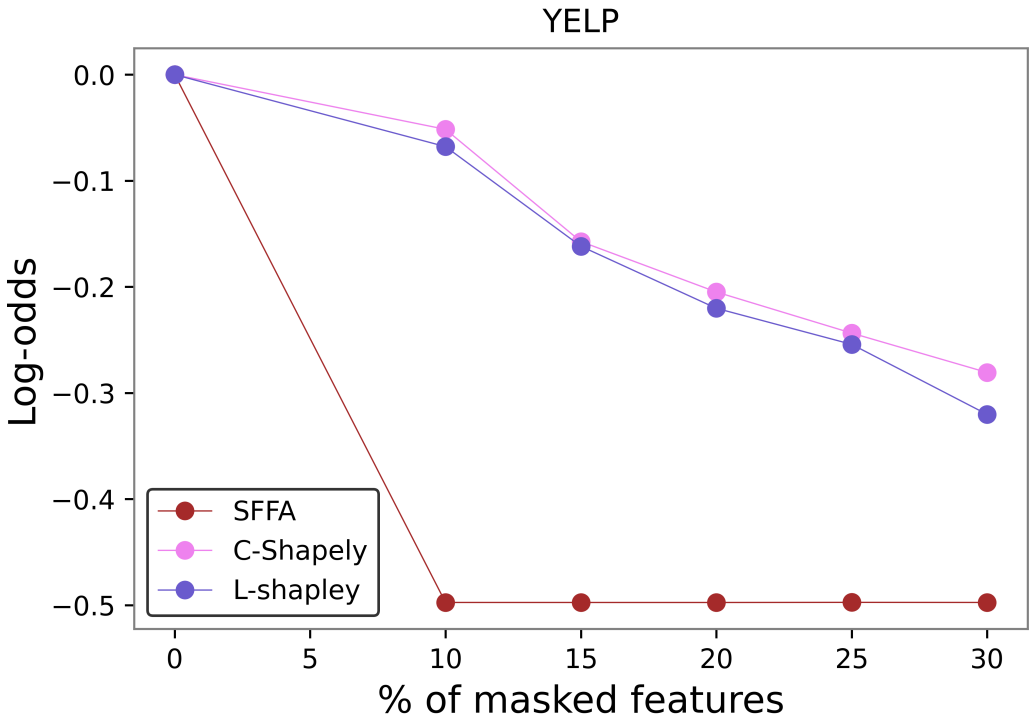


Figure B.12: Log-odds as a function of masked tokens on RoBERTa(YELP).

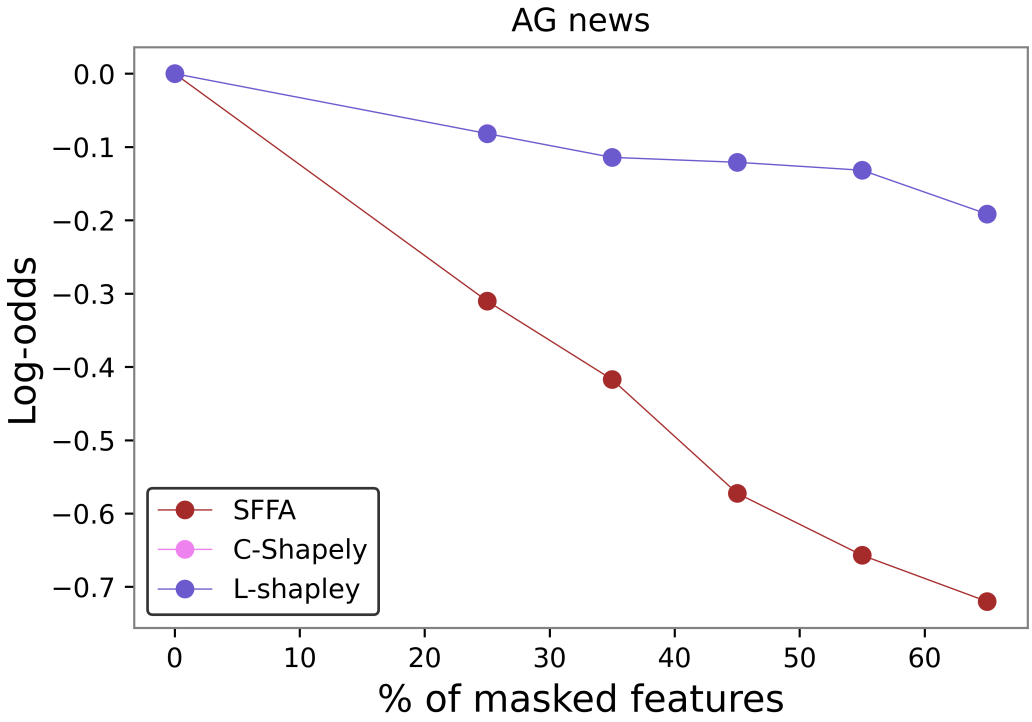


Figure B.13: Log-odds as a function of masked tokens on RoBERTa(AG news). L-shapley and C-shapley acheived similar scores in terms of log-odds.