APPROVAL SHEET

This

is submitted in partial fulfillment of the requirements
for the degree of

Author:

Advisor:

Advisor:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Committee Member:

Accepted for the School of Engineering and Applied Science:

[illegible]

Jennifer L. West, School of Engineering and Applied Science

Abstract

Natural language processing (NLP) has illuminated the world by enabling understanding, communication, and interaction between computers and humans. NLP has a wide range of practical applications in various fields, such as processing and extracting information from unstructured data, translating different languages to support communication, and building dialog systems or virtual assistants for social good. The development of NLP is driven by neural network models. Over the past decades, NLP models have grown ever larger and more sophisticated and demonstrated impressive learning abilities to handle a variety of tasks. On the other hand, the interpretability of NLP models has diminished due to the incremental complexity of neural networks and the limited access to their inner working or training data. The lack of interpretability has raised much concern about the trustworthiness and reliability of NLP models in real-world applications. Besides, the black-box nature of neural network models has hindered humans from understanding them, finding their weaknesses, and avoiding unexpected failures.

In this dissertation, I cultivate neural model interpretability for trustworthy NLP. Specifically, I integrate interpretation techniques into model development, covering three main phases of a model life cycle—training, testing, and debugging. During training, I build interpretable models by designing learning strategies to make model prediction behavior transparent and reasonable. In the testing phase, with limited access to a black box model, I develop explanation methods to explain the model decision-making on each test example. I evaluate explanations (e.g., informativeness), as complementary to traditional evaluations on predictions (e.g., accuracy), to understand model prediction behavior. Finally, I diagnose and debug models (e.g., robustness) through the lens of explanations and develop solutions to improve them. My research has the potential to benefit NLP and AI developers, providing them with a better understanding of neural network models and helping them build trustworthy and reliable intelligent systems.

To everyone who led me here.

Acknowledgements

A little dream sparked in my mind five years ago—to pursue a Ph.D. To this day, it still feels surreal that I am almost there. I would like to extend my sincere gratitude to all those who have helped and supported me along the way.

First and foremost, I want to express my deepest appreciation to my advisor, Dr. Yangfeng Ji, for his constant guidance, help, support, and encouragement throughout my Ph.D. journey. Yangfeng has been instrumental in leading me towards the research area that I am truly passionate about and thoroughly enjoy working on. Under his mentorship, I have not only developed research skills, but also been trained to become a future professor with well-rounded capabilities in teaching, communication, service, etc. His belief in my abilities has been a source of great inspiration and motivation, and his encouragement and support to explore new ideas have helped me overcome numerous obstacles and challenges. I am fortunate to have had the opportunity to learn from him, and to have benefited from his exemplary qualities as a meticulous researcher, considerate supervisor, and advocate for work-life balance. I believe all those I learned from him will be a life-long treasure in my future career and life.

I am grateful for having Dr. Aidong Zhang, Dr. Haifeng Xu, Dr. Cong Shen, Dr. Alexander (Sasha) M. Rush, and Dr. Noah A. Smith, as my thesis committee members. Aidong and Haifeng have been on my Ph.D. committee since my qualification exam. Their valuable suggestions and comments have been of immense help to me throughout my research. I first met Cong many years ago when I was a Masters student in his class in China. It was an honor to work with him again at UVA as he provided expert guidance and instruction for my thesis. Sasha asked insightful questions and offered sagacious comments and instructions that helped improve my thesis. He also generously supported me in my job search with helpful advice and assistance. Noah, whom I consider my grand advisor, shared sparkling thoughts and practical suggestions on research and future directions with me. His passion and dedication to research have always been inspiring me. I sincerely appreciate all my committee members for their kind support, expert guidance, and precious time devoted to my thesis. Their professionalism has been crucial in shaping the outcome of my research.

In addition to my advisor and committee, I was very fortunate to work with many brilliant researchers, including Dr. Yanjun Qi at UVA, Dr. Yejin Choi, Dr. Swabha Swayamdipta, Dr. Faeze Brahman, Dr. Xiang Ren during my internship at Allen Institute for AI, Dr. Guoqing Zheng, Dr. Srinagesh Sharma, Dr. Ahmed Hassan Awadallah during my internship at Microsoft Research, and Dr. Song Feng, Dr. Chulaka Gunasekara, Dr. Hui Wan, Jatin Ganhotra and Sachindra Joshi during my internship at IBM Research. I thank them for teaching me a lot, broadening my research horizons, and sharing their experiences.

My thanks also go to my other amazing collaborators: Dr. Arshdeep Sekhon, Wanyu Du, Guangtao Zheng, Ruixuan Tang, Sanchit Sinha, Dr. Aaron Chan, Zhe Wang, Aman Shrivastava, Zhiyuan Zeng, Brihi Joshi, and Wyatt Lake. They generously shared ideas, actively engaged in discussions, and diligently devoted to our research projects. Working with them was a pleasure, and we all learned a great deal from each other.

I am proud to be a member of the Information and Language Processing (ILP) lab, which provides a supportive and welcoming research environment. I would like to thank all my labmates (Wanyu Du, Aidan San, Stephanie Schoch, Hannah Chen, Dane Williamson...) for their support and help in my research and daily life. I am also grateful to receive plenty of advice and support from many senior Ph.D. students, such as Dr. Meiyi Ma, Dr. Xiang Li, Dr. Mengdi Huai, Dr. Lu Lin, Dr. Renqin Cai, Dr. Dongkuan Xu, etc. They are role models to me, and I have benifited a lot from them in research and career development. Furthermore, I want to thank my friends, Wanyu Du, Jing Ma, Sanxing Chen, Wenbo Pang, Zhe Wang, Haozhe Zhang...Their companionship, care, and encouragement make my Ph.D. life lovely and enjoyable.

Last but not least, I owe a lot to the unconditional support and endless love from my family. I appreciate my parents, S.J. Chen and G.Y. Gong, who raised me up with efforts and gave me tremendous trust and encouragement to pursue my dreams. I want to extend my special thank to my partner Guangtao Zheng. We embarked on this academic journey together, from our Masters to Ph.D. He has been my labmate, teammate, collaborator, boyfriend, and now husband. Through all the ups and downs, he has been my rock, offering unwavering support and love. I could not have made it this far without him by my side.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

With the rapid development of deep learning techniques and neural network models, natural language processing (NLP) has entered a new era where computers are able to understand, process, and generate natural languages and communicate and interact with humans [5, 6, 7, 8]. NLP has spread its applications in various fields, such as analyzing the sentiments expressed in customer reviews and social media, translating different languages to support communication, extracting and summarizing useful information from large amounts of unstructured data, and building chatbots and virtual assistants for customer support, helpdesk services, and other applications that require human-like interactions.

Despite the promising advances in NLP, neural language models have grown ever larger and more sophisticated, from recurrent neural networks [9, 10], convolutional neural networks [11, 12], to large pre-trained language models containing millions and billions of parameters [13, 14, 15]. In addition to the incremental complexity, the limited access to models' inner-working or training data also blocks humans from understanding them. In other words, deep neural networks are black-box models, lacking interpretability. The black-box nature can result in severe pitfalls in real-world applications. For example, a medical chatbot shows unexpected behavior, providing harmful advice to suicidal patients. [1] Improving interpretability is crucial for helping AI developers debug and diagnose models, find their weaknesses and vulnerability, and develop trustworthy and reliable intelligent systems [16, 17, 18, 19]. Especially, in some high-stakes scenarios (e.g., health care and criminal justice), people would not use deep neural networks if they do not trust them [1, 20, 21].

[1] `https://www.artificialintelligence-news.com/2020/10/28/medical-chatbot-openai-gpt3-patient-kill-themselves/`

The recent literature on improving model interpretability broadly falls into two categories: (1) explaining black-box models from a post-hoc manner [22]; (2) building interpretable neural network models [23]. For the first direction, most work focuses on attributing single-feature contributions to model predictions, such as perturbation-based methods [1, 24], gradient-based methods [25, 26], attention-based methods [27, 28], and information bottleneck-based methods [29, 30]. Natural languages are generally complex: containing negations, transitions, etc. For example, the sentiment of "a waste of good performance" is negative. Current language models (e.g., BERT [13]) are able to capture the semantic meaning of composite sentences by encoding context information. This requires explanations to detect feature interactions in model processing input tokens, e.g., "a waste of" flips the sentiment of "good performance" from positive to negative, and provide a comprehensive picture showing model decision-making based on feature interactions. In other words, it is crucial to generate explanations that are faithful to models and reflect their true reasoning processes. For the second direction, building interpretable models is essential for trustworthiness since they have transparent decision making. In practice, interpretable neural network models have been applied to solve specific tasks, such as recidivism prediction with categorical data [31] and MNIST digit recognition with simple image data [32]. However, designing interpretable language models for text data with complex representations (e.g. word embeddings) is more challenging and requires much engineering effort [33]. In fact, it is almost impossible to design interpretable models from scratch that can achieve comparable performance as existing large pre-trained language models. Thus, balancing model performance and interpretability is a critical question in NLP.

With model explanations, many research questions arise, such as *How faithful are explanations to model predictions?*, *To what extent do explanations help users understand model predictions?*, *What insights can we get from explanations to understand model behavior?*, etc. All those questions involve one specific research problem—explanation evaluation. Generally, explanations are evaluated along two dimensions: (1) faithfulness to model predictions; (2) plausibility to human understanding [34]. Faithfulness measures to which extent explanations reflect the decision making of models. Plausibility evaluates to which extent explanations help humans understand model predictions. Although these two evaluations cover basic properties of explanations, other properties are also significant, such as informativeness in justifying model reasoning, factuality in reflecting true knowledge, sensitivity under input perturbations, etc. However, there is still a lack of holistic formalism for explanation evaluation.

Building upon a better understanding of models, it is natural to move to the next stage—diagnosing and debugging models through the lens of explanations [18, 35]. For example, analyzing explanations can help identify spurious correlations captured by models [36]. And aligning model explanations with human annotations can guide models to focus on right reasons to make correct predictions [37]. Besides, exploring the interplay between model interpretability with other properties (e.g., robustness, fairness) is a promising direction for developing trustworthy and reliable intelligent systems.

1.2 Dissertation Statement

This dissertation explores model interpretability and integrates interpretation techniques into different stages of model development—training, testing, and debugging—for trustworthy NLP.

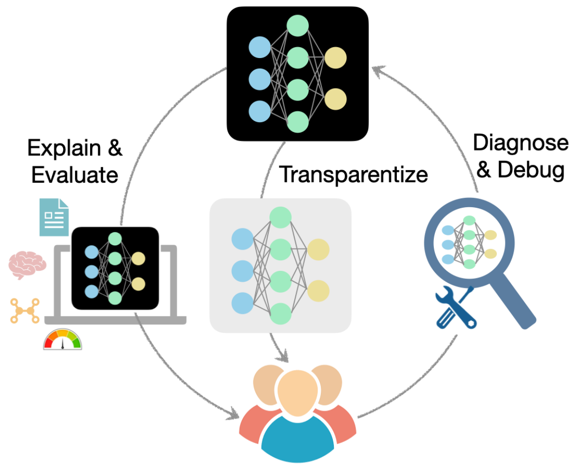


Figure 1.1: Three threads of the dissertation. (1) generating and evaluating explanations for model understanding; (2) transparentizing model decision-making for interpretability; (3) diagnosing and debugging models through the lens of explanations.

As Figure 1.1 shows, this dissertation is composed of three threads: (1) generating and evaluating explanations for model understanding; (2) transparentizing model decision-making for interpretability; (3) diagnosing and debugging models through the lens of explanations. The three threads cover three main phases of a model life cycle—training, testing, and debugging. During training, we have full access to build interpretable models. Without designing an interpretable model from scratch, I transparentize a black box model by controlling its prediction beahvior to be transparent and interpretable. The core idea is to teach models to focus on task-specific important features to make predictions, hence improving their interpretability. Given limited access to a model's inner-working or training, a feasible way to understand it is to explain its predictions in a post-hoc manner by inferring the relationships between its inputs and outputs. I develop explanation methods to explain model decision-making on each test example. Besides, I design evaluation metrics to assess explanations and get insights into model prediction behavior. With explanations, I diagnose and debug neural language models, especially their robustness and fairness. I identify model pathologies and develop solutions to address them. The interpretation techniques I have developed bridge the trustworthy gap between models and humans. Next, I elaborate the main contributions of this dissertation.

1.3 Contributions

This dissertation advances neural model interpretability for trustworthy NLP along three directions:

- ***Building interpretable neural language models by transparentizing their decision-making***. I design learning strategies to make model decision making transparent and interpretable. The core idea is to teach models to focus on task-specific important features to make predictions. For example, if a model learns to predict "an interesting movie" as positive by relying on "interesting", its prediction would be trustworthy. I propose two data augmentation methods that create additional training examples to help models learn task-specific important features: one utilizes a predefined word list as external knowledge to identify important features; the other one identifies important features by perturbing input texts via adversarial attacks. In addition, I propose inserting a variational word mask layer into a neural network, after the input layer (e.g., word embedding layer), which learns to restrict the information of irrelevant or noisy features flowing to subsequent network layers, hence forcing the model to focus on important features to make predictions. This variational word mask layer is plug-and-play and automatically learns important features without resorting to external knowledge or human annotations.

- ***Explaining neural language models and evaluating their explanations for model understanding***. I develop explanation methods to explain black box models by inferring the relationships between their inputs and outputs. To ensure the generated explanations are qualified to explain models and understandable to humans, I evaluate explanations in several aspects (e.g., faithfulness, informativeness). An explanation usually highlights important features in the input to explain the model's prediction. For example, if a model predicts the sentiment of "good performance" as positive and the explanation highlights "good", the prediction would be trusted. However, natural languages are generally complex, containing negations and transitions. For example, "a waste of" can flip the sentiment of "good performance" from positive to negative. To capture feature interactions, I propose a hierarchical explanation, visualizing the process of different granularities of features (e.g., words, phrases) interacting with each other within a model towards the final prediction. The hierarchical explanation demonstrates both faithfulness to model predictions and plausibility to humans by providing a comprehensive picture of model decision making. I further design an efficient explanation dedicated to models that take in multiple input texts, because computing feature interactions across sentences is computationally inefficient. I propose to implicitly detect word correlations by grouping correlated words from input texts together and measure their overall contribution to the corresponding NLP tasks. This method significantly decreases the computational complexity and fills a void in explaining sentence-pair modeling tasks (e.g., natural language inference). In addition to explaining prediction labels, explaining predictive uncertainty is also important for model understanding. Especially when a model makes a correct prediction with low confidence, people may doubt it and wonder what causes the uncertainty. My work is among the first in NLP, arguing a comprehensive explanation should explain both prediction label and uncertainty. I propose a simple method to generate uncertainty explanations by extracting uncertain words (e.g., negations, those against model prediction labels) in inputs and demonstrate their necessity in helping users understand model prediction behavior.

In evaluating model explanations, an open research question is to what extent an explanation explains a label, or how much new information (e.g., background knowledge, reasoning) an explanation supplies to explain a label beyond the original input. Take commonsense question-answering for example, the input question is "Why do people go hiking?", and the model prediction is "enjoy nature". An explanation "Hiking means the activity of going for long walks especially across country, or in the nature. People who go hiking enjoy nature." contains more information than a vacuous one "People go hiking to enjoy nature", though both satisfy existing evaluations (e.g., faithfulness, plausibility). I propose an information-theoretic metric to quantify the new information in an explanation that supports a given label beyond the information already available in the input or the label. This metric demonstrates consistent evaluations with human judgements and offers deeper insights into a model's reasoning and prediction processes.

- ***Diagnosing and debugging models via explanations for trustworthy NLP***. I identify model robustness and fairness issues through the lens of explanations. I discover the discrepancy between models' decision-making on original examples and their adversarial counterparts via explanations. For example, a model correctly predicts an example “a fantastic movie” as positive based on “fantastic”, while for its adversarial counterpart “a marvelous movie”, the model makes a positive prediction based on a neutral word “movie”. This reveals a model robustness issue. To address the problem, I propose a feature-level adversarial training method, teaching models to behave consistently on predicting original/adversarial example pairs by focusing on the corresponding important features (e.g., fantastic and marvelous), hence improving model robustness to adversarial attacks. In addition, I discover many model pathologies in few-shot settings (with scarce training data) through explanations. For example, pre-trained language models have strong prediction bias across labels; while fine-tuning with a few examples can mitigate the prediction bias, the model prediction behavior might be pathological (e.g., capturing spurious features from data). This provides insights for future research on building more reliable and trustworthy NLP models under data scarcity.

The research in this dissertations is expected to benefit NLP and AI developers, providing them with a better understanding of neural network models and helping them build trustworthy and reliable intelligent systems

List of publications

- *Building interpretable neural language models by transparentizing their decision-making*

1. **Hanjie Chen** and Yangfeng Ji. Improving the explainability of neural sentiment classifiers via data augmentation. *NeurIPS 2019 Workshop on Robust AI in Financial Services*, 2019.

2. **Hanjie Chen** and Yangfeng Ji. Learning Variational Word Masks to Improve the Interpretability of Neural Text Classifiers. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4236–4251, Online. Association for Computational Linguistics, 2020.

- *Explaining neural language models and evaluating their explanations for model understanding*

3. **Hanjie Chen**, Guangtao Zheng, and Yangfeng Ji. Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5578–5593, Online. Association for Computational Linguistics, 2020.

4. **Hanjie Chen**, Song Feng, Jatin Ganhotra, Hui Wan, Chulaka Gunasekara, Sachindra Joshi, and Yangfeng Ji. Explaining Neural Network Predictions on Sentence Pairs via Learning Word-Group

Masks. *In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, pages 3917–3930, Online. Association for Computational Linguistics, 2021.

5. **Hanjie Chen**, Wanyu Du, and Yangfeng Ji. Explaining predictive uncertainty by looking back at model explanations. *AAAI Workshop on Uncertainty Reasoning and Quantification in Decision Making*, 2023.

6. **Hanjie Chen**, Faeze Brahman, Xiang Ren, Yangfeng Ji, Yejin Choi, and Swabha Swayamdipta. REV: Information-theoretic evaluation of free-text rationales. *arXiv preprint arXiv:2210.04982*, 2022.

- *Diagnosing and debugging models via explanations for trustworthy NLP*

7. **Hanjie Chen** and Yangfeng Ji. Adversarial Training for Improving Model Robustness? Look at Both Prediction and Interpretation. *In Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI)*, Vancouver, BC, Canada, Feb. 2022.

8. **Hanjie Chen**, Guoqing Zheng, Ahmed Hassan Awadallah, and Yangfeng Ji. Pathologies of Pre-trained Language Models in Few-shot Fine-tuning. *ACL Workshop on Insights from Negative Results in NLP*, Dublin, Ireland, May 2022.

1.4 Dissertation Outline

For the rest of the dissertation, I first discuss the background in Chapter 2, including interpretable model design and development, post-hoc explanations, explanation evaluations, and interpretability for model diagnosis and debugging. Then I elaborate the three threads of my research. Chapter 3 summarizes my work on improving model interpretability via data augmentation and variational word masks. Chapter 4 introduces the hierarchical explanation, efficient explanation, and uncertainty explanation methods. Chapter 5 proposes an information-theoretic evaluation metric for free-text rationales. Chapter 6 demonstrates a novel feature-level adversarial training method for improving model robustness and reports model pathologies in few-shot fine-tuning. Finally, I conclude the dissertation and discuss the outlook for future work in Chapter 7.

Chapter 2

Background

Over the past years, neural model interpretability has been explored for cultivating trustworthy AI. This chapter provides a literature review on model interpretability from the following perspectives: interpretable model design and development, post-hoc explanations, explanation evaluations, and interpretability for model diagnosis and debugging.

2.1 Interpretable Model Design and Development

Building interpretable models is essential for trustworthiness. An interpretable model obeys a domain-specific set of constraints to allow it (or its predictions, or the data) to be more easily understood by humans [23]. For example, the constraints can be applied to regularize model structures to be transparent, such as linear additive models. A complex interpretable model can be composed of a stack of interpretable components (e.g., linear models, decision trees) under the constrain of transparent decision-making, such as generalized additive models [38]. The constraints can also be applied to regularize model predictions to be more interpretable. A representative example is to enforce sparsity in model predictions, such as restricting the number of features the model focuses on or the number of neurons being activated [39]. Those constraints make a model simpler and more comprehensible to humans.

Despite the promising side of building interpretable models, there is much debate on the trade-off between model interpretability and prediction performance [17]. Intuitively, sophisticated deep neural networks have better capability of capturing semantic information of inputs and handling more complicated tasks compared to simple models. However, deep neural networks are black-box models. Their lack of interpretability can result in catastrophic problems, especially in high-stakes applications, such as health care, criminal justice, etc. Although adding constrains into the development of black-box models can make them more transparent and interpretable, their prediction performance can drop significantly. For example, enforcing sparsity in input features or activated neurons can cause information loss or blocks in the transmission of information. Additionally, simply applying interpretability regularizations to model training can force the model to find suboptimal solutions [23].

Designing interpretable models is particularly challenging for NLP [32]. On the one hand, text data are usually represented by high-dimensional representations (e.g., word embeddings). NLP models are designed to encode the complex information and capture the high-level semantic meaning of input texts to make predictions. It is not trivial to enforce sparsity in input representations or intermediate computations within the model. On the other hand, simple models or a stack of simple networks may fall short in solving complex NLP tasks, such as commonsense question-answering [40] and natural language inference [41], which require understanding and reasoning on texts. Considering the prevalence of large pre-trained language models in NLP [13, 14, 15], it is not reasonable to keep people from using them. Additionally, it is almost impossible to design an interpretable model from scratch that can reach the same performance as the state-of-the-art language models. Some work builds self-explaining models based on existing neural networks by incorporating interpretable layers [33]. The (globally or locally) interpretable layers identify the most influential concepts in the training set for a given example or quantify the contribution of each local input concept by computing a relevance score relative to the predicted label. Although self-explaining models provide explanations for their predictions, they are partially interpretable or transparent and require thoughtful engineering design.

Another line of works improve interpretability by regularizing model prediction behavior [42, 43, 44, 45]. For example, human-annotated rationales are applied to regularize model explanations [42, 43]. The idea is to align model decision-making with human reasoning, hence making model prediction more interpretable. Those methods mostly rely on human annotations or external resources, which are expensive and time-consuming for collection. Despite the enhancement of interpretability, model prediction performance may drop due to the inconsistency between model decision-making and human reasoning [34]. Besides, many human annotations have quality issues and are not sufficient for solving the task or guiding model learning [46].

Without relying on human annotations, my work proposes to automatically learn task-specific important features and teach models to focus on those features to make predictions, hence improving their interpretability in Chapter 3.

2.2 Post-hoc Explanations

A main direction in explainable AI is explaining neural model predictions in the post-hoc manner. Specifically, a post-hoc explanation is produced to explain why the model makes the prediction. Post-hoc explanations broadly fall into two categories: model-agnostic (black-box) explanations and model-specific (white-box) explanations [39]. The former explains a model by inferring the relationship between its inputs and outputs without requiring access to the model's internals [1, 24]. The latter is dedicated to specific models and requires access to the model inside or additional information from the model (e.g., attention weights, gradients) [25, 47].

Model-agnostic explanations are generally applied to any black-box models regardless of their architectures or implementations. Many model-agnostic explanation methods are based on input perturbations. For example, Li et al. [48] propose Leave-one-out to identify feature attributions by erasing a certain feature and observing model prediction probability change. This method offers insights into which part of the input is important to model prediction. However, leaving one feature out at each time cannot quantify the influence of multiple features on model predictions. To address this limitation, a line of Shapley-based methods are proposed, such as Sampling Shapley [2], KernelSHAP [24], and L/C-Shapley [49]. The Shapley value [50] originally stems from coalitional game theory, providing an axiomatic solution to attribute the contribution of each player in a game in a fair way. Shapley-based methods adapt the Shapley value to quantify feature contributions to model predictions. They compute feature contributions by considering coalitions between features, hence providing more faithful explanations than Leave-one-out. LIME proposed by Ribeiro et al. [1] estimates individual feature attributions locally by linear approximation from perturbed examples. Specifically, an interpretable proxy model (e.g., linear model, decision tree) is applied to approximate the black-box model in a local region. Then the explanation for the proxy model is used to explain the black-box model locally. Although model-agnostic methods are useful, they pose certain risks as we cannot ensure that the explanation accurately reflects the model's true reasoning processes.

Model-specific explanations are designed for specific types of models. For example, Murdoch et al. [51] propose a contextual decomposition (CD) method for explaining Long Short Term Memory networks (LSTMs) [9]. The idea is to decompose the output of a LSTM into two parts—those resulting solely from the given feature and those involving other features. CD is able to capture the contributions of combinations of features (e.g., words) to the prediction of an LSTM model. A hierarchical version of CD called agglomerative contextual decomposition (ACD) is further proposed to aggregate feature attributions and form a hierarchical explanation [3]. Jin et al. [52] identify the limitations of CD and ACD in calculating phrase interactions in a formal context and propose to quantify context independent importance of words and phrases. Attention-based methods are proposed to explain models that contain attention layers [47, 53, 27, 28]. Although attention weights explain the workings of neural models to some extent, there has been a debate casting doubt on the explanatory power of attentions [54]. Jain and Wallace [28] question the fundamental of attention explanations based on their poor correlations to other explanations and model predictions. Wiegreffe and Pinter [55] challenge some viewpoints of Jain and Wallace [28], arguing "attention is not not explanation". To this day, the debate is still continuing. Gradient-based methods attribute feature contributions by computing the gradients of the output towards the input [56, 25, 26, 57]. The fundamental hypothesis is that a larger gradient indicates a more influential feature to the model prediction. Gradient-based methods are known for their implementation efficiency. However, due to the need for multiple forward and back propagation calculations, these methods may produce noisy results (e.g., fuzzy and blurry saliency maps). Adebayo et al. [58] report that some gradient-based methods fail in explaining the tasks that are sensitive to either data or model.

Another line of works explain neural models by analyzing the functions of different parts (e.g., layers, neurons) of neural networks [59, 60, 61]. For example, we can comprehend layers by evaluating their capacity to aid in resolving diverse problems that differ from the ones the network was initially trained to tackle. In addition, the role of individual neurons can be understood qualitatively, by creating visualizations of the input patterns that maximize the response of a single unit, or quantitatively, by testing the ability of a unit to solve a transfer problem [62]. Closely related to characterizing individual neurons, some works interprets the representations from neural network models to understand their prediction behavior [61]. Meng et al. [63] propose a causal intervention method to identify neuron activations that are important for model predictions and reveal the important role of mid-layer feed-forward modules in storing factual associations. Developing explanations from model internals is beneficial for many other directions, such as model editing, diagnosing and debugging, and efficiency. However, the limited access to some pre-trained language models (e.g., GPT-3 [15]) poses challenge and difficulty.

Another way of explaining neural network models is from the information-theoretic perspective [29, 64, 65, 66]. For example, Chen et al. [65] propose an information-based framework to learn instancewise informative features that have the maximal mutual information with the model output variable. Schulz et al. [29] proposes an information bottleneck framework to learn to attribute a relevance score to each individual input feature. The main challenge of information-based methods is to solve the optimization problem with mutual information items.

This dissertation focuses more on model-agnostic explanations. In addition to the limitations discussed above, most work focuses on identifying individual feature attributions (e.g., word-level saliency maps). Since natural languages are generally complex—containing negations and transitions—explanations should be able to capture feature interactions in model processing input tokens (as discussed in Chapter 1). Previous work proposed to generate higher-level explanations (e.g., phrase-level and hierarchical explanations). For example, Tsang et al. [67] generate hierarchical explanations by considering the interactions between any features with exhaustive search, which is computationally expensive. Singh et al. [3] propose ACD which utilizes CD scores [51] for feature importance evaluation and employ a hierarchical clustering algorithm to aggregate features together for hierarchical explanation. Lundberg et al. [68] calculate features interactions via SHAP interaction values along a given tree structure. Chen and Jordan [69] utilize a linguistic tree structure to capture the contributions beyond individual features for text classification. Different from previous methods that require decomposition of neural network layers or hierarchical structures available, I propose a model-agnostic method in Section 4.1 to construct hierarchical explanations solely based on feature interaction detection without resorting external structural information.

For sentence-pair classification tasks (e.g., natural language inference), computing feature interactions between all word pairs is computationally inefficient [67]. The methods [3, 52] that only consider the interactions between adjacent words are not applicable to sentence pair modeling tasks as critical interactions usually form between words from different sentences. To address these issues, I propose to implicitly detect correlated words from a sentence pair and distribute them into a group and learn the group importance in Section 4.2. Then weighted word attributions are computed based word distributions and group importance.

In addition to explaining prediction labels, explaining predictive uncertainty is significant for comprehensively understanding model prediction behavior. Predictive uncertainty estimation of pretrained language models is an important measure of how likely people can trust their predictions [70, 71]. However, little is known about what makes a model prediction uncertain. Explaining predictive uncertainty is an important complement to explaining prediction labels in helping users understand model decision making and gaining their trust on model predictions, while has been largely ignored in prior works. In Section 4.3, I propose to explain the predictive uncertainty of pre-trained language models by extracting uncertain words from existing model explanations. I find the uncertain words are those identified as making negative contributions to prediction labels, while actually explaining the predictive uncertainty. Uncertainty explanations are indispensable to explaining models and helping humans understand model prediction behavior.

2.3 Explanation Evaluations

Evaluating model explanations is critical for ensuring the information being provided is accurate and trustworthy. Generally, explanations are evaluated from two perspectives—faithfulness to models and plausibility to humans [34]. Faithfulness measures to which extent rationales reflect the true reasoning process of models, while plausibility assesses how convincing rationales are to humans.

The evaluation of faithfulness can be further divided into comprehensiveness and sufficiency [72]. Comprehensiveness quantifies the influence of a feature by leaving it out and then observing the model predicted probability change on the same class. Intuitively, the model is expected to be less confident in its prediction once important features are removed from the input. Sufficiency evaluates the degree to which the important features identified by the explanation are sufficient for the model to make the prediction. Most automatic evaluations assess model explanations in these two aspects [73, 72, 34]. Beyond evaluating important features, the degradation test [74, 29] considers both most important features and least important features and their influence on model predictions.

The assessment of plausibility involves human evaluations. Doshi-Velez and Kim [16] design binary forced choice, forward simulation, and counterfactual simulation. Specifically, humans are asked to compare the quality of pairs of explanations, predict the model's outputs based on explanations, or guess what has to be changed to change the model's predictions. Hase and Bansal [75] conduct human studies by separating explained instances from test instances and blocking users from guessing model (correct) predictions solely based on inputs. Lage et al. [76] conduct carefully controlled human-subject experiments across different tasks to understand what factors make models interpretable. Furthermore, Arora et al. [77] evaluate explanations via a crowdsourcing study, allowing participants to interact with models.

In addition to faithfulness and plausibility, the robustness of explanations is also crucial for evaluation. Many works have demonstrated the vulnerability of explanations to input or model perturbations [78, 79, 80, 81]. Particularly, Wang et al. [82] reveal that gradient-based explanations are fragile to manipulations, such as increasing the gradient on the stop words or the first input word. Sinha et al. [83] propose a method to perturb input words to identify fragile explanations. Tang et al. [84] further investigate the source of vulnerability by disentangling models and explanation methods. Other evaluation perspectives include the ability of explanations in helping a student model simulate a teacher model [85] or bridging the communication between a classifier and a layperson [86].

With the appearance of natural language explanations (or free-text rationales) [42, 87, 88, 46, 89]—explaining models in the form of natural language, some evaluation metrics (e.g., sufficiency, comprehensiveness) have become inapplicable because they are only applied to features identified within the input. Some evaluation metrics proposed for natural language explanations focus on the association between explanations and labels [90, 91]. Specifically, the utility of an explanation is evaluated based on how much it helps a model proxy predict the given label, which is inspired by human simulatability [16]. Chan et al. [92] further propose a framework to evaluate the automatic metrics. However, none of them consider measuring the amount of additional new information (e.g., background or commonsense knowledge) in the explanation, beyond what is contained in the input or the label. Sun et al. [93] conduct a human study on the additional knowledge provided by natural language explanations. In Chapter 5, I propose an information-theoretic metric to quantify the new information in natural language explanations.

2.4 Interpretability for Model Diagnosis and Debugging

Model interpretability offers an access for people to understand, diagnose, debug, and improve models. For example, explanations have been utilized to regularize models' prediction behavior by forcing them to align with human-annotated rationales [94, 37, 35, 18]. Explanations have also been leveraged to debug and improve model robustness, fairness and some other properties [95, 96].

Neural networks have shown vulnerability to adversarial examples which are formed by applying small but intentionally worst-case perturbations to their original counterparts [97, 98, 99]. Ross and Doshi-Velez [100] proposed to improve model robustness by regularizing input gradients. Boopathy et al. [95] and Chen et al. [101] showed that regularizing interpretation discrepancy between original and adversarial examples can improve model robustness. The above methods were proposed to defence adversarial attacks in image domain, while not directly applicable to NLP due to the discreteness of text data. Most adversarial examples in text domain are generated by heuristically manipulating input texts, such as replacing words with their synonyms [102, 103]. A common way to improve model robustness is adversarial training which follows two steps: (1) collecting adversarial examples by attacking a target model and (2) fine-tuning the model on the augmented dataset with these adversarial examples [103, 104]. The objective of traditional adversarial training is making a model produce the same correct predictions on original/adversarial example pairs. Nevertheless, a robust model should behave consistently on predicting similar texts beyond producing the same predictions. Regularizing model prediction behavior should be considered in improving model robustness during adversarial training. In Section 6.1, I propose to improve model robustness through the lens of explanations by making models produce the same predictions based on the same reasons.

Pre-trained language models like BERT [13], RoBERTa [105] and GPT-3 [106] have shown remarkable adaptation performance on downstream tasks due to their learning ability and prior knowledge gained during pre-training [107]. Moreover, this adaptation power is also expected when the number of task-specific examples is limited, which is likely to happen in the real world, such as data privacy or expense constraints. To fulfill this goal, much effort has been put into improving model adaptation performance in few-shot or zero-shot settings [108, 109, 110]. However, there is a lack of work on analyzing model adaptation behavior or explicit evidence supporting that the performance gain is truly based on useful information. Utama et al. [111] disclosed that models obtained from few-shot prompt-based fine-tuning tend to adopt inference heuristics (i.e. lexical overlap) to make predictions on sentence pair classification tasks. Ma et al. [112] revealed the performance variance of entailment-based models across zero-shot text classification tasks. Zhao et al. [113] discovered the instability of model performance towards different prompts in few-shot learning. These works either focused on a specific scheme (e.g. prompt) in few-shot learning or a common problem (e.g. lexical overlap) associated with a type of sentence pair classification tasks. Differently, I study a more general adaptation process of pre-trained language models in standard few-shot fine-tuning. Besides, I explain model adaptation behavior via post-hoc explanations and discover many pathologies of pre-trained language models in few-shot settings in Section 6.2.

In terms of fairness, neural language models can rely on shortcuts (e.g., dataset biases, spurious correlations) to make predictions [114, 115, 116]. Some shortcut features (e.g., negation words) are highly associated with a specific label, resulting in biased and unfair model predictions [111, 117, 118]. It is promising to identify and mitigate shortcut features or spurious correlations via model explanations.

Chapter 3

Improving Model Interpretability

I build interpretable models by teaching them to focus on critical information to make predictions—making their prediction behavior interpretable. In this chapter, I introduce two data augmentation methods in Section 3.1 and the variational word masks (VMASK) method in Section 3.2. These methods improve the interpretability of *existing* models while maintaining their prediction performance.

3.1 Improving the Interpretability of Neural Sentiment Classifiers via Data Augmentation

Sentiment analysis is one of the most widely-used applications of natural language processing (NLP), where neural sentiment classifiers help enterprises gauge pubic opinion, conduct market research, monitor brand and product reputation, and understand customer experiences [119, 120, 121]. The recent development of neural network modeling has largely boosted the prediction performance (e.g., accuracy) on sentiment classification [122, 123, 124, 125], while the nonlinearity of neural network models hinders the understanding on predictions. A fair question with no easy answer for neural sentiment classifiers is *why the prediction on this text is positive (or negative)?* Moreover, the lack of interpretability on model prediction will raise the issue of trustworthy and fairness of sentiment classifiers in practice [62, 126].

To address the interpretability issue of neural classifiers, various approaches have been developed recently to provide model-agnostic explanations on predictions [1, 127, 24]. Particularly, this work focuses on local explanations, which aims to explain predictions for individual data. The most

Review	a truly moving experience, and a perfect example of how art when done right can help heal, clarify, and comfort.
Pred. A	Positive
Exp. A	a, moving, can
Pred. B	Positive
Exp. B	perfect, comfort, truly

Table 3.1: Explanations generated by the local explanation method LIME from two neural sentiment classifiers. The ground-truth sentiment polarity of the text is positive and *both* models give the right prediction.

common way of generating local explanations in sentiment classification is identifying the important part of a text associated with predicted sentiment polarity [128, 65, 129]. For example, a widely-adopted local explanation method LIME [1] can identify a set of keywords as an explanation.

Table 3.1 presents an example of movie reviews and the explanations based on two neural sentiment classifiers. Although both classifiers give the right prediction on this example, the explanation A is harder to be *interpreted* than the explanation B, in terms of why the prediction is positive. This difference on the interpretability of explanations leads us to trust more on prediction B than A, which will eventually discriminate the practical values of these two sentiment classifiers.

In general, a prediction explanation can be generated by using any local explanation method [1, 130, 131]. However, the real challenge in practice is that whether an explanation is easy to be interpreted, as demonstrated in Table 3.1. By noticing the connection and difference between explanations and their interpretability, we would like to study the problem on **improving the interpretability of neural sentiment classifiers**. We consider this as a learning problem and propose to resolve it with some data augmentation methods. The goal is to increase the interpretability of existing neural sentiment classifiers while maintaining similar prediction performance.

In this work, we explore the strategy of using data augmentation to improve the interpretability of neural sentiment classifiers. We propose two data augmentation methods: one with a predefined sentiment word list as external knowledge and the other with adversarial examples. Experiments on the two base models and three benchmark datasets show that the proposed methods improve the model interpretability with respect to both automatic and human evaluation.

3.1.1 Data Augmentation Methods

The basic idea is to teach the models to make predictions based on critical information. In the scenario of sentiment classification, the task is to teach model to make predictions by grasping

Original text	the only problem is that, by the end, no one in the audience or the film seems to really care
Da-Ek	the only that , by the end, one in the audience or the film seems to care
Adversarial example	the only difficulty is that, by the end, no one in the audience or the movie seems to really caring
DA-Adv	the only is that, by the end, no one in the audience or the seems to really
Original text	michel piccoli's moving performance is this films reason for being
Da-Ek	michel piccoli's this films reason for being
Adversarial example	michel piccoli's moving play is this movie reason for being
DA-Adv	michel piccoli's moving is this reason for being

Table 3.2: Some examples of the augmented data created by Da-Ek and DA-Adv.

sentiment words. This section presents two proposed methods for data augmentation and a unified method of using augmented data for training.

Augmenting via External Knowledge

The first method is called data augmentation with external knowledge (Da-Ek). We propose a simple method to create some examples that are similar to training examples with respect to their surface forms, but those examples do not belong to any of the predefined classes $\mathcal{Y}$. To be specific, the augmented examples for sentiment analysis are the examples that are similar to original training examples but have no sentiment polarity, as illustrated in Table 3.2.

A simple way to create an augmented example $\tilde{\boldsymbol{x}}$ is that, for a given sentence $\boldsymbol{x}$, removing words $\{\boldsymbol{x}_i\}$ from $\boldsymbol{x}$ if $\boldsymbol{x}_i$ belongs to a predefined sentiment word list. In this work, we use the words listed in the SentiWordNet corpus [132] and their sentiment polarity scores. For a given sentence $\boldsymbol{x}$, removing $\boldsymbol{x}_i$ from $\boldsymbol{x}$ if $\boldsymbol{x}_i$ is in the word list will create an augmented example $\tilde{\boldsymbol{x}}$. Table 3.2 presents two examples of the original text and its augmented counterpart after removing words with clear sentiment polarity. For some simple texts, removing sentiment words will cause their augmented counterparts to be incomplete sentences, which can still be used as augmented data points. For example, if we remove the sentiment word in text `I like this movie`, then the augmented training example is `I this movie`. However, with the training framework proposed in Section 3.1.2, this augmented example will help the model to emphasize the sentiment prediction on the original sentence.

There is a critical distinction between the augmented examples created by Da-Ek and the example from the *neutral* class in sentiment classification. In multi-class sentiment classification, there is often a class with an average sentiment score called the neutral class. The major difference is that texts from a neutral class still have sentiment, or at least contains some sentiment words. For example, the movie review `The Cockettes provides a window into a subculture hell-bent on expressing itself in every way imaginable.` is a neutral but not augmented example. With words like `hell-bent` and `imaginable`, it shows sentiment inclination of this movie review even though it is not strong. To construct an augmented example from this text, the proposed method still needs to remove the sentiment words. Empirical results show that adding neutral examples can only lead to a minor improvement on interpretability.

Augmenting with Adversarial Examples

This method is called data augmentation with adversarial examples or DA-Adv. We adopt the method proposed by Alzantot et al. [102] to generate adversarial examples, which may have similar surface forms and semantic meanings to training examples. To be specific, this method aims to minimize the number of modified words between the original and adversarial examples, and maintain semantic and syntactic similarity by substituting only a few synonyms. A well-known challenge on generating adversarial examples in text data is that texts are discrete, which causes the difficulty in generating adversarial examples by using the popular gradient-based methods [98, 133, 134]. Alzantot et al. [102] developed an attack algorithm via genetic algorithms. In each generation, a group of candidate examples are generated by substituting synonyms, and those most fit within the context surrounding are selected by the Google 1-billion words language model [135]. The candidates that can successfully attack the model to flip prediction polarity are adversarial examples. Like many other adversarial attack methods, there is a budget about how many words can be replaced. Beyond that budget limit will cause a fail attack. In our case, it means not every text can get an adversarial example.

As adversarial examples can flip model predictions, the replaced words from original texts must be critical to sentiment prediction. Similar to the previous data augmentation method, we can construct augmented examples by taking the replaced words as the sentiment words in Da-Ek.

Comparison. Table 3.2 presents some examples of two data augmentation methods. With Da-Ek, we have a high-precision method for data augmentation. If any word in a text matches one entry in the SentiWordNet, then it is very likely to be a sentiment word. However, the word list in the SentiWordNet is predefined and definitely not comprehensive. The missing sentiment words imply Da-Ek could be a data augmentation method with low recall. With DA-Adv, we have a low-precision method for data augmentation. Words identified by adversarial attacks can be sentiment words or simply can be non-sentiment words that are sensitive to neural sentiment classifiers. Besides, finding adversarial examples is very time consuming, as further explained in Section 3.1.4. But DA-ADV has the potential to extend this method to other text classification tasks, where we do not have a pre-defined word list.

3.1.2 Learning with Augmented Examples

We extend the training set $\mathcal{D} = \{(\boldsymbol{x}^{(k)}, y^{(k)})\}$ as by adding the augmented examples $\{(\tilde{\boldsymbol{x}}^{(k')}, \text{AUG})\}$ generated by either DA-EK or DA-ADV and extend it as $\widetilde{\mathcal{D}} = \mathcal{D} \cup \{(\tilde{\boldsymbol{x}}^{(k')}, \text{AUG})\}$. Similarly, the label set $\mathcal{Y}$ is also extended as $\widetilde{\mathcal{Y}} = \mathcal{Y} \cup \{\text{AUG}\}$. Note that, the proposed methods only create augmented examples for the training set and development set. No modification is on the test set.

Once we have the extended training and development set, learning a neural sentiment classifier with data augmentation is straightforward. Specifically, we optimize the following loss function

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\widetilde{\mathcal{D}}} \mathcal{L}(\hat{y}^{(k)}, y^{(k)}), \tag{3.1}$$

to achieve the best prediction accuracy on the *augmented* development set, where $\hat{y}^{(k)}$ is decoded from the decision function defined in Equation 3.2 with extended label set $\widetilde{\mathcal{Y}}$, $\mathcal{L}(\cdot, \cdot)$ is the cross-entropy loss, and $\boldsymbol{\theta}$ is the collection of parameters, which is the same as the base model. During test, with no augmented example, the trained neural classifier simply ignores any prediction on the label AUG and picks the label from $\mathcal{Y}$ that maximizes the decision value in Equation 3.2.

3.1.3 Experimental Setup

This section describes the experimental setup used in this work. We test the proposed data augmentation methods with two neural sentiment classifiers, a convolutional neural network in [11, CNN] and a recurrent neural network with LSTM [9, RNN], on three benchmark datasets, SST [136], MR [137], and IMDB [138]. Local explanations were generated from LIME [1] with model predictions and the cosine similarity method based on text representations.

Datasets. We use three sentiment benchmark datasets for evaluation.

- **SST.** This dataset was proposed in [136] for sentence-level sentiment classification. We used the SST-2, which is the 2-class version of this dataset. There are 6,920 examples in the training set, 872 examples in the development set, and 1,821 examples in the test set. For data augmentation with DA-EK, additional 1,624 and 229 augmented examples were added to the training and development sets respectively. With DA-ADV, 4,885 and 539 augmented examples were added to the training and development sets respectively.

- **MR.** This dataset was proposed by Pang and Lee [137]. These reviews in this dataset were divided into 9,596 training and 1066 test examples. In our experiments, 90% of the training examples are used for training, and the rest is used as development set. With DA-EK, we added additional 4,318 and 480 augmented examples to the training and development sets respectively.

- **IMDB.** This dataset was proposed by Maas et al. [138]. These reviews in this dataset were divided into 25,000 training and 25,000 test examples. We split 90% of the training examples for training, and the rest as the development set. With DA-EK, additional 11,250 and 1,250 augmented examples were added to the training and development sets respectively.

Neural sentiment classifiers. In this work, we use a convolutional neural network in [11, CNN] and a recurrent neural network with LSTM [9, RNN] as our baseline models.

The CNN consists of an input layer that takes word embeddings as inputs, a convolutional layer followed by a max-pooling layer for composing word embeddings into text representations, and a softmax layer for classification. For a given text $\boldsymbol{x}$, $\boldsymbol{f}(\cdot)$ denotes the representation function in CNN, which maps $\boldsymbol{x}$ into a d-dimensional numeric vector $\boldsymbol{f}(\boldsymbol{x})$ as text representation. The decision function is defined as

$$\boldsymbol{h}(\boldsymbol{x}, y) = \boldsymbol{u}_y^{\top}\boldsymbol{f}(\boldsymbol{x}), \tag{3.2}$$

where $y \in \mathcal{Y}$ is the class label, and $\mathbf{u}_y \in \mathbb{R}^d$ is the corresponding classification weight vector. For prediction, we use $\hat{y} = \text{argmax}_y\, \boldsymbol{h}(\boldsymbol{x}, y)$.

The RNN consists of uni-directional one-layer LSTM. For a given text, the last hidden state of this RNN is used as the text representation $\boldsymbol{f}(\boldsymbol{x})$. The same decision function defined in Equation 3.2 is employed for sentiment prediction.

Even though the main focus of this work is on model explainability, the prerequisite is to match the classification performance in prior work with the similar model architectures. Here are some implementation details that we adopted from prior work [11]. For both the CNN, CNN-EK and CNN-ADV, we used a single convolutional layer with filters of the window sizes ranging from 3 to 5. For both RNN and RNN-EK, we used a single layer LSTM. For all of the models, the input parameters were initialized with the 300-dimensional pretrained word embeddings [139, word2vec] and all other parameters were randomly initialized with the default method in PyTorch. Hyperparameters, including kernel size (for CNN only), hidden size (for RNN only), learning rate, minibatch size, etc., were tuned separately on the development set for different datasets. We used Adam [140] to update the parameters.

Local Explanation Generation

To generate local explanations, we adopt the LIME proposed by Ribeiro et al. [1] to generate explanations on model predictions. Besides, we also suggest another way of generating local explanations based on the cosine similarity between word representations and text representations.

LIME with model predictions. The basic idea of the LIME is that, for a given example $\boldsymbol{x}$, it finds an explanation based on a locally linear approximation $\boldsymbol{g}(\boldsymbol{z}^{(l)}, y)$ of the decision function $\boldsymbol{h}(\boldsymbol{z}, y)$, in which $\boldsymbol{z}$ is a perturbation of $\boldsymbol{x}$ obtained by subsampling the words from $\boldsymbol{x}$. Given a set of subsamples $\{\boldsymbol{z}^{(l)}\}$ from $\boldsymbol{x}$, the loss function of the LIME is defined as

$$L(\boldsymbol{h}, \boldsymbol{g}) = \sum_{i=1}^{q} D_{\boldsymbol{x},\boldsymbol{z}^{(l)}} (\boldsymbol{h}(\boldsymbol{z}^{(l)}, y) - \boldsymbol{g}(\boldsymbol{z}^{(l)}, y))^2, \tag{3.3}$$

where the linear approximation function $\boldsymbol{g}$ is usually defined as $\boldsymbol{g}(\boldsymbol{z}, y) = \boldsymbol{w}_y^\top \boldsymbol{z}$. $D_{\boldsymbol{x},\boldsymbol{z}^{(l)}}$ measures the similarity between $\boldsymbol{x}$ and $\boldsymbol{z}^{(l)}$,

$$D_{\boldsymbol{x},\boldsymbol{z}^{(l)}} = \exp\left(\frac{-d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{z}^{(l)}))^2}{\sigma^2}\right), \tag{3.4}$$

with $d(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{z}^{(l)})$ as the cosine distance between the latent representations of $\boldsymbol{x}$ and $\boldsymbol{z}^{(l)}$, as suggested in [1].

Optimizing Equation 3.3 will try to match the decision values from linear approximation $\boldsymbol{g}(\boldsymbol{z}^{(l)}, y)$ with $\boldsymbol{h}(\boldsymbol{z}^{(l)}, y)$ and also produce a set of linear weights $\{\boldsymbol{w}_y\}_{y\in\mathcal{Y}}$ associated with $\mathcal{Y}$. The values of $\{\boldsymbol{w}_{y,i}\}$ indicate the importance of $\{\boldsymbol{x}_i\}$. If the predicted label is $\hat{y}$, then top t words according to $\{\boldsymbol{w}_{\hat{y},i}\}$ will be selected as an explanation of $\boldsymbol{x}$ on the corresponding prediction.

Cosine similarity on text representations. For a given text $\boldsymbol{x}$, the basic idea of using cosine similarity generating explanations is to measure the similarity between its text representation $\boldsymbol{f}(\boldsymbol{x})$ and word representations $\boldsymbol{f}(\boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ is the embedding of the i-th word in text $\boldsymbol{x}$. In this way, we can find the most similar words with respect to the text representation $\boldsymbol{f}(\boldsymbol{x})$, and choose the top t words as an explanation. The underlying assumption of this idea is that, if a text representation

$\boldsymbol{f}(\boldsymbol{x})$ could facilitate sentiment prediction, the sentiment polarity indicated by the top t similar words should be consistent with its overall sentiment polarity.

To compute cosine similarity, we first need to map all the word embeddings $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ into the text representation space using $\boldsymbol{f}(\cdot)$. Then, the similarity between a text and the i-th word within this text is measured by the cosine value of these two vectors,

$$\text{cos-sim}(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}_i)) = \frac{\langle \boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x}_i) \rangle}{\|\boldsymbol{f}(\boldsymbol{x})\|_2 \cdot \|\boldsymbol{f}(\boldsymbol{x}_i)\|_2}. \tag{3.5}$$

After applying Equation 3.5 to every word in text $\boldsymbol{x}$, then we pick the top t words with respect to their cosine similarities as an explanation of $\boldsymbol{f}(\boldsymbol{x})$.

3.1.4 Experiments

Although there are different ways to evaluate prediction explanations as suggested in prior work [1, 62], the interpretability of explanations should be the most of important criterion. As argued by Gilpin et al. [62], a good explanation should be easily interpretable and "*simple enough for a person to understand using a vocabulary that is meaningful to the user*". For sentiment classification, as demonstrated in the running example (Table 3.1), a good explanation on sentiment prediction should be easy enough for a human user to understand together with the prediction. Following this intuition, we define the interpretability measurement for both automatic evaluation and human evaluation.

Automatic Evaluation

Our automatic evaluation method measures the interpretability of an local explanation (consisting of a set of keywords) by predicting its sentiment polarity and comparing with the model prediction. Specifically, for each keyword in an explanation, we retrieve its sentiment scores from the SentiWordNet. SentiWordNet offers three scores for a sentiment word: a positive score, a negative score, and a neutral score. For the word `truly`, its positive score is 0.625, negative score is 0 and neutral score is 0.375, which indicates that it is a word with positive polarity. On the other hand, the positive score of word `a` is 0 and its neutral score is 1. With the sentiment scores of these words, the overall scores of an explanation is the accumulation of the sentiment scores of its keywords.

Consider the local explanations in Table 3.1, the sentiment scores of explanation A with respect to the positive sentiment polarity is 0 and the positive socre of explanation B is 0.625. Under this simple automatic evaluation measurement, the explanation A is easier to be interpreted than explanation B, which is consistent with our expectation.

			Coherence	
Dataset	Model	Accuracy	LIME	Cos-Sim
SST	Cnn	0.85	0.65	0.58
	Cnn-Ek	0.85	0.70	0.60
	Cnn-Adv	0.85	0.68	0.59
	Rnn	0.84	0.64	0.61
	Rnn-Ek	0.84	0.66	0.62
MR	Cnn	0.81	0.63	0.50
	Cnn-Ek	0.80	0.66	0.55
	Cnn-Adv	0.80	0.65	0.55
	Rnn	0.80	0.64	0.59
	Rnn-Ek	0.80	0.65	0.60
IMDB	Cnn	0.90	0.76	0.23
	Cnn-Ek	0.90	0.80	0.53
	Cnn-Adv	0.90	0.78	0.47
	Rnn	0.87	0.78	0.74
	Rnn-Ek	0.87	0.81	0.78

Table 3.3: The classification and explainability evaluation results of different models on SST, MR and IMDB. The models trained with augmented data from Da-Ek are named with -Ek. The model trained with the augmented data from DA-Adv is named with -Adv.

To quantificationally evaluate prediction explanations, we propose the *coherence score* defined as follow: for a given test example, depending on the sentiment polarities predicted by the model, indicated by the explanation and the ground truth, it will be counted as a coherent case, if it satisfies one of the two conditions:

- Condition 1: if the sentiment polarity indicated by the explanation is not none and is consistent with the model prediction; or
- Condition 2: if the sentiment polarity indicated by the explanation is none and the model prediction is not the same as the ground truth.

The coherence between the model prediction and its prediction in condition 1 is obvious. About condition 2, we consider that an explanation with no sentiment polarity is also coherent with a wrong prediction. Intuitively, an explanation with no sentiment polarity explains why the prediction is wrong. For a collection of explanations, the coherence score is the ratio of the number of coherent cases to the total number of instances.

Results

Table 3.3 shows the prediction accuracies and coherence scores of different models on the all three datasets. As indicated in the third column, the models trained with augmented data, including both Da-Ek and DA-Adv, maintain the prediction accuracies comparing to their counterparts. This observation matches our expectation that data augmentation for improving interpretability should not hurt prediction accuracy.

More important, all models trained with augmented data outperform the base models with respect to the coherence score (column 4 and 5). Comparing the coherence scores with the same base model and the same dataset, we found that, in most of the cases, both data augmentation methods help improve the coherence score, regardless which explanation generation we use. Comparing the scores across multiple datasets and models, we also notice that the improvement on LIME-based explanations has a smaller variance than the explanations generated by the cosine similarity method. We suspect that this is because local explanations are always tied with model predictions, while the cosine similarity method only use text representations to generate explanations.

As shown in the experiments with the Cnn-Adv, data augmentation with adversarial examples does provide some benefit to improve the coherence scores of the Cnn model on all of the three datasets. The state-of-the-art method [102] generating adversarial examples can be extended to other neural calssifiers (e.g. Rnn) and text classification tasks in the future work.

Human Evaluation

We propose the coherence score and use it to automatically evaluate the local explanations. Even though it is easy to compute, the major limitation is from the pre-defined list of sentiment words. For a specific test example, this evaluation method will fail if the sentiment words in the generated explanation are not the SentiWordNet word list. Furthermore, as discussed in the beginning of this section, interpretability is about whether an explanation is understandable to human users. Human evaluation is necessary if we would like to measure the interpretability improvement. Besides the human evaluation results can provide further justification of the coherence scores from automatic evaluation.

To conduct a human evaluation task, we random pick 100 test examples from the SST and MR datasets. Explanations of these examples are generated by LIME based on the Cnn and Cnn-Ek models. We have 7 graduate students with proficient English skills as volunteers to evaluate the quality of these explanations.

Dataset	Model	Human Evaluation	Automatic Evaluation
SST	Cnn	0.85	0.56
	Cnn-Ek	0.92	0.63
MR	Cnn	0.84	0.55
	Cnn-Ek	0.90	0.60

Table 3.4: Human and automatic evaluation results on the sets of the SST and MR datasets. The augmented examples are from Da-Ek and the explanations are generated by LIME.

With a given test example with an explanation pair generated from the Cnn and Cnn-Ek models respectively, a human evaluator needs to a two-step evaluation. First, for each explanation, the human evaluator needs to analyze whether it can interpret the corresponding prediction, and mark with a score ("1" for coherent, "0" for incoherent) according to the two conditions, only with the evaluator himself to give the sentiment polarity of the explanation. Then, for the explanation pair, the evaluator will be asked to pick which one better explains the corresponding model prediction. Note that, two explanations within each pair are presented to our human evaluators randomly to eliminate any possible bias.

Finally, the human evaluation score is calculated as the ratio of the sum of the scores to the number of examples. We also calculate the coherence scores on the 100 test examples and compare them with the human evaluation scores.

Results

Table 3.4 presents both the human evaluation scores and also the coherence scores. On both datasets, human evaluation scores indicate that data augmentation with additional examples improves the interpretability of Cnn. As shown in Table 3.5, the explanations from Cnn-Ek are more interpretable and the sentiment polarity indicated by these two explanations are clear. In addition, the comparison between the human evaluation and the automatic evaluation also shows the coherence scores are positively correlated with the human evaluation scores, which provides a justification for our automatic evaluation measurement.

We also notice that the coherence scores are constantly lower than the human evaluation scores, even though the computations of these two scores are similar. One possible reason is that the pre-defined sentiment word list from the SentiWordNet is not comprehensive enough, while human evaluators can always tell which explanation is better than the other.

Input text	Models	Prediction	Keywords	Indication
[Pos] at about 95 minutes, treasure planet maintains a brisk pace as it races through the familiar story	Cnn	Positive	treasure, pace, a	None
	Cnn-Ek	Positive	brisk, treasure, familiar	Positive
[Neg] unfortunately, they're sandwiched in between the most impossibly dry account of kahlo's life imaginable	Cnn	Negative	dry, 're, sandwiched	None
	Cnn-Ek	Negative	unfortunately, dry, account	Negative

Table 3.5: Examples of the explanations generated by LIME from the Cnn and Cnn-Ek models, where the ground truth of each input text is marked in front of it as "Pos" or "Neg".

3.1.5 Conclusion

We showed that the interpretability of neural sentiment classifiers can be improved by training with augmented data. We proposed two data augmentation methods by employing a predefined word list and adversarial examples respectively. In this work, we focused on the interpretability of local explanations, which were generated by LIME and the cosine similarity method. Then, the improvement of model interpretability was assessed with both automatic evaluation and human evaluation. Experiments showed that the proposed data augmentation methods could successfully improve the model interpretability.

3.2 Learning Variational Word Masks to Improve the Interpretability of Neural Text Classifiers

Neural network models have achieved remarkable performance on text classification due to their capacity of representation learning on natural language texts [12, 122, 141, 13]. However, the lack of understanding of their prediction behaviors has become a critical issue for reliability and trustworthiness and hindered their applications in the real world [17, 1, 34]. Many explanation methods have been proposed to provide post-hoc explanations for neural networks [1, 24, 25], but they are only able to explain model predictions and cannot help improve their interpretability.

Ex.	Model	Text & Explanation
1	A	An exceedingly clever piece of cinema
	B	An exceedingly clever piece of cinema
2	A	It becomes gimmicky instead of compelling
	B	It becomes gimmicky instead of compelling

Table 3.6: Model A and B are two neural text classifiers with similar network architectures. They all make correct sentiment predictions on both texts (ex. 1: positive; ex. 2: negative). Two post-hoc explanation methods, LIME [1] and SampleShapley [2], are used to explain the model predictions on example 1 and 2 respectively. Top three important words are shown in pink or blue for model A and B. Whichever post-hoc method is used, explanations from model B are easier to understand because the sentiment keywords "clever" and "gimmicky" are highlighted.

In this work, we consider interpretability as an intrinsic property of neural network models. Furthermore, we hypothesize that neural network models with similar network architectures could have different levels of interpretability, even though they may have similar prediction performance. Table 3.6 shows explanations extracted from two neural text classifiers with similar network architectures.[1] Although both models make correct predictions of the sentiment polarities of two input texts (positive for example 1 and negative for example 2), they have different explanations for their predictions. In both examples, no matter which explanation generation method is used, explanations from model B are easier to be interpreted regarding the corresponding predictions. Motivated by the difference of interpretability, we would like to investigate the possibility of *building more interpretable neural classifiers with a simple modification on input layers.* The proposed method does not demand significant efforts on engineering network architectures [142, 143]. Also, unlike prior work on improving interpretability [44, 144], it does not require pre-defined important attributions or pre-collected explanations.

[1]The similarity will be detailed in Section 3.2.4 and more examples are provided in Table 3.10.

Specifically, we propose variational word masks (VMASK) that are inserted into a neural text classifier, after the word embedding layer, and trained jointly with the model. VMASK learns to restrict the information of globally irrelevant or noisy word-level features flowing to subsequent network layers, hence forcing the model to focus on important features to make predictions. Experiments in Section 3.2.5 show that this method can improve model interpretability and prediction performance. As VMASK is deployed on top of the word-embedding layer and the major network structure keeps unchanged, it is *model-agnostic* and can be applied to any neural text classifiers.

The contribution of this work is three-fold: (1) we proposed the VMASK method to learn global task-specific important features that can improve both model interpretability and prediction accuracy; (2) we formulated the problem in the framework of information bottleneck (IB) [145, 146] and derived a lower bound of the objective function via the variational IB method [147]; and (3) we evaluated the proposed method with three neural network models, CNN [11], LSTM [9], and BERT [13], on seven text classification tasks via both quantitative and qualitative evaluations.

3.2.1 Interpretable Text Classifier with Word Masks

For an input text $\boldsymbol{x} = [x_1, \cdots, x_T]$, where x_t $(t \in \{1, \ldots, T\})$ indicates the word or the word index in a predefined vocabulary. In addition, we use $\boldsymbol{x}_t \in \mathbb{R}^d$ as the word embedding of x_t. A neural text classifier is denoted as $f_{\boldsymbol{\theta}}(\cdot)$ with parameter $\boldsymbol{\theta}$, which by default takes $\boldsymbol{x}$ as input and generates a probability of output $\boldsymbol{Y}$, $p(\boldsymbol{Y}|\boldsymbol{x})$, over all possible class labels. In this work, beyond prediction accuracy, we also expect the neural network model to be more interpretable, by focusing on important words to make predictions.

To help neural network models for better feature selection, we add a random layer $\boldsymbol{R}$ after the word embeddings, where $\boldsymbol{R} = [R_{x_1}, \ldots, R_{x_T}]$ has the same length of $\boldsymbol{x}$. Each $R_{x_t} \in \{0, 1\}$ is a binary random variable associated with the word type x_t instead of the word position. This random layer together with word embeddings form the input to the neural network model, i.e.,

$$\boldsymbol{Z} = \boldsymbol{R} \odot \boldsymbol{x}, \tag{3.6}$$

where $\odot$ is an element-wise multiplication and each $\boldsymbol{Z}_t = R_{x_t} \cdot \boldsymbol{x}_t$. Intuitively, $\boldsymbol{Z}$ only contains a subset of $\boldsymbol{x}$, which is selected randomly by $\boldsymbol{R}$. Since $\boldsymbol{R}$ is applied directly on the words as a sequence of 0-1 masks, we also call it the word mask layer in this work.

To ensure $\boldsymbol{Z}$ has enough information on predicting $\boldsymbol{Y}$ while contains the least redundant information from $\boldsymbol{x}$, we follow the standard practice in the information bottleneck theory [145], and write the objective function as

$$\max_{\boldsymbol{Z}} I(\boldsymbol{Z};\boldsymbol{Y}) - \beta \cdot I(\boldsymbol{Z};\boldsymbol{X}), \tag{3.7}$$

where $\boldsymbol{X}$ as a random variable representing a generic word sequence as input, $\boldsymbol{Y}$ is the one-hot output random variable, $I(\cdot;\cdot)$ is the mutual information, and $\beta \in \mathbb{R}_+$ is a coefficient to balance the two mutual information items. This formulation reflects our exact expectation on $\boldsymbol{Z}$. The main challenge here is to compute the mutual information.

3.2.2 Variational Word Masks

Inspired by the variational information bottleneck proposed by Alemi et al. [147], instead of computing $p(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z})$, we start from an approximation distribution $q(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z})$. Then, with a few assumptions specified in the following, we construct a tractable lower bound of the objective in Equation 3.7.

For $I(\boldsymbol{Z};\boldsymbol{Y})$ under q, we have $I(\boldsymbol{Z};\boldsymbol{Y}) = \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y},\boldsymbol{z}) \log(q(\boldsymbol{y}|\boldsymbol{z})/q(\boldsymbol{y}))$. By replacing $\log q(\boldsymbol{y}|\boldsymbol{z})$ with the conditional probability derived from the true distribution $\log p(\boldsymbol{y}|\boldsymbol{z})$, we introduce the constraint between $\boldsymbol{Y}$ and $\boldsymbol{Z}$ from the distribution and also obtain a lower bound of $I(\boldsymbol{Z};\boldsymbol{Y})$,

$$\begin{aligned} I(\boldsymbol{Z};\boldsymbol{Y}) &\geq \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y},\boldsymbol{z}) \log p(\boldsymbol{y}|\boldsymbol{z}) + H_q(\boldsymbol{Y}) \\ &= \sum_{\boldsymbol{y},\boldsymbol{z},\boldsymbol{x}} q(\boldsymbol{x},\boldsymbol{y}) q(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{y}|\boldsymbol{z}) + H_q(\boldsymbol{Y}), \end{aligned} \tag{3.8}$$

where $H_q(\cdot)$ is entropy, and the last step uses $q(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) = q(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})$, which is a factorization based on the conditional dependency [2].

Given a specific observation $(\boldsymbol{x}^{(i)},\boldsymbol{y}^{(i)})$, we define the empirical distribution $q(\boldsymbol{X}^{(i)},\boldsymbol{Y}^{(i)})$ as a multiplication of two Delta functions $q(\boldsymbol{X}^{(i)}=\boldsymbol{x}^{(i)},\boldsymbol{Y}^{(i)}=\boldsymbol{y}^{(i)}) = \delta_{\boldsymbol{x}^{(i)}}(\boldsymbol{x}) \cdot \delta_{\boldsymbol{y}^{(i)}}(\boldsymbol{y})$. Then, Equation 3.8 can be further simplified as

$$\begin{aligned} I(\boldsymbol{Z};\boldsymbol{Y}^{(i)}) &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}|\boldsymbol{x}^{(i)}) \log p(\boldsymbol{y}^{(i)}|\boldsymbol{z}) \\ &= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}^{(i)})}[\log p(\boldsymbol{y}^{(i)}|\boldsymbol{z})]. \end{aligned} \tag{3.9}$$

[2] $\boldsymbol{Y} \leftrightarrow \boldsymbol{X} \leftrightarrow \boldsymbol{Z}$: $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are independent given $\boldsymbol{X}$.

Similarly, for $I(\boldsymbol{Z};\boldsymbol{X})$ under q, we have an upper bound of $I(\boldsymbol{Z};\boldsymbol{X})$ by replacing $p(\boldsymbol{Z}|\boldsymbol{X})$ with a predefined prior distribution $p_0(\boldsymbol{Z})$

$$\begin{aligned} I(\boldsymbol{Z};\boldsymbol{X}) &\leq \mathbb{E}_{q(\boldsymbol{x})}[\mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x})\|p_0(\boldsymbol{z})]] \\ &= \mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x}^{(i)})\|p_0(\boldsymbol{z})], \end{aligned} \tag{3.10}$$

where $\mathrm{KL}[\cdot\|\cdot]$ denotes Kullback-Leibler divergence. The simplification in the last step is similar to Equation 3.9 with the empirical distribution $q(\boldsymbol{X}^{(i)})$.

Substituting (3.10) and (3.9) into Equation 3.7 gives us a lower bound $\mathcal{L}$ of the informaiton bottleneck

$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}^{(i)})}[\log p(\boldsymbol{y}^{(i)}|\boldsymbol{z})] - \beta \cdot \mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x}^{(i)})\|p_0(\boldsymbol{z})]. \tag{3.11}$$

The learning objective is to maximize Equation 3.11 with respect to the approximation distribution $q(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}) = q(\boldsymbol{X},\boldsymbol{Y})q(\boldsymbol{Z}|\boldsymbol{X})$. As a classification problem, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are both observed and $q(\boldsymbol{X},\boldsymbol{Y})$ has already been simplified as an empirical distribution, the only one left in the approximation distribution is $q(\boldsymbol{Z}|\boldsymbol{X})$. Similarly to the objective function in variational inference [147, 148], the first term in $\mathcal{L}$ is to make sure the information in $q(\boldsymbol{Z}|\boldsymbol{X})$ for predicting $\boldsymbol{Y}$, while the second term in $\mathcal{L}$ is to regularize $q(\boldsymbol{Z}|\boldsymbol{X})$ with a predefined prior distribution $p_0(\boldsymbol{Z})$.

The last step of obtaining a practical objective function is to notice that, given $\boldsymbol{X}_t^{(i)} = \boldsymbol{x}_t^{(i)}$ every $\boldsymbol{Z}_t$ can be redefined as

$$\boldsymbol{Z}_t = R_{x_t} \cdot \boldsymbol{x}_t^{(i)}, \tag{3.12}$$

where $R_{x_t} \in \{0,1\}$ is a standard Bernoulli distribution. Then, $\boldsymbol{Z}$ can be reparameterized as $\boldsymbol{Z} = \boldsymbol{R} \odot \boldsymbol{x}^{(i)}$ with $\boldsymbol{R} = [R_{x_1},\ldots,R_{x_T}]$. The lower bound $\mathcal{L}$ can be rewritten with the random variable $\boldsymbol{R}$ as

$$\begin{aligned} \mathcal{L} = &\mathbb{E}_{q(\boldsymbol{r}|\boldsymbol{x}^{(i)})}[\log p(\boldsymbol{y}^{(i)}|\boldsymbol{R},\boldsymbol{x}^{(i)})] \\ &- \beta \cdot \mathrm{KL}[q(\boldsymbol{R}|\boldsymbol{x}^{(i)})\|p_0(\boldsymbol{R})]. \end{aligned} \tag{3.13}$$

Note that, although β is inherited from the information bottleneck theory, in practice it will be used as a tunable hyper-parameter to address the notorious posterior collapse issue [149, 150].

Proof of the Two Bounds for the Information Bottleneck

The following derivation is similar to the variational information bottleneck, where the difference is that our starting point is the approximation distribution $q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ instead of the true distribution $p(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$.

The lower bound for $I(\boldsymbol{Z}; \boldsymbol{Y})$.

$$\begin{aligned} I(\boldsymbol{Z}, \boldsymbol{Y}) &= \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log \frac{q(\boldsymbol{y}, \boldsymbol{z})}{q(\boldsymbol{y})q(\boldsymbol{z})} \\ &= \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log \frac{q(\boldsymbol{y}|\boldsymbol{z})}{q(\boldsymbol{y})} \\ &= \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log q(\boldsymbol{y}|\boldsymbol{z}) \\ &\quad + H_q(\boldsymbol{Y}), \end{aligned} \tag{3.14}$$

where $H_q(\cdot)$ represents entropy. Now, if we replace $\log q(\boldsymbol{y}|\boldsymbol{z})$ with the conditional probability derived from the true distribution $\log p(\boldsymbol{y}|\boldsymbol{z})$, we have

$$\begin{aligned} &\sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log q(\boldsymbol{y}|\boldsymbol{z}) \\ &= \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log \frac{q(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{y}|\boldsymbol{z})}{p(\boldsymbol{y}|\boldsymbol{z})} \\ &= \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log p(\boldsymbol{y}|\boldsymbol{z}) + \mathrm{KL}[q(\boldsymbol{y}|\boldsymbol{z}) \| p(\boldsymbol{y}|\boldsymbol{z})] \\ &\geq \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log p(\boldsymbol{y}|\boldsymbol{z}), \end{aligned} \tag{3.15}$$

where $\mathrm{KL}[\cdot\|\cdot]$ denotes Kullback-Leibler divergence. Therefore, we can obtain a lower bound of the mutual information

$$\begin{aligned} I(\boldsymbol{Z}, \boldsymbol{Y}) &\geq \sum_{\boldsymbol{y},\boldsymbol{z}} q(\boldsymbol{y}, \boldsymbol{z}) \log p(\boldsymbol{y}|\boldsymbol{z}) + H_q(\boldsymbol{Y}) \\ &= \sum_{\boldsymbol{y},\boldsymbol{z},\boldsymbol{x}} q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \log p(\boldsymbol{y}|\boldsymbol{z}) + H_q(\boldsymbol{Y}) \\ &= \sum_{\boldsymbol{y},\boldsymbol{z},\boldsymbol{x}} q(\boldsymbol{x}, \boldsymbol{y}) q(\boldsymbol{z}|\boldsymbol{x}) \log p(\boldsymbol{y}|\boldsymbol{z}) + H_q(\boldsymbol{Y}), \end{aligned} \tag{3.16}$$

where the last step uses $q(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = q(\boldsymbol{x})q(\boldsymbol{y}|\boldsymbol{x})q(\boldsymbol{z}|\boldsymbol{x})$, which is a factorization based on the conditional dependency [3].

[3] $\boldsymbol{Y} \leftrightarrow \boldsymbol{X} \leftrightarrow \boldsymbol{Z}$: $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are independent given $\boldsymbol{X}$.

Since $q(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z})$ is the approximation defined by ourselves, given a specific observation $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, the empirical distribution $q(\boldsymbol{X}^{(i)}, \boldsymbol{Y}^{(i)})$ is simply defined as a multiplication of two Delta functions

$$q(\boldsymbol{X}^{(i)} = \boldsymbol{x}^{(i)}, \boldsymbol{Y}^{(i)} = \boldsymbol{y}^{(i)}) = \delta_{\boldsymbol{x}^{(i)}}(\boldsymbol{x}) \cdot \delta_{\boldsymbol{y}^{(i)}}(\boldsymbol{y}). \tag{3.17}$$

Then, Equation 3.16 with $\boldsymbol{X}^{(i)}$ and $\boldsymbol{Y}^{(i)}$ can be further simplified as

$$\begin{aligned} I(\boldsymbol{Z}; \boldsymbol{Y}^{(i)}) &\geq \sum_{\boldsymbol{z}} q(\boldsymbol{z}|\boldsymbol{x}^{(i)}) \log p(\boldsymbol{y}^{(i)}|\boldsymbol{z}) \\ &= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}^{(i)})}[\log p(\boldsymbol{y}^{(i)}|\boldsymbol{z})] \end{aligned} \tag{3.18}$$

The upper bound for $I(\boldsymbol{Z}; \boldsymbol{X})$.

$$\begin{aligned} I(\boldsymbol{Z}, \boldsymbol{X}) &= \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log \frac{q(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{x})q(\boldsymbol{z})} \\ &= \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log \frac{q(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})} \\ &= \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log q(\boldsymbol{z}|\boldsymbol{x}) \\ &\quad - \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log q(\boldsymbol{z}) \end{aligned} \tag{3.19}$$

By replacing $q(\boldsymbol{z})$ with a prior distribution of $\boldsymbol{z}$, $p_0(\boldsymbol{z})$, we have

$$\sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log q(\boldsymbol{z}) \geq \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log p_0(\boldsymbol{z}). \tag{3.20}$$

Then we can obtain an upper bound of the mutual information

$$\begin{aligned} I(\boldsymbol{Z}, \boldsymbol{X}) &\leq \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log q(\boldsymbol{z}|\boldsymbol{x}) \\ &\quad - \sum_{\boldsymbol{x},\boldsymbol{z}} q(\boldsymbol{x}, \boldsymbol{z}) \log p_0(\boldsymbol{z}) \\ &= \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x}) \| p_0(\boldsymbol{z})] \\ &= \mathbb{E}_{q(\boldsymbol{x})}[\mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x}) \| p_0(\boldsymbol{z})]] \\ &= \mathrm{KL}[q(\boldsymbol{z}|\boldsymbol{x}^{(i)}) \| p_0(\boldsymbol{z})]. \end{aligned} \tag{3.21}$$

Connections

The idea of modifying word embeddings with the information bottleneck method has recently shown some interesting applications in NLP. For example, Li and Eisner [151] proposed two ways to transform word embeddings into new representations for better POS tagging and syntactic parsing. According to Equation 3.6, VMASK can be viewed as a simple linear transformation on word embeddings. The difference is that $\{R_{x_t}\}$ is defined on the vocabulary, therefore can be used to represent the global importance of word x_t. Recall that $R_{x_t} \in \{0, 1\}$, from a slightly different perspective, Equation 3.6 can be viewed as a generalized method on word-embedding dropout [152]. Although there are two major differences: (1) in Gal and Ghahramani [152] all words share the same dropout rate, while in VMASK every word has its own dropout rate specified by $q(R_{x_t}|\boldsymbol{x}_t)$, i.e. $1 - \mathbb{E}[q(R_{x_t}|\boldsymbol{x}_t)]$; (2) the motivation of word-embedding dropout is to force a model not to rely on single words for prediction, while VMASK is to learn a task-specific importance for every word.

Another implementation for making word masks sparse is by adding L_0 regularization [128, 153, 154], while in the objective Equation 3.13, we regularize masks with a predefined prior distribution $p_0(\boldsymbol{R})$ as described in Section 3.2.3.

3.2.3 Model Specification and Training

We resort to mean-field approximation [155] to simplify the assumption on our q distribution. For $q_{\boldsymbol{\phi}}(\boldsymbol{R}|\boldsymbol{x})$, we have $q_{\boldsymbol{\phi}}(\boldsymbol{R}|\boldsymbol{x}) = \prod_{t=1}^{T} q_{\phi}(R_{x_t}|\boldsymbol{x}_t)$, which means the random variables are mutually independent and each governed by $\boldsymbol{x}_t$. We use the amortized variational inference [148] to represent the posterior distribution $q_{\boldsymbol{\phi}}(R_{x_t}|\boldsymbol{x}_t)$ with using an inference network [156]. In this work, we adopt a single-layer feedforward neural network as the inference network, whose parameters $\boldsymbol{\phi}$ are optimized with the model parameters $\boldsymbol{\theta}$ during training.

Following the same factorization as in $q_{\boldsymbol{\phi}}(\boldsymbol{R}|\boldsymbol{x})$, we define the prior distribution $p_0(\boldsymbol{R})$ as $p_0(\boldsymbol{R}) = \prod_{t=1}^{T} p_0(R_{x_t})$ and each of them as $p_0(R_{x_t}) = \text{Bernoulli}(0.5)$. By choosing this non-informative prior, it means every word is initialized with no preference to be important or unimportant, and thus has the equal probability to be masked or selected. As $p_0(\boldsymbol{R})$ is a uniform distribution, we can further simplify the second term in Equation 3.13 as a conditional entropy,

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_q[\log p(\boldsymbol{y}^{(i)}|\boldsymbol{R}, \boldsymbol{x}^{(i)})] + \beta \cdot H_q(\boldsymbol{R}|\boldsymbol{x}^{(i)}). \tag{3.22}$$

Datasets	C	L	*#train*	*#dev*	*#test*
IMDB	2	268	20K	5K	25K
SST-1	5	18	8544	1101	2210
SST-2	2	19	6920	872	1821
Yelp	2	138	500K	60K	38K
AG News	4	32	114K	6K	7.6K
TREC	6	10	5000	452	500
Subj	2	23	8000	1000	1000

Table 3.7: Summary statistics for the datasets, where C is the number of classes, L is average sentence length, and # counts the number of examples in the *train/dev/test* sets.

We apply stochastic gradient descent to solve the optimization problem (Equation 3.22). Particularly in each iteration, the first term in Equation 3.22 is approximated with a single sample from $q(\boldsymbol{R}|\boldsymbol{x}^{(i)})$ [156]. However, sampling from a Bernoulli distribution (like from any other discrete distributions) causes difficulty in backpropagation. We adopt the Gumbel-softmax trick [157, 158] to utilize a continuous differentiable approximation and tackle the discreteness of sampling from Bernoulli distributions. During training, We use Adam [140] for optimization and KL cost annealing [149] to avoid posterior collapse.

For a given word x_t and its word embedding $\boldsymbol{x}_t$, in training stage, the model samples each r_{x_t} from $q(R_{x_t}|\boldsymbol{x}_t)$ to decide to either keep or zero out the corresponding word embedding $\boldsymbol{x}_t$. In inference stage, the model takes the multiplication of the word embedding $\boldsymbol{x}_t$ and the expectation of the word mask distribution, i.e. $\boldsymbol{x}_t \cdot \mathbb{E}[q(R_{x_t}|\boldsymbol{x}_t)]$, as input.

3.2.4 Experiment Setup

The proposed method is evaluated on seven text classification tasks, ranging from sentiment analysis to topic classification, with three typical neural network models, a long short-term memories [9, LSTM], a convolutional neural network [11, CNN], and BERT [13].

Datasets. We adopt seven benchmark datasets: movie reviews IMDB [138], Stanford Sentiment Treebank with fine-grained labels SST-1 and its binary version SST-2 [136], Yelp reviews [12], AG's News [12], 6-class question classification TREC [159], and subjective/objective classification Subj [137]. For the datasets (e.g. IMDB, Subj) without standard train/dev/test split, we hold out a proportion of training examples as the development set. Table 3.7 shows the statistics of the datasets.

Models. The CNN model [11] contains a single convolutional layer with filter sizes ranging from 3 to 5. The LSTM [9] has a single unidirectional hidden layer. Both models are initialized with 300-dimensional pretrained word embeddings [139]. We fix the embedding layer and update other parameters on different datasets to achieve the best performance respectively. We use the pretrained BERT-base model with 12 transformer layers, 12 self-attention heads, and the hidden size of 768. We fine-tune it with different downstream tasks, and then fix the embedding layer and train the mask layer with the rest of the model together.

Baselines and Competitive Methods. As the goal of this work is to propose a novel training method that improves both prediction accuracy and interpretability, we employ two groups of models as baselines and competitive systems. Models trained with the proposed method are named with suffix "-VMASK". We also provide two baselines: (1) models trained by minimizing the cross-entropy loss (postfixed with "-base") and (2) models trained with ℓ_2-regularization (postfixed with "-ℓ_2"). The comparison with these two baseline methods mainly focuses on prediction performance as no explicit training strategies are used to improve interpretability.

Besides, we also propose two competitive methods: models trained with the explanation framework "Learning to Explain" [65] (postfixed with "-L2X") and the "Information Bottleneck Attribution" [29] (postfixed with "-IBA"). L2X and IBA were originally proposed to find feature attributions as post-hoc explanations for well-trained models. We integrated them in model training, working as the mask layer to directly generate mask values for input features (L2X) or restrict information flow by adding noise (IBA). In our experiments, all training methods worked with random dropout ($\rho = 0.2$) to avoid overfitting.

- The explanation framework of L2X [65] is a neural network which learns to generate importance scores $\boldsymbol{w} = [w_1, w_2, \cdots, w_T]$ for input features $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$. The neural network is optimized by maximizing the mutual information between the selected important features and the model prediction, i.e. $I(\boldsymbol{x}_S; y)$, where $\boldsymbol{x}_S$ contains a subset of features from $\boldsymbol{x}$. In our experiments, we adopt a single-layer feedforward neural network as the interpreter to generate importance scores for an input text, and multiply each word embedding with its importance score, $\boldsymbol{x}' = \boldsymbol{w} \odot \boldsymbol{x}$. The weighted word embedding matrix $\boldsymbol{x}'$ is sent to the rest of the model to produce an output y'. We optimize the interpreter network with the original model by minimizing the cross-entropy loss between the final output and the ground-truth label, $\mathcal{L}_{ce}(y_t; y')$.

- We adopt the Readout Bottleneck of IBA which utilizes a neural network to predict mask values $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_T]$, where $\lambda_t \in [0, 1]$. The information of a feature $\boldsymbol{x}_t$ is restricted by adding noise, i.e. $\boldsymbol{z}_t = \lambda_t \boldsymbol{x}_t + (1 - \lambda_t)\boldsymbol{\epsilon}_t$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mu_{\boldsymbol{x}_t}, \sigma^2_{\boldsymbol{x}_t})$. And $\boldsymbol{z}$ is learned by optimizing the objective function Equation 3.7. By assuming the variational approximation $q(\boldsymbol{z})$ as a Gaussian distribution, the mutual information can be calculated explicitly [29] . We still use a single-layer feedforward neural network as the Readout Bottleneck to generate continuous mask valuses $\boldsymbol{\lambda}$ and construct $\boldsymbol{z}$ for model to make predictions. The Readout Bottleneck is trained jointly with the original model by minimizing the sum of the cross-entropy loss $\mathcal{L}_{ce}(y_t; y)$ and an upper bound $\mathcal{L}_I = \mathbb{E}_{\boldsymbol{x}}[\mathrm{KL}[p(\boldsymbol{z}|\boldsymbol{x})\|q(\boldsymbol{z})]]$ of the mutual information $I(\boldsymbol{Z}; \boldsymbol{X})$. See Schulz et al. [29] for the proof of the upper bound.

Models	Methods	IMDB	SST-1	SST-2	Yelp	AG News	TREC	Subj
CNN	CNN-base	89.06	46.32	85.50	94.32	91.30	92.40	92.80
	CNN-ℓ_2	89.12	46.01	85.56	94.46	91.28	90.62	92.39
	CNN-L2X	78.94	37.92	80.01	83.14	84.36	61.00	82.40
	CNN-IBA	88.31	41.40	84.24	93.82	91.37	89.80	91.80
	CNN-VMASK	**90.10**	**48.92**	**85.78**	**94.53**	**91.60**	**93.02**	**93.50**
LSTM	LSTM-base	88.39	43.84	83.74	95.06	91.03	90.40	90.20
	LSTM-ℓ_2	88.40	43.91	83.36	95.00	91.09	90.20	89.10
	LSTM-L2X	67.45	36.92	75.45	77.12	77.53	46.00	81.80
	LSTM-IBA	88.48	42.99	83.53	94.74	91.14	85.40	89.50
	LSTM-VMASK	**90.07**	**44.12**	**84.35**	**95.41**	**92.19**	**90.80**	**91.20**
BERT	BERT-base	91.80	53.43	92.25	96.42	93.59	96.40	95.10
	BERT-ℓ_2	91.75	52.08	92.25	96.41	93.52	96.80	94.80
	BERT-L2X	71.75	39.23	74.03	87.14	82.59	93.20	86.10
	BERT-IBA	91.66	53.80	92.24	96.27	93.45	96.80	95.60
	BERT-VMASK	**93.04**	**54.53**	**92.26**	**96.80**	**94.24**	**97.00**	**96.40**

Table 3.8: Prediction accuracy (%) of different models with different training strategies on the seven datasets.

3.2.5 Results and Discussion

We trained the three models on the seven datasets with different training strategies. Table 3.8 shows the prediction accuracy of different models on test sets. The models trained with VMASK outperform the ones with similar network architectures but trained differently. The results show that VMASK can help improve the generalization power.

Except the base models and the models trained with the proposed method, the records of other three competitors are mixed. For example, the traditional ℓ_2-regularization cannot always help improve accuracy, especially for the BERT model. Although the performance with IBA is slightly better than with L2X, training with them does not show a constant improvement on a model's prediction accuracy.

To echo the purpose of improving model interpretability, the rest of this section will focus on evaluating the model interpretability quantitatively and qualitatively.

Quantitative Evaluation

We evaluate the local interpretability of VMASK-based models against the base models via the AOPC score [129, 73] and the global interpretability against the IBA-based models via post-hoc accuracy [65]. Empirically, we observed the agreement between local and global interpretability, so there is no need to exhaust all possible combinations in our evaluation.

Methods	Models	IMDB	SST-1	SST-2	Yelp	AG News	TREC	Subj
LIME	CNN-base	14.47	7.59	16.50	10.69	5.66	15.28	9.77
	CNN-VMASK	**14.74**	**8.63**	**18.86**	**11.38**	**9.03**	14.81	**12.40**
	LSTM-base	14.34	8.76	17.03	8.72	7.00	11.95	9.67
	LSTM-VMASK	**15.10**	**9.52**	**22.14**	**9.70**	**7.39**	11.97	**11.68**
	BERT-base	10.63	36.00	35.89	6.30	7.00	59.22	13.08
	BERT-VMASK	**12.64**	36.16	**46.87**	6.49	**8.47**	**60.37**	**17.82**
SampleShapley	CNN-base	15.53	7.63	13.15	13.57	9.88	14.97	8.84
	CNN-VMASK	15.53	**8.33**	**15.95**	**15.06**	9.98	**15.03**	**12.88**
	LSTM-base	15.80	7.91	22.38	10.55	6.62	11.90	11.66
	LSTM-VMASK	**16.48**	**9.73**	22.52	**10.99**	**7.65**	11.86	**12.74**
	BERT-base	12.97	42.06	43.16	18.06	7.21	57.69	33.22
	BERT-VMASK	**13.18**	**44.57**	**50.44**	18.17	**10.02**	**58.26**	**34.22**

Table 3.9: AOPCs (%) of LIME and SampleShapley in interpreting the base and VMASK-based models on the seven datasets.

Local interpretability: AOPC We adopt two model-agnostic explanation methods, LIME [1] and SampleShapley [2], to generate local explanations for base and VMASK-based models, where "local" means explaining each test data individually. The area over the perturbation curve (AOPC) [129, 73] metric is utilized to evaluate the faithfulness of explanations to models. It calculates the average change of prediction probability on the predicted class over all test data by deleting top n words in explanations. We adopt this metric to evaluate the model interpretability to post-hoc explanations. Higher AOPC scores are better.

For TREC and Subj datasets, we evaluate all test data. For each other dataset, we randomly pick up 1000 examples for evaluation due to computation costs. Table 3.9 shows the AOPCs of different models on the seven datasets by deleting top 5 words identified by LIME or SampleShapley. The

AOPCs of VMASK-based models are significantly higher than that of base models on most of the datasets, indicating that VMASK can improve model's interpretability to post-hoc explanations. The results on the TREC dataset are very close because top 5 words are possible to include all informative words for short sentences with the average length of 10.

Global Interpretability: Post-hoc accuracy The expectation values $\{\mathbb{E}[q(R_{x_t}|\boldsymbol{x}_t)]\}$ represent the global importance of words for a specific task. To measure the interpretability of a model itself (aka, global interpretability), we adopt the post-hoc accuracy [65] to evaluate the influence of global task-specific important features on the predictions of VMASK- and IBA-based models. For each test data, we select the top k words based on their global importance scores for the model to make a prediction, and compare it with the original prediction made on the whole input text

$$\text{post-hoc-acc}(k) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}[y_m(k) = y_m],$$

where M is the number of examples, y_m is the predicted label on the m-th test data, and $y_m(k)$ is the predicted label based on the top k important words.

Figure 3.1 shows the results of VMASK- and IBA-based models on the seven datasets with k ranging from 1 to 10. VMASK-based models (solid lines) outperform IBA-based models (dotted lines) with higher post-hoc accuracy, which indicates our proposed method is better on capturing task-specific important features. For CNN-VMASK and LSTM-VMASK, using only top two words can achieve about 80% post-hoc accuracy, even for the IMDB dataset, which has the average sentence length of 268 tokens. The results illustrate that VMASK can identify informative words for model predictions. We also noticed that BERT-VMASK has lower post-hoc accuracy than the other two models. It is probably because BERT tends to use larger context with its self-attentions for predictions. This also explains that the post-hoc accuracies of BERT-VMASK on the IMDB and SST-1 datasets are catching up slowly with k increasing.

Qualitative Evaluation

Visualizing post-hoc local explanations. Table 3.10 shows some examples of LIME explanations for different models on the IMDB dataset. We highlight the top three important words identified by LIME, where the color saturation indicates word attribution. The pair of base and VMASK-based models make the same and correct predictions on the input texts. For VMASK-based models, LIME can capture the sentiment words that indicate the same sentiment polarity as the

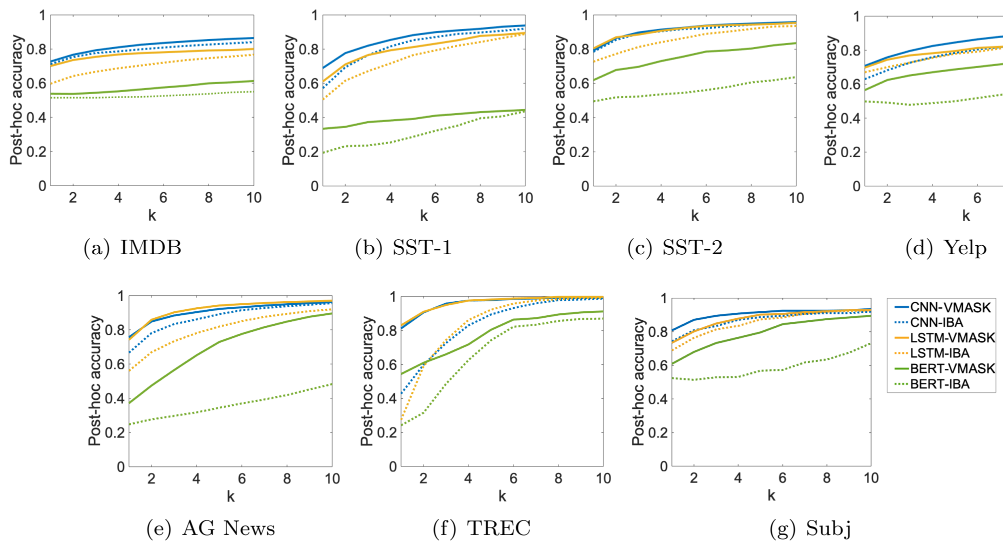


Figure 3.1: Post-hoc accuracy of VMASK- and IBA-based models on the seven datasets.

Models	Texts	Prediction
CNN-base	Primary plot , primary direction , poor interpretation .	negative
CNN-VMASK	Primary plot , primary direction , poor interpretation .	negative
LSTM-base	John Leguizamo 's freak is one of the funniest one man shows I 've ever seen . I recommend it to anyone with a good sense of humor .	positive
LSTM-VMASK	John Leguizamo 's freak is one of the funniest one man shows I 've ever seen . I recommend it to anyone with a good sense of humor .	positive
BERT-base	Great story , great music . A heartwarming love story that ' s beautiful to watch and delightful to listen to . Too bad there is no soundtrack CD .	positive
BERT-VMASK	Great story , great music . A heartwarming love story that ' s beautiful to watch and delightful to listen to . Too bad there is no soundtrack CD .	positive

Table 3.10: Examples of the explanations generated by LIME for different models on the IMDB dataset, where the top three important words are highlighted. The color saturation indicates word attribution.

prediction. While for base models, LIME selects some irrelevant words (e.g. "plot", "of", "to") as explanations, which illustrates the relatively lower interpretability of base models to post-hoc explanations.

Visualizing post-hoc global explanations. We adopt SP-LIME proposed by Ribeiro et al. [1] as a third-party global interpretability of base and VMASK-based models. Without considering the rectriction on the number of explanations, we follow the method to compute feature global importance from LIME local explanations by calculating the sum over all local importance scores of a feature as its global importance. To distinguish it from the global importance learned by VMASK, we call it *post-hoc global importance*.

Table 3.11 lists the top three post-hoc global important words of base and VMASK-based models on the IMDB dataset. For VMASK-based models, the global important features selected by SP-LIME are all sentiment words. While for base models, some irrelevant words (e.g. "performances", "plot", "butcher") are identified as important features, which makes model predictions unreliable.

Models	Words
CNN-base	excellent, performances, brilliant
CNN-VMASK	excellent, fine, favorite
LSTM-base	plot, excellent, liked
LSTM-VMASK	excellent, favorite, brilliant
BERT-base	live, butcher, thrilling
BERT-VMASK	powerful, thrilling, outstanding

Table 3.11: Post-hoc global important words selected by SP-LIME for different models on the IMDB dataset.

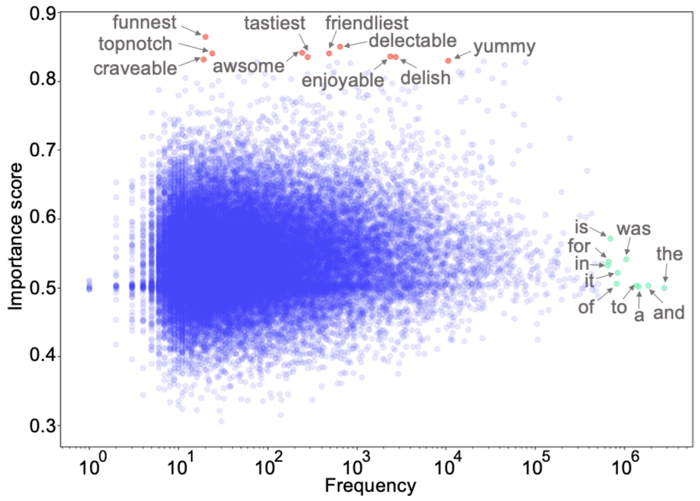
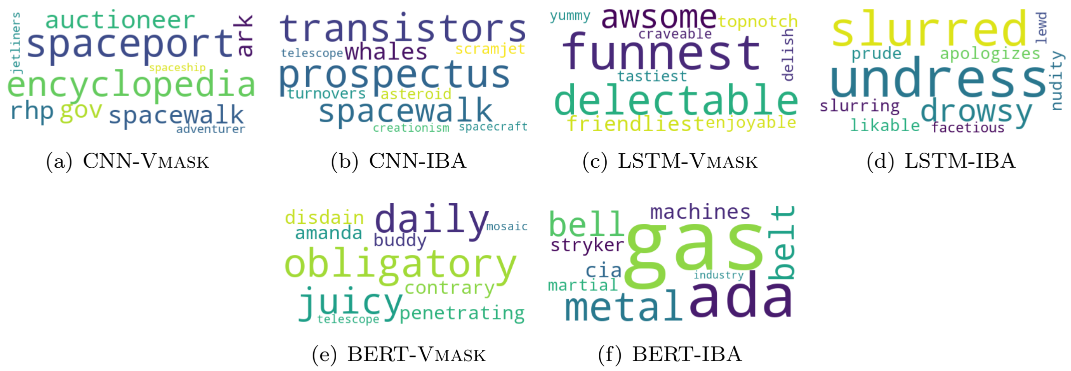


Figure 3.2: Scatter plot of word global importance and frequency (in log scale) of LSTM-VMASK on the Yelp dataset, where red dots represent top 10 important sentiment words and green dots represent top 10 high-frequency words.

Frequency-importance correlation. We compute the Pearson correlation coefficients between word frequency and global word importance of VMASK-based models. The results show that they are not significantly correlated, which indicates that VMASK is not simply learning to select high-frequency words. Figure 3.2 further verifies this by ploting the expectation ($\mathbb{E}[q(R_{x_t}|\boldsymbol{x}_t)]$) of word masks from the LSTM-Vmask trained on Yelp and the word frequency from the same dataset. Here, we visualize the top 10 high-frequency words and top 10 important words based the expectation of word masks. The global importance scores of the sentiment words are over 0.8, even for some low-frequency words (e.g. “funnest”, “craveable”), while that of the high-frequency words are all around 0.5, which means the Vmask-based models are less likely to focus on the irrelevant words to make predictions.



(a) CNN-Vmask (b) CNN-IBA (c) LSTM-Vmask (d) LSTM-IBA (e) BERT-Vmask (f) BERT-IBA

Figure 3.3: Word clouds of top 10 important words, where (a) is CNN-Vmask on the AG News dataset, (b) is CNN-IBA on the AG News dataset, (c) is LSTM-Vmask on the Yelp dataset, (d) is LSTM-IBA on the Yelp dataset, (e) is BERT-Vmask on the Subj dataset, and (f) is BERT-IBA on the Subj dataset.

Task-specific important words. Figure 3.3 visualizes top 10 important words for the Vmask- and IBA-based models on three datasets via word clouds. We can see that the selected words by Vmask are consistent with the corresponding topic, such as “funnest”, “awsome” for sentiment analysis, and “encyclopedia”, “spaceport” for news classification, while IBA selects some irrelevant words (e.g. “undress”, “slurred”).

3.2.6 Conclusion

In this work, we proposed an effective method, Vmask, learning global task-specific important features to improve both model interpretability and prediction accuracy. We tested Vmask with three different neural text classifiers on seven benchmark datasets, and assessed its effectiveness via both quantitative and qualitative evaluations.

Chapter 4

Explaining Neural Language Models

Explaining neural networks is important for helping users understand model decision-making and gaining their trust on model predictions. In this chapter, I propose a hierarchical explanation method (HEDGE) by detecting feature interactions for text classification in Section 4.1 and introduce the Group Masks (GMASK) method for explaining neural network predictions on sentence-pair modeling tasks in Section 4.2. I also propose the uncertainty explanation to explain model predictive uncertainty beyond predicted label in Section 4.3.

4.1 Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection

In NLP, most of existing work on local explanation generation focuses on producing word-level or phrase-level explanations by quantifying contributions of individual words or phrases to a model prediction [1, 24, 51]. Figure 4.1 (a) and (b) present a word-level and a phrase-level explanation generated by the LIME [1] and the Contextual Decomposition (CD) [51] respectively for explaining sentiment classification. Both explanations provide scores to quantify how a word or a phrase contributes to the final prediction. For example, the explanation generated by LIME captures a keyword `waste` and the explanation from CD identifies an important phrase `waste of`. However, neither of them is able to explain the model decision-making in terms of how words and phrases are interacted with each other and composed together for the final prediction. In this example, since

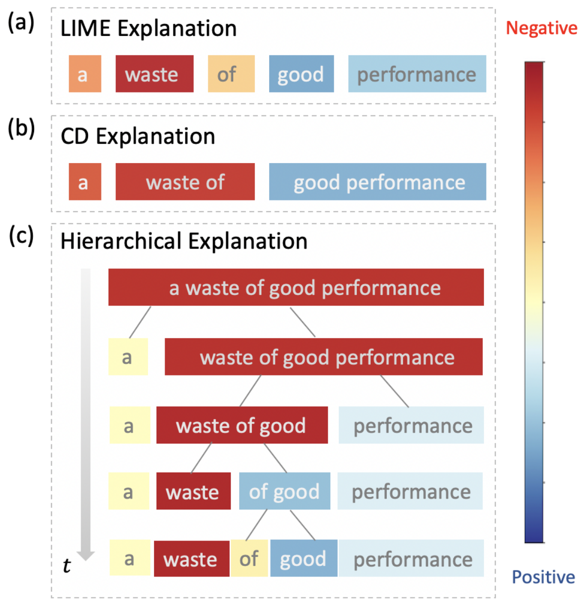


Figure 4.1: Different explanations for a NEGATIVE movie review `a waste of good performance`, where the color of each block represents the contribution of the corresponding word/phrase/clause (feature) to the model prediction. From the hierarchical explanation, we obtain a set of features in each timestep (t), where the most important one is `waste of good`.

the final prediction is NEGATIVE, one question that we could ask is that how the word `good` or a phrase related to the word `good` contributes to the model prediction. An explanation being able to answer this question will give users a better understanding on the model decision-making and also more confidence to trust the prediction.

The goal of this work is to reveal prediction behaviors of a text classifier by detecting feature (e.g., words or phrases) interactions with respect to model predictions. For a given text, we propose a model-agnostic approach, called HEDGE (for Hierarchical Explanation via Divisive Generation), to build hierarchical explanations by recursively detecting the weakest interactions and then dividing large text spans into smaller ones based on the interactions. As shown in Figure 4.1 (c), the hierarchical structure produced by HEDGE provides a comprehensive picture of how different granularity of features interacting with each other within the model. For example, it shows how the word `good` is dominated by others in the model prediction, which eventually leads to the correct prediction. Furthermore, the scores of text spans across the whole hierarchy also help identify the most important feature `waste of good`, which can be served as a phrase-level explanation for the model prediction.

4.1.1 Generating Hierarchical Explanations

For a classification task, let $\boldsymbol{x} = (x_1, \ldots, x_n)$ denotes a text with n words and $\hat{y}$ be the prediction label from a well-trained model. Furthermore, we define $\mathcal{P} = \{\boldsymbol{x}_{(0,s_1]}, \boldsymbol{x}_{(s_1,s_2]}, \ldots, \boldsymbol{x}_{(s_{P-1},n]}\}$ be a partition of the word sequence with P text spans, where $\boldsymbol{x}_{(s_i,s_{i+1}]} = (x_{s_i+1}, \ldots, x_{s_{i+1}})$. For a given text span $\boldsymbol{x}_{(s_i,s_{i+1}]}$, the basic procedure of HEDGE is to divide it into two smaller text spans $\boldsymbol{x}_{(s_i,j]}$ and $\boldsymbol{x}_{(j,s_{i+1}]}$, where j is the dividing point ($s_i < j < s_{i+1}$), and then evaluate their contributions to the model prediction $\hat{y}$.

Algorithm 1 describes the whole procedure of dividing $\boldsymbol{x}$ into different levels of text spans and evaluating the contribution of each of them. Starting from the whole text $\boldsymbol{x}$, the algorithm first divides $\boldsymbol{x}$ into two segments. In the next iteration, it will pick one of the two segments and further split it into even smaller spans. As shown in Algorithm 1, to perform the top-down procedure, we need to answer the questions: for the next timestep, which text span the algorithm should pick to split and where is the dividing point?

Algorithm 1 Hierarchical Explanation via Divisive Generation

```
Input: text $\boldsymbol{x}$ with length $n$, and predicted label $\hat{y}$
Initialize the original partition $\mathcal{P}_0 \leftarrow \{\boldsymbol{x}_{(0,n]}\}$
Initialize the contribution set $\mathcal{C}_0 = \emptyset$
Initialize the hierarchy $\mathcal{H} = [\mathcal{P}_0]$
for $t = 1, \ldots, n-1$ do
  Find $\boldsymbol{x}_{(s_i,s_{i+1}]}$ and $j$ by solving Equation 4.1
  Update the partition
  $\mathcal{P}'_t \leftarrow \mathcal{P}_{t-1} \backslash \{\boldsymbol{x}_{(s_i,s_{i+1}]}\}$
  $\mathcal{P}_t \leftarrow \mathcal{P}'_t \cup \{\boldsymbol{x}_{(s_i,j]}, \boldsymbol{x}_{(j,s_{i+1}]}\}$
  $\mathcal{H}.add(\mathcal{P}_t)$
  Update the contribution set $\mathcal{C}$ with
  $\mathcal{C}'_t \leftarrow \mathcal{C}_{t-1} \cup \{(\boldsymbol{x}_{(s_i,j]}, \psi(\boldsymbol{x}_{(s_i,j]}))\}$
  $\mathcal{C}_t \leftarrow \mathcal{C}'_t \cup \{(\boldsymbol{x}_{(j,s_{i+1}]}, \psi(\boldsymbol{x}_{(j,s_{i+1}]}))\}$
end for
Output: $\mathcal{C}_{n-1}$, $\mathcal{H}$
```

Both questions can be addressed via the following optimization problem:

$$\min_{\boldsymbol{x}_{(s_i,s_{i+1}]} \in \mathcal{P}} \min_{j \in (s_i,s_{i+1})} \phi(\boldsymbol{x}_{(s_i,j]}, \boldsymbol{x}_{(j,s_{i+1}]} \mid \mathcal{P}), \tag{4.1}$$

where $\phi(\boldsymbol{x}_{(s_i,j]}, \boldsymbol{x}_{(j,s_{i+1}]} \mid \mathcal{P})$ defines the interaction score between $\boldsymbol{x}_{(s_i,j]}$ and $\boldsymbol{x}_{(j,s_{i+1}]}$ given the current partition $\mathcal{P}$. The detail of this score function will be explained in Section 4.1.2.

For a given $\boldsymbol{x}_{(s_i,s_{i+1}]} \in \mathcal{P}$, the inner optimization problem will find the *weakest* interaction point to split the text span $\boldsymbol{x}_{(s_i,s_{i+1}]}$ into two smaller ones. It answers the question about where the dividing point should be for a given text span. A trivial case of the inner optimization problem is on a text span with length 2, since there is only one possible way to divide it. The outer optimization answers the question about which text span should be picked. This optimization problem can be solved by simply enumerating all the elements in a partition $\mathcal{P}$. A special case of the outer optimization problem is at the first iteration $t = 1$, where $\mathcal{P}_0 = \{\boldsymbol{x}_{(0,n]}\}$ only has one element, which is the whole input text. Once the partition is updated, it is then added to the hierarchy $\mathcal{H}$.

The last step in each iteration is to evaluate the contributions of the new spans and update the contribution set $\mathcal{C}$ as in line 9 of the Algorithm 1. For each, the algorithm evaluates its contribution to the model prediction with the feature importance function $\psi(\cdot)$ defined in Equation 4.5. The final output of Algorithm 1 includes the contribution set $\mathcal{C}_{n-1}$ which contains all the produced text spans in each timestep together with their importance scores, and the hierarchy $\mathcal{H}$ which contains all the partitions of $\boldsymbol{x}$ along timesteps. A hierarchical explanation can be built based on $\mathcal{C}_{n-1}$ and $\mathcal{H}$ by visualizing the partitions with all text spans and their importance scores along timesteps, as Figure 4.1 (c) shows.

4.1.2 Detecting Feature Interaction

For a given text span $\boldsymbol{x}_{(s_i,s_{i+1}]} \in \mathcal{P}$ and the dividing point j, the new partition will be $\mathcal{N} = \mathcal{P} \backslash \{\boldsymbol{x}_{(s_i,s_{i+1}]}\} \cup \{\boldsymbol{x}_{(s_i,j]}, \boldsymbol{x}_{(j,s_{i+1}]}\} = \{\boldsymbol{x}_{(0,s_1]}, \ldots, \boldsymbol{x}_{(s_i,j]}, \boldsymbol{x}_{(j,s_{i+1}]}, \ldots, \boldsymbol{x}_{(s_{P-1},n]}\}$. We consider the effects of other text spans in $\mathcal{N}$ when calculate the interaction between $\boldsymbol{x}_{(s_i,j]}$ and $\boldsymbol{x}_{(j,s_{i+1}]}$, since the interaction between two words/phrases is closely dependent on the context [160, 161]. We adopt the Shapley interaction index from coalition game theory [162, 163, 164] to calculate the interaction. For simplicity, we denote $\boldsymbol{x}_{(s_i,j]}$ and $\boldsymbol{x}_{(j,s_{i+1}]}$ as j_1 and j_2 respectively. The interaction score is defined as [68],

$$\phi(j_1, j_2 \mid \mathcal{P}) = \sum_{S \subseteq \mathcal{N} \backslash \{j_1, j_2\}} \frac{|S|!(P - |S| - 1)!}{P!} \gamma(j_1, j_2, S), \tag{4.2}$$

where S represents a subset of text spans in $\mathcal{N} \backslash \{j_1, j_2\}$, $|S|$ is the size of S, and $\gamma(j_1, j_2, S)$ is defined as follows,

$$\gamma(j_1, j_2, S) = \mathbb{E}[f(\boldsymbol{x}') \mid S \cup \{j_1, j_2\}] - \mathbb{E}[f(\boldsymbol{x}') \mid S \cup \{j_1\}] - \mathbb{E}[f(\boldsymbol{x}') \mid S \cup \{j_2\}] + \mathbb{E}[f(\boldsymbol{x}') \mid S], \tag{4.3}$$

where $\boldsymbol{x}'$ is the same as $\boldsymbol{x}$ except some missing words that are not covered by the given subset (i.g. S), $f(\cdot)$ denotes the model output probability on the predicted label $\hat{y}$, and $\mathbb{E}[f(\boldsymbol{x}') \mid S]$ is the expectation of $f(\boldsymbol{x}')$ over all possible $\boldsymbol{x}'$ given S. In practice, the missing words are usually replaced with a special token `<pad>`, and $f(\boldsymbol{x}')$ is calculated to estimate $\mathbb{E}[f(\boldsymbol{x}')|S]$ [49, 165, 24]. We also adopt this method in our experiments. Another way to estimate the expectation is to replace the missing words with substitute words randomly drawn from the full dataset, and calculate the empirical mean of all the sampling data [2, 166], which has a relatively high computational complexity.

With the number of text spans (features) increasing, the exponential number of model evaluations in Equation 4.2 becomes intractable. We calculate an approximation of the interaction score based on the assumption [49, 3, 52]: a word or phrase usually has strong interactions with its neighbours in a sentence. The computational complexity can be reduced to polynomial by only considering m neighbour text spans of j_1 and j_2 in $\mathcal{N}$. The interaction score is rewritten as

$$\phi(j_1, j_2 \mid \mathcal{P}) = \sum_{S \subseteq \mathcal{N}_m \setminus \{j_1, j_2\}} \frac{|S|!(M - |S| - 2)!}{(M-1)!} \gamma(j_1, j_2, S), \tag{4.4}$$

where $\mathcal{N}_m$ is the set containing j_1, j_2 and their neighbours, and $M = |\mathcal{N}_m|$. In experiments, we set $m = 2$, which performs well. The performance can be further improved by increasing m, but at the cost of increased computational complexity.

4.1.3 Quantifying Feature Importance

To measure the contribution of a feature $\boldsymbol{x}_{(s_i, s_{i+1}]}$ to the model prediction, we define the importance score as

$$\psi(\boldsymbol{x}_{(s_i, s_{i+1}]}) = f_{\hat{y}}(\boldsymbol{x}_{(s_i, s_{i+1}]}) - \max_{y' \neq \hat{y}, y' \in \mathcal{Y}} f_{y'}(\boldsymbol{x}_{(s_i, s_{i+1}]}), \tag{4.5}$$

where $f_{\hat{y}}(\boldsymbol{x}_{(s_i, s_{i+1}]})$ is the model output on the predicted label $\hat{y}$; $\max_{y' \neq \hat{y}, y' \in \mathcal{Y}} f_{y'}(\boldsymbol{x}_{(s_i, s_{i+1}]})$ is the highest model output among all classes excluding $\hat{y}$. This importance score measures how far the prediction on a given feature is to the prediction boundary, hence the confidence of classifying $\boldsymbol{x}_{(s_i, s_{i+1}]}$ into the predicted label $\hat{y}$. Particularly in text classification, it can be interpreted as the contribution to a specific class $\hat{y}$.

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

Table 4.1: The classification accuracy of different models on the SST and IMDB datasets.

4.1.4 Experiments

The proposed method is evaluated on text classification tasks with three typical neural network models, a long short-term memories [9, LSTM], a convolutional neural network [11, CNN], and BERT [13], on the SST [136] and IMDB [138] datasets, via both automatic and human evaluations.

Setup

We adopt the SST-2 [136] which has 6920/872/1821 examples in the *train/dev/test* sets with binary labels. The IMDB [138] also has binary labels with 25000/25000 examples in the *train/test* sets. We hold out 10% of the training examples as the development set.

The proposed method is evaluated on text classification tasks with three typical neural network models, a long short-term memories [9, LSTM], a convolutional neural network [11, CNN], and BERT [13]. The CNN model [11] includes a single convolutional layer with filter sizes ranging from 3 to 5. The LSTM [9] has a single layer with 300 hidden states. Both models are initialized with 300-dimensional pretrained word embeddings [139]. We use the pretrained BERT model[1] with 12 transformer layers, 12 self-attention heads, and the hidden size of 768, which was then fine-tuned with different downstream tasks to achieve the best performance. Table 4.1 shows the best performance of the models on both datasets in our experiments, where BERT outperforms CNN and LSTM with higher classification accuracy.

Quantitative Evaluation

We adopt two metrics from prior work on evaluating word-level explanations: the area over the perturbation curve (AOPC) [129, 73] and the log-odds scores [167, 49], and define a new evaluation metric called *cohesion-score* to evaluate the interactions between words within a given text span. The first two metrics measure local fidelity by deleting or masking top-scored words and comparing the probability change on the predicted label. They are used to evaluate Equation 4.5 in quantifying

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

Datasets	Methods	LSTM		CNN		BERT	
		AOPC	Log-odds	AOPC	Log-odds	AOPC	Log-odds
SST	Leave-one-out	0.441	-0.443	0.434	-0.448	0.464	-0.723
	CD	0.384	-0.382	-	-	-	-
	LIME	0.444	-0.449	0.473	-0.542	0.134	-0.186
	L-Shapley	0.431	-0.436	0.425	-0.459	0.435	-0.809
	C-Shapley	0.423	-0.425	0.415	-0.446	0.410	-0.754
	KernelSHAP	0.360	-0.361	0.387	-0.423	0.411	-0.765
	SampleShapley	0.450	-0.454	0.487	-0.550	0.462	-0.836
	HEDGE	**0.458**	**-0.466**	**0.494**	**-0.567**	**0.479**	**-0.862**
IMDB	Leave-one-out	0.630	-1.409	0.598	-0.806	0.335	-0.849
	CD	0.495	-1.190	-	-	-	-
	LIME	0.764	-1.810	0.691	-1.091	0.060	-0.133
	L-Shapley	0.637	-1.463	0.623	-0.950	0.347	-1.024
	C-Shapley	0.629	-1.427	0.613	-0.928	0.331	-0.973
	KernelSHAP	0.542	-1.261	0.464	-0.727	0.223	-0.917
	SampleShapley	0.757	-1.597	0.707	-1.108	0.355	-1.037
	HEDGE	**0.783**	**-1.873**	**0.719**	**-1.144**	**0.411**	**-1.126**

Table 4.2: AOPCs and log-odds scores of different interpretation methods in explaining different models on the SST and IMDB datasets.

feature contributions to the model prediction. The cohesion-score 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.

AOPC. By deleting top $k\%$ words, AOPC calculates the average change in the prediction probability on the predicted class over all test data as follows,

$$\text{AOPC}(k) = \frac{1}{N}\sum_{i=1}^{N}\{p(\hat{y} \mid \boldsymbol{x}_i) - p(\hat{y} \mid \tilde{\boldsymbol{x}}_i^{(k)})\}, \tag{4.6}$$

where $\hat{y}$ is the predicted label, N is the number of examples, $p(\hat{y} \mid \cdot)$ is the probability on the predicted class, and $\tilde{\boldsymbol{x}}_i^{(k)}$ is constructed by dropping the $k\%$ top-scored words from $\boldsymbol{x}_i$. Higher AOPCs are better, which means that the deleted words are important for model prediction. To compare with other word-level explanation generation methods under this metric, we select word-level features from the bottom level of a hierarchical explanation and sort them in the order of their estimated importance to the prediction.

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

$$\text{Log-odds}(r) = \frac{1}{N}\sum_{i=1}^{N} \log \frac{p(\hat{y} \mid \tilde{\boldsymbol{x}}_i^{(r)})}{p(\hat{y} \mid \boldsymbol{x}_i)}. \tag{4.7}$$

The notations are the same as in Equation 4.6 with the only difference that $\tilde{\boldsymbol{x}}_i^{(r)}$ is constructed by replacing the top r% word features with the special token ⟨`pad`⟩ in $\boldsymbol{x}_i$. Under this metric, lower log-odds scores are better.

Cohesion-score. We propose cohesion-score to justify an important text span identified by HEDGE. Given an important text span $\boldsymbol{x}_{(a,b]}$, we randomly pick a position in the word sequence $(x_1, \ldots, x_a, x_{b+1}, \ldots, x_n)$ and insert a word back. The process is repeated until a shuffled version of the original sentence $\bar{\boldsymbol{x}}$ is constructed. The cohesion-score is the difference between $p(\hat{y} \mid \boldsymbol{x})$ and $p(\hat{y} \mid \bar{\boldsymbol{x}})$. Intuitively, the words in an important text span have strong interactions. By perturbing such interactions, we expect to observe the output probability decreasing. To obtain a robust evaluation, for each sentence $\boldsymbol{x}_i$, we construct Q different word sequences $\{\bar{\boldsymbol{x}}_i^{(q)}\}_{q=1}^{Q}$ and compute the average as

$$\text{Cohesion-score} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q}\sum_{q=1}^{Q}(p(\hat{y} \mid \boldsymbol{x}_i) - p(\hat{y} \mid \bar{\boldsymbol{x}}_i^{(q)})), \tag{4.8}$$

where $\bar{\boldsymbol{x}}_i^{(q)}$ is the q^{th} perturbed version of $\boldsymbol{x}_i$, Q is set as 100, and the most important text span in the contribution set $\mathcal{C}$ is considered. Higher cohesion-scores are better.

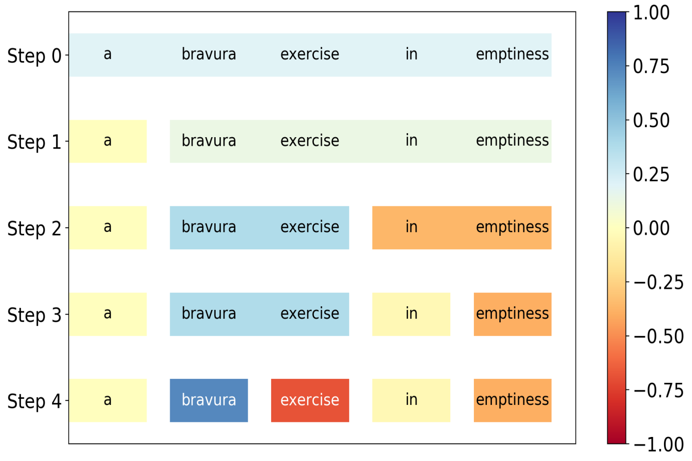
Results

We compare HEDGE with several competitive baselines, namely Leave-one-out [48], LIME [1], CD [51], Shapley-based methods, [49, L/C-Shapley], [24, KernelSHAP], and [2, SampleShapley], using AOPC and log-odds metrics; and use cohesion-score to compare HEDGE with another hierarchical explanation generation method ACD [3].

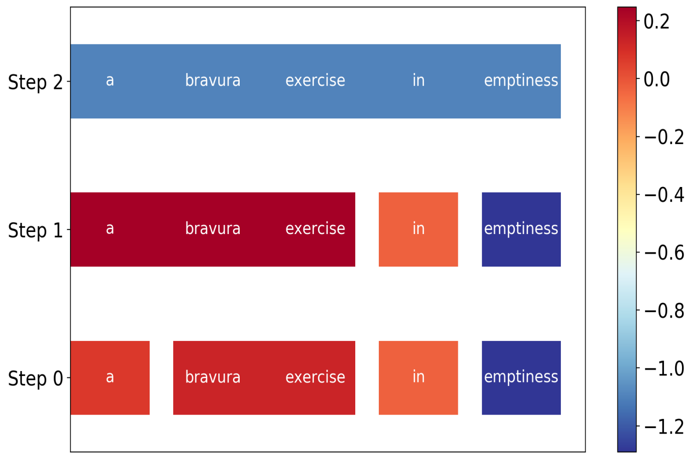
Methods	Models	Cohesion-score	
		SST	IMDB
HEDGE	CNN	0.016	0.012
	BERT	0.124	0.103
	LSTM	0.020	0.050
ACD	LSTM	0.015	0.038

Table 4.3: Cohesion scores of HEDGE and ACD in interpreting different models on the SST and IMDB datasets. For ACD, we adopt the existing application from the original paper [3] to explain LSTM on text classification.

The AOPCs and log-odds scores on different models and datasets are shown in Table 4.2, where $k = r = 20$. For the IMDB dataset, we tested on a subset with 2000 randomly selected samples due to computation costs.



(a) Hedge for LSTM on the SST.



(b) ACD for LSTM on the SST.

Figure 4.2: Compare Hedge with ACD in interpreting the LSTM model on a negative movie review from the SST dataset, where LSTM makes a wrong prediction (positive). The importance scores of Hedge and CD scores are normalized for comparison.

Hedge achieves the best performance on both evaluation metrics. SampleShapley also achieves a good performance with the number of samples set as 100, but the computational complexity is 200 times than Hedge. Other variants, L/C-Shapley and KernelSHAP, applying approximations to Shapley values perform worse than SampleShapley and Hedge. LIME performs comparatively to SampleShapley on the LSTM and CNN models, but is not fully capable of interpreting the deep neural network BERT. The limitation of context decomposition mentioned by Jin et al. [52]

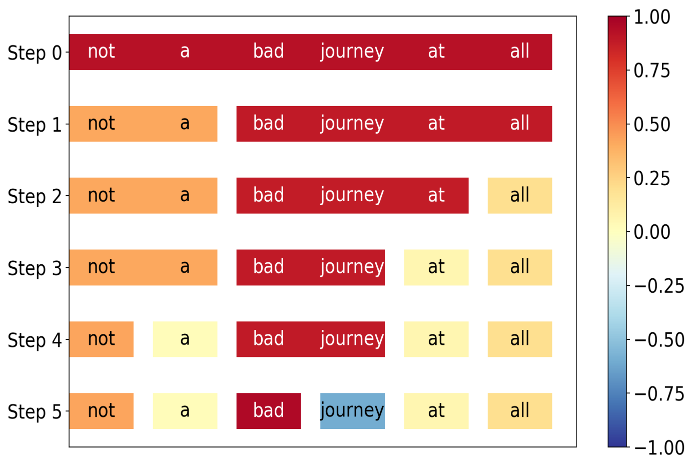
is validated by the worst performance of CD in identifying important words. We also observed an interesting phenomenon that the simplest baseline Leave-one-out can achieve relatively good performance, even better than HEDGE when k and r are small. And we suspect that is because the criteria of Leave-one-out for picking single keywords matches the evaluation metrics. Overall, experimental results demonstrate the effectiveness of Equation 4.5 in measuring feature importance. And the computational complexity is only $\mathcal{O}(n)$, which is much smaller than other baselines (e.g. SampleShapley, and L/C-Shapley with polynomial complexity).

Table 4.3 shows the cohesion-scores of HEDGE and ACD with different models on the SST and IMDB datasets. HEDGE outperforms ACD with LSTM, achieving higher cohesion-scores on both datasets, which indicates that HEDGE is good at capturing important phrases. Comparing the results of HEDGE on different models, the cohesion-scores of BERT are significantly higher than LSTM and CNN. It indicates that BERT is more sensitive to perturbations on important phrases and tends to utilize context information for predictions.

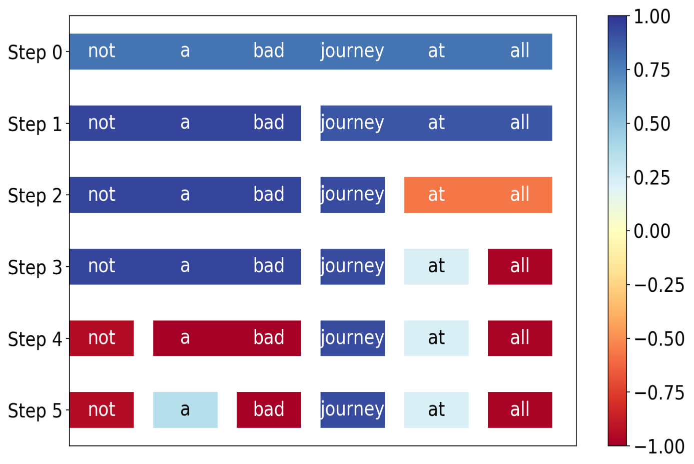
Qualitative Analysis

For qualitative analysis, we present two typical examples. In the first example, we compare HEDGE with ACD in interpreting the LSTM model. Figure 4.2 visualizes two hierarchical explanations, generated by HEDGE and ACD respectively, on a negative movie review from the SST dataset. In this case, LSTM makes a wrong prediction (POSITIVE). Figure 4.2(a) shows HEDGE correctly captures the sentiment polarities of `bravura` and `emptiness`, and the interaction between them as `bravura exercise` flips the polarity of `in emptiness` to positive. It explains why the model makes the wrong prediction. On the other hand, ACD incorrectly marks the two words with opposite polarities, and misses the feature interaction, as Figure 4.2(b) shows.

In the second example, we compare HEDGE in interpreting two different models (LSTM and BERT). Figure 4.3 visualizes the explanations on a positive movie review. In this case, BERT gives the correct prediction (POSITIVE), while LSTM makes a wrong prediction (NEGATIVE). The comparison between Figure 4.3(a) and 4.3(b) shows the difference of feature interactions within the two models and explains how a correct/wrong prediction was made. Specifically, Figure 4.3(b) illustrates that BERT captures the key phrase `not a bad` at step 1, and thus makes the positive prediction, while LSTM (as shown in Figure 4.3(a)) misses the interaction between `not` and `bad`, and the negative word `bad` pushes the model making the NEGATIVE prediction. Both cases show that HEDGE is capable of explaining model prediction behaviors, which helps humans understand the decision-making.



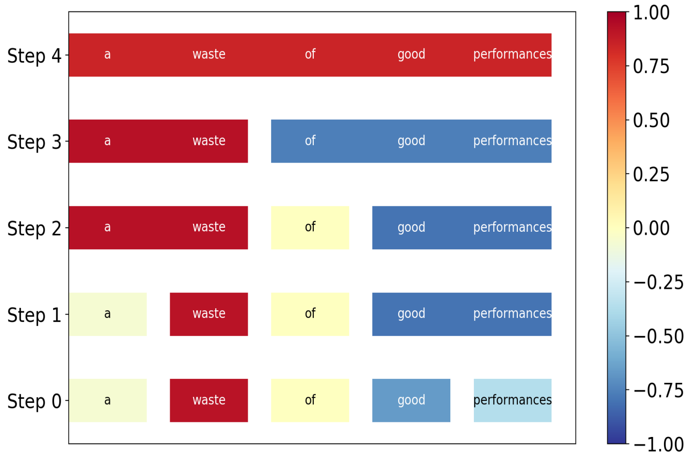
(a) Hedge for LSTM on SST.



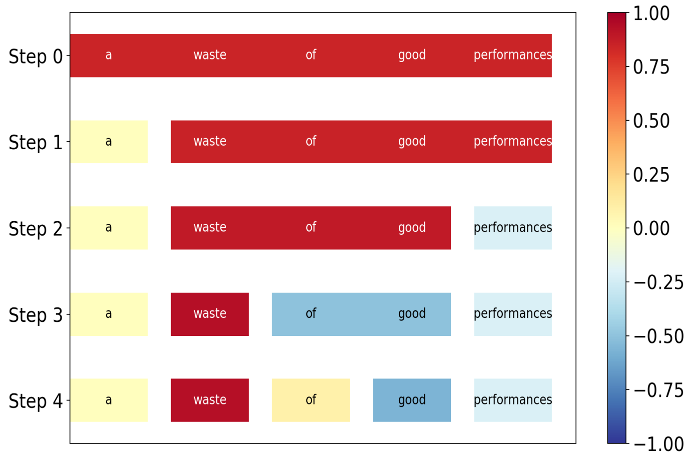
(b) Hedge for BERT on SST.

Figure 4.3: Compare Hedge in interpreting different models (LSTM and BERT) on a positive movie review from the SST dataset, where BERT makes the correct prediction (positive), while LSTM makes a wrong prediction (negative). Hedge explains that BERT captures the important phrase `not a bad` for making the correct prediction, while LSTM ignores it and is misled by the negative word `bad`.

Comparison between top-down and bottom-up approaches. Given the sentence `a waste of good performance` for example, Figure 4.4 shows the hierarchical interpretations for the LSTM model using the bottom-up and top-down approaches respectively. Figure 4.4(a) shows that the interaction between `waste` and `good` can not be captured until the last (top) layer, while the important phrase `waste of good` can be extracted in the intermediate layer by top-down algorithm. We can see that `waste` flips the polarity of `of good` to negative, causing the model predicting negative as well. Top-down segmentation performs better than bottom-up in capturing feature interactions. The reason is that the bottom layer contains more features than the top layer, which incurs larger errors in calculating interaction scores. Even worse, the calculation error will propagate and accumulate during clustering.



(a) Bottom-up clustering.



(b) Top-down segmentation.

Figure 4.4: Hierarchical interpretations for the LSTM model using the bottom-up and top-down approaches respectively. Red and blue colors represent the negative and positive sentiments respectively.

Human Evaluation

We had 9 human annotators from the Amazon Mechanical Turk (AMT) for human evaluation. The features (e.g., words or phrases) with the highest importance score given by HEDGE and other baselines are selected as the explanations. Note that HEDGE and ACD can potentially give very long top features which are not user-friendly in human evaluation, so we additionally limit the maximum length of selected features to five. We provided the input text with different explanations in the user interface (as shown in Figure 4.5) and asked human annotators to guess the model's prediction [129] from {"Negative", "Positive", "N/A"} based on each explanation, where "N/A" was selected when annotators cannot guess the model's prediction. We randomly picked 100 movie reviews from the IMDB dataset for human evaluation.

There are two dimensions of human evaluation. We first compare HEDGE with other baselines using the predictions made by the same LSTM model. Second, we compare the explanations generated by HEDGE on three different models: LSTM, CNN, and BERT. We measure the number of human annotations that are *coherent* with the actual model predictions, and define the *coherence score* as the ratio between the coherent annotations and the total number of examples.

Table 4.4 shows the coherence scores of eight different interpretation methods for LSTM on the IMDB dataset. HEDGE outperforms other baselines with higher coherence score, which means that HEDGE can capture important features which are highly consistent with human interpretations. LIME is still a strong baseline in providing interpretable explanations, while ACD and Shapley-based methods perform worse. Table 4.5 shows both the accuracy and coherence scores of different models. HEDGE succeeds in interpreting black-box models with relatively high coherence scores. Moreover, although BERT can achieve higher prediction accuracy than the other two models, its coherence score is lower, manifesting a potential tradeoff between accuracy and interpretability of deep models.

Methods	Coherence Score
Leave-one-out	0.82
ACD	0.68
LIME	0.85
L-Shapley	0.75
C-Shapley	0.73
KernelSHAP	0.56
SampleShapley	0.78
HEDGE	**0.89**

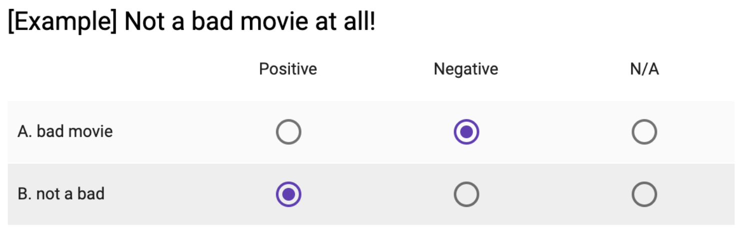
Table 4.4: Human evaluation of different interpretation methods with LSTM model on the IMDB dataset.

Models	Accuracy	Coherence scores
LSTM	0.87	0.89
CNN	0.90	0.84
BERT	0.97	0.75

Table 4.5: Human evaluation of HEDGE with different models on the IMDB dataset.

4.1.5 Conclusion

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



[Example] Not a bad movie at all!

	Positive	Negative	N/A
A. bad movie	○	◉	○
B. not a bad	◉	○	○

Figure 4.5: Interfaces of Amazon Mechanical Turk where annotators are asked to guess the model's prediction based on different explanations.

4.2 Explaining Neural Network Predictions on Sentence Pairs via Learning Word-Group Masks

Explaining deep neural networks is critical for revealing their prediction behaviors and enhancing the trustworthiness of applying them in real-world applications. Many methods have been proposed to explain neural network models from the post-hoc manner that generates faithful explanations based on model predictions [1, 24, 25, 168]. Most existing work focuses on identifying word attributions [169, 48, 170] for NLP tasks. Knowing which individual features are important might not be enough for explaining model behaviors. Then, other recent work exploits feature interactions as explanations [3, 171, 172]. However, they could suffer computation inefficiency while computing interactions between all word pairs, and they also fall short for identifying multiple important words correlated from different input sources for predictions. Such intuitions are particularly important for explaining sentence pair modeling tasks such as natural language inference (NLI) [173] and paraphrase identification (PI) [174].

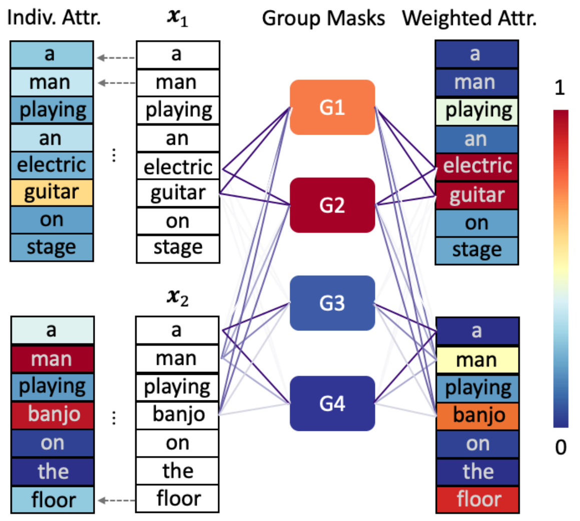


Figure 4.6: An illustration of obtaining individual word attributions (Indiv. Attr.) and weighted word attributions (Weighted Attr.), where the color of each block represents word importance or group importance, and the color saturation of purple lines indicates the probability of a word belonging to a specific group.

Figure 4.6 shows an example of NLI, where the model makes correct prediction as CONTRADICTION. The first column visualizes individual word attributions to the prediction, where the top four important words are `man`, `banjo`, `guitar`, `a`. However, the correlations between them are unclear and intuitively `man` and `a` are irrelevant to the model prediction, which makes the explanation untrustworthy. A good explanation should be able to capture correlated words between the sentence pair, and identify their importance to the model prediction.

In this work, we propose Group Masks (GMASK), a model-agnostic approach that considers the importance of correlated words from two input sentences. In particular, it distributes correlated words into a group and learns the group importance. In Figure 4.6, the input words are distributed in four groups with importance $G2 > G1 > G3 > G4$. The color saturation of purple lines represents the probability of a word belonging to a group. Different from individual word attributions, GMASK assigns `electric`, `guitar`, and `banjo` into important groups ($G2/G1$), while `man` and `a` into unimportant groups ($G3/G4$). The weighted word attributions computed as the weighted sum of group importance identify the important words `electric`, `guitar` from $\boldsymbol{x}_1$ and `banjo` from $\boldsymbol{x}_2$, which explains the model prediction.

The contribution of this work is three-fold: (1) we introduce GMASK method to explain sentence pair modeling tasks by learning weighted word attributions based on word correlations; (2) we propose a sampling-based method to solve the optimization objective of GMASK; and (3) we evaluate the proposed method with two types neural network models (decomposable attention model [175] and BERT [13]), for two types of sentence pair modeling tasks on four datasets. Experiments show the superiority of GMASK in generating faithful explanations compared to other competitive methods.

4.2.1 GMASK

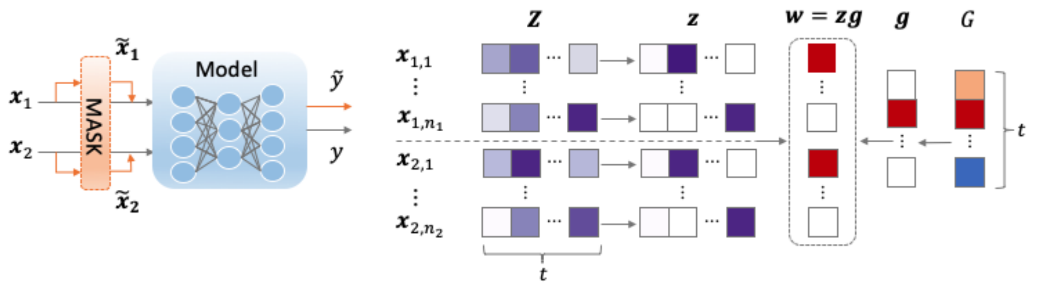


Figure 4.7: The left part shows that masks are applied on the word embedding layer, selecting important words for the neural network model. The outputs y and $\tilde{y}$ are corresponding to the original input $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]^T$ and masked input $\tilde{\boldsymbol{x}} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2]^T$ respectively. The right part shows the sampling process of GMASK. For $\boldsymbol{Z}$, the color saturation of purple blocks represents the probability of a word belonging to a specific group (i.e. $\phi_{i,j}(\iota)$). $\boldsymbol{z}$ is a sample of $\boldsymbol{Z}$ with binary values. For G, the color of each block represents group importance. $\boldsymbol{g}$ is a one-hot vector sampled from G, indicating which group being selected. $\boldsymbol{w}$ is a sample of word masks obtained by multiplying $\boldsymbol{z}$ and $\boldsymbol{g}$.

This section introduces the proposed GMASK method. GMASK implicitly learns word correlations, and distributes correlated words from different input sentences into a group. GMASK learns the importance of each group by randomly masking out groups of words. Finally, the weighted word attributions are computed based on word group distributions and group importance.

Explaining Models with Word Masks

As the left part of Figure 4.7 shows, the word masks are applied on input word embeddings, learning to select important words to explain the model prediction. For each input data, we generate a post-hoc explanation by learning a set of mask values which represent the word attributions.

For sentence pair modeling tasks, the input contains two sentences $\boldsymbol{x}_1 = [\boldsymbol{x}_{1,1}^T, \dots, \boldsymbol{x}_{1,n_1}^T]^T$ and $\boldsymbol{x}_2 = [\boldsymbol{x}_{2,1}^T, \dots, \boldsymbol{x}_{2,n_2}^T]^T$, where $\boldsymbol{x}_{i,j} \in \mathbb{R}^d$ $(i \in \{1,2\},\ j \in \{1, \dots, n_i\})$ represents the word embedding and n_1 and n_2 are the number of words in the two texts respectively. We denote the neural network model as $f(\cdot)$ which takes $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as input and outputs a prediction label $y = f(\boldsymbol{x})$, where $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]^T$. To explain the model prediction, we learn a set of word masks $\boldsymbol{W} = [W_{1,1}, \dots, W_{1,n_1}, W_{2,1}, \dots, W_{2,n_2}]^T$ to identify important words by multiplying the masks with input word embeddings,

$$\tilde{\boldsymbol{x}} = \boldsymbol{W} \odot \boldsymbol{x}, \tag{4.9}$$

where $\odot$ is an element-wise multiplication, the masked input $\tilde{\boldsymbol{x}} = [\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2]^T$, $\tilde{\boldsymbol{x}}_{i,j} = W_{i,j} \cdot \boldsymbol{x}_{i,j}$ $(i \in \{1,2\},\ j \in \{1, \dots, n_i\})$, and $W_{i,j} \in \{0, 1\}$ is a binary random variable with 1 and 0 indicating to select or mask out the word $\boldsymbol{x}_{i,j}$ respectively. To generate an effective explanation, the word masks $\boldsymbol{W}$ should have the following properties: (1) correctly selecting important words for the model prediction; (2) removing as many irrelevant words as possible to keep the explanation concise; (3) selecting or masking out correlated words together from the sentence pair.

Previous work on learning individual word masks only focuses on the first two properties [176, 154]. To satisfy the third property, We propose GMASK to implicitly detect word correlations and distribute the correlated words into a group (e.g. `electric`, `guitar`, and `banjo` are assigned to $G1$ or $G2$ in Figure 4.6), and learn a group mask for these words. Specifically, we decompose each $W_{i,j}$ in $\boldsymbol{W}$ into two random variables,

$$W_{i,j} = \sum_{\iota=1}^{t} \delta(Z_{i,j}, \iota)\delta(G, \iota), \tag{4.10}$$

where t is the predefined number of groups, and we will introduce how to pick up a t subsequently. $Z_{i,j} \in \{1, \ldots, t\}$ indicates the word $\boldsymbol{x}_{i,j}$ belonging to which group, and $G \in \{1, \ldots, t\}$ indicates which group takes the mask value 1, which means all words in this group are selected as important words, while other words in the rest groups are masked out. $\delta(a, b)$ is the Delta function with $\delta(a, b) = 1$ when $a = b$, and 0 otherwise. The conditional dependency of $\boldsymbol{W}$, $\boldsymbol{Z}$, and G can be represented as a graphical model [2]. The problem of learning $\boldsymbol{W}$ is equivalent to learning $\boldsymbol{Z}$ and G, that is learning word distributions among the groups and group importance. According to $\delta(Z_{i,j}, \iota)$ and $\delta(G, \iota)$, the word masks $\boldsymbol{W}$ will keep or mask out all words in group ι, which satisfies the third property.

Learning GMASK

We formulate the problem of learning GMASK by optimizing the following objective in terms of the three properties,

$$\max_{\Phi, \Psi} \mathbb{E}[p(y \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{g})] - \gamma_1 \mathcal{L}_{\boldsymbol{Z}} - \gamma_2 \mathcal{L}_G, \tag{4.11}$$

where Φ and Ψ are parameters of $\boldsymbol{Z}$ and G respectively, and $\boldsymbol{z}$ and $\boldsymbol{g}$ are samples of $\boldsymbol{Z}$ and G respectively. We denote $\mathcal{L}_{\boldsymbol{Z}}$ and $\mathcal{L}_G$ as regularizations on $\boldsymbol{Z}$ and G respectively, which are applied to make the learned masks satisfy the required properties. We will introduce the two regularization terms subsequently. $\gamma_1, \gamma_2 \in \mathbb{R}_+$ are coefficients.

Optimizing the first term in Equation 4.11 is to make the word masks $\boldsymbol{W}$ satisfy the first property, that is the model outputs the same prediction on the selected words as on the whole text. Given $\boldsymbol{Z}$ and G, we have word masks $\boldsymbol{W}$, and multiply them with input word embeddings, and obtain the masked input $\tilde{\boldsymbol{x}}$ as in Equation 4.9. The model output on $\tilde{\boldsymbol{x}}$ is $\tilde{y} = f(\tilde{\boldsymbol{x}})$. If the masks correctly select important words, the predicted label on the selected words should be equal to that on the whole input text. We can optimize the first term by minimizing the cross entropy loss ($\mathcal{L}_{ce}(\cdot, \cdot)$) between $\tilde{y}$ and y. The objective Equation 4.11 can be rewritten as

$$\min_{\Phi, \Psi} \mathcal{L}_{ce}(y, \tilde{y}) + \gamma_1 \mathcal{L}_{\boldsymbol{Z}} + \gamma_2 \mathcal{L}_G. \tag{4.12}$$

The last two terms in the optimization objective are to make word masks satisfy the second and third properties. We regularize $\boldsymbol{Z}$ to encourage each group contains some words from different sentences. We regularize G to ensure only one or few groups are identified as important (with relatively large probabilities). Optimizing the cross entropy loss with the two regularization terms

[2] $\boldsymbol{Z} \rightarrow \boldsymbol{W} \leftarrow G$: $\boldsymbol{Z}$ and G are dependent given $\boldsymbol{W}$.

can make the word masks select the important group of words, where the words are selected from the input sentence pair and are correlated.

Regularizations on $\boldsymbol{Z}$ and G. As each $Z_{i,j}$ ($i \in \{1,2\}$, $j \in \{1,\dots,n_i\}$) indicates a word belonging to a specific group, it follows categorical distribution with probabilities $[\phi_{i,j}(1),\dots,\phi_{i,j}(t)]$, where t is the predefined number of groups, and $\phi_{i,j}(\iota)$ ($\iota \in \{1,\dots,t\}$) represents the probability of the word in group ι. Then we denote the parameters of $\boldsymbol{Z}$ as Φ,

$$\Phi = \begin{bmatrix} \phi_{1,1}(1) & \cdots & \phi_{1,1}(t) \\ \vdots & \cdots & \vdots \\ \phi_{1,n_1}(1) & \cdots & \phi_{1,n_1}(t) \\ \phi_{2,1}(1) & \cdots & \phi_{2,1}(t) \\ \vdots & \cdots & \vdots \\ \phi_{2,n_2}(1) & \cdots & \phi_{2,n_2}(t) \end{bmatrix} \tag{4.13}$$

To ensure that each group contains some words from both input sentences, and also avoid assigning a bunch of words into one group, we distribute the words in each sentence evenly among all groups. Then each group implicitly captures the words from different sentences. We can regularize $\boldsymbol{Z}$ to achieve this goal. We sum each column of Φ along the upper half rows and lower half rows respectively, and obtain two vectors by taking averages, $\boldsymbol{\phi}^U = \frac{1}{n_1}[\sum_{j=1}^{n_1}\phi_{1,j}(1),\dots,\sum_{j=1}^{n_1}\phi_{1,j}(t)]$, $\boldsymbol{\phi}^L = \frac{1}{n_2}[\sum_{j=1}^{n_2}\phi_{2,j}(1),\dots,\sum_{j=1}^{n_2}\phi_{2,j}(t)]$. Then $\boldsymbol{\phi}^U$ and $\boldsymbol{\phi}^L$ are the distributions of two discrete variables Z^U and Z^L, which also represent the word distributions of the two input sentences among groups. To make the distributions of words even, we maximize the entropy of Z^U and Z^L, and have

$$\mathcal{L}_{\boldsymbol{Z}} = -(H(Z^U) + H(Z^L)), \tag{4.14}$$

where $H(\cdot)$ is entropy.

$G \in \{1,\dots,t\}$ also follows categorical distribution with probabilities $\Psi = [\psi(1),\dots,\psi(t)]$, where $\psi(\iota)$ ($\iota \in \{1,\dots,t\}$) represents the probability of group ι being selected. According to the relation of $\boldsymbol{W}$, $\boldsymbol{Z}$, G in Equation 4.10, the word masks only keep the words assigned to the selected group. To ensure one or few groups have relatively large probabilities to be selected, we regularize G by minimizing its entropy, that is $\mathcal{L}_G = H(G)$. The final optimization objective is

$$\min_{\Phi,\Psi} \mathcal{L}_{ce}(y,\tilde{y}) - \gamma_1(H(Z^U) + H(Z^L)) + \gamma_2 H(G). \tag{4.15}$$

Optimization via sampling. We adopt a sampling based method to solve Equation 4.15 by learning the parameters of $\boldsymbol{Z}$ and G (i.e. $\{\Phi, \Psi\}$). As the right part of Figure 4.7 shows, we sample a $\boldsymbol{z}$ from the categorical distributions of $\boldsymbol{Z}$, where each row $\boldsymbol{z}_{i,j}$ is a one-hot vector, indicating the word $\boldsymbol{x}_{i,j}$ assigned to a specific group. And we sample a $\boldsymbol{g}$ from the categorical distribution of G, which is a vertical one-hot vector, indicating the selected group. Then we obtain a sample of word masks by multiplying $\boldsymbol{z}$ and $\boldsymbol{g}$, i.e. $\boldsymbol{w} = \boldsymbol{z} \cdot \boldsymbol{g}$, where the mask values corresponding to the words in the selected group are 1, while the rest are 0. We apply the masks on the input word embeddings and optimize Equation 4.15 via stochastic gradient descent.

There are two challenges of the learning process: discreteness and large variance. We apply the Gumbel-softmax trick [157, 158] to address the discreteness of sampling from categorical distributions in backpropagation. We do the sampling multiple times and generate a batch of masked inputs of the original input data to decrease the variance in probing the model, and train for multiple epochs until the learnable parameters $\{\Phi, \Psi\}$ reach stable values.

Weighted word attributions. After training, we learn the parameters of $\boldsymbol{Z}$, i.e. Φ, where each element $\phi_{i,j}(\iota) \in (0,1)$ $(i \in \{1,2\},\ j \in \{1,\dots,n_i\},\ \iota \in \{1,\dots,t\})$ represents the probability of word $\boldsymbol{x}_{i,j}$ belonging to group ι. We also learn the parameters of G, i.e. Ψ, where each element $\psi(\iota) \in (0,1)$ represents the importance of group ι. According to the definition of word masks $\boldsymbol{W}$, we know that each mask variable $W_{i,j}$ follows Bernoulli distribution, and the probability of $W_{i,j}$ taking 1 is denoted as $\theta_{i,j}$. We can compute $\theta_{i,j}$ based on the relation of $W_{i,j}$, $Z_{i,j}$ and G in Equation 4.10, that is

$$\theta_{i,j} = \sum_{\iota=1}^{t} \phi_{i,j}(\iota)\psi(\iota). \tag{4.16}$$

We can see that $\theta_{i,j}$ is the expectation of $W_{i,j}$, representing the weighted attribution of the word $\boldsymbol{x}_{i,j}$ to the model predicition. Hence, we have a set of weighted word attributions $\Theta = [\theta_{1,1}, \dots, \theta_{1,n_1}, \theta_{2,1}, \dots, \theta_{2,n_2}]^T$ for extracting important words as an explanation.

Complexity. For a set of n words, computing interactions between all word pairs costs $O(n^2)$ and aggregating words step by step to form a tree structure even costs more time [3, 171]. GMASK circumvents the feature interaction detection by learning word groups. The complexity is $O(nt + t)$, where t is the number of groups and usually $t \ll n$ in practice.

Implementation Specification

We initialize the parameters of all categorical distributions ($\{\Phi, \Psi\}$) with $\frac{1}{t}$, which means all words have the same importance and do not have any preference to be in a specific group at the start of training. To stabilize the learning process, we sample 100 - 1000 examples (depending on the model and datasets) and train at most 100 epochs until converge. The coefficients γ_1 and γ_2 are hyperparameters. We empirically found $\gamma_1 = 10$ and $\gamma_2 = 1$ work well in our experiments.

In our pilot experiments, we found that preliminarily filtering out some noisy or irrelevant words can help decrease the learnable parameters, hence accelerating the training process. Specifically, we adopt a simple word mask method from [176] to select a set of individual words for an input sentence pair before running GMASK. This simple method, denoted as IMASK, will learn individual word attributions as masks $\boldsymbol{R} = \{R_{i,j}\}_{i \in \{1,2\},\ j \in \{1,\ldots,n_i\}} \in \{0,1\}^{n_1+n_2}$ regardless any correlation. Then, based on the expected values of $\boldsymbol{R}$, we preliminarily select top k words for GMASK to further learn weighted word attributions. Within these top k words, assume k_1 words from the first input text and k_2 words from the second text, then we will set the number of groups as $t = \min(k_1, k_2)$, so that at least one group contains words from both sentences. k is a hyper-parameter associated with the average length of input texts. In the experiments, we set $k = 10$. Note that, the IMASK method adopted here can also be used as a baseline method for comparison.

4.2.2 Experimental Setup

We evaluate GMASK with two kinds of neural network models, decomposable attention model (DAttn) [175] and BERT [13], for two types of sentence pair modeling tasks on four datasets. We compare our method with four baselines.

Datasets. e-SNLI [42] is natural language inference task, where the model predicts the semantic relationship between two input sentences as entailment, contradiction, or neutral. Quora [177], QQP [178] and MRPC [179] are paraphrase identification tasks, where the model decides whether two input texts are semantically equivalent or not. Table 4.6 shows the statistics of the four datasets. We adopt the data splits of e-SNLI [42] from the ERASER benchmark [3]. We adopt the data splits of Quora released by Wang et al. [177]. The data splits of QQP [178] and MRPC [179] are from the GLUE benchmark [4].

[3]https://www.eraserbenchmark.com/
[4]https://gluebenchmark.com/

Datasets	C	L	V	*#train*	*#dev*	*#test*
e-SNLI	3	10.2	64291	549K	9K	9K
Quora	2	11.5	85249	384K	10K	10K
QQP	2	11.1	126266	364K	40K	391K
MRPC	2	22	15547	3668	408	1725

Table 4.6: Summary statistics for the datasets, where C is the number of classes, L is average sentence length, V is vocab size, and # counts the number of examples in the *train/dev/test* sets.

Models	e-SNLI	Quora	QQP	MRPC
DAttn	86.62	86.78	85.00	68.30
BERT	90.38	90.48	89.00	83.70

Table 4.7: The prediction accuracy (%) of different models on the four datasets.

Models. We adopt the decomposable attention model (DAttn) [175] and BERT [13] model, and fine-tune the models on each downstream task to achieve the best performance, as Table 4.7 shows. The test results on QQP and MRPC are scored by the GLUE benchmark.

Baselines. We compare GMASK with four baseline methods: (1) LIME [1]-fitting a local linear model with perturbations to approximate the neural network and produce word attributions; (2) L2X [65]-constructing a network to learn feature attributions by maximizing the mutual information between the selected features and model output; (3) IBA (*Per-Sample* framework) [29] - learning feature attributions by optimizing the information bottleneck which restricts feature information flow by adding noise; (4) IMASK-learning individual word masks. Note that here we use standalone IMASK as one of the baselines, as oppose to applying it for selecting preliminary important words for GMASK.

4.2.3 Results and Discussion

We compare the faithfulness of generated post-hoc explanations via both quantitative and qualitative evaluations.

Quantitative Evaluation

We adopt three metrics from prior work to evaluate the faithfulness of learned feature attributions: AOPC score [129, 73], post-hoc accuracy [65, 176], and degradation score [74, 29]. We evaluate explanations on all test data for the MRPC dataset, and on 2000 examples randomly selected from the test set for other three datasets due to computational complexity.

Models	Methods	e-SNLI	Quora	QQP	MRPC
DAttn	LIME	0.286	0.120	0.079	0.064
	L2X	0.299	0.128	0.079	0.035
	IBA	0.354	0.137	**0.104**	**0.109**
	IMASK	0.324	0.140	0.087	0.064
	GMASK	**0.361**	**0.142**	0.095	0.091
BERT	LIME	0.221	0.153	0.110	0.062
	L2X	0.310	0.119	0.134	0.083
	IBA	0.282	0.199	0.144	0.114
	IMASK	0.292	0.232	0.139	0.130
	GMASK	**0.319**	**0.309**	**0.181**	**0.200**

Table 4.8: AOPC scores of different explanation methods with the DAttn and BERT models on the four datasets.

AOPC score

We adopt the area over the perturbation curve (AOPC) [129, 73] metric to evaluate the comprehensiveness of explanations to models. It calculates the average change of prediction probability on the predicted class over all examples by removing top $1 \dots u$ words in explanations.

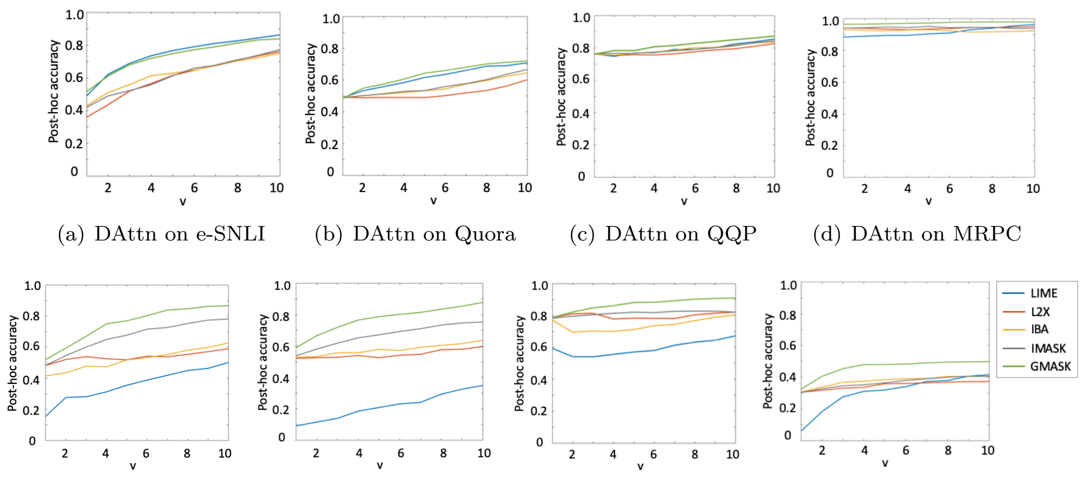
$$\text{AOPC} = \frac{1}{U+1} \langle \sum_{u=1}^{U} p(y|\boldsymbol{x}) - p(y|\boldsymbol{x}_{\backslash 1 \dots u}) \rangle_{\boldsymbol{x}}, \tag{4.17}$$

where $p(y|\boldsymbol{x}_{\backslash 1 \dots u})$ is the probability for the predicted class when words $1 \dots u$ are removed and $\langle \cdot \rangle_{\boldsymbol{x}}$ denotes the average over all test examples. Higher AOPC score indicates better explanations.

Table 4.8 shows the results of AOPC scores of different explanation methods when $U = 10$. GMASK outperforms other baseline methods on most of the datasets. Especially for the BERT model, GMASK achieves significantly higher AOPC scores than other methods, indicating that BERT tends to rely on word correlations to make predictions. IBA and IMASK, either learning continuous or binary individual word masks, perform better than learning word attributions via an additional network (L2X) or using linear approximation (LIME).

Post-hoc Accuracy

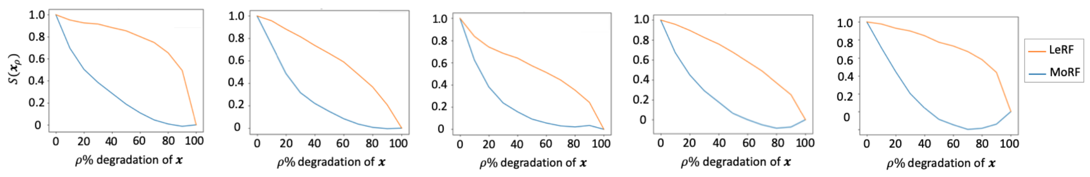
The post-hoc accuracy [65, 176] evaluates the sufficiency of important words to the model prediction. For each test data, we select top v important words based on word attributions for the model to make a prediction, and compare it with the original prediction made on the whole input text. We



(a) DAttn on e-SNLI (b) DAttn on Quora (c) DAttn on QQP (d) DAttn on MRPC

(e) BERT on e-SNLI (f) BERT on Quora (g) BERT on QQP (h) BERT on MRPC

Figure 4.8: Post-hoc accuracy of different explanation methods with the DAttn and BERT models on the four datasets.



(a) LIME (b) L2X (c) IBA (d) IMASK (e) GMASK

Figure 4.9: Degradation test of different explanation methods with the DAttn model on the e-SNLI dataset.

compute the post-hoc accuracy on M examples,

$$\text{post-hoc-acc}(v) = \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}[y_v^{(m)} = y^{(m)}],$$

where $y^{(m)}$ is the predicted label on the m-th test data, and $y_v^{(m)}$ is the predicted label based on the top v important words. Higher post-hoc accuracy indicates better explanations.

Figure 4.8 shows the results of post-hoc accuracy of different explanation methods where we increase v from 1 to 10. Similar to the results of the AOPC scores, GMASK achieves higher post-hoc accuracy than other methods for both DAttn and BERT models.

The explanations of GMASK for the BERT model achieve about 80% post-hoc accuracy on all datasets except the MRPC dataset. This is only by relying on top 4 important words, which means that GMASK captures informative words for model predictions. The post-hoc accuracies of the BERT model on the MRPC dataset are lower than those on other three datasets because the average sentence length of MRPC is twice as long as the others, indicating that BERT tends to use larger context for predictions. The post-hoc accuracies of the DAttn model on the MRPC dataset are extremely high for all the explanation methods. The reason is that the prediction accuracy of DAttn model on the MRPC dataset is relatively low (Table 4.7). Any random words picked up by explanations could make the model output wrong predictions since the original predictions on the whole texts are also wrong, hence causing high post-hoc accuracy.

Models	Methods	e-SNLI	Quora	QQP	MRPC
DAttn	LIME	0.502	0.070	0.091	1.367
	L2X	0.366	0.002	0.036	1.779
	IBA	0.423	0.110	0.197	2.775
	IMASK	0.436	0.152	0.214	2.037
	GMASK	**0.620**	**0.178**	**0.238**	**2.790**
BERT	LIME	0.188	0.192	0.087	0.018
	L2X	0.303	0.168	0.173	-0.003
	IBA	0.166	0.038	0.176	0.050
	IMASK	0.369	0.303	0.172	0.251
	GMASK	**0.576**	**0.726**	**0.707**	**0.533**

Table 4.9: Degradation scores of different explanation methods with the DAttn and BERT models on the four datasets.

Degradation Test

Degradation test [74, 29] evaluates the ranking of importance by removing the most important words or least important words first, and observing model prediction probability drop on the predicted class. We draw two curves as shown in Figure 4.9, one with the most relevant words removed first (MoRF) and another one with the least relevant words removed first (LeRF). x-axis is the percentage of words removed (degradation proportion), and y-axis is the normalized model output probability as

$$S(\boldsymbol{x}_\rho) = \frac{p(y|\boldsymbol{x}_\rho) - p(y|\boldsymbol{x}_o)}{p(y|\boldsymbol{x}) - p(y|\boldsymbol{x}_o)}, \tag{4.18}$$

where $\boldsymbol{x}$ is the original input, y is the predicted label, $\boldsymbol{x}_\rho$ means $\rho\%(\rho \in [0, 100])$ degradation of $\boldsymbol{x}$, and $\boldsymbol{x}_o$ is full degradation. We compute the averages of $p(y|\boldsymbol{x}_\rho)$, $p(y|\boldsymbol{x})$, and $p(y|\boldsymbol{x}_o)$ over all test examples. The degradation score is calculated as the integral between the MoRF and LeRF curves,

$$\text{degra-score} = \int_{\rho=0}^{100} \frac{S^L(\boldsymbol{x}_\rho) - S^M(\boldsymbol{x}_\rho)}{100} d\rho, \tag{4.19}$$

Models	Methods	Texts
DAttn	LIME	a man playing an electric guitar on stage . a man playing banjo on the floor .
	L2X	a man playing an electric guitar on stage . a man playing banjo on the floor .
	IBA	a man playing an electric guitar on stage . a man playing banjo on the floor .
	IMASK	a man playing an electric guitar on stage . a man playing banjo on the floor .
	GMASK	a man playing an electric guitar on stage . a man playing banjo on the floor .
BERT	LIME	why are vikings portrayed wearing horned helmets ? why did vikings have horns on their helmets ?
	L2X	why are vikings portrayed wearing horned helmets ? why did vikings have horns on their helmets ?
	IBA	why are vikings portrayed wearing horned helmets ? why did vikings have horns on their helmets ?
	IMASK	why are vikings portrayed wearing horned helmets ? why did vikings have horns on their helmets ?
	GMASK	why are vikings portrayed wearing horned helmets ? why did vikings have horns on their helmets ?

Table 4.10: Examples of different explanations, where the top four important words are highlighted. The important words in the first and second sentences are highlighted in pink and blue colors respectively. The color saturation indicates word attribution. The first example is from the e-SNLI dataset, and the DAttn model makes a correct prediction as CONTRADICTION. The second example is from the Quora dataset, and the BERT model makes a correct prediction as PARAPHRASES.

where $S^L(\boldsymbol{x}_\rho)$ and $S^M(\boldsymbol{x}_\rho)$ are normalized model outputs by removing the least or most important words respectively. Higher degradation score is better.

Table 4.9 shows the results of degradation scores of different explanation methods. GMASK shows superiority to other baseline methods under this metric. Figure 4.9 shows the degradation test results of DAttn model on the e-SNLI dataset. GMASK can distinguish both important and unimportant words, while IBA does not learn the correct order of unimportant words. LIME does not perform well in identifying important words, but captures the correct order of unimportant words. The MoRF and LeRF curves of L2X and IMASK are relatively symmetric, but not as expanded as GMASK.

Qualitative Evaluation

Table 4.10 shows different explanations on two examples from e-SNLI and Quora respectively. For the first example, the DAttn model makes a correct prediction as CONTRADICTION. For the second example, the BERT model also makes a correct prediction as PARAPHRASES. We highlight the top four important words, where the words in the first and second sentences are in pink and blue colors respectively. The color saturation indicates word attribution.

For the first example, LIME and IBA mainly capture the important words from the first sentence, while ignoring the ones in the second sentence (e.g. `banjo`, `floor`). On the contrary, L2X focuses on the words in the second sentence, while ignoring the important word `guitar` in the first sentence. IMASK picks up two irrelevant words `man` and `a` as important words, which can not explain the model prediction. GMASK correctly identifies top four important words and captures two correlated words `guitar` and `banjo` from the two input sentences respectively.

For the second example, only GMASK captures the two important correlated words `horned` and `horns`, which explains why the BERT model predicts the two input questions as paraphrases. LIME captures the overlapped word `helmets` in the two sentences, while L2X only selects some irrelevant words. Both IBA and IMASK identify a question mark as the important word, which is untrustworthy to the model prediction.

4.2.4 Conclusion

We focused on sentence pair modeling and proposed an effective method, GMASK, learning group masks for correlated words and calculating weighted word attributions. We tested GMASK with two different neural network models on four datasets, and assessed its effectiveness via both quantitative and qualitative evaluations.

4.3 Explaining Predictive Uncertainty by Looking Back at Model Explanations

Pre-trained language models [e.g., BERT; 13] have been indispensable to natural language processing (NLP) due to their remarkable performance [105, 180, 107, 15]. Predictive uncertainty estimation of pre-trained language models is an important measure of how likely people can trust their predictions [70, 71].

A typical way of measuring predictive uncertainty is to calibrate model outputs with the true correctness likelihood [181, 182, 113], so that the output probabilities well represent the confidence of model predictions. In this case, higher prediction confidence indicates lower uncertainty [71, 183].

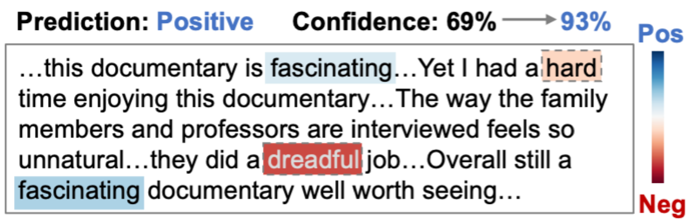

Prediction: Positive
Confidence: 69% → 93%
…this documentary is fascinating…Yet I had a hard time enjoying this documentary…The way the family members and professors are interviewed feels so unnatural…they did a dreadful job…Overall still a fascinating documentary well worth seeing…
Pos
Neg

Figure 4.10: An illustration of model explanation for sentiment classification, where the model makes the correct prediction (POSITIVE) with a relatively low confidence 69%. The top and bottom salient words with respect to the predicted label are highlighted in blue and red colors respectively, indicating different sentiment polarities. Darker color implies larger attribution. Removing the two bottom salient words in dashed boxes can improve the model prediction confidence to 93%.

However, little is known about what makes a model prediction uncertain. Explaining predictive uncertainty is important to understanding model prediction behavior and complementary to explaining prediction labels for gaining users' trust, while has been largely ignored [184]. Most works on model explanations focus on explaining a model from the post-hoc manner by identifying important features in inputs that contribute to model predicted labels [1, 24, 25, 185, 186]. Figure 4.10 shows an example of model explanation for sentiment classification, where the model makes the correct prediction (POSITIVE) with a relatively low confidence 69%. The top two salient words highlighted in blue color explain the predicted label. However, users may still wonder what compromises the prediction confidence?

This work is the first to explain model predictive uncertainty in NLP. Specifically, this work is based on a simple observation that bottom salient words in model explanations (e.g., `dreadful` and `hard` in Figure 4.10) identified as making negative contributions to predicted labels actually explain model predictive uncertainty. The two bottom salient words in Figure 4.10 indicate the opposite sentiment (NEGATIVE) to the model predicted label. Removing them can improve the model prediction confidence from 69% to 93%. We argue that both top and bottom salient words are indispensable to explaining model predictions. We name top salient words as *important words*, explaining model predicted labels; and bottom salient words as *uncertain words*, explaining model predictive uncertainty. In other words, a comprehensive prediction explanation should consist of *label explanation* with important words and *uncertainty explanation* with uncertain words.

The goal of this work is to demonstrate **the benefits of comprehensive explanations and the necessity of including uncertainty explanations**. In the empirical study, we adopt two explanation methods, Leave-one-out [48] and Sampling Shapley [187], to explain two pre-trained language models, BERT [13] and RoBERTa [105] on three tasks. Experiments show the effectiveness of the two methods in identifying uncertain words for explaining model predictive uncertainty. Besides, human evaluations illustrate the indispensability of uncertainty explanations in helping humans understand model prediction behavior.

4.3.1 Explaining Predictive Uncertainty

In this work, we consider models that are calibrated, such that their prediction confidence is aligned with their prediction probability. Let $f(\cdot)$ denote a model. Given an input $\boldsymbol{x} = [x_1, \ldots, x_N]$ consisting of N words, the model prediction probabilities on $\boldsymbol{x}$ over classes are $[f_1(\boldsymbol{x}), \ldots, f_C(\boldsymbol{x})]$, where $f_c(\boldsymbol{x}) = P(y = c \mid \boldsymbol{x})$ and C is the total number of classes. As model f is calibrated, the probability on the predicted class $\hat{y}$, i.e. $f_{\hat{y}}(\boldsymbol{x})$, represents the model prediction confidence on this label. As prediction confidence and predictive uncertainty are negative correlated (higher confidence implies lower uncertainty), we explain model predictive uncertainty by answering the question: *What drags model prediction confidence down?* We answer the question based on a simple observation on model prediction explanations [1, 48, 24].

When a feature is identified with negative contribution, removing it can improve model prediction confidence, as shown in Figure 4.10. Similar to the definition of prediction explanation, we consider this feature explains predictive uncertainty. Furthermore, given a ranking of input word contributions produced by an explanation method, we name top-ranked words as *important words*, explaining model predicted labels; and bottom words (with negative contributions) as *uncertain words*, explaining model predictive uncertainty. In other words, a comprehensive prediction explanation should consist of *label explanation* with important words and *uncertainty explanation* with uncertain words. As mentioned before, the goal of this study is to demonstrate the benefits of comprehensive explanations and the necessity of including uncertainty explanations. In this work, we focus on extracting uncertain words from *existing* explanation methods, with the expectation of stimulating further research on explaining predictive uncertainty in NLP.

4.3.2 Explanation Methods

With the previous discussion, we adopt two perturbation-based explanation methods, Leave-one-out [48] and Sampling Shapley [187], for uncertainty explanations. Other explanation methods can be easily adapted to explaining predictive uncertainty.

Leave-one-out (LOO). This method evaluates the effect of each word on model prediction by leaving it out and observing the output probability change on the predicted class. We define a contribution score for each word as

$$S_i = f_{\hat{y}}(\boldsymbol{x}) - f_{\hat{y}}(\boldsymbol{x}_{\setminus i}), \tag{4.20}$$

where $\boldsymbol{x}_{\setminus i}$ denotes the input with the word $\boldsymbol{x}_i$ removed. The contribution score S_i quantifies how much the model prediction confidence decreases when $\boldsymbol{x}_i$ is left out.

Sampling Shapley (SS). Leave-one-out is simple but ignores coalitions between words when quantifying their contributions. The Shapley value [50] stems from coalitional game theory provides an axiomatic solution to attribute the contribution of each word in a fair way. However, the exponential complexity ($O(2^N)$) of computing Shapley value is intractable. Sampling Shapley [187] provides a solvable approximation to Shapley value via sampling. This method computes feature contributions in a more sophisticated way by considering coalitions between words. Specifically, for a word $\boldsymbol{x}_i$, its contribution score is computed as

$$S_i = \frac{1}{M} \sum_{m=1}^{M} f_{\hat{y}}(\boldsymbol{x}_{\setminus i}^{(m)} \cup \{x_i\}) - f_{\hat{y}}(\boldsymbol{x}_{\setminus i}^{(m)}), \tag{4.21}$$

where M is the number of samples, and $\boldsymbol{x}_{\setminus i}^{(m)} \subseteq \boldsymbol{x}_{\setminus i}$ contains a subset of words in $\boldsymbol{x}_{\setminus i}$. The contribution score quantifies the overall contribution of the word $\boldsymbol{x}_i$ to the predicted label over M ensembles. In experiments, we set $M = 200$.

For each prediction, both methods produce an explanation with input word contributions, from which we extract important and uncertain words as label and uncertainty explanations respectively.

Datasets	L	#*train*	#*dev*	#*test*	Label distribution
IMDB	231	20K	5K	25K	Positive: *train*(10036), *dev*(2414), *test*(12535) Negative: *train*(9956), *dev*(2583), *test*(12451)
Toxics	68	96K	32K	32K	Toxic: *train*(9245), *dev*(3069), *test*(3048) Nontoxic: *train*(86447), *dev*(29059), *test*(28818)
Politics	34	70K	7.8K	19K	Democratic: *train*(36222), *dev*(3982), *test*(10240) Republican: *train*(33796), *dev*(3789), *test*(9189)

Table 4.11: Summary statistics of the datasets, where L is average sentence length, and # counts the number of examples in the *train/dev/test* sets. For label distribution, the number of examples with a specific label in *train/dev/test* is noted in bracket.

Models	IMDB	Toxics	Politics
BERT	91.29	96.96	91.20
RoBERTa	93.36	96.75	91.32

Table 4.12: Prediction accuracy (%) of different models on the test sets.

4.3.3 Setup

Models and datasets. We evaluate two pre-trained language models, BERT [13] and RoBERTa [105], on three tasks, including sentiment analysis, toxic comments detection and political bias classification. For sentiment analysis, we utilize the IMDB [138] dataset which contains positive and negative movie reviews. For toxic comments detection, we test on the Wikipedia Toxicity Corpus (Toxics) [188]. The task is to detect whether a comment is toxic or nontoxic. For political bias classification, we adopt the Senator Tweets dataset (Politics) [5], which collects all tweets made by US senators during 2021-2022. The task is to recognize the political bias of each tweet as Democratic or Republican. Table 4.11 shows the statistics of the datasets. We fine-tune the models on the three datasets and report their prediction performance in Table 4.12.

Posterior calibration. A common way of measuring predictive uncertainty is by calibrating model outputs with the true correctness likelihood, so that the predictive probabilities well represent the confidence of model predictions being correct [181, 182, 70, 113]. Lower prediction confidence indicates higher uncertainty [71, 183]. We follow the post-calibration methods and adopt the temperature scaling [181, 113] to calibrate the pre-trained language models (BERT and RoBERTa) in our experiments.

Specifically, we use the development set to learn a temperature T which corrects model output probabilities by dividing non-normalized logits before the softmax function. Then the learned T is

[5]`https://huggingface.co/datasets/m-newhauser/senator-tweets`

Models	IMDB	Toxics	Politics
BERT:			
T	4.59	1.95	4.2
pre-ECE	8.45	2.36	8.56
post-ECE	2.85	0.89	3.83
RoBERTa:			
T	2.76	2.16	3.98
pre-ECE	6.36	2.90	8.45
post-ECE	2.50	1.13	4.29

Table 4.13: Posterior calibration results. T is the learned temperature. pre-ECE and post-ECE represent the ECEs on test sets before and after calibration respectively.

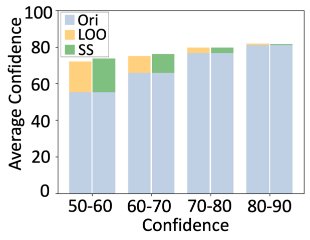
applied to modify model outputs on the test set. In experiments, we linearly search for an optimal temperature T between $[0, 10]$ with a granularity of 0.01, which empirically performs well. We evaluate model calibration with Expected Calibration Error (ECE) [181]. The ECE measures the difference between prediction confidence and accuracy, i.e.

$$ECE = \sum_{k=1}^{K} \frac{|B_k|}{n} |acc(B_k) - conf(B_k)|, \tag{4.22}$$

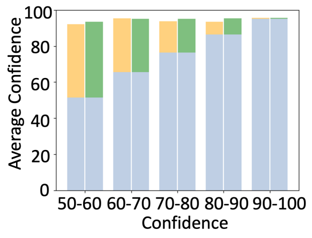
where the total n predictions are partitioned into K equally-spaced bins, B_k represents the predictions fall into the kth bin, $acc(\cdot)$ and $conf(\cdot)$ compute the average accuracy and confidence in each bin respectively. For a perfect calibration, $acc(B_k) = conf(B_k)$, $k \in \{1, \ldots, K\}$. In this work, we set $K = 10$. We report the learned temperature scalars and ECEs before and after calibration in Table 4.13. Temperature scaling performs effectively in decreasing model calibration errors. This enables us to further explain prediction uncertainty based on calibrated confidence. We apply temperature scaling to correct model outputs in experiments.

4.3.4 Experiments

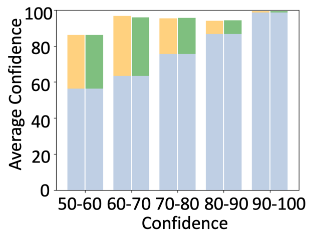
In our experiments, we focus on the three research questions: (1) How effectively existing model explanation methods can identify uncertain words? (2) What insights we can obtain from uncertainty explanations in addition to label explanations? (3) Whether users appreciate uncertainty explanations in understanding model prediction behavior?



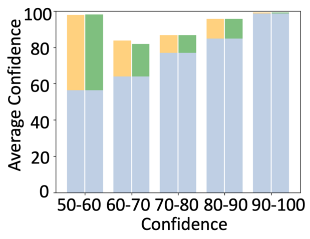
(a) BERT, IMDB



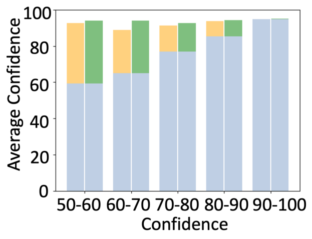
(b) RoBERTa, IMDB



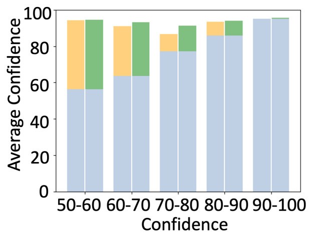
(c) BERT, Toxics



(d) RoBERTa, Toxics



(e) BERT, Politics



(f) RoBERTa, Politics

Figure 4.11: Average confidence (%) changes with uncertain words removed. X-axis shows different bins of original confidence. Ori: original confidence; LOO: Leave-one-out; SS: Sampling Shapley.

Quantitative Evaluation

For each dataset, we randomly select 1000 test examples and generate explanations for model predictions on them (see visualizations in Table 4.16). The following two results answer the research question (1) and (2) respectively.

Existing model explanation methods effectively identify uncertain words that limit model prediction confidence. We extract top k uncertain words identified by model explanations and remove them from inputs and then compute the average prediction confidence change in each bin of original confidence. We empirically set $k = 10$ for IMDB and $k = 5$ for Toxics and Politics based on their average sentence lengths in Table 4.11. Figure 4.11 shows that both LOO and SS capture uncertain words that limit prediction confidence. Overall, SS performs better than LOO in identifying uncertain words.

Important words and negations can result in uncertain predictions. We analyze feature statistics of model explanations via local mutual information (LMI) [189, 190]. LMI quantifies the association between a feature (an important/uncertain word) and a prediction label in model explanations.

To understand which features contribute to model predictions and which features cause prediction uncertainty, we follow [189, 190] and analyze feature statistics of model explanations via local mutual

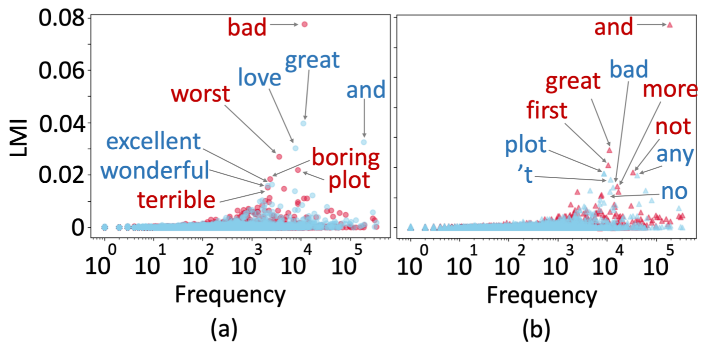


Figure 4.12: LMI distributions based on important words (a) and uncertain words (b). The x-axis represents word frequency in the vocabulary built on the IMDB dataset. We use blue and red colors to distinguish features associated with the POSITIVE and NEGATIVE labels respectively. Top 5 tokens in each distribution are pointed out.

information (LMI). LMI quantifies the association between a feature and a prediction label in model explanations. We compute LMI based on top 5 important and uncertain words in prediction and uncertainty explanations respectively. Specifically, for each group of features, we can get a set of unique features, $E = \{e\}$. The LMI between a feature e and a prediction label y is

$$\mathrm{LMI}(e, y) = p(e, y) \cdot \log\left(\frac{p(y \mid e)}{p(y)}\right), \tag{4.23}$$

where $p(y \mid e) = \frac{count(e,y)}{count(e)}$, $p(y) = \frac{count(y)}{|E|}$, $p(e, y) = \frac{count(e,y)}{|E|}$, and $|E|$ is the number of occurrences of all features in E. Then we can get a distribution of LMI over all tokens in the vocabulary ($\{w\}$) built on the dataset, i.e.

$$P_{\mathrm{LMI}}(w, y) = \begin{cases} \mathrm{LMI}(w, y) & \text{if token } w \in E \\ 0 & \text{else} \end{cases} \tag{4.24}$$

We normalize the LMI distribution by dividing each value with the sum of all values. Table 6.12 records top 10 tokens in different LMI distributions of model explanations. Leave-one-out captures more noisy tokens (e.g., special tokens, punctuations) than Sampling Shapley. Both BERT and RoBERTa focus on some task-irrelevant features (e.g., stop words) to make predictions, which reveals the problem of model prediction behavior. We leave this problem to our future work.

We analyze explanations generated by SS for RoBERTa on the IMDB dataset. Figure 4.12 shows LMI distributions based on important and uncertain words in explanations respectively. Some important words for model predictions on a specific label (e.g., `great` for POSITIVE, `bad` for NEGATIVE in (a)) become uncertain words for the other label in (b). This indicates models may get confused by

Model	Dataset	LOO		SS	
		Label	Unc	Label	Unc
BERT	IMDB	63.33	**86.67**	67.50	**87.50**
	Toxics	60.00	**86.67**	86.67	**90.00**
	Politics	66.67	**83.33**	82.50	75.00
RoBERTa	IMDB	66.67	**83.33**	80.00	**86.67**
	Toxics	80.00	**83.33**	80.00	**86.67**
	Politics	63.33	60.00	75.00	65.00

Table 4.14: Human prediction performance (%) on label explanations (Label) and uncertainty explanations (Unc).

important words corresponding to different labels in inputs. Besides, negation words (e.g., `not`, `no`) pointed out in (b) are not shown in (a), which means they may not be used by models for making predictions but can highly cause model predictive uncertainty. We observe similar results on other datasets in Table 4.15.

Human Evaluation

To answer the research question (3), we conduct human evaluation on both important and uncertain words in model explanations through the Amazon Mechanical Turk (AMT).

We conduct human evaluation on both important and uncertain words in model explanations through the Amazon Mechanical Turk (AMT). For each dataset, we randomly select 30 test examples to generate explanations for each pre-trained language model. Each explanation (with 2-3 important and uncertain words extracted respectively) is assessed by 5 workers. We pay the workers $0.3 for assessing each explanation. We have collected 900 annotations in total.

For each explanation, we ask the worker to answer the following 5 questions:

1. **Prediction on label explanations (multiple choices)**: Given the model input text, can you guess the model prediction label based on the highlighted tokens?

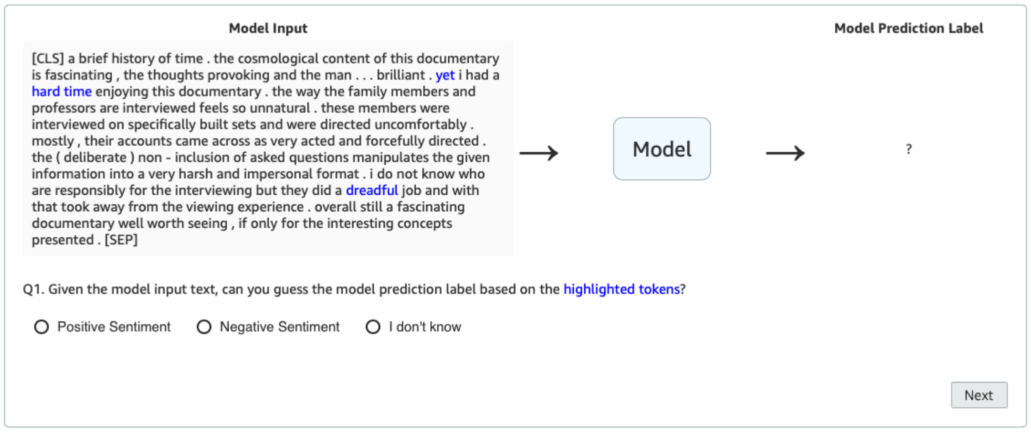
2. **Rating on label explanations (1-5 Liker scale)**: Given the model input text and model prediction label, how much do you think the highlighted tokens make sense to you?

3. **Prediction on uncertainty explanations (multiple choices)**: Given the model input text and model prediction probability, do you think removing the highlighted tokens can further increase the model prediction probability or not?

4. **Rating on uncertainty explanations (1-3 Liker scale)**: How much do you think the current model prediction probability could be changed by removing the highlighted tokens?

5. **Comparison on label explanations and uncertainty explanations (multiple choices)**: Which type of model explanations can help you better understand the model prediction?

Figure 4.13 and Figure 4.14 show the interfaces of human evaluation on Q1 and Q3 respectively.



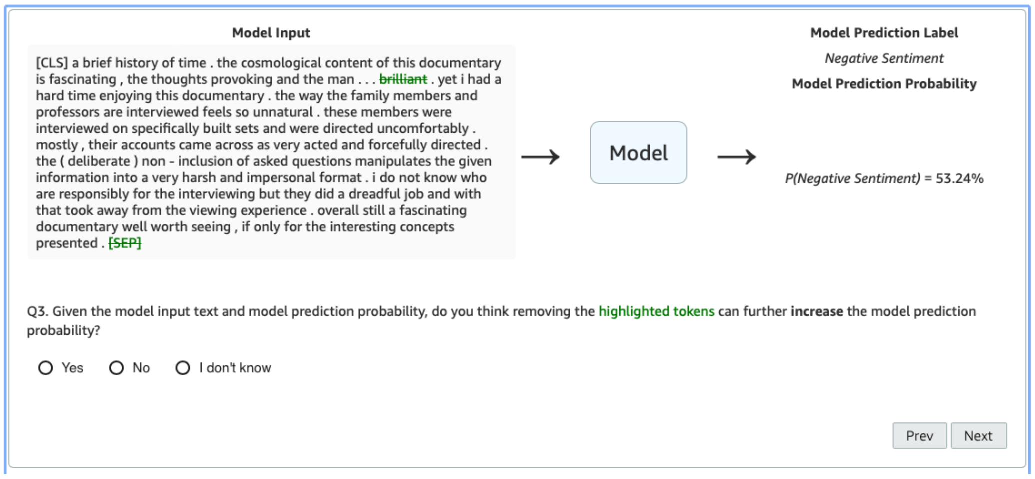
Model Input

[CLS] a brief history of time . the cosmological content of this documentary is fascinating , the thoughts provoking and the man . . . brilliant . yet i had a hard time enjoying this documentary . the way the family members and professors are interviewed feels so unnatural . these members were interviewed on specifically built sets and were directed uncomfortably . mostly , their accounts came across as very acted and forcefully directed . the (deliberate) non - inclusion of asked questions manipulates the given information into a very harsh and impersonal format . i do not know who are responsibly for the interviewing but they did a dreadful job and with that took away from the viewing experience . overall still a fascinating documentary well worth seeing , if only for the interesting concepts presented . [SEP]

Q1. Given the model input text, can you guess the model prediction label based on the highlighted tokens?

Positive Sentiment Negative Sentiment I don't know

Figure 4.13: Interface of human evaluation on important words highlighted in blue color.



Model Input

[CLS] a brief history of time . the cosmological content of this documentary is fascinating , the thoughts provoking and the man . . . ~~brilliant~~ . yet i had a hard time enjoying this documentary . the way the family members and professors are interviewed feels so unnatural . these members were interviewed on specifically built sets and were directed uncomfortably . mostly , their accounts came across as very acted and forcefully directed . the (deliberate) non - inclusion of asked questions manipulates the given information into a very harsh and impersonal format . i do not know who are responsibly for the interviewing but they did a dreadful job and with that took away from the viewing experience . overall still a fascinating documentary well worth seeing , if only for the interesting concepts presented . ~~[SEP]~~

Model Prediction Label

Negative Sentiment

Model Prediction Probability

P(Negative Sentiment) = 53.24%

Q3. Given the model input text and model prediction probability, do you think removing the highlighted tokens can further **increase** the model prediction probability?

Yes No I don't know

Figure 4.14: Interface of human evaluation on uncertain words highlighted in green color.

The following two observations illustrate the effectiveness and indispensability of uncertainty explanations.

Humans perform better on understanding uncertainty explanations than label explanations. First, we provide inputs with important words highlighted and ask evaluators to guess model prediction labels. Then we show model predictions with confidence and ask evaluators whether removing uncertain words can improve prediction confidence or not. Table 4.14 shows the results of human performance on predicting model prediction labels and confidence change. Overall, humans have better performance on understanding model predictive uncertainty based on uncertain words. This indicates the effectiveness of uncertainty explanations in helping users understand model predictions. Besides, SS produces more understandable explanations to humans than LOO. This is also reflected by the evaluation results where evaluators score (from 1-5) the quality of explanations with the average values 3.7 and 4.0 for LOO and SS respectively.

Humans prefer to see uncertainty explanations in addition to label explanations. We ask evaluators to vote whether they want to include uncertainty explanations in addition to label explanations for understanding model decision making. Most (71%) evaluators prefer to see uncertainty explanations. Besides, evaluators mark 72.6% of uncertainty explanations identify the words that largely limit model prediction confidence. This implies that uncertainty explanations are indispensable to explaining model prediction behavior.

Table 4.16 shows visualizations of different model explanations with both important and uncertain words highlighted.

4.3.5 Related Work

The problem of predictive uncertainty estimation has been well studied [191, 192, 193, 194, 195, 182, 183]. However, little is known about what causes predictive uncertainty. Extensive literatures on model explanations focus on explaining model predicted labels, while ignoring predictive uncertainty [1, 24, 25, 185, 186]. However, explaining predictive uncertainty is an important complement to explaining predicted labels for improving model trustworthiness [184, 196].

There is limited work on studying the source of predictive uncertainty in NLP. For example, previous works on explaining uncertainty estimates mainly focus on tabular and image data [184, 197, 196]. Feng et al. [198] observed that prediction confidence increases with input reduction, while focusing on model pathologies as reduced inputs lack predictive information. Differently, we focus on identifying uncertain words in inputs for explaining model predictive uncertainty. To the best of our knowledge, this is the first work on explaining predictive uncertainty of pre-trained language models in NLP.

4.3.6 Conclusion

In this paper, we propose to explain model prediction uncertainty by extracting uncertain words from existing model explanations. We adopt two explanation methods to explain BERT and RoBERTa on three tasks, including sentiment analysis, toxic comments detection, and political bias classification.

Experiments show the effectiveness of uncertainty explanations in explaining models and helping humans understand model predictions.

Model	Dataset	Label	Leave-one-out		Sampling Shapley	
			Important	Uncertain	Important	Uncertain
BERT	IMDB	Pos	this great best film a good excellent and it wonderful	i movie this was to just the would but not	great best and excellent love wonderful good this very enjoyed	movie just would bad but nothing not could off plot
		Neg	this worst movie bad not but no terrible just nothing	but is and not the it a this 't great	bad worst this movie just boring terrible awful not nothing	and great very it is good in not his seen
	Toxics	Tox	you fuck hell fucking bullshit idiot dick suck stupid gay	the are so fuck of good have wow love for	you fuck hell gay fucking bullshit idiot dick suck stupid	the can in please if so certainly because know help
		NTox	please to i if not the of wikipedia can is thank	you your i the and a to please me is	please the can to if for in of thank use	you a i your me the my it van and
	Politics	Dec	and to climate must this child our so the in	the and to of we in american is americans i	this must climate that health now today more every to	the a is for american back ensure work and not
		Rep	democrats the is border bid great and communist inflation fox	to and the i this our my with in for	the a is bid and border for communist his fox	that this to more must you today every now your
RoBERTa	IMDB	Pos	this best and great not but good I film is	the not I is a for this and no was	great and love excellent wonderful best very amazing brilliant perfect	any plot 't bad no nothing movie much never this
		Neg	this not bad worst boring just the even and no	bad 't not and plot butwas a to me	bad worst plot boring terrible nothing stupid much no waste	and great first not more special very love life moments
	Toxics	Tox	you fuck stupid HELL suck Fuck YOU You fucking shit	to but if or Go an reported thanks ipedia should	you fuck You stupid HELL suck Fuck YOU fucking shit	to for reported OF but the about in help need
		NTox	to the Please article of please for Thank and in	you your is I a Your vandal not my me	to the for article use in please Please of If	you your a me is my You vandal I are
	Politics	Dec	and the to this in our a must for will	the americ an in to for is act this a	this and climate care child today that workers how families	americ is the will of back family they not would
		Rep	bid en is democr the border americ us great to	to the and our my i of in this a	americ border democr bid is spending great not inflation would	and this that to our it more my you families

Table 4.15: Top 10 tokens in different LMI distributions of model explanations. Important: statistics of top salient words in explanations; Uncertain: statistics of bottom salient words in explanations; Pos: postive; Neg: negative; Tox: toxic; NTox: nontoxic; Dec: democratic; Rep: republican. Warning: this table contains toxic tokens.

Model/Dataset	Method	Prediction	Explanation
BERT/IMDB	LOO	Negative (0.69 → 0.80)	i found it to be a complete disappointment . if i had of known this movie was going to be as stupid as it was , i would have stayed home and done something more entertaining ... the plot was a great idea , just could have been done in a much better way .
RoBERTa/IMDB	SS	Positive (0.51 → 0.86)	Not the best of the Lone Star series, but it moves along quickly with good performances. Introduced as "Singin' Sandy" in the main title, John Wayne as a 'singing cowboy' isn't successful...
BERT/Toxics	LOO	Nontoxic (0.83 → 0.97)	oh , and i have a question . why was the article on brad christian , a famous magician , deleted because of vandalism instead of simply restored ? i believe that many users on this site are biased towards magicians . i have come to the conclusion that wiki is a useless site that does nothing to help anyone . you are welcome to ban me longer , and i understand completely if you do , but this site is the worst piece of garbage i have ever found !
RoBERTa/Toxics	SS	Nontoxic (0.71 → 0.94)	You are so full of shit. First of all, you aren't an admin, and for the sake of this site I hope you never will be. I know I will personally work against you if you ever decide to try for one. But I digress as you are not an administrator, and especially since you have no access to checkuser, you cannot determine who is or is not a sockpuppet nor do you have the authorization to place a tag on a user page .
BERT/Politics	SS	Democratic (0.64 → 0.95)	fantastic news . star plastics was founded in ravenswood and is continuing to invest in west virginia . this expansion will lead to economic development and growth in jackson county , and shows that wv is the perfect place for companies large and small .
RoBERTa/Politics	LOO	Republican (0.85 → 0.94)	happy national day , taiwan. your commitment to democracy and market economics is an effective model that can be relied upon to solve our collective problem .

Table 4.16: Visualizations of prediction explanations for different models on different datasets, where top two important and uncertain words are highlighted in blue and red colors respectively. The prediction confidence changes are shown in brackets when the highlighted uncertain words are removed. LOO: Leave-one-out; SS: Sampling Shapley. Warning: some examples may be offensive or upsetting.

Chapter 5

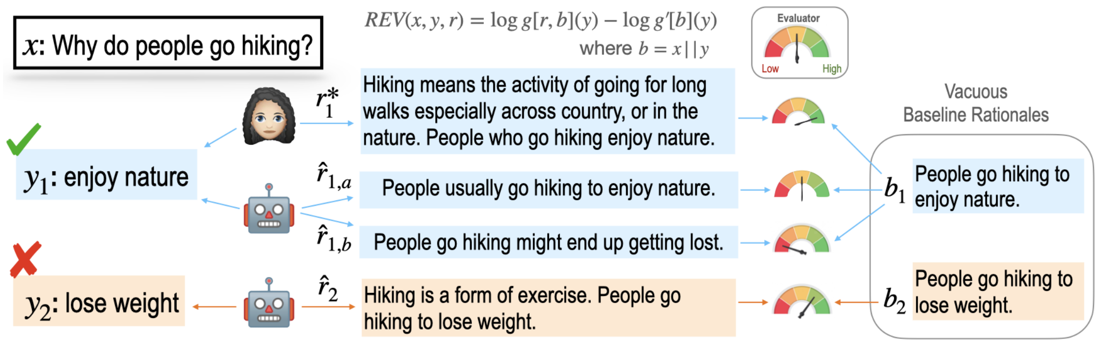
Evaluating Model Explanations

Free-text rationale (or natural language explanation) is a promising step towards explainable NLP, yet their evaluation remains an open research problem. Existing metrics have mostly focused on measuring the association between the rationale and a given label. I argue that an ideal metric should focus on the new information uniquely provided in the rationale that is otherwise not provided in the input or the label. In this chapter, I investigate this research problem from an information-theoretic perspective using conditional $\mathcal{V}$-information [4]. I propose a metric called REV ($\underline{\text{R}}$ationale $\underline{\text{E}}$valuation with conditional $\underline{\mathcal{V}}$-information), that can quantify the new information in a rationale supporting a given label *beyond* the information already available in the input or the label.

5.1 REV: Information-Theoretic Evaluation of Free-Text Rationales

Model explanations have been indispensable for trust and interpretability in natural language processing (NLP) [1, 199, 17, 185, 186]. Free-text rationales, which explain a model prediction in natural language, have been especially appealing due to their flexibility in eliciting the reasoning process behind the model's decision making [42, 87, 88, 46, 89], making them closer to human explanations. However, current automatic evaluation of free-text rationales remains narrowly focused. Existing metrics primarily measure the extent to which a rationale can help a (proxy) model predict the label it explains (i.e., accuracy based) [90, 91]. Yet, these metrics offer little understanding of the *new information* contained in the rationale, as added to the original input, that could *explain why the label is selected*—the very purpose a rationale is designed to serve. For instance, the two rationales

r_1^* and $\hat{r}_{1,a}$ in Fig. 5.1 would be considered equally valuable under existing metrics, even though they supply different amount of novel and relevant information.



x: Why do people go hiking?
$REV(x, y, r) = \log g[r, b](y) - \log g'[b](y)$
where $b = x || y$
Evaluator
Low
High
y_1: enjoy nature
y_2: lose weight
r_1^*
$\hat{r}_{1,a}$
$\hat{r}_{1,b}$
$\hat{r}_2$
Hiking means the activity of going for long walks especially across country, or in the nature. People who go hiking enjoy nature.
People usually go hiking to enjoy nature.
People go hiking might end up getting lost.
Hiking is a form of exercise. People go hiking to lose weight.
Vacuous Baseline Rationales
b_1 People go hiking to enjoy nature.
b_2 People go hiking to lose weight.

Figure 5.1: Our evaluation framework for different free-text rationales (r). r_1^* is a human-written rationale, $\hat{r}_{1,a}$ and $\hat{r}_{1,b}$ are two generated rationales for the true label y_1. Our metric, REV, based on CVI [4] is able to distinguish all three rationales by measuring how much new and relevant information each adds over a vacuous rationale, b; performance-based evaluations can only distinguish between $\hat{r}_{1,a}$ and $\hat{r}_{1,b}$. For an (arguably) incorrect label, y_2, REV still gives a positive score highlighting that $\hat{r}_2$ is able to provide new information for why it supports y_2. Prediction accuracy can be augmented with REV to provide a fuller interpretability of model decisions.

In this paper, we overcome this shortcoming by introducing an automatic evaluation for free-text rationales along two dimensions: (1) whether the rationale supports (i.e., is predictive of) the intended label, and (2) how much *new information* does it provide to justify the label, **beyond** what is contained in the input. For example, rationale $\hat{r}_{1,b}$ in Fig. 5.1 violates (1) because it is not predictive of the label, "enjoy nature". Rationale $\hat{r}_{1,a}$ does support the label but contains no new information that justifies it, *beyond* what is stated in the input x; thus, it violates (2). Rationale r_1^* is satisfied along both dimensions: it supports the label and does so by providing new and relevant information, beyond what is in the input. Our proposed evaluation is designed to penalize both $\hat{r}_{1,a}$ and $\hat{r}_{1,b}$, while rewarding rationales like r_1^*.

We introduce REV[1], which adapts an information-theoretic framework from Xu et al. [200] for evaluating free-text rationales along the two dimensions mentioned above. Specifically, REV is based on conditional $\mathcal{V}$-information [4], which quantifies the degree of information contained in a representation, *beyond* another (baseline) representation, accessible to a model family, $\mathcal{V}$. As our baseline representation, we consider any vacuous rationale which simply combines an input with a given label, without providing any new information relevant to answering why the label was chosen. REV adapts conditional $\mathcal{V}$-information to evaluate rationales, where the representation is obtained via an evaluation model trained to produce a label given the rationale. Other metrics do not

[1]For Rationale Evaluation with conditional $\mathcal{V}$-information.

take into consideration vacuous rationales, and are hence unable to measure new and label-relevant information in rationales.

Our experiments present evaluations with REV for rationales under two reasoning tasks, commonsense question-answering (CQA) [40] and natural language inference (NLI) [41], across four benchmarks. Several quantitative evaluations demonstrate the capabilities of REV in providing evaluations along new dimensions for free-text rationales, while being more consistent with human judgements compared to existing metrics. We also provide comparisons to demonstrate the sensitivity of REV to various degrees of input perturbations. Additionally, our evaluation with REV offers insights into why rationales obtained through chain-of-thought prompting [201] do not necessarily improve prediction performance.

5.1.1 REV: Information-Theoretic Evaluation of Rationales

We introduce a new metric, REV, Rationale Evaluation with conditional $\mathcal{V}$-information, for evaluation of free-text rationales on the proposed dimensions (§5.1.3), based on the information-theoretic framework of conditional $\mathcal{V}$-information (§5.1.2).

We consider the setting where we have input $X \in \mathcal{X}$, label $Y \in \mathcal{Y}$, and free-text rationale $R \in \mathcal{R}$ generated for label Y. A common strategy to evaluate rationales R is through an evaluator $f \in \mathcal{V}$ based on how much R helps f predict Y given X. The evaluator $f : Z \rightarrow Y$ maps a variable Z to a label distribution. The definition of Z depends on the evaluation framework; e.g., Z can be a concatenation of X and R. The evaluator f is trained on a set of input, label and rationale triples $\mathcal{D}_{\text{train}} = \{(x_j, y_j, r_j)\}$, and applied to $\mathcal{D}_{\text{test}} = \{(x_i, y_i, r_i)\}$ for evaluation. The utility of R is formulated as the difference between the performance of the evaluator on predicting Y with R, and without it, i.e.

$$\text{Perf}[f(Y|X, R)] - \text{Perf}[f(Y|X)]. \tag{5.1}$$

A larger performance gap indicates a better rationale. Existing metrics [90, 91] compute the performance gap based on prediction accuracies, measuring how much R can help the evaluator correctly predict Y given X.

However, accuracy-based evaluation can only indicate whether or not a rationale is predictive of a label, but cannot quantify how much *new information the rationale provides to justify the label.* Fig. 5.1 illustrates this issue via an example. Accuracy-based evaluation can distinguish between $\hat{r}_{1,a}$ and $\hat{r}_{1,b}$ since $\hat{r}_{1,a}$ supports y_1 and $\hat{r}_{1,b}$ does not. However, it is unable to distinguish between r_1^* and $\hat{r}_{1,a}$ (since both are predictive of y_1), despite the fact that $\hat{r}_{1,a}$ does not provide any unique and relevant information to answer why the label should be y_1. In practice, vacuous rationales such as $\hat{r}_{1,a}$ are commonly seen in model generations [93, 202]. This calls for an evaluation metric which is able to identify and penalize such vacuous rationales.

5.1.2 An Information-Theoretic Perspective on Rationale Evaluation

The key quantity of interest for our evaluation of rationale R is the amount of new information expressed in R (e.g., background knowledge, reasoning process) that can justify a label Y. The mutual information between R and Y, $I(Y;R)$, can be helpful for evaluating this quantity. However, we are not interested in the information that is already captured in the input X. A **vacuous** rationale, such as $\hat{r}_{1,a}$ in Fig. 5.1, which simply combines the input X and the label, Y, captures all the information in X and Y without specifying any new information to help understand why Y has been chosen for X; let us denote such rationales as $B \in \mathcal{B}$. Thus, we argue that a good evaluation metric must be able to measure the amount of relevant, new information contained in a rationale *beyond* what is contained in any vacuous rationale, B, that leads to the prediction of Y. Then the new information in R beyond what is available in B can be grounded with conditional mutual information [203] as follows,

$$I(Y;R \mid B) = I(Y;R,B) - I(Y;B), \tag{5.2}$$

where the difference of two information quantities demonstrates the performance gap in Equation 5.1. Directly computing mutual information, however, is challenging because true distributions of random variables are usually unknown, and we do not have unbounded computation. A recently introduced information-theoretic framework called $\mathcal{V}$-information circumvents this by restricting the computation to certain predictive model families, $\mathcal{V}$ [200]. Our approach to evaluate rationales extends this framework, following [4], as described below.

Conditional $\mathcal{V}$-information. Given a model family $\mathcal{V}$ that maps two random variables R and Y, $\mathcal{V}$-information defines the usable information that can be extracted from R by models in $\mathcal{V}$ to predict Y, i.e. $I_{\mathcal{V}}(R \to Y)$. If $\mathcal{V}$ generalizes to the set of all possible functions, then $\mathcal{V}$-information is mutual information [203]. In practice, it is feasible to estimate the usable information from R about Y by selecting any neural model without frozen parameters as $\mathcal{V}$.[2]

[2]Please see [200] for a detailed discussion of properties such as optional ignorance that a predictive family $\mathcal{V}$ must follow.

Following conditional mutual information in information theory [204], $\mathcal{V}$-information has been extended to conditional $\mathcal{V}$-information (CVI) [4]. CVI quantifies the $\mathcal{V}$-usable information in R about Y conditioned on a variable B, i.e.

$$I_{\mathcal{V}}(R \rightarrow Y \mid B) = H_{\mathcal{V}}(Y \mid B) - H_{\mathcal{V}}(Y \mid R, B). \tag{5.3}$$

Here B is any vacuous rationale that leads to the prediction of Y. In this work, we consider B simply as the combination of X and Y. We leave analyzing how different baseline construction impacts our metric to future work. $H_{\mathcal{V}}(\cdot \mid \cdot)$ is the conditional $\mathcal{V}$-entropy [200, 4, 205], defined as

$$\begin{aligned} H_{\mathcal{V}}(Y \mid B) &= \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f[b](y)] \\ H_{\mathcal{V}}(Y \mid R, B) &= \inf_{f \in \mathcal{V}} \mathbb{E}[-\log f[r,b](y)], \end{aligned} \tag{5.4}$$

where $f[b]$ and $f[r,b]$ produce a probability distribution over the labels given b and $[r,b]$ as inputs respectively. Further, we consider pointwise CVI for evaluating individual samples, (r, y, b) as

$$PI_{\mathcal{V}}(r \rightarrow y \mid b) = \log g[r,b](y) - \log g'[b](y), \tag{5.5}$$

where $g \in \mathcal{V}$ s.t. $\mathbb{E}[-\log g[r,b](y)] = H_{\mathcal{V}}(Y \mid R, B)$ and $g' \in \mathcal{V}$ s.t. $\mathbb{E}[-\log g'[b](y)] = H_{\mathcal{V}}(Y \mid B)$.

Properties of conditional $\mathcal{V}$-information. As proved by Hewitt et al. [4], CVI has several useful properties:

1. *Non-Negativity*: $I_{\mathcal{V}}(R \rightarrow Y \mid B) \geq 0$.

2. *Independence*: If Y and B are jointly independent of R, then $I_{\mathcal{V}}(R \rightarrow Y \mid B) = 0$.

3. *Monotonicity*: If $\mathcal{U} \subseteq \mathcal{V}$, then $H_{\mathcal{V}}(Y \mid B) \leq H_{\mathcal{U}}(Y \mid B)$.

An implication from *Monotonicity* is complex models (e.g., pre-trained language models) can do better than simpler ones (e.g., linear models) in estimating $\mathcal{V}$-usable information. Experiments in Section 5.3 show the amount CVI varies across different model architectures, while strong models usually capture more usable information. Since CVI measures the additional $\mathcal{V}$-usable information in R about Y beyond what's already extracted from B by models in $\mathcal{V}$, it grounds the goal of the proposed metric REV.

5.1.3 Computing REV for Rationale Evaluation

Building on the framework of CVI, we propose a new metric REV, for Rationale Evaluation with conditional $\underline{\mathcal{V}}$-information. We compute REV over a given test set, $\mathcal{D}_{\text{test}} = \{(x_i, y_i, r_i)\}$, by estimating CVI over the set with evaluators. For a test example (x, y, r), the REV score denoted as $\text{REV}(x, y, r)$ is computed based on Equation 5.5, where b is constructed by combining x and y.

$$\text{REV}(x, y, r) = PI_{\mathcal{V}}(r \to y \mid b) \tag{5.6}$$

The REV score for the test corpus $\mathcal{D}_{\text{test}}$, is given by the average pointwise REV score:

$$\text{REV} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_i \text{REV}(x_i, y_i, r_i). \tag{5.7}$$

Algorithm 2 shows the process of computing both pointwise and aggregate REV scores. The higher the REV score, the more additional (*new* and *relevant*) information the rationale r contains to explain the label beyond the baseline rationale b. $\text{REV}(x, y, r)$ can take positive, negative, or zero values. When $\text{REV}(x_i, y_i, r_i) > 0$, the rationale supplies additional information for supporting the label (e.g., r_1^* in Fig. 5.1); when $\text{REV}(x_i, y_i, r_i) = 0$, the rationale provides no additional information beyond the baseline (e.g., $\hat{r}_{1,a}$ in Fig. 5.1); and when $\text{REV}(x_i, y_i, r_i) < 0$, the rationale does *not* support the label (e.g., $\hat{r}_{1,b}$ in Fig. 5.1).

Algorithm 2 Computing REV Scores

```
1: Input: evaluation models $g$ and $g'$, test set $\mathcal{D}_{\text{test}} = \{(x_i, y_i, r_i)\}$
2: Initialize an empty list $\mathcal{S}$
3: for $(x_i, y_i, r_i) \in \mathcal{D}_{\text{test}}$ do
4:     Construct the baseline rationale $b_i$
5:     $\text{REV}(x_i, y_i, r_i)$
       $= \log g[r_i, b_i](y_i) - \log g'[b_i](y_i)$
6:     $\mathcal{S}$.add($\text{REV}(x_i, y_i, r_i)$)
7: end for
8: $\text{REV} = \text{sum}(\mathcal{S})/|\mathcal{S}|$
9: Output: $\mathcal{S}$, REV
```

Task	Input	Label	Baseline Rationale
CQA	Where can personal mushrooms be kept fresh?	refrigerator	Personal mushrooms can be kept fresh in the refrigerator.
NLI	Premise: A dog running in the surf. Hypothesise: A dog is at the beach.	entailment	A dog running in the surf indicates a dog is at the beach.

Table 5.1: Examples of constructed baseline rationales for CQA and NLI tasks.

Note that REV can assign a positive score to a rationale for an incorrect prediction as long as the rationale supports it and provides additional information beyond a vacuous baseline rationale (e.g., $\hat{r}_2$ in Fig. 5.1). Thus, REV cannot be seen as a replacement for prediction accuracy, but rather as an orthogonal metric to interpret the usefulness of a generated rationale for the model decision.

Constructing a Baseline with Vacuous Rationales Given an input x and a label y, we construct a baseline rationale b by converting x and y into a declarative sentence. For the CQA task, we adopt a pre-trained T5-3B model fine-tuned on a set of (*question*, *answer*, *declarative sentence*) tuples [206] [3] annotated by Demszky et al. [207]. For the NLI task, we first use a template to convert (*premise*, *hypothesis*, *label*) tuple into a baseline rationale: "*premise* `implies` / `contradicts` / `is not related to` *hypothesis*". Then we apply a pre-trained model [4] to paraphrase the baseline rationale. This can avoid evaluators to capture the template pattern. Table 5.1 shows some examples of constructed vacuous rationales.

Training evaluation models, g and g' We train two models, g and g', which take $[r, b]$ and b as inputs respectively. [5] In particular, we use pre-trained language models [e.g., T5; 208] and fine-tune them on the training set $\mathcal{D}_{train} = \{(x, y^*, r^*)\}$, where $\{y^*\}$ and $\{r^*\}$ are gold labels and human-annotated rationales, respectively. We construct baseline rationales $\{b^*\}$ based on $\{(x, y^*)\}$. The objective is to maximize the log-likelihood of y^* given $[r^*, b^*]$ or b^*. After training, the evaluation models are applied to evaluate a rationale-label pair (y, r) w.r.t. an input x. The rationale-label pair (y, r) can be model-generated and the label may not be ground-truth (e.g., y_2 in Fig. 5.1), while REV is still able to provide an assessment on the rationale along the two dimensions.

[3] `https://github.com/jifan-chen/QA-Verification-Via-NLI`
[4] `https://huggingface.co/humarin/chatgpt_paraphraser_on_T5_base`
[5] $[r, b]$ is the concatenation of r and b.

5.2 Experimental Setup

We outline our experimental setup by describing the reasoning tasks and datasets (§5.2.1), followed by the task and evaluation models (§5.2.2), and the baseline metrics for comparison (§5.2.3).

5.2.1 Datasets

We explore two reasoning tasks, namely CommonsenseQA (CQA) and Natural Language Inference (NLI) across four datasets, all containing human-annotated free-text rationales. For CQA task, we use ECQA [209], CoS-E (v1.11) [6] [88] and QuaRTz [210]. Both ECQA and CoS-E originate from the CommonsenseQA dataset [40], where each commonsense question is paired with 5 candidate choices and the task is to select an answer from the candidates. ECQA contains higher quality free-text rationales compared to CoS-E, in terms of comprehensiveness, coherence, non-redundancy, etc. [209, 93]. QuaRTz is an open-domain reasoning task about textual qualitative relationships. Each instance contains a situated qualitative question, two answer options and a knowledge statement. The task is to select an answer from the two options to the question based on the textual qualitative knowledge. We use the knowledge statement as a free-text rationale since it explains why the answer is to the question. For NLI task, we use e-SNLI [42] which is an extension of SNLI [41] with augmented free-text human-written rationales. The task is to predict the entailment relationship between a premise and a hypothesis as entailment, contradiction, or neutral. Table 5.2 shows the summary statistics of the four datasets.[7]

Datasets	*#train*	*#dev*	*#test*
ECQA	7598	1090	2194
CoS-E	8766	975	1221
QuaRTz	2696	384	784
e-SNLI	54933	9842	9824

Table 5.2: Summary statistics of the datasets, where # counts the number of examples in the *train/dev/test* sets.

5.2.2 Task and Evaluation Models

Task models We choose T5 Large [208] as the task model (finetuned on ground truth labels and rationales) to produce generated rationale-label pairs under three settings:

[6]We use the version v1.11 where each question is paired with 5 answer choices, for comparison with ECQA.

[7]Since CoS-E does not provide rationales for instances in the test set, we use the original development set as the test set and hold out 10% of training data as the new development set. We follow Hase et al. [90] and randomly sample 10% of training data to form the training set for finetuning our models.

- $XY^*\to R$: Given an input text and the gold label, generate a rationale.
- $X\to YR$: Given an input text, generate a label followed by a rationale. Since T5 decodes tokens sequentially, each R is generated conditioned on the predicted Y.
- $X\to RY$: Given an input text, generate a rationale followed by a label. Here, we compute a likelihood for each candidate Y conditioned on R, and then select the most probable candidate. This operation can improve the model prediction accuracy, while weakening the consistency and relevance between the generated rationales and predicted labels.

After training, we collect three types of rationale-label pairs by applying the three task models on the test set of each dataset. In addition to these three settings, we also evaluate ground-truth labels paired with crowd-sourced rationales ($Y^*;R^*$).

Evaluation models Our evaluation models, g and g' (see Equation 5.6 in §5.1.2), are based on T5 Large trained on gold rationale-label pairs of the respective dataset.

We use Huggingface Transformers [211] to access all task models and evaluators. We train each model for up to 30 epochs with a learning rate $5e-6$ and a batch size 8. All experiments were performed on a single NVIDIA RTX 8000 GPU. Table 5.3 shows input-output formattings of different task models for different tasks.

Type	Input	Output
$XY^*\to R$	CQA: [question] question [choice] choice-1 ... [choice] choice-n [answer] gold label [rationale] NLI: [premise] premise [hypothesis] hypothesis [answer] gold label [rationale]	rationale <eos>
$X\to YR$	CQA: [question] question [choice] choice-1 ... [choice] choice-n [answer] NLI: [premise] premise [hypothesis] hypothesis [answer]	label [rationale] rationale <eos>
$X\to RY$	CQA: [question] question [choice] choice-1 ... [choice] choice-n [rationale] NLI: [premise] premise [hypothesis] hypothesis [rationale]	rationale [answer] label <eos>

Table 5.3: The input-output formatting of different task models.

We apply REV to evaluate different types of free-text rationales w.r.t. labels on the ECQA dataset. Figure 5.2 shows REV scores of the four types of rationale-label pairs evaluated by the four evaluators. The ranking of the four groups of rationale-label pairs is consistent across the four evaluators, i.e. $Y^*;R^* > XY^*\to R > X\to YR > X\to RY$. This ranking is also consistent with human evaluation in §4.3.4. Since ECQA contains high-quality crowdsourced rationales [209], it is expected

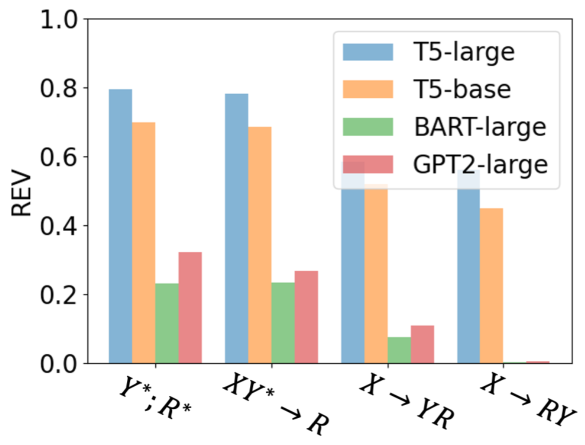


Figure 5.2: REV for evaluating rationale-label pairs on the ECQA dataset with different evaluator architectures.

that the REV of gold rationale-label pairs (Y*;R*) is the highest. The REV of XY*→R is close to that of Y*;R*, indicating the task model (T5 Large) can produce good quality rationales when it is prompted with ground-truth labels. All four evaluators agree that the generated rationales of X→YR contain more additional background information for explaining the predicted labels than those of X→RY. This is consistent with our design of the X→RY in §5.2.3, where the generated rationales and labels have weakened relevance. For each type of rationale-label pairs, the four evaluators capture different amount of conditional $\mathcal{V}$-information, while T5 Large consistently outperforms other three models. In the reported experiments §5.3, we use T5 Large as the evaluator.

5.2.3 Other Metrics for Rationale Evaluation

We compare with two existing automatic metrics for free-text rationale evaluation: LAS [90] and RQ [91]. Analogous to our evaluation models, both approaches use proxy models; we use the same architecture (T5 Large) across metrics in our reported results.

Leakage-Adjusted Simulatability (LAS) Hase et al. [90] evaluate the quality of free-text rationales via a proxy model, trained with the task model outputs as labels and original input texts combined with rationales as input sequences. The metric computes the difference between its prediction accuracy on the predicted label when the rationale is included into the input vs. when it is not, $\mathbb{1}[\hat{y} \mid x, \hat{r}] - \mathbb{1}[\hat{y} \mid x]$, averaged over examples grouped based on whether they leak labels or not. The final LAS score is given by the macro average across groups.

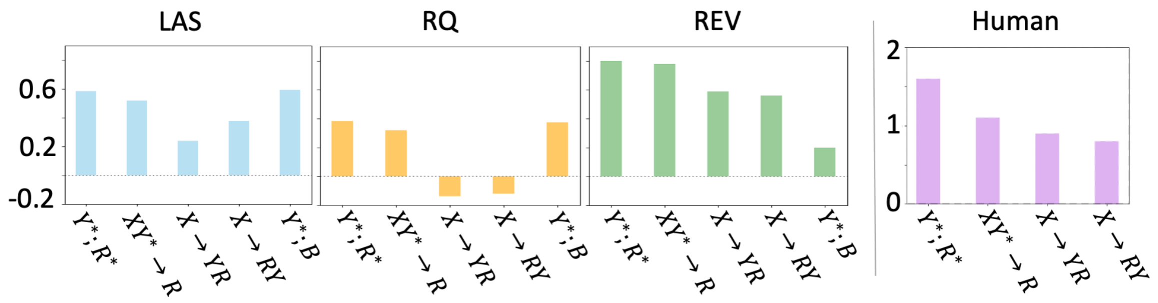


Figure 5.3: Left: automatic evaluation results of LAS, RQ and REV for rationale-label pairs on the ECQA test set. Right: human evaluation for rationale-label pairs on 230 randomly selected examples from the ECQA test set.

Rationale Quality (RQ) Wiegreffe et al. [91] propose a variant of the simulatability in Hase et al. [90]. The main difference is that gold labels are used to train the model proxy and evaluate rationale quality. Specifically, the quality of a rationale $\hat{r}$ is measured as $\mathbb{1}[y^* \mid x, \hat{r}] - \mathbb{1}[y^* \mid x]$, where y^* is the gold label. Similarly, RQ is the average score over all test examples.

5.3 Experiments

We first compare REV with existing metrics (§5.3.1) and human judgments (§5.3.3) on the ECQA dataset, as well as show REV on other CQA and NLI benchmarks. We then test the sensitivity of different metrics to input perturbations (§5.3.4). Next, we apply REV to generations via few-shot prompting (5.3.5).

5.3.1 Comparison Between Evaluation Metrics

We compare REV to LAS and RQ, in evaluating different rationale-label pairs on the ECQA dataset. In addition to XY*→R, X→YR, X→RY, and (Y*;R*), we also explore the evaluation on vacuous baseline rationales (Y*;B), which simply combine inputs and labels with no additional information. Note that the scores obtained from different metrics are not directly comparable due to different comparison scales and criteria (e.g., log-probability vs. accuracy). We mainly focus on the ranking over different types of rationale-label pairs. The results averaged over 4 random seeds are shown in the left part of Fig. 5.3.

All three metrics agree that the crowdsourced rationales (Y*;R*) in the ECQA have the highest quality. While by definition, REV for vacuous rationales (Y*;B) is low, both LAS and RQ scores for these rationales are quite high, showing that these metrics are incapable of measuring the amount of additional information in rationales. Intuitively, we expect weaker rationale-label consistency in

X→RY setting compared to X→YR, as the labels are forcefully selected among the candidates as opposed to being freely generated by the task model (§5.2.2). While REV is able to capture this intuition and ranks X→YR higher than X→RY, LAS and RQ have a different ranking.

Qualitative Analysis of Different Metrics Table 5.4 shows the qualitative analysis of different metrics on the four types of rationale-label pairs (Y^*;R^*, XY^*→R, X→YR, X→RY) on the ECQA dataset. REV provides more accurate evaluations on those examples than LAS and RQ.

Additional Analysis on Label-Related Sentences In some cases, a rationale contains the given label and provides new information related to the label, but does not necessarily explain why the label is selected for the input. To evaluate such rationales, we randomly select 250 gold labels in ECQA and extract their related sentences from a large-scale knowledge base—GenericsKB [212]. Those sentences contain the labels, while providing little or irrelevant new information to explain the labels w.r.t. the inputs. We use them as trivial rationales for evaluation. The average REV scores for those trivial rationales and their crowdsourced counterparts are 0.26 and 1.14 respectively, indicating the effectiveness of REV in identifying the new and relevant information in rationales. Table 5.5 shows the REV scores of some examples and the corresponding crowdsourced rationales. The results show that REV can distinguish the new information in different rationales and penalize meaningless rationales. Overall, REV gives higher scores to crowdsourced rationales than trivial sentences from GenericsKB.

5.3.2 Evaluation on Different Datasets

Next, we apply REV to evaluate crowdsourced and model generated rationale-label pairs (Y^*;R^*, XY^*→R, X→YR, X→RY) across different datasets. For each dataset, the evaluation models are trained on the training set with gold labels and crowdsourced rationales. The results are shown in Table 5.6. We observe that the gold rationales in the ECQA dataset achieve higher REV score than those in CoS-E. This observation is in line with the known quality issues of crowdsourced rationales in CoS-E [209, 93]. Moreover, training the evaluation models with CoS-E results in lower REV for all models, compared to training with ECQA. Interestingly, model-generated rationales (XY^*→R) have higher REV scores than crowdsourced rationales for CoS-E (see examples in Table 5.7). QuaRTz has better quality of rationales compared to ECQA, CoS-E, and e-SNLI. In the case of e-SNLI, the problem is severe as most of the crowdsourced or generated rationales do not provide reasoning but rather follow a label-specific template e.g., *A implies (that) B* [46, 89].

Qualitative Analysis of CoS-E Rationales Table 5.7 shows the exemplar of REV scores for crowdsourced and model-generated ($XY^* \to R$) rationales for CoS-E. The main observation is model-generated rationales ($XY^* \to R$) generally support labels, though provide limited new information, while many crowdsourced rationales in CoS-E are noisy or uninformative. Specifically, compared to the crowdsourced rationales in CoS-E, we observe that $XY^* \to R$ can produce better rationales that support the labels, which also corresponds to higher REV scores. However, the new information contained in those rationales is still limited (please see examples). A possible reason is the task model ($XY^* \to R$) hardly learns to produce more informative rationales when trained using lower quality rationales from CoS-E, known quality issue as reported in prior work [209, 93].

Qualitative Analysis of Negative REV Scores Table 5.8 shows some examples of $X \to RY$ with negative REV scores on the ECQA dataset. When REV < 0, we observe in most cases the rationale does not support the given label, while indicating other labels, or something even beyond the label candidates (e.g., “helicopter” in the second example). Or they could repeat the input (e.g., the first example). The same observation holds for other types of rationale-label pairs.

5.3.3 Human Evaluation

To understand how REV correlates with human judgments of rationales, we conduct a crowdsourcing experiment via Amazon Mechanical Turk. We randomly sample 230 examples from the ECQA test set and ask workers to evaluate the four types of rationale-label pairs ($Y^*;R^*$, $XY^* \to R$, $X \to YR$, $X \to RY$) for each example. We do not consider ($Y^*;B$) because we have trained workers to recognize baseline rationales as vacuous. We selected workers located in Australia, Canada, the UK, or the US, with a past HIT approval rate of $> 98\%$ and > 5000 HITs approved. Each instance is assessed by 3 workers. We pay the workers \$0.08 for assessing each instance.

We present workers with a question (input text), an answer (label) and an explanation (rationale), and ask them whether the explanation justifies the answer (*yes/no*). If they answer *yes*, we further ask them to evaluate the amount of additional information supplied by the explanation that explains *why* the answer might have been chosen for the question. The workers choose from *none / little / some / enough*, corresponding to a 4-point Likert-scale. [8]. We collect 3 annotations per instance and use majority vote to decide whether the rationale can justify the label. If *yes*, we take the average over the 3 human-annotated scores as the amount of information. Otherwise, we give a score of -1.

[8]The four options correspond to 0/1/2/3 respectively.

Figure 5.7 shows the instructions we provide to workers. In Figure 5.8, we show three examples, illustrating when the explanation (rationale) does not justify the answer (label), when the explanation supports the answer while not supplying additional information, and when the explanation supports the answer and provides additional information. Figure 5.9 shows the interface of the actual hit for human evaluation.

For each instance, we provide a question (input), an answer (label), and an explanation (rationale), and ask the workers to answer the following two questions:

1. *Does the Explanation justify the given Answer?* (yes or no) The question is to ask workers to judge whether the rationale supports the label or not.

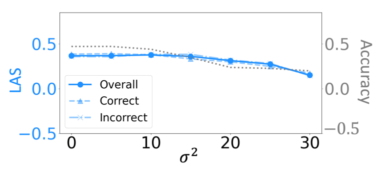
2. *If yes, how much additional information does the Explanation have to justify the Answer beyond just reiterating what is stated in Question and Answer?* (No additional info, Little additional info, Some additional info, Enough additional info) We only ask this question if the workers choose "yes" for the first question. We design this question to ask workers to evaluate the extent to which the rationale provides additional information for justifying the label beyond repeating it w.r.t. the input.

The results are shown in the right part of Fig. 5.3, where the ranking of the four types of rationale-label pairs is $Y^*;R^* > XY^* \rightarrow R > X \rightarrow YR > X \rightarrow RY$. While LAS and RQ rank X→RY better than X→YR (see the left part of Fig. 5.3), the ranking from REV is more consistent with human judgments, suggesting its effectiveness in evaluating rationale quality.

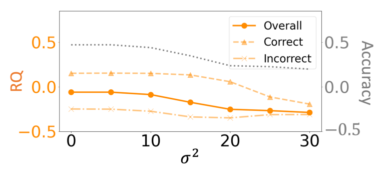
5.3.4 Is REV Sensitive to Input Perturbations?

We test the sensitivity of all automatic metrics to input (X) perturbations in the task model, under two settings: X→YR and X→RY. Following Wiegreffe et al. [91], we add zero-mean Gaussian noise $\mathcal{N}(0, \sigma^2)$ to input word embeddings during inference, inducing task models to produce progressively degenerate rationales and labels. A good metric should be sensitive to the change of rationales and labels and reflect their relationships under input perturbations. REV and RQ show similar trends as for X→RY in Fig. 5.4 (b) and (c). However, LAS is less sensitive to noise for both joint models, X→RY and X→YR, in Fig. 5.4 (a) and (d). Since the proxy model for LAS is trained on the task models' predicted labels and generated rationales, it can overfit to the degenerate rationale-label pairs under input perturbations, hence being less sensitive to input noise during inference.

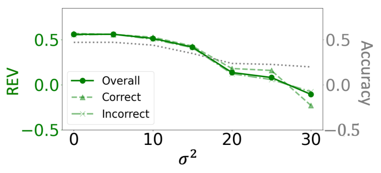
The largest differences between REV and RQ are for X→YR. We observe the task model can predict incorrect labels and then make up reasonable-sounding rationales for its wrong predictions



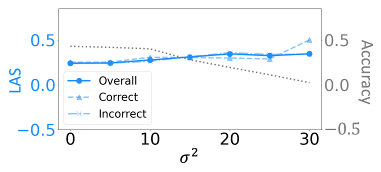
(a) X→RY, LAS



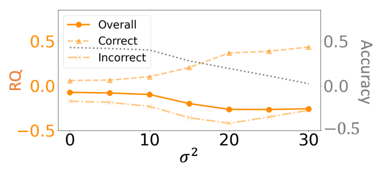
(b) X→RY, RQ



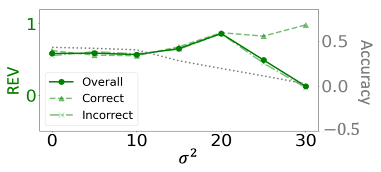
(c) X→RY, REV



(d) X→YR, LAS



(e) X→YR, RQ



(f) X→YR, REV

Figure 5.4: Sensitivity test results of REV, LAS and RQ for X→RY and X→YR on the ECQA dataset. The X-axis shows different levels of noise (σ^2). We plot the curve of Accuracy (model prediction accuracy) vs. Noise in gray dashed line. We also separate the evaluation results on populations on which the model predictions are correct ("Correct") or incorrect ("Incorrect") in addition to the overall evaluation on all test examples ("Overall").

under certain input perturbations; prior work also reports this finding [87, 91]. REV does not drop under a certain amount of input perturbations (e.g., $\sigma^2 \leq 20$) in Fig. 5.4 (f), likely because the generated rationales still provide new information for describing both correct and incorrect labels (also see the example in Table 5.9). However, as the noise exceeds the certain level, REV decreases indicating that the task model is no longer able to make up rationales for very noisy inputs. On the other hand, the behaviors of RQ and REV are quite different in Fig. 5.4 (e) and (f). Since RQ is computed based on gold labels (§5.2.3), it has reduced sensitivity to input perturbations. When the prediction accuracy decreases, the overall evaluation of RQ is dominated by the results on incorrect predictions, as shown in Fig. 5.4 (e). Table 5.9 shows some examples from the sensitivity test.

5.3.5 Evaluating Rationales in Few-shot Prompting

We test the ability of REV in evaluating rationales generated by few-shot prompting, and get insights on the reasoning and prediction processes of large language models (e.g., GPT-3).

GPT-3 Rationales for Gold Labels. Wiegreffe et al. [213] collected 250 high quality free-text rationales generated by few-shot prompting with GPT-3 [15] for CQA (given gold labels). Each example was assessed by 3 crowdworkers. We focus on two aspects of their annotations: "supports the gold label" and "amount of information". Crowdworkers provide a *yes* / *no* answer to justify whether a rationale supports the corresponding gold label. Only when the answer is *yes*, they are further asked to evaluate the amount of information contained in the rationale for justifying the

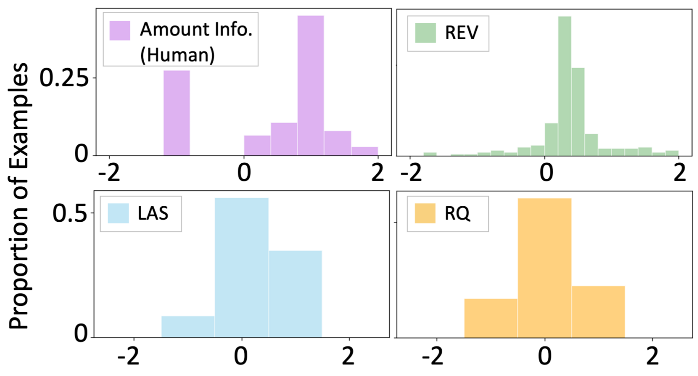


Figure 5.5: Histograms of human-annotated amount of information and pointwise REV, LAS and RQ scores on GPT-3 few-shot prompted rationales for gold labels.

label. The amount of information is roughly categorized into 3 levels: "Not Enough", "Enough", "Too Much", each annotated with a Likert-scale score.[9] In Fig. 5.5, we compare human annotation scores for amount of information[10] with the pointwise scores obtained by three automatic metrics, LAS, RQ, and REV. For automatic metrics, the evaluation models of REV and the proxy models of LAS and RQ are trained on the ECQA training set with gold labels and human-annotated rationales (§5.2.2). We observe that REV provides finer-grained assessment of the information contained in rationales compared to LAS and RQ which only take {-1, 0, 1} values. When LAS and RQ are zero, it is unclear whether the rationale supports the label or not because the model proxy may predict the label based on the input only. The judgments of REV on whether rationales support labels (REV > 0) are close to human judgments (i.e., 80% agreement). The support rates of LAS and RQ are relatively low, i.e. 35% and 23%, while a large portion (56% and 60% respectively) corresponds to a zero LAS / RQ score.

Chain of Thought Rationales. Wei et al. [201] propose *chain of thought prompting* to teach large language models to produce intermediate reasoning steps (rationales) before prediction, which improves their prediction performance on a range of reasoning tasks (e.g., arithmetic and symbolic reasoning). However, the reported improvement is trivial for CQA [201], which motivates us to evaluate the intermediate rationales w.r.t. model predictions. We apply REV to analyze the generated rationales during intermediate reasoning steps and final predicted labels from GPT-3 text-davinci-002 [15] and LaMDA 137B [214].[11]

[9]The original human-annotated scores w.r.t. the three levels are: -1, 0, 1. Since Wiegreffe et al. [213] suggest "a value of 0 is preferred to a value of 1", we map the scores {-1, 0, 1} to {0, 1, 2} accordingly. The value "-1" is then given to examples annotated as "not supporting gold labels".

[10]We take majority vote to decide "supports the gold label", and average "amount of information" over 3 workers.

[11]Available at `https://github.com/jasonwei20/chain-of-thought-prompting`

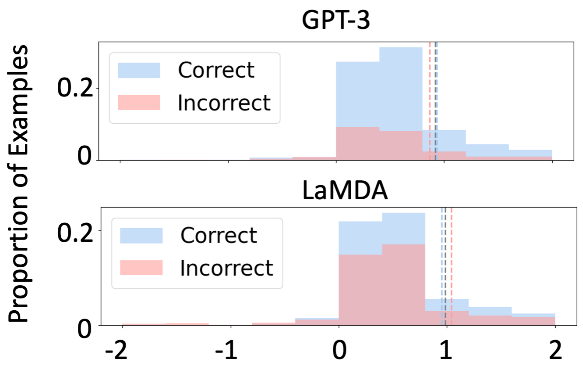


Figure 5.6: Distributions of REV for rationales w.r.t. correct and incorrect predictions produced by GPT-3 and LaMDA respectively. The average REV scores over all instances, correctly predicted instances and incorrectly predicted instances are marked by gray, blue and red dashed lines respectively.

Figure 5.6 shows the distributions of REV for correctly and incorrectly predicted instances from GPT-3 and LaMDA, respectively. For both GPT-3 and LaMDA, the REV distributions of correct and incorrect predictions are similar and most instances have positive REV scores. The results demonstrate the causality between the models' intermediate reasoning process and their final predictions, no matter whether the predicted labels are correct or incorrect. The average REV scores (blue/red dashed lines) over correct and incorrect predictions are close, especially for GPT-3. This is consistent with our observation that most generated rationales from the two models are describing their predicted labels. The prediction accuracy of GPT-3 is much higher than that of LaMDA (77% vs. 59%), while the average REV scores (gray dashed lines) over all instances are close (0.92 vs. 0.99). An insight we obtain is that the generated intermediate reasoning steps (rationales) support models' predictions (consistent REV scores), but cannot guarantee their correctness (discrepant accuracies between GPT-3 and LaMDA). This partially explains the minor improvement of chain of thought prompting on CQA.

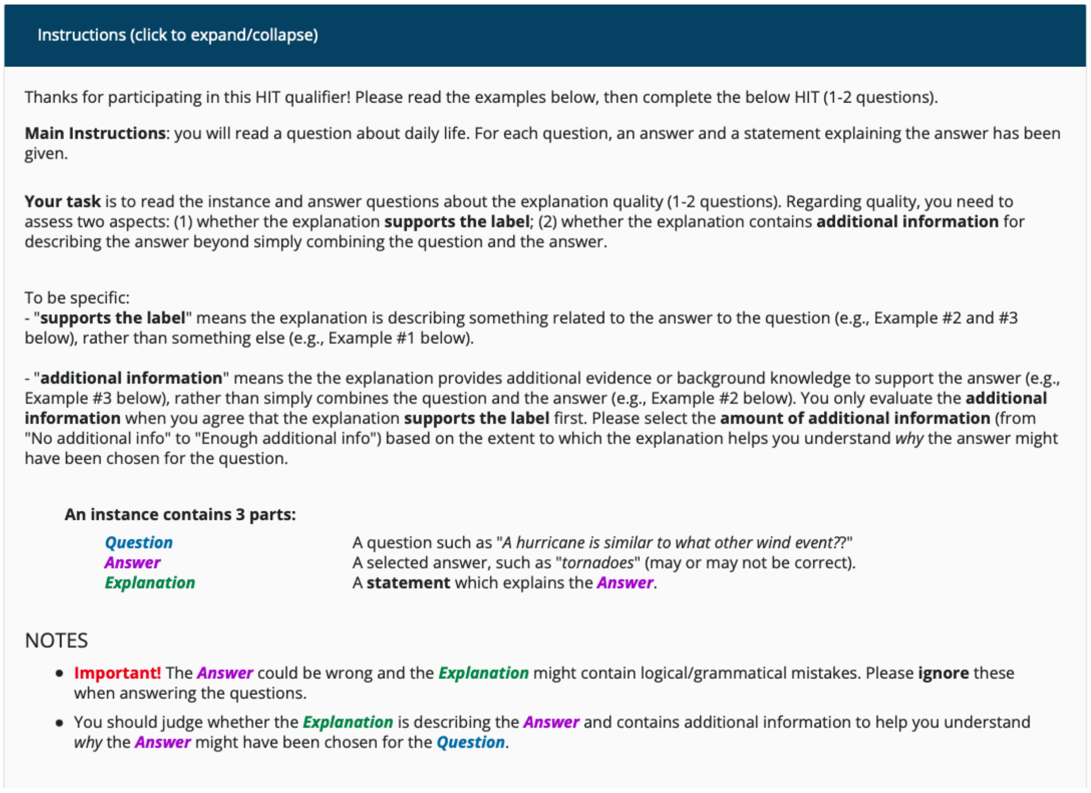
5.4 Related Work

Self-rationalized models serve interpretability by providing rationales for their predictions. Model rationales broadly fall into two categories: extractive rationales and free-text rationales. Extractive rationales contain some important features extracted from input texts that make models produce final predictions [128, 72, 215, 216]. Free-text rationales are produced by generative models in the form of natural language. Compared to extractive rationales, free-text rationales explain model predictions in a more human-like way and fill the gap in explaining reasoning tasks [42, 87, 88, 46, 89].

Evaluations on extractive rationales have been well studied, generally from two perspectives — faithfulness and plausibility [72, 85, 217]. Faithfulness measures to which extent rationales reflect the true reasoning process of models, while plausibility evaluates how convincing rationales are to humans [34]. Other perspectives include the ability of rationales in helping a student model simulate a teacher model [85] or bridging the communication between a classifier and a layperson [86]. However, most evaluation metrics (e.g., sufficiency, comprehensiveness) proposed for extractive rationales are not applicable to free-text rationales because they are not a part of inputs. Existing automatic metrics for free-text rationales focus on rationale-label association, and measure the utility of a rationale based on how much it helps a model proxy predict the given label (inspired by human simulatability [16]) [90] or the gold label [91] given the input. Chan et al. [92] further propose a framework to evaluate the automatic metrics. However, none of them consider measuring the amount of additional information in free-text rationales. Sun et al. [93] conduct a human study on the additional knowledge provided by free-text rationales. This work is the first that proposes an automatic metric to quantify the new information in free-text rationales.

5.5 Conclusion

In this paper, we propose an information-theoretic metric, REV, to evaluate free-text rationale. REV measures if a rationale contains new information that is relevant for the label of interest, beyond the information contained in the input. We show the advantage of REV in evaluating different types of rationale-label pairs compared to existing metrics. We demonstrate that the evaluation of free-text rationales with REV is consistent with human judgments. REV also offers insights on evaluating rationales generated via few-shot prompting.



Instructions (click to expand/collapse)

Thanks for participating in this HIT qualifier! Please read the examples below, then complete the below HIT (1-2 questions).

Main Instructions: you will read a question about daily life. For each question, an answer and a statement explaining the answer has been given.

Your task is to read the instance and answer questions about the explanation quality (1-2 questions). Regarding quality, you need to assess two aspects: (1) whether the explanation **supports the label**; (2) whether the explanation contains **additional information** for describing the answer beyond simply combining the question and the answer.

To be specific:
- "**supports the label**" means the explanation is describing something related to the answer to the question (e.g., Example #2 and #3 below), rather than something else (e.g., Example #1 below).

- "**additional information**" means the the explanation provides additional evidence or background knowledge to support the answer (e.g., Example #3 below), rather than simply combines the question and the answer (e.g., Example #2 below). You only evaluate the **additional information** when you agree that the explanation **supports the label** first. Please select the **amount of additional information** (from "No additional info" to "Enough additional info") based on the extent to which the explanation helps you understand *why* the answer might have been chosen for the question.

An instance contains 3 parts:

Question	A question such as "*A hurricane is similar to what other wind event??*"
Answer	A selected answer, such as "*tornadoes*" (may or may not be correct).
Explanation	A **statement** which explains the ***Answer***.

NOTES

- **Important!** The ***Answer*** could be wrong and the ***Explanation*** might contain logical/grammatical mistakes. Please **ignore** these when answering the questions.
- You should judge whether the ***Explanation*** is describing the ***Answer*** and contains additional information to help you understand *why* the ***Answer*** might have been chosen for the ***Question***.

Figure 5.7: The instructions of human evaluation in the user interface on AMT.

Type	Question	Label	Rationale	Metric		
				REV	LAS	RQ
$Y^*;R^*$	If you have a ticket and you are planning to eat hot dogs, where would you go?	baseball stadium	Hot dogs can be eaten at baseball stadium. When you go to a baseball stadium, you have a ticket and you may plan to eat hot dogs.	0.32	0	0
	How does a person go to space?	space shuttle	People go to space by a vehicle specially designed to travel to space. That vehicle is called a space shuttle.	0.32	0	0
	What is a dangerous outdoor activity for children?	sun themselves	Sunning themselves is a dangerous activity Children should not sun themselves	0.28	1	1
$XY^*\rightarrow R$	Where are old pictures kept?	attic	Attic is a place where old pictures are kept.	0.34	1	0
	What would you be if you comfort friend?	friendly	Comforting friend is a good thing.	0.43	0	1
	What do customers do to a waiter after the waiter serves customers?	pay to	Paying to a waiter is the action of paying. Waiters get paid to serve customers.	0.10	-1	0
$X\rightarrow YR$	Where is there likely to be more than one desk drawer?	desk	Desk drawer is a drawer used for storing office supplies. There is likely to be more than one desk drawer in office.	-4.75	-1	1
	What leads to someone's death when they are very depressed?	suicide	Suicide is the act of committing suicide. When someone is very depressed, suicide leads to their death.	0.57	0	0
	Where are you normally when you take a bath?	hotel room	Hotel room is a place where people stay. Bathing is normally done in hotel rooms.	0.32	0	-1
$X\rightarrow RY$	What is likely heard by those going to a party?	laughter	People go to a party to meet new people. People are likely to hear laughter at the party.	-0.10	1	0
	What would you do if you have excitement and do not want to stay in your house?	go to gym	Go to gym is to go to a place where you can express information. If you have excitement and do not want to stay in your house, then you would go somewhere.	0.53	1	0
	If you're caught committing murder, an injection can lead to your own what?	die	An injection can lead to one's own death. If you're caught committing murder, you can be injected into your own body and die.	1.46	0	0

Table 5.4: Pointwise evaluation of REV, LAS and RQ on different types of rationale-label pairs. Incorrect labels are colored red.

Source	Input	Label	Rationale	REV
Crowdsourced	What form of government is most associated with kingdoms?	monarchy	Monarchy is a form of government with the monarch at the head. Monarchy is a form of government mostly associated with kingdoms.	0.65
	Bailey liked playing games against other people. He found it exhilarating. What might Bailey like about games?	competitiveness	When a game is played against someone, it is a competition and it promotes competitiveness. Games are competitive in nature when it involves people against each other.	0.37
	How is a dog likely to communicate with another dog?	bark	Bark is the sharp explosive cry of a dog, fox, or seal. The dog is likely to communicate with another dog with a bark.	2.11
	Where would you put a car near your house?	driveway	Driveway is a place near the house. A car can be put in the driveway.	0.48
GenericsKB	What form of government is most associated with kingdoms?	monarchy	Monarchies are countries.	-0.94
	Bailey liked playing games against other people. He found it exhilarating. What might Bailey like about games?	competitiveness	Competitiveness also means education, research and innovation including in the area of environment.	-0.14
	How is a dog likely to communicate with another dog?	bark	Bark is covering.	-4.37
	Where would you put a car near your house?	driveway	Driveways are located in cars.	0.43

Table 5.5: Exemplar of REV scores for crowdsourced rationales and label-related sentences from GenericsKB for ECQA.

Datasets	Rationale-label pairs			
	$Y^*;R^*$	$XY^*\rightarrow R$	$X\rightarrow YR$	$X\rightarrow RY$
ECQA	0.7943	0.7806	0.5840	0.5599
CoS-E	0.2415	0.4050	0.2308	0.1198
QuaRTz	1.3919	1.3696	1.3449	1.0170
e-SNLI	0.0752	0.0079	0.0055	0.0047

Table 5.6: REV scores of different types of rationale-label pairs on the four datasets.

Type	Input	Label	Rationale	REV
Crowdsourced	The goal was to hit the target, but a projectile ball can't hit anything if it isn't in what?	motion	if you stand still you get hit	-0.14
	When you get together with friends to watch film, you might do plenty of this?	have fun	when the working day is done	-0.27
	They dealt with combustible mixtures in their experiments, this is why they kept a fire extinguisher where?	chemistry lab	mixtures mixing fruitsa	-0.17
$XY^* \to R$	The goal was to hit the target, but a projectile ball can't hit anything if it isn't in what?	motion	a projectile ball can't hit anything if it's not in motion	0.09
	When you get together with friends to watch film, you might do plenty of this?	have fun	when you get together with friends to watch film, you might do plenty of fun	1.47
	They dealt with combustible mixtures in their experiments, this is why they kept a fire extinguisher where?	chemistry lab	chemistry labs deal with combustible mixtures in their experiments.	0.74

Table 5.7: Exemplar of REV scores for crowdsourced and model-generated (XY*→R) rationales for CoS-E.

Input	Label	Rationale	REV
What do people call it when they are going for run?	falling down	People call it run when they are going for run.	-1.06
What enables most people to transport themselves?	own cars	People who believe in god are able to transport themselves through helicopter.	-0.19
Where would you expect to find popcorn in a public place?	movie theater	Popcorn can be found in a public place.	-1.27
What are you usually at when you sit on a bench on a curb?	city	Ohio is a state in the United States. You are usually at street corner when you sit on bench on curb.	-0.27

Table 5.8: Exemplar of negative REV scores for rationale-label pairs from X→RY on the ECQA dataset.

X→YR
Question: What is likely to happen if you're doing housework all day?
Choices: boredom, tiredness, get tired, backache, get a clean house
Gold label: get tired
Gold rationale: Get tired means no longer wanting someone or wanting to do something because you are bored with it or annoyed by it. Get tired is likely to happen if you're doing housework all day.

$\boldsymbol{\sigma^2 = 0, \mathbf{REV} = 1.36}$
Output: get tired [rationale] If you're doing housework all day, it's likely to happen if you are getting tired.

$\boldsymbol{\sigma^2 = 5, \mathbf{REV} = 1.36}$
Output: get tired [rationale] If you're doing housework all day, it's likely to happen if you are getting tired.

$\boldsymbol{\sigma^2 = 10, \mathbf{REV} = 1.36}$
Output: get tired [rationale] If you're doing housework all day, it is likely to happen if you are getting tired.

$\boldsymbol{\sigma^2 = 15, \mathbf{REV} = 1.29}$
Output: backache [rationale] Backache is a condition of being tired. You are likely to get tired if you're doing housework all day.

$\boldsymbol{\sigma^2 = 20, \mathbf{REV} = 0.72}$
Output: boredom [rationale] Boredom is a state of being or being unable to do something. People tend to get bored easily when they are bored.

$\boldsymbol{\sigma^2 = 25, \mathbf{REV} = 1.33}$
Output: backache [rationale] Backache is the state of being unable to move backwards or forwards in a straight line. One cannot move forward if they are not able to do so.

$\boldsymbol{\sigma^2 = 30, \mathbf{REV} = 0}$
Output: banishing oneself from one's own body by removing it from the body of another person. This is a way of preventing one from being harmed by other people. It is possible to get rid of people from their own bodies if they are living in the same house. That is why they have to live together.

X→RY
Question: What happens when not paying attention when bathing?
Choices: dry skin, using soap, reading newspaper, shampoo, slip and fall
Gold label: slip and fall
Gold rationale: Bathing is generally done in a bathroom. Bathrooms could be slippery due to water. If not careful slipperiness can cause one to slip and fall.

$\boldsymbol{\sigma^2 = 0, \mathbf{REV} = 1.28}$
Output: Bathing can cause slip and fall when one is not paying attention. Slip and Fall is a common occurrence when people do not pay attention when bathing. [answer] slip and fall

$\boldsymbol{\sigma^2 = 5, \mathbf{REV} = 1.28}$
Output: Bathing can cause slip and fall when one is not paying attention. Slip and Fall is a common occurrence when people do not pay attention when bathing. [answer] slip and fall

$\boldsymbol{\sigma^2 = 10, \mathbf{REV} = 1.24}$
Output: Bathing can cause slip and fall when not paying attention. Slipping and falling is a result of not being attentive when bathing. [answer] slip and fall

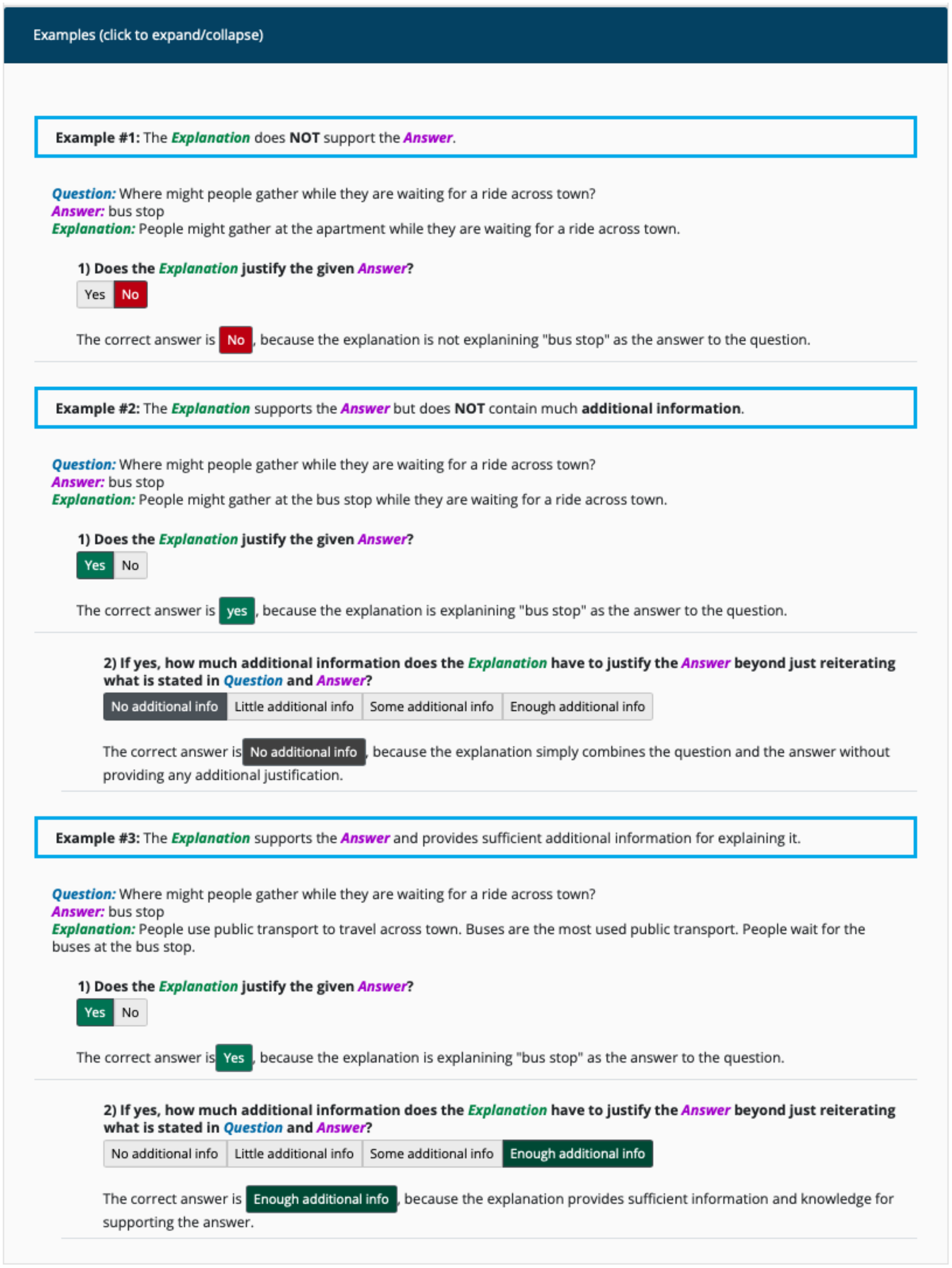
$\boldsymbol{\sigma^2 = 15, \mathbf{REV} = 1.20}$
Output: Bathing is a process of washing the body. Not paying attention means not paying enough attention to the bathing process. [answer] slip and fall

$\boldsymbol{\sigma^2 = 20, \mathbf{REV} = -0.15}$
Output: Bathing is a process of washing and rinsing. Paying attention means paying attention to something. [answer] reading newspaper

$\boldsymbol{\sigma^2 = 25, \mathbf{REV} = -0.17}$
Output: Paying attention to what you are doing is a sign of being aware of what is going on around you. Knowing what to do is to pay attention. [answer] dry skin

$\boldsymbol{\sigma^2 = 30, \mathbf{REV} = -0.85}$
Output: Paying attention to one's own needs is a basic human need. One needs to be aware of the needs of others in order to meet them...engavengendersengendoenv -end-engagingendentenant [answer] slip and fall

Table 5.9: Examples of the two task models' outputs under different levels of noise in sensitivity test.


Examples (click to expand/collapse)

Example #1: The ***Explanation*** does **NOT** support the ***Answer***.

Question: Where might people gather while they are waiting for a ride across town?
Answer: bus stop
Explanation: People might gather at the apartment while they are waiting for a ride across town.

1) Does the *Explanation* justify the given *Answer*?
Yes [✓ No]

The correct answer is No, because the explanation is not explanining "bus stop" as the answer to the question.

Example #2: The ***Explanation*** supports the ***Answer*** but does **NOT** contain much **additional information**.

Question: Where might people gather while they are waiting for a ride across town?
Answer: bus stop
Explanation: People might gather at the bus stop while they are waiting for a ride across town.

1) Does the *Explanation* justify the given *Answer*?
[✓ Yes] No

The correct answer is yes, because the explanation is explanining "bus stop" as the answer to the question.

2) If yes, how much additional information does the *Explanation* have to justify the *Answer* beyond just reiterating what is stated in *Question* and *Answer*?
[✓ No additional info] Little additional info Some additional info Enough additional info

The correct answer is No additional info, because the explanation simply combines the question and the answer without providing any additional justification.

Example #3: The ***Explanation*** supports the ***Answer*** and provides sufficient additional information for explaining it.

Question: Where might people gather while they are waiting for a ride across town?
Answer: bus stop
Explanation: People use public transport to travel across town. Buses are the most used public transport. People wait for the buses at the bus stop.

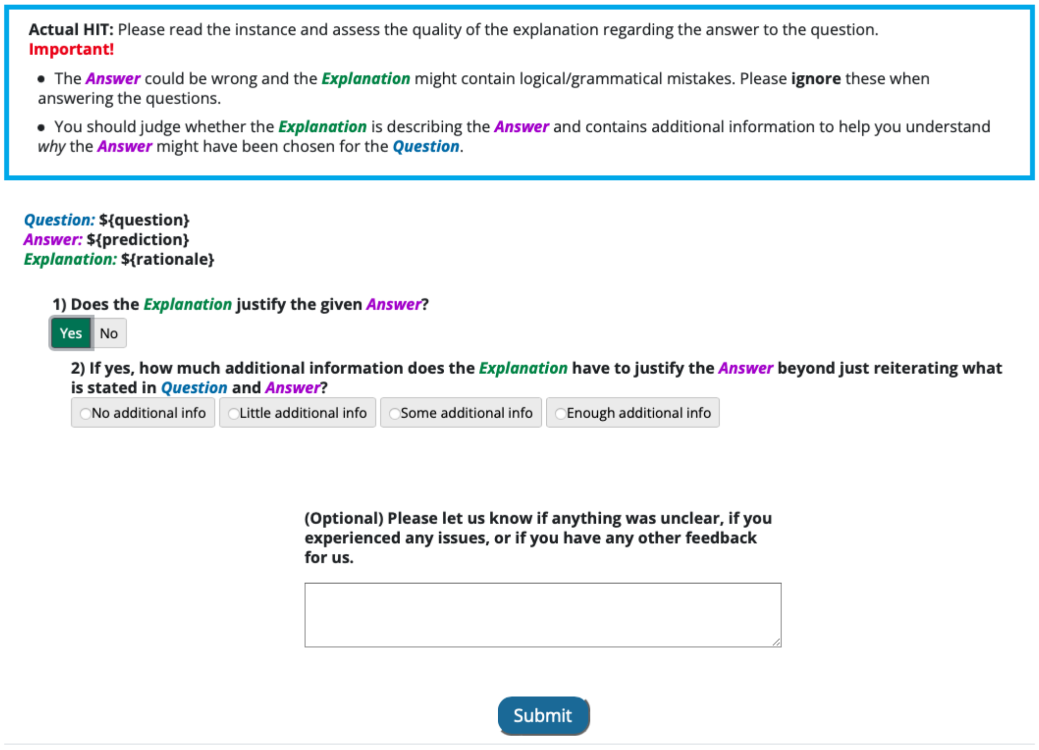
1) Does the *Explanation* justify the given *Answer*?
[✓ Yes] No

The correct answer is Yes, because the explanation is explanining "bus stop" as the answer to the question.

2) If yes, how much additional information does the *Explanation* have to justify the *Answer* beyond just reiterating what is stated in *Question* and *Answer*?
No additional info Little additional info Some additional info [✓ Enough additional info]

The correct answer is Enough additional info, because the explanation provides sufficient information and knowledge for supporting the answer.

Figure 5.8: Exemplars provided to worker in the user interface on AMT.



Actual HIT: Please read the instance and assess the quality of the explanation regarding the answer to the question.
Important!

- The ***Answer*** could be wrong and the ***Explanation*** might contain logical/grammatical mistakes. Please **ignore** these when answering the questions.
- You should judge whether the ***Explanation*** is describing the ***Answer*** and contains additional information to help you understand *why* the ***Answer*** might have been chosen for the ***Question***.

Question: **${question}**
Answer: **${prediction}**
Explanation: **${rationale}**

1) Does the *Explanation* justify the given *Answer*?

[✓ Yes] No

2) If yes, how much additional information does the *Explanation* have to justify the *Answer* beyond just reiterating what is stated in *Question* and *Answer*?

○ No additional info ○ Little additional info ○ Some additional info ○ Enough additional info

(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any other feedback for us.

Figure 5.9: The actual hit of human evaluation in the user interface on AMT.

Chapter 6

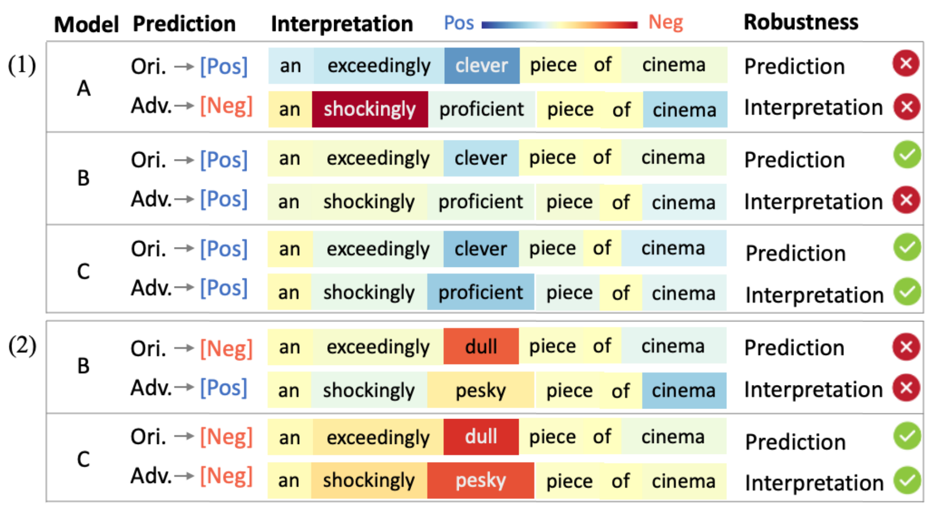
Diagnosing and Debugging Models

The ultimate goal of explaining neural networks or improving their interpretability is to better understand, diagnose, debug, and improve them. In this chapter, I introduce a novel feature-level adversarial training method (FLAT) to improve model robustness via explanations in Section 6.1 and report my discoveries of model pathologies in few-shot fine-tuning in Section 6.2.

6.1 Improving Model Robustness via Interpretations

Neural language models are vulnerable to adversarial examples generated by adding small perturbations to input texts [218, 219, 102]. Adversarial examples can be crafted in several ways, such as character typos [220, 221], word substitutions [102, 222, 103, 223], sentence paraphrasing [224, 225], and malicious triggers [226]. In this work, we focus on word substitution-based attacks, as the generated adversarial examples largely maintain the original semantic meaning and lexical and grammatical correctness compared to other attacks [227].

Previous methods on defending this kind of attacks via adversary detection and prevention [228, 229] or certifiably robust training [230, 231] either circumvent improving model predictions on adversarial examples or scale poorly to complex neural networks [232]. Alternatively, adversarial training [103, 104] improves model robustness via two steps—collecting adversarial examples by attacking a target model, and fine-tuning the model on the augmented dataset with these adversarial examples. However, existing adversarial training only focuses on making a model produce the same correct predictions on an original/adversarial example pair, while ignores the consistency between model decision-makings on the two similar texts.



Model
Prediction
Interpretation
Pos
Neg
Robustness
(1)
A
Ori. → [Pos]
Adv. → [Neg]
an exceedingly clever piece of cinema
an shockingly proficient piece of cinema
Prediction
Interpretation
B
Ori. → [Pos]
Adv. → [Pos]
an exceedingly clever piece of cinema
an shockingly proficient piece of cinema
Prediction
Interpretation
C
Ori. → [Pos]
Adv. → [Pos]
an exceedingly clever piece of cinema
an shockingly proficient piece of cinema
Prediction
Interpretation
(2)
B
Ori. → [Neg]
Adv. → [Pos]
an exceedingly dull piece of cinema
an shockingly pesky piece of cinema
Prediction
Interpretation
C
Ori. → [Neg]
Adv. → [Neg]
an exceedingly dull piece of cinema
an shockingly pesky piece of cinema
Prediction
Interpretation

Figure 6.1: Illustration of different model robustness with respect to predictions and interpretations on (1) a POSITIVE movie review and (2) a NEGATIVE movie review (Ori.), and their adversarial counterparts (Adv.). Model B makes the same correct predictions on Ori. and Adv. in (1), while the discrepant interpretations reveal its vulnerability which is attacked by another adversarial example in (2). Only model C is robust with the same predictions and consistent interpretations on both original/adversarial example pairs.

To illustrate the necessity of maintaining consistent model decision-makings (reflected by interpretations) during adversarial training, Figure 6.1 shows both the predictions and their corresponding interpretations of different models on original/adversarial example pairs. The interpretations were generated by IG [25], which visualizes the attribution of each input feature (word/token) to the model prediction. Figure 6.1 (1) shows the predictions and interpretations of model A, B, and C on a POSITIVE movie review and its adversarial counterpart. Model A is not robust as its prediction on the adversarial example is flipped and the interpretation is totally changed. Although model B makes the same predictions on the original and adversarial examples, its interpretations reveal that these predictions are based on different key features: for the original example, it is a sentiment word `clever`; for the adversarial example, it is a neutral word `cinema`. The interpretation discrepancy reveals the vulnerability of model B, as shown in Figure 6.1 (2), where we craft another adversarial attack. Model B fails to recognize `dull` and `pesky` as the same important, and makes a wrong prediction on the NEGATIVE adversarial example based on `cinema`. Only model C is robust as it behaves consistently on predicting both original/adversarial example pairs. Note that we look at model robustness through the lens of interpretations, while leaving the problem of trustworthiness or robustness of an interpretation method itself out as that is beyond the scope of this work.

Based on the previous discussion, we argue that a robust model should have consistent prediction behaviors on original/adversarial example pairs, that is making the same predictions (***what***) based on the same reasons (***how***) which are reflected by consistent interpretations, as the word saliency maps of model C in Figure 6.1. However, traditional adversarial training does not regularize model prediction behavior for improving model robustness. To train a robust model, we propose a fine-grained feature-level adversarial training named FLAT. FLAT learns global word importance via variational word masks [176] and regularizes the importance scores of the replaced words and their substitutions in original/adversarial example pairs during training. FLAT teaches the model to behave consistently on predicting original/adversarial example pairs by focusing on the corresponding important words based on their importance scores, hence improving the model robustness to adversarial examples.

The contribution of this work is three-fold: (1) we argue that adversarial training should improve model robustness by making the model produce the same predictions on original/adversarial example pairs with consistent interpretations; (2) we propose a new training strategy, feature-level adversarial training (FLAT), to achieve this goal by regularizing model prediction behaviors on original/adversarial example pairs to be consistent; and (3) we evaluate the effectiveness of FLAT in improving the robustness of four neural network models, LSTM [9], CNN [11], BERT [13], and DeBERTa [233], to two adversarial attacks on four text classification tasks. The models trained via FLAT also show better robustness than baseline models on unforeseen adversarial examples across six different attacks.

6.1.1 Method

This section introduces the proposed FLAT method. FLAT aims at improving model robustness by making a model behave consistently on predicting original/adversarial example pairs. To achieve this goal, FLAT leverages variational word masks to select the corresponding words (e.g. `fantastic` and `marvelous` in Figure 6.2) from an original/adversarial example pair for the model to make predictions. To ensure the correctness of model predictions, variational word masks learn global word importance during training and play as a bottleneck teaching the model to make predictions based on important words. Besides, FLAT regularizes the global importance of the replaced words in an original example and their substitutions in the adversarial counterpart so that the model would recognize the corresponding words as the same important (or unimportant), as Figure 6.2 shows.

Preliminaries. Given an input $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ ($i \in \{1, \ldots, n\}$) denotes the word embedding, the model $f_{\boldsymbol{\theta}}(\cdot)$ with parameter $\boldsymbol{\theta}$ outputs a prediction label $y = f_{\boldsymbol{\theta}}(\boldsymbol{x})$ for text classification tasks. An adversarial example $\boldsymbol{x}'$ is crafted from $\boldsymbol{x}$ under some constraints, such as

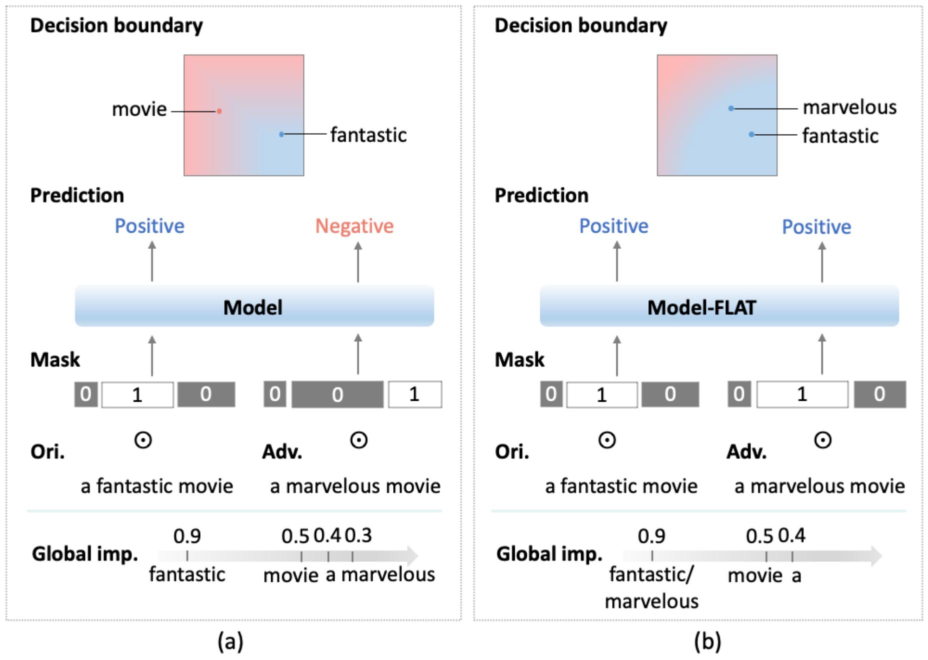


Figure 6.2: (a) The model with variational word masks trained on the standard training set. As `marvelous` is not recognized as the same important as its synonym `fantastic` and masked out, the model makes a wrong prediction based on a neutral word `movie`. (b) FLAT increases the global importance of `marvelous` and teaches the model to make the same correct predictions on the original/adversarial example pair by focusing on `fantastic` and `marvelous` respectively.

maintaining the original semantic meaning. For word substitution-based adversarial attacks, an adversarial example replaces some words $\{\boldsymbol{x}_i\}$ in the original example $\boldsymbol{x}$ with their synonyms $\{\boldsymbol{x}'_i\}$. The adversarial example fools the model to output a different label, i.e. $y' = f_{\boldsymbol{\theta}}(\boldsymbol{x}') \neq y$.

We obtain a set of adversarial examples $\mathcal{D}' = \{(\boldsymbol{x}'^{(m)}, y^{(m)})\}$ by attacking the model on the original dataset $\mathcal{D} = \{(\boldsymbol{x}^{(m)}, y^{(m)})\}$. During adversarial training, the model is trained on both original and adversarial examples ($\mathcal{D} \cup \mathcal{D}'$). In addition to improving model prediction accuracy on adversarial examples, adversarial training should also make the model produce the same predictions on the similar texts with consistent decision-makings. Failing to do this would make the model vulnerable to unforeseen adversarial examples crafted with the substitution words in some other contexts. To achieve this goal, we propose the feature-level adversarial training (FLAT) method.

Feature-level Adversarial Training

Recall the goal of FLAT is to train a robust model with consistent prediction behaviors on original/adversarial example pairs. There are two desiderata for FLAT:

1. Global feature importance scores $\boldsymbol{\phi}$. To teach the model to recognize the replaced words in an original example and their substitutions in the adversarial counterpart as the same important

(or unimportant) for predictions, FLAT needs to learn the global importance score $\phi_{\boldsymbol{x}_i}$ of a word $\boldsymbol{x}_i$. Note that the "global" means the importance score is solely dependent on the word (embedding).

2. Feature selection function $g_{\boldsymbol{\phi}}(\cdot)$. To guide the model to make predictions based on the corresponding important words in the original and adversarial example respectively, FLAT needs a feature selection function $g_{\boldsymbol{\phi}}(\cdot)$. $g_{\boldsymbol{\phi}}(\boldsymbol{x})$ selects important words from $\boldsymbol{x}$ based on their global importance scores in $\boldsymbol{\phi}$. The selected words are then forwarded to the model to output a prediction, i.e. $y = f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{x}))$.

FLAT leverages variational word masks [176] to learn global feature importance scores and select important features for model predictions, which will be introduced subsequently.

With the two desiderata, the objective of FLAT is formulated as

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \quad \mathcal{L}_{pred} + \gamma \mathcal{L}_{imp} \tag{6.1}$$

$$\mathcal{L}_{pred} = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathcal{L}(f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{x})), y)] + \mathbb{E}_{(\boldsymbol{x}',y)\sim\mathcal{D}'}[\mathcal{L}(f_{\boldsymbol{\theta}}(g_{\boldsymbol{\phi}}(\boldsymbol{x}')), y)] \tag{6.2}$$

$$\mathcal{L}_{imp} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\sim\mathcal{D}\cup\mathcal{D}'}[\textstyle\sum_{i,\boldsymbol{x}_i\neq\boldsymbol{x}'_i} |\phi_{\boldsymbol{x}_i} - \phi_{\boldsymbol{x}'_i}|] \tag{6.3}$$

where $\mathcal{L}(\cdot,\cdot)$ denotes cross entropy loss. $\boldsymbol{\phi}$ is a learnable vector with the same dimension as the predefined vocabulary, where $\phi_{\boldsymbol{x}_i} \in (0,1)$ represents the global importance of the word $\boldsymbol{x}_i$. $\gamma \in \mathbb{R}_+$ is a coefficient. $\mathcal{L}_{imp}$ regularizes the global importance scores of the replaced words $\{\boldsymbol{x}_i\}$ and their substitutes $\{\boldsymbol{x}'_i\}$ in an original/adversarial example pair $(\boldsymbol{x}, \boldsymbol{x}')$ by pushing $\phi_{\boldsymbol{x}_i}$ and $\phi_{\boldsymbol{x}'_i}$ close. With similar importance scores, the associated word pair $(\boldsymbol{x}_i, \boldsymbol{x}'_i)$ would be selected by $g_{\boldsymbol{\phi}}(\cdot)$ or not simultaneously. $\mathcal{L}_{pred}$ encourages the model to make the same and correct predictions on the original and adversarial example based on the selected important words $g_{\boldsymbol{\phi}}(\boldsymbol{x})$ and $g_{\boldsymbol{\phi}}(\boldsymbol{x}')$ respectively. By optimizing the objective, the model learns to behave consistently on predicting similar texts, hence having better robustness to adversarial attacks.

Learning with Variational Word Masks

FLAT fulfills the two desiderata by training the model ($f_{\boldsymbol{\theta}}(\cdot)$) with variational word masks [176]. Specifically, variational word masks learn global word importance $\boldsymbol{\phi}$ during training and select important words for the model to make predictions by masking out irrelevant or unimportant words. For an input $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]$, a set of masks $\boldsymbol{W} = [W_{\boldsymbol{x}_1}, \ldots, W_{\boldsymbol{x}_n}]$ are generated based on $\boldsymbol{\phi}$, where

$W_{\boldsymbol{x}_i} \in \{0, 1\}$ is a binary random variable with 0 and 1 indicating to mask out or select the word $\boldsymbol{x}_i$ respectively. The word importance score $\phi_{\boldsymbol{x}_i}$ is the expectation of $W_{\boldsymbol{x}_i}$, that is the probability of the word $\boldsymbol{x}_i$ being selected. The feature selection function $g_{\boldsymbol{\phi}}(\cdot)$ is defined as

$$g_{\boldsymbol{\phi}}(\boldsymbol{x}) = \boldsymbol{W} \odot \boldsymbol{x}, \tag{6.4}$$

where $\odot$ denotes element-wise multiplication.

To ensure the model concentrating on a few important words to make predictions, we regularize $\boldsymbol{W}$ by maximizing its entropy conditioned on $\boldsymbol{x}$. The intuition is that most words in the vocabulary are irrelevant or noisy features (e.g. stop words) to text classification tasks [176]. The regularization on $\boldsymbol{W}$ will push the importance scores of most irrelevant words close to 0.5, while making a few important words have relatively high importance scores (close to 1), and the rest unimportant words have low scores (close to 0). Under this constraint, we rewrite the prediction loss $\mathcal{L}_{pred}$ in the objective (6.1) as

$$\begin{aligned}\mathcal{L}_{pred} = & \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathbb{E}_q[\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{W} \odot \boldsymbol{x}), y)] - \beta H_q(\boldsymbol{W} \mid \boldsymbol{x})] \\ & + \mathop{\mathbb{E}}_{(\boldsymbol{x}',y)\sim\mathcal{D}'}[\mathbb{E}_{q'}[\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{W}' \odot \boldsymbol{x}'), y)] - \beta H_{q'}(\boldsymbol{W}' \mid \boldsymbol{x}')],\end{aligned}$$

where $q = q_{\boldsymbol{\phi}}(\boldsymbol{W} \mid \boldsymbol{x})$ and $q' = q_{\boldsymbol{\phi}}(\boldsymbol{W}' \mid \boldsymbol{x}')$ denote the distributions of word masks on the original example $\boldsymbol{x}$ and adversarial example $\boldsymbol{x}'$ respectively, $H_q(\cdot \mid \cdot)$ is the conditional entropy, and $\beta \in \mathbb{R}_+$ is a coefficient.

Connection

FLAT degrades to traditional adversarial training when all words are regarded as equal important (all mask values are 1), and no constraint is added to regularize the importance scores of associated words in original/adversarial example pairs. Traditional adversarial training simply updates the model on the augmented dataset $\mathcal{D} \cup \mathcal{D}'$ by optimizing

$$\min_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{(\boldsymbol{x},y)\sim\mathcal{D}}[\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)] + \mathop{\mathbb{E}}_{(\boldsymbol{x}',y)\sim\mathcal{D}'}[\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}')), y)]. \tag{6.5}$$

With no constraint on model prediction behavior on predicting similar texts, the model robustness is not guaranteed, especially to unforeseen adversarial attacks, as the results shown in experiments.

Datasets	C	L	*#train*	*#dev*	*#test*
SST2	2	19	6920	872	1821
IMDB	2	268	20K	5K	25K
AG	4	32	114K	6K	7.6K
TREC	6	10	5000	452	500

Table 6.1: Summary statistics of the datasets, where C is the number of classes, L is the average sentence length, # counts the number of examples in the *train/dev/test* sets.

Implementation Specification

We utilize the amortized variational inference [156] to approximate word mask distributions, and learn the parameter $\boldsymbol{\phi}$ (global word importance) via an inference network which is a single-layer feedforward neural network. For simplicity, we assume the word masks are mutually independent and each mask is dependent on the word embedding, that is $q_{\boldsymbol{\phi}}(\boldsymbol{W} \mid \boldsymbol{x}) = \prod_{i=1}^{n} q_{\boldsymbol{\phi}}(W_{\boldsymbol{x}_i} \mid \boldsymbol{x}_i)$. We optimize the inference network with the model jointly via stochastic gradient descent, and apply the Gumbel-softmax trick [157, 158] to address the discreteness of sampling binary masks from Bernoulli distributions in backpropagation [176]. In the inference stage, we multiply each word embedding and its global importance score for the model to make predictions.

We first train a base model on the original dataset, and attack the model by manipulating the original training data and collect adversarial examples. Then we train the model on both original and adversarial examples via FLAT. We repeat the attacking and training processes 3-5 times (depending on the model and dataset) until convergence. Note that in each iteration, we augment the original training data with new adversarial examples generated by attacking the latest checkpoint.

6.1.2 Experimental Setup

The proposed method is evaluated with four neural network models in defending two adversarial attacks on four text classification tasks.

Datasets. The four text classification datasets are: Stanford Sentiment Treebank with binary labels SST2 [136], movie reviews IMDB [138], AG's News (AG) [12], and 6-class question classification TREC [159]. For the datasets (e.g. IMDB) without standard train/dev/test split, we hold out a proportion of training examples as the development set. Table 6.1 shows the statistics of the datasets.

Models	SST2	IMDB	AG	TREC
LSTM-base	84.40	88.03	91.08	90.80
LSTM-adv(Textfooler)	82.32	88.79	90.29	87.60
LSTM-adv(PWWS)	82.59	88.37	91.16	89.60
LSTM-FLAT (Textfooler)	**84.79**	**89.17**	91.00	91.00
LSTM-FLAT (PWWS)	83.69	88.52	**91.37**	**91.20**
CNN-base	**84.18**	88.63	91.32	**91.20**
CNN-adv(Textfooler)	82.15	88.81	90.99	89.20
CNN-adv(PWWS)	83.42	88.89	91.30	90.00
CNN-FLAT (Textfooler)	83.09	88.89	**91.64**	89.20
CNN-FLAT (PWWS)	83.31	**88.99**	91.03	89.20
BERT-base	91.32	91.71	93.59	**97.40**
BERT-adv(Textfooler)	91.38	92.50	90.30	96.00
BERT-adv(PWWS)	90.88	**93.14**	93.38	95.20
BERT-FLAT (Textfooler)	**91.54**	92.78	**94.07**	96.20
BERT-FLAT (PWWS)	91.05	93.11	93.09	96.60
DeBERTa-base	94.18	93.80	93.62	96.40
DeBERTa-adv(Textfooler)	94.40	92.86	92.84	95.60
DeBERTa-adv(PWWS)	**94.78**	94.17	92.96	96.40
DeBERTa-FLAT (Textfooler)	94.29	**94.29**	**94.29**	96.40
DeBERTa-FLAT (PWWS)	94.12	94.26	93.82	96.40

Table 6.2: Prediction accuracy (%) of different models on standard test sets.

Models. We evaluate the proposed method with a recurrent neural network [9, LSTM], a convolutional neural network [11, CNN], and two state-of-the-art transformer-based models—BERT [13] and DeBERTa [233]. The LSTM and CNN are initialized with 300-dimensional pretrained word embeddings [139]. We adopt the base versions of both BERT and DeBERTa.

Attack methods. We adopt two adversarial attacks, Textfooler [103] and PWWS [222]. Both methods check the lexical correctness and semantic similarity of adversarial examples with their original counterparts. The adversarial attacks are conducted on the TextAttack benchmark [234] with default settings. During adversarial training, we attack all training data for the SST2 and TREC datasets to collect adversarial examples, while randomly attacking 10K training examples for the IMDB and AG datasets due to computational costs.

6.1.3 Results

We train the four models on the four datasets with different training strategies. The base model trained on the clean data is named with suffix "-base". The model trained via traditional adversarial training is named with suffix "-adv". The model trained via the proposed method is named with suffix "-FLAT". For fairness, traditional adversarial training repeats the attacking and training processes the same times as FLAT. Table 6.2 shows the prediction accuracy of different models on standard test sets. The attack method used for generating adversarial examples during training is noted in brackets. For example, "CNN-FLAT (Textfooler)" means the CNN model trained via FLAT with adversarial examples generated by Textfooler attack. Different from previous defence methods [235, 236] that hurt model performance on clean data, adversarial training ("adv" and "FLAT") does not cause significant model performance drop, and even improves prediction accuracy in some cases. Besides, we believe that producing high-quality adversarial examples for model training would further improve model prediction performance, and leave this to our future work. The rest of this section will focus on evaluating model robustness from both prediction and interpretation perspectives. The evaluation results are recorded in Table 6.3.

		SST2			IMDB			AG			TREC		
Attacks	Models	AA	KT	TI	AA	KT	TI	AA	KT	TI	AA	KT	TI
Textfooler	LSTM-base	5.05	0.46	0.68	0.16	0.53	0.46	45.00	0.76	0.81	44.40	0.62	0.89
	LSTM-adv	12.36	0.49	0.68	29.18	0.60	0.58	48.39	0.76	0.82	51.20	0.51	0.87
	LSTM-FLAT	**17.76**	**0.58**	**0.75**	**31.38**	**0.66**	**0.65**	**54.16**	**0.82**	**0.86**	**55.20**	**0.68**	**0.90**
	CNN-base	1.98	0.46	0.69	3.72	0.64	0.56	8.74	0.55	0.62	45.20	0.68	0.91
	CNN-adv	2.53	0.52	0.72	16.04	0.71	0.65	15.84	0.55	0.62	52.60	0.71	0.92
	CNN-FLAT	**37.07**	**0.70**	**0.82**	**32.62**	**0.76**	**0.75**	**25.18**	**0.61**	**0.67**	**62.20**	**0.87**	**0.96**
	BERT-base	4.72	0.35	0.56	3.84	0.38	0.33	11.84	0.39	0.48	37.60	0.44	0.87
	BERT-adv	5.60	0.33	0.56	23.28	0.34	0.25	30.67	0.25	0.40	40.00	0.52	0.89
	BERT-FLAT	**12.41**	**0.44**	**0.64**	**28.35**	**0.46**	**0.38**	**32.29**	**0.45**	**0.53**	**55.00**	**0.58**	**0.90**
	DeBERTa-base	5.22	0.64	0.76	2.82	0.71	0.72	12.12	0.60	0.63	39.00	0.69	0.92
	DeBERTa-adv	7.96	0.60	0.73	8.38	0.81	0.77	25.70	0.61	0.62	42.80	0.69	0.93
	DeBERTa-FLAT	**11.59**	**0.70**	**0.79**	**24.62**	**0.83**	**0.78**	**31.62**	**0.62**	**0.65**	**49.60**	**0.73**	**0.94**
PWWS	LSTM-base	11.64	0.51	0.71	0.29	0.55	0.48	54.53	0.82	0.86	54.40	0.66	0.90
	LSTM-adv	18.73	0.57	0.74	23.68	0.63	0.61	61.17	0.84	0.88	64.20	0.61	0.88
	LSTM-FLAT	**19.66**	**0.60**	**0.75**	**25.00**	**0.69**	**0.67**	**62.41**	**0.85**	**0.89**	**67.80**	**0.79**	**0.94**
	CNN-base	8.29	0.53	0.72	4.36	0.72	0.59	18.86	0.57	0.64	54.20	0.71	0.91
	CNN-adv	12.63	0.57	0.73	20.64	0.72	0.68	33.21	0.56	0.63	63.00	0.76	0.92
	CNN-FLAT	**14.83**	**0.58**	**0.74**	**20.70**	**0.73**	**0.69**	**71.37**	**0.91**	**0.93**	**65.60**	**0.77**	**0.93**
	BERT-base	11.70	0.37	0.57	7.08	0.36	0.32	32.34	0.28	0.40	51.60	0.52	0.87
	BERT-adv	14.44	0.37	0.58	18.32	0.33	0.29	33.38	0.29	0.40	65.20	0.45	0.86
	BERT-FLAT	**14.61**	**0.44**	**0.64**	**25.08**	**0.41**	**0.36**	**49.16**	**0.30**	**0.42**	**68.20**	**0.64**	**0.90**
	DeBERTa-base	14.17	0.72	0.81	7.04	0.82	0.80	31.30	0.65	0.71	52.80	0.73	0.94
	DeBERTa-adv	15.16	0.65	0.76	18.66	0.81	0.78	53.02	0.65	0.70	63.60	0.64	0.91
	DeBERTa-FLAT	**23.23**	**0.75**	**0.83**	**26.58**	**0.84**	**0.81**	**55.14**	**0.67**	**0.72**	**66.40**	**0.80**	**0.95**

Table 6.3: Model robustness to adversarial attacks in terms of predictions and interpretations. AA: after-attack accuracy (%); KT: Kendall's Tau order rank correlation; TI: top-k intersection ($k = 5$).

Prediction Robustness

We evaluate the prediction robustness of well-trained models by attacking them with adversarial examples crafted from original test examples. The model prediction accuracy on adversarial examples is denoted as after-attack accuracy [103]. In Table 6.3, we omit the attack name in naming a model

("-adv" or "-FLAT") as it is trained with adversarial examples generated by the corresponding attack method (Textfooler or PWWS).

Table 6.3 shows that base models are easily fooled by adversarial examples, achieving much lower after-attack accuracy than other models ("-FLAT" and "-adv") trained with adversarial examples. FLAT consistently outperforms traditional adversarial training, indicating the effectiveness of regularizing model prediction behavior during adversarial training in improving prediction robustness. All the models show better prediction robustness on multiclass topic classification tasks (AG and TREC) than on binary sentiment classification tasks (SST2 and IMDB). Besides, the after-attack accuracy on the IMDB dataset is the lowest for most of the base models (especially LSTM-base). We suspect that IMDB has longer average text length than other datasets, which is easier to find successful adversarial examples. FLAT improves the after-attack accuracy of base models $15\% - 30\%$ on the IMDB dataset.

Interpretation Consistency

Beyond prediction robustness, model robustness can also be evaluated by comparing its decision-makings on predicting original/adversarial example pairs, i.e. interpretation consistency. Note that we obtain interpretations via local post-hoc interpretation methods that identify feature (word/token) attributions to the model prediction per example. We adopt two interpretation methods, IG [25] and LIME [1], which are the representatives from two typical categories, white-box interpretations and black-box interpretations, respectively. IG computes feature attributions by integrating gradients of points along a path from a baseline to the input. LIME explains neural network predictions by fitting a local linear model with input perturbations and producing word attributions. For IG, we evaluate all test examples and their adversarial counterparts. For LIME, we randomly pick up 1000 example pairs for evaluation due to computational costs. We evaluate interpretation consistency under two metrics, Kendall's Tau order rank correlation [101, 95] and Top-k intersection [101, 78]. For both metrics, we compute the interpretation consistency on corresponding labels and take the average over all classes as the overall consistency. Table 6.3 reports the results of IG interpretations. The results of LIME interpretations have similar tendency.

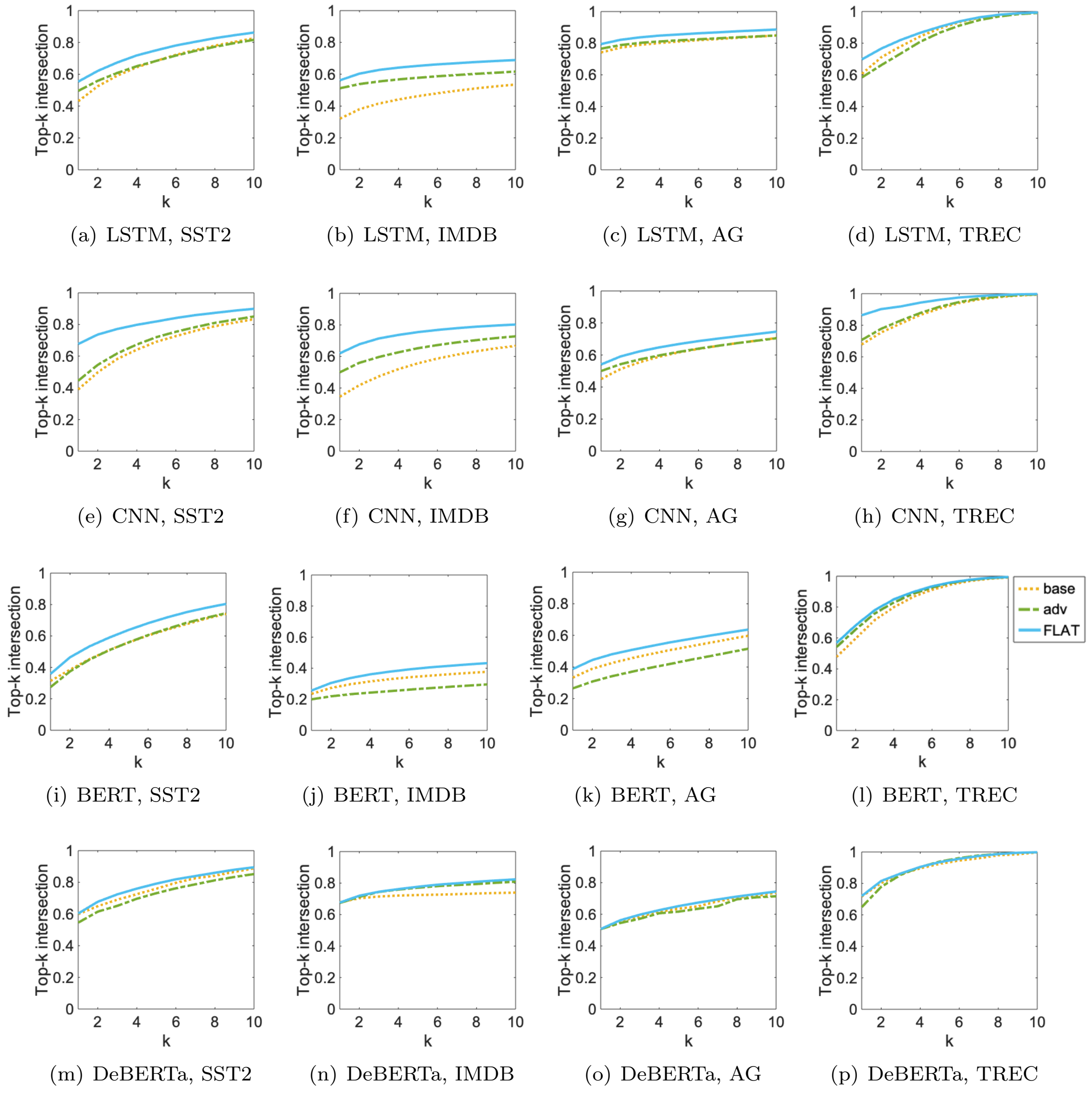
Kendall's Tau order rank correlation. We adopt this metric to compare the overall rankings of word attributions between different interpretations. Higher Kendall's Tau order rank correlation indicates better interpretation consistency. The models ("-FLAT") outperform other baseline models ("-adv" and "-base") with higher Kendall's Tau order rank correlations, showing that FLAT teaches models to behave consistently on predicting similar texts. However, traditional adversarial training cannot guarantee the model robustness being improved as the interpretation discrepancy is even worse than that of base models in some cases, such as LSTM-adv and LSTM-base on the TREC dataset under the Textfooler attack. As FLAT consistently improves model interpretation consistency, no matter which interpretation method (IG or LIME) is used for evaluation, we believe the model robustness has been improved.

Top-k intersection. We adopt this metric to compute the proportion of intersection of top k important features identified by the interpretations of original/adversarial example pairs. Note that we treat synonyms as the "same" words. Higher top-k intersection indicates better interpretation consistency. Table 6.3 records the results of IG interpretations when $k = 5$. The full results of top-k intersection with k increasing from 1 to 10 are in Figure 6.3. Similar to the results of Kendall's Tau order rank correlation, the models ("-FLAT") outperform other baseline models ("-adv" and "-base") with higher top-k intersection rates, showing that they tend to focus on the same words (or their synonyms) in original/adversarial example pairs to make predictions.

6.1.4 Discussion

Visualization of interpretations. Interpretations show the robustness of models ("-FLAT") in producing the same predictions on original/adversarial example pairs with consistent decision-makings. Figure 6.4 visualizes the IG interpretations of LSTM- and CNN-based models on a POSITIVE and NEGATIVE SST2 movie review respectively. The adversarial examples of the two movie reviews were generated by Textfooler. The base models ("-base") were fooled by adversarial examples. Although LSTM-adv correctly predicted the POSITIVE original/adversarial example pair, its interpretations are discrepant with `treat` and `is` identified as the top important word respectively. For the NEGATIVE adversarial example, CNN-adv failed to recognize `bad` and `wicked` as synonyms and labeled them with opposite sentiment polarities, which explains its wrong prediction. Both LSTM-FLAT and CNN-FLAT correctly predicted the original/adversarial example pairs with consistent interpretations.

Transferability of model robustness. The models trained via FLAT show better robustness than baseline models across different attacks. We test the robustness transferability of different models, where "-adv" and "FLAT" were trained with adversarial examples generated by Textfooler, to six unforeseen adversarial attacks: PWWS [222], Gene [102], IGA [237], PSO [238], Clare [104],



(a) LSTM, SST2 (b) LSTM, IMDB (c) LSTM, AG (d) LSTM, TREC

(e) CNN, SST2 (f) CNN, IMDB (g) CNN, AG (h) CNN, TREC

(i) BERT, SST2 (j) BERT, IMDB (k) BERT, AG (l) BERT, TREC

(m) DeBERTa, SST2 (n) DeBERTa, IMDB (o) DeBERTa, AG (p) DeBERTa, TREC

Figure 6.3: Top-k intersection of IG interpretations for different models on the four datasets under the Textfooler attack with k increasing from 1 to 10.

and BAE [223], which generate adversarial examples in different ways (e.g. WordNet swap [239], BERT masked token prediction). Table 6.4 shows the after-attack accuracy of different models on the SST2 test set. The models trained via FLAT achieve higher after-attack accuracy than baseline models, showing better robustness to unforeseen adversarial examples.

Ablation study. The regularizations on word masks and global word importance scores in the objective (6.1) are important for improving model performance. We take the LSTM-FLAT model trained with Textfooler adversarial examples on the SST2 dataset for evaluation. The optimal hy-

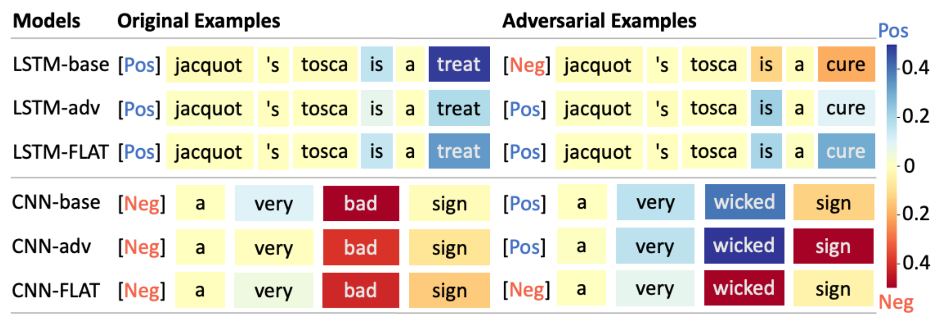

Models
Original Examples
Adversarial Examples
LSTM-base [Pos] jacquot 's tosca is a treat [Neg] jacquot 's tosca is a cure
LSTM-adv [Pos] jacquot 's tosca is a treat [Pos] jacquot 's tosca is a cure
LSTM-FLAT [Pos] jacquot 's tosca is a treat [Pos] jacquot 's tosca is a cure
CNN-base [Neg] a very bad sign [Pos] a very wicked sign
CNN-adv [Neg] a very bad sign [Pos] a very wicked sign
CNN-FLAT [Neg] a very bad sign [Neg] a very wicked sign
Pos
0.4
0.2
0
0.2
0.4
Neg

Figure 6.4: Visualization of IG interpretations. The model predictions are in "[ ]". The color of each block represents the word attribution to the model prediction.

Models	PWWS	Gene	IGA	PSO	Clare	BAE
LSTM-base	11.64	20.26	9.83	5.88	3.02	36.52
LSTM-adv	15.38	25.65	17.02	5.60	3.90	36.35
LSTM-FLAT	**20.48**	**33.44**	**24.22**	**6.53**	**5.55**	**39.87**
CNN-base	8.29	20.32	7.85	5.60	1.48	37.12
CNN-adv	8.68	16.42	6.26	5.60	1.04	35.48
CNN-FLAT	**42.56**	**55.02**	**46.35**	**10.38**	**17.57**	**48.38**
BERT-base	11.70	32.24	9.72	6.26	0.86	35.31
BERT-adv	13.01	34.49	10.87	6.64	1.04	36.74
BERT-FLAT	**15.93**	**35.31**	**15.93**	**9.50**	**5.29**	**37.56**
DeBERTa-base	14.17	37.12	12.19	6.75	0.55	38.61
DeBERTa-adv	17.52	37.18	12.85	7.96	1.07	40.14
DeBERTa-FLAT	**21.80**	**48.16**	**28.17**	**13.01**	**1.37**	**44.54**

Table 6.4: After-attack accuracy (%) of different models to different attacks on the SST2 test set.

perparameters are $\beta = 0.1$, $\gamma = 0.001$. We study the effects by setting β, γ, or both as zero. Table 6.5 shows the results. Only with both regularizations, the model can achieve good prediction performance on the clean test data (standard accuracy) and adversarial examples (after-attack accuracy). We observed that when $\beta = 0$, all masks are close to 1, failing to learn feature importance. When $\gamma = 0$, the model cannot recognize some words and their substitutions as the same important, which is reflected by the larger variance of L1 norm on the difference between the global importance of 1000 randomly sampled words and 10 of their synonyms, as Figure 6.5 shows.

Correlations. The learned global word importance, word frequency, and word substitution frequency in adversarial examples do not show strong correlations with each other. We take the LSTM-FLAT trained with Textfooler on the SST2 dataset for analysis. As the scatter plots in Figure 6.6 show, any two of the three do not have strong correlations. Figure 6.6 (a) shows that the replaced words are not based on their frequency. Figure 6.6 (b) and (c) show that global word importance scores were learned during training, not trivially based on word frequency or substitution frequency.

Hyperparameters	SA	AA
$\beta = 0.1$, $\gamma = 0.001$	84.79	17.76
$\beta = 0.1$, $\gamma = 0$	83.96	9.99
$\beta = 0$, $\gamma = 0.001$	84.34	8.18
$\beta = 0$, $\gamma = 0$	84.40	8.35

Table 6.5: The effects of FLAT regularizations on model performance. SA: standard accuracy (%); AA: after-attack accuracy (%)

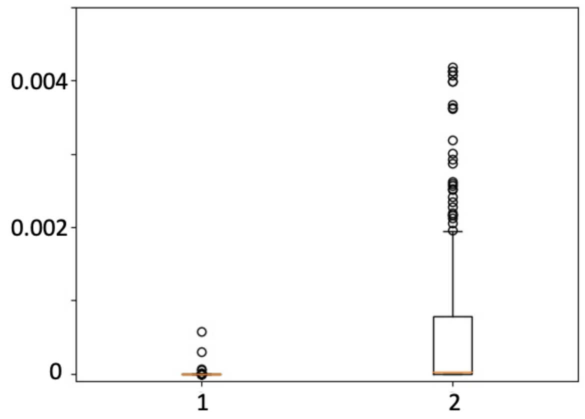


Figure 6.5: Box plot of the L1 norm on the difference between the global importance scores of 1000 randomly sampled words and 10 of their synonyms. 1: $\beta = 0.1$, $\gamma = 0.001$; 2: $\beta = 0.1$, $\gamma = 0$.

It is expected the words that have high substitution frequency in adversarial examples have high importance scores. In addition, FLAT also identifies some important words that are low-frequency or even not replaced by adversarial examples.

6.1.5 Conclusion

In this work, we looked into the robustness of neural network models from both prediction and interpretation perspectives. We proposed a new training strategy, FLAT, to regularize a model prediction behavior so that it produces the same predictions on original/adversarial example pairs

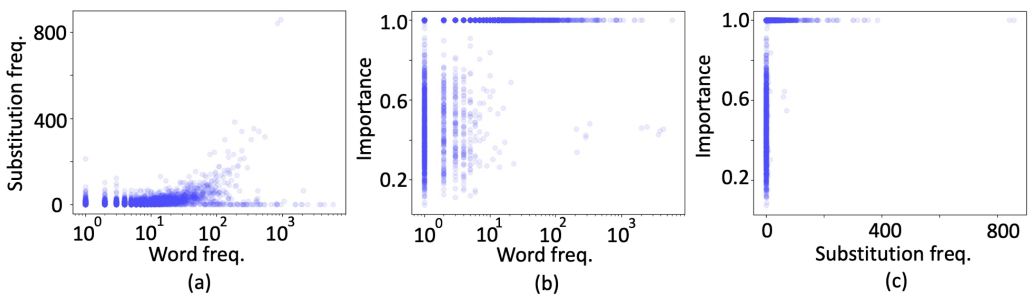


Figure 6.6: Scatter plots: (a) substitution frequency vs. word frequency; (b) global importance vs. word frequency; (c) global importance vs. substitution frequency.

with consistent interpretations. Experiments show the effectiveness of FLAT in improving model robustness to two adversarial attacks on four text classification tasks.

6.2 Pathologies of Pre-trained Language Models in Few-shot Fine-tuning

Pre-trained language models [106, 105, 13] have shown impressive adaptation ability to dowstream tasks, achieving considerable performance even with scarce task-specific training data, i.e., few-shot adaptation [14, 108, 109]. Existing few-shot adaptation techniques broadly fall in fine-tuning and few-shot learning [240, 241, 110]. Specifically, fine-tuning includes directly tuning pre-trained language models with few task-specific examples or utilizing a natural-language prompt to transform downstream tasks to masked language modeling task for better mining knowledge from pre-trained models [242, 243, 244]. Few-shot learning leverages unlabeled data or auxiliary tasks to provide additional information for facilitating model training [245, 246, 247].

Although much success has been made in adapting pre-trained language models to dowstream tasks with few-shot examples, some issues have been reported. Utama et al. [111] found that models obtained from few-shot prompt-based fine-tuning utilize inference heuristics to make predictions on sentence pair classification tasks. Zhao et al. [113] discovered the instability of model performance towards different prompts in few-shot learning. These works mainly look at prompt-based fine-tuning and discover some problems.

This work looks into direct fine-tuning and provides a different perspective on understanding model adaptation behavior via post-hoc explanations [187, 25]. Specifically, post-hoc explanations identify the important features (tokens) contribute to the model prediction per example. We model the statistics of important features over prediction labels via local mutual information (LMI) [189, 190]. We track the change of feature statistics with the model adapting from pre-trained to fine-tuned and compare it with the statistics of few-shot training examples. This provides insights on understanding model adaptation behavior and the effect of training data in few-shot settings.

We evaluate two pre-trained language models, BERT [13] and RoBERTa [105], on three tasks, including sentiment classification, natural language inference, and paraphrase identification. For each task, we test on both in-domain and out-of-domain datasets to evaluate the generalization of model adaptation performance. We discover some interesting observations, some of which may have been overlooked in prior work: (1) without fine-tuning, pre-trained models show strong prediction bias across labels; (2) fine-tuning with a few examples can mitigate the prediction bias, but the model prediction behavior may be pathological by focusing on non-task-related features (e.g. stop words); (3) models adjust their prediction behaviors on different labels asynchronously; (4) models can capture the shallow patterns of training data to make predictions. The insight drawn from the

Datasets	C	L	*#train*	*#dev*	*#test*	Label distribution
IMDB	2	268	19992	4997	24986	Positive: *train*(10036), *dev*(2414), *test*(12535) Negative: *train*(9956), *dev*(2583), *test*(12451)
Yelp	2	138	500000	60000	38000	Positive: *train*(250169), *dev*(29831), *test*(19000) Negative: *train*(249831), *dev*(30169), *test*(19000)
SNLI	3	14	549367	4921	4921	Entailment: *train*(183416), *dev*(1680), *test*(1649) Contradiction: *train*(183187), *dev*(1627), *test*(1651) Neutral: *train*(182764), *dev*(1614), *test*(1651)
MNLI	3	22	391176	4772	4907	Entailment: *train*(130416), *dev*(1736), *test*(1695) Contradiction: *train*(130381), *dev*(1535), *test*(1631) Neutral: *train*(130379), *dev*(1501), *test*(1581)
QQP	2	11	363178	20207	20215	Paraphrases: *train*(134141), *dev*(7435), *test*(7447) Nonparaphrases: *train*(229037), *dev*(12772), *test*(12768)
TPPDB	2	15	42200	4685	4649	Paraphrases: *train*(11167), *dev*(941), *test*(880) Nonparaphrases: *train*(31033), *dev*(3744), *test*(3769)

Table 6.6: Summary statistics of the datasets, where C is the number of classes, L is average sentence length, and # counts the number of examples in the *train/dev/test* sets. For label distribution, the number of examples with the same label in *train/dev/test* is noted in bracket.

above observations is that pursuing model performance with fewer examples is dangerous and may cause pathologies in model prediction behavior. We argue that future research on few-shot fine-tuning or learning should do sanity check on model prediction behavior and ensure the performance gain is based on right reasons.

6.2.1 Setup

Tasks. We consider three tasks: sentiment classification, natural language inference, and paraphrase identification. For sentiment classification, we utilize movie reviews IMDB [138] as the in-domain dataset and Yelp reviews [12] as the out-of-domain dataset. For natural language inference, the task is to predict the semantic relationship between a premise and a hypothesis as entailment, contradiction, or neutral. The Stanford Natural Language Inference (SNLI) corpus [41] and Multi-Genre Natural Language Inference (MNLI) [248] are used as the in-domain and out-of-domain datasets respectively. The task of paraphrase identification is to judge whether two input texts are semantically equivalent or not. We adopt the Quora Question Pairs (QQP) [249] as the in-domain dataset, while using the TwitterPPDB (TPPDB) [250] as the out-of-domain dataset. Table 6.6 shows the statistics of the datasets.

Model	r	In-domain			Out-of-domain		
		IMDB	SNLI	QQP	Yelp	MNLI	TPPDB
BERT	SS	0.41	0.82	0.61	0.53	0.77	0.40
	IG	0.08	0.34	0.19	0.12	0.31	0.10
	Attn	0.07	0.35	0.28	0.12	0.26	0.14
	IMASK	0.09	0.28	0.25	0.09	0.25	0.08
RoBERTa	SS	0.25	0.86	0.53	0.28	0.84	0.28
	IG	0.02	0.36	0.21	0.04	0.38	0.09
	Attn	0.02	0.33	0.26	0.03	0.23	0.09
	IMASK	0.02	0.18	0.18	0.03	0.17	0.05

Table 6.7: AOPC scores of different explanation methods in explaining different models.

Models. We evaluate two pre-trained language models, BERT [13] and RoBERTa [105]. For each task, we train the models on the in-domain training set with different ratio ($r\%, r \in [0, 1]$) of clean examples and then test them on in-domain and out-of-domain test sets.

Explanations. We explain model prediction behavior via post-hoc explanations which identify important features (tokens) in input texts that contribute to model predictions. We test four explanation methods: sampling Shapley (SS) [187], integrated gradients (IG) [25], attentions (Attn) [251], and individual word masks (IMASK) in Section 4.2.2. For each dataset, we randomly select 1000 test examples to generate explanations due to computational costs. We evaluate the faithfulness of these explanation methods via the AOPC metric [129, 73]. Higher AOPC score indicates better explanations. We report the results of AOPC scores when U = 10 in Table 6.7. Sampling Shapley consistently outperforms other three explanation methods in explaining different models on both in-domain and out-of-domain datasets. In the following experiments, we adopt it to explain model predictions.

6.2.2 Experiments

We report the prediction results (averaged across 5 runs) of BERT and RoBERTa trained with different ratio ($r\% : 0 \sim 1\%$) of in-domain training examples on both in-domain and out-of-domain test sets in Table 6.9. Overall, training with more examples, BERT and RoBERTa achieve better prediction accuracy on both in-domain and out-of-domain test sets.

We look into the predictions of models from pre-trained to fine-tuned and analyze model prediction behavior change during adaptation via post-hoc explanations. We observe that pre-trained models without fine-tuning show strong prediction bias across labels. The models fine-tuned with a few examples can quickly mitigate the prediction bias by capturing non-task-related features, leading

Models	IMDB	SNLI	QQP	Yelp	MNLI	TPPDB
BERT	Pos	Neu	Pa	Pos	Neu	Pa
RoBERTa	Pos	Con	Pa	Pos	Con	Pa

Table 6.8: The majority labels of original pre-trained models on different datasets. Pos: postive, Con: contradiction, Neu: neutral, Pa: paraphrases.

Model	r	In-domain						Out-of-domain					
		IMDB		SNLI		QQP		Yelp		MNLI		TPPDB	
		Acc	PB	Acc	PB	Acc	PB	Acc	PB	Acc	PB	Acc	PB
BERT	0	49.73	0.97	35.30	0.65	45.10	0.46	49.86	0.98	32.95	0.95	44.44	0.85
	0.01	-	-	48.45	0.20	65.33	0.45	-	-	34.77	0.92	80.25	0.35
	0.05	60.31	0.41	63.20	0.08	69.82	0.16	61.61	0.09	37.58	0.95	86.26	0.14
	0.1	70.76	0.13	69.13	0.12	73.65	0.04	67.11	0.41	38.27	0.93	86.69	0.07
	0.5	84.71	0.05	77.63	0.06	79.06	0.02	88.19	0.08	55.37	0.45	87.27	0.03
	1	85.46	0.05	80.33	0.06	80.16	0.05	89.09	0.03	58.81	0.34	85.22	0.07
RoBERTa	0	50.17	1.00	33.55	1.00	36.84	1.26	50.00	1.00	33.24	1.02	18.93	1.62
	0.01	-	-	36.27	0.61	66.26	0.54	-	-	32.48	1.00	81.07	0.38
	0.05	58.11	0.61	68.03	0.13	71.64	0.09	58.47	0.71	42.41	0.88	82.30	0.21
	0.1	78.58	0.10	77.04	0.07	76.82	0.04	76.59	0.37	54.72	0.75	83.54	0.21
	0.5	89.56	0.01	83.84	0.04	81.91	0.05	92.54	0.08	66.90	0.37	85.67	0.06
	1	90.34	0.01	85.43	0.03	83.19	0.05	93.76	0.01	70.47	0.20	85.78	0.08

Table 6.9: Prediction accuracy and bias of BERT and RoBERTa trained with different ratio (r%) of in-domain training examples on both in-domain and out-of-domain test sets. Acc: accuracy (%), PB: prediction bias. For PB, darker pink color implies larger prediction bias. Note that we do not consider $r = 0.01$ for IMDB and Yelp datasets because the number of training examples is too small.

to a plausible performance gain. We further quantify the prediction behavior change by comparing the feature statistics of model explanations and training data. We discover that the models adjust their prediction behavior on minority labels first rather than learning information from all classes synchronously and can capture the shallow patterns of training data, which may result in pathologies in predictions.

Prediction bias in pre-trained models

In our pilot experiments, we find the predictions of pre-trained models without fine-tuning are biased across labels (see an example of confusion matrix in Figure 6.7). Original pre-trained models tend to predict all examples with a specific label on each dataset. We denote the specific label as the majority label and the rest labels as minority labels. The results of majority labels are in Table 6.8.

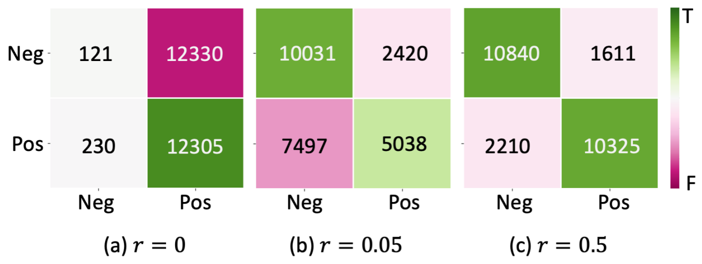


Figure 6.7: Confusion matrix of BERT (with different r) on the IMDB dataset. "Neg" and "Pos" represent negative and positive labels respectively. Vertical and horizontal dimensions show ground-truth and predicted labels respectively. Green and pink colors represent true or false predictions respectively. Darker color indicates larger number.

We propose a metric, prediction bias (PB), to quantify the bias of model predictions across labels,

$$\text{PB} = \left| \frac{T_{i_1} - T_{i_2}}{T_{i_1} + T_{i_2}} - \frac{D_{i_1} - D_{i_2}}{D_{i_1} + D_{i_2}} \right|, \tag{6.6}$$
$$i_1 = \underset{i \in \{1,\ldots,C\}}{\text{argmax}} (T_i), i_2 = \underset{i \in \{1,\ldots,C\}}{\text{argmin}} (T_i)$$

where i_1 and i_2 are the majority and most minority labels respectively. T_i and D_i denote the numbers of model predictions and test examples on label i respectively, and C is number of classes. The range of PB is $[0, 2]$. PB takes 0 if the label distribution of model predictions is consistent with that of data. For balanced dataset, the upper bound of PB is 1, that is all examples are predicted as one label. For imbalanced dataset, PB takes 2 in an extreme case, where the dataset only contains one label of examples, while the model wrongly predicts them as another label. We consider data bias because some datasets (e.g. QQP and TPPDB) have imbalanced label distributions.

The results in Table 6.9 show that both pre-trained BERT and RoBERTa have strong prediction bias on all of the datasets. The prediction bias decreases with models fine-tuned with more examples.

Models make biased predictions by focusing on non-task-related features. To understand which features are associated with model prediction labels, we follow Schuster et al. [189], Du et al. [190] and analyze the statistics of model explanations via local mutual information (LMI). Specifically, we select top k important features in each explanation and get a set of important features ($E = \{e\}$) over all explanations. We empirically take $k = 10$ for the IMDB and Yelp datasets and $k = 6$ for other datasets based on their average sentence lengths. The LMI between a

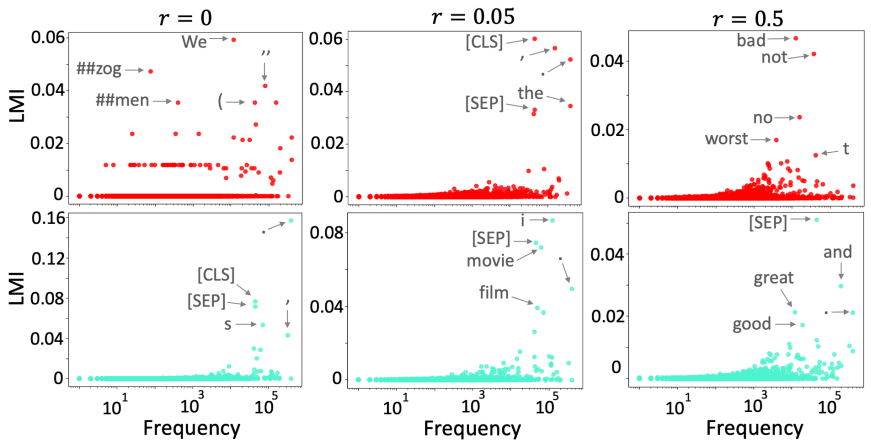


Figure 6.8: LMI distributions based on explanation statistics of BERT on the IMDB dataset with different r. The horizontal axis represents tokens in vocabulary in the ascending order of frequency. The upper and lower plots are on the negative and positive labels respectively. Top 5 tokens are pointed in each plot.

Model	r	IMDB				SNLI						QQP			
		Ori		Data		Ori			Data			Ori		Data	
		Neg	Pos	Neg	Pos	En	Con	Neu	En	Con	Neu	NPa	Pa	NPa	Pa
BERT	0.01	-	-	-	-	0.71	0.43	0.33	0.70	0.42	0.51	0.67	0.32	0.93	0.45
	0.05	2.26	0.45	0.90	0.63	0.58	0.60	0.47	0.31	0.17	0.16	0.49	0.14	0.23	0.22
	0.1	2.00	0.76	0.80	0.54	0.56	0.82	0.45	0.30	0.42	0.46	0.46	0.53	0.19	0.37
	0.5	1.39	0.80	1.16	0.52	0.70	1.51	0.94	0.14	0.54	0.46	0.31	0.67	0.08	0.21
	1	1.21	1.60	0.68	0.86	0.80	1.02	0.65	0.14	0.48	0.52	0.21	1.01	0.00	0.42
RoBERTa	0.01	-	-	-	-	-	0.96	-	0.76	0.52	0.56	-	0.08	0.54	0.36
	0.05	-	0.66	0.17	0.72	-	0.62	-	0.50	0.32	0.67	-	0.43	0.22	0.35
	0.1	-	1.03	0.69	0.71	-	1.05	-	0.22	0.57	0.45	-	1.27	0.17	0.59
	0.5	-	1.33	0.81	0.42	-	2.07	-	0.21	0.60	0.55	-	1.01	0.15	0.69
	1	-	1.41	0.86	0.62	-	0.30	-	0.17	0.32	0.23	-	0.42	0.27	0.23

Table 6.10: The KL divergence between LMI distributions on in-domain datasets. The columns of "Ori" and "Data" show the results with original pre-trained models' explanations or few-shot training data as the reference respectively. Neg: negative, Pos: postive, En: entailment, Con: contradiction, Neu: neutral, NPa: nonparaphrases, Pa: paraphrases. Darker color indicates larger KL divergence.

feature e and a particular label y is

$$\text{LMI}(e, y) = p(e, y) \cdot \log \left(\frac{p(y \mid e)}{p(y)} \right), \tag{6.7}$$

where $p(y \mid e) = \frac{count(e,y)}{count(e)}$, $p(y) = \frac{count(y)}{|E|}$, $p(e, y) = \frac{count(e,y)}{|E|}$, and $|E|$ is the number of occurrences of all features in E. Then we can get a distribution of LMI over all tokens in the vocabulary ($\{w\}$)

Model	r	Yelp				MNLI						TPPDB			
		Ori		Data		Ori			Data			Ori		Data	
		Neg	Pos	Neg	Pos	En	Con	Neu	En	Con	Neu	NPa	Pa	NPa	Pa
BERT	0.01	-	-	-	-	0.35	0.09	0.29	0.40	0.33	0.76	0.74	0.16	1.55	0.18
	0.05	2.20	0.69	0.66	0.43	0.43	0.45	0.41	0.76	0.27	0.63	0.87	0.02	0.58	0.03
	0.1	2.06	0.79	0.37	0.45	0.46	0.49	0.41	0.61	0.40	1.25	0.67	0.21	0.53	0.00
	0.5	1.61	0.93	0.73	0.52	0.92	1.70	0.78	0.82	0.91	1.02	0.93	0.09	0.37	0.04
	1	0.73	1.94	0.46	0.83	0.73	1.31	0.55	0.76	0.69	1.14	0.46	0.54	0.33	0.11
RoBERTa	0.01	-	-	-	-	-	0.95	-	0.33	0.84	0.95	-	0.00	1.55	0.00
	0.05	-	0.38	0.14	0.62	-	0.26	-	0.89	0.22	1.07	-	0.26	1.43	0.39
	0.1	-	0.96	0.30	0.47	-	0.18	-	1.05	0.10	0.62	-	0.39	0.72	0.36
	0.5	-	1.70	0.66	0.43	-	0.70	-	0.87	0.70	0.79	-	0.59	0.79	0.48
	1	-	1.91	0.65	0.78	-	0.18	-	0.72	0.66	0.51	-	0.64	0.95	0.47

Table 6.11: The KL divergence between LMI distributions on out-of-domain datasets. The columns of "Ori" and "Data" show the results with original pre-trained models' explanations or few-shot training data as the reference respectively. Neg: negative, Pos: postive, En: entailment, Con: contradiction, Neu: neutral, NPa: nonparaphrases, Pa: paraphrases. Darker color indicates larger KL divergence.

built upon the dataset, i.e.

$$P_{\text{LMI}}(w, y) = \begin{cases} \text{LMI}(w, y) & \text{if token } w \in E \\ 0 & \text{else} \end{cases} \tag{6.8}$$

We normalize the LMI distribution by dividing each value with the sum of all values.

Figure 6.8 shows LMI distributions of BERT on the IMDB dataset with different r, where top 5 tokens are pointed in each plot (see Table 6.12 for more results on other datasets). When $r = 0$, we can see that BERT makes biased predictions on the positive label (in Table 6.8) by focusing on some non-task-related high-frequency tokens. The top features associated with the negative label include some relatively low-frequency tokens (e.g. ##men, ##zog) which may have been seen by the model during pre-training.

Models adjust prediction bias by capturing non-task-related features on minority labels. Fine-tuning BERT with a few examples ($r = 0.05$, exactly 9 examples) from IMDB can quickly mitigate the prediction bias along with a plausible improvement on prediction accuracy (in Table 6.9). However, Figure 6.8 (the middle upper plot) shows that the model captures non-task-related high-frequency tokens to make predictions on the minority label (negative), implying the performance gain is not reasonable. Only when the model is fine-tuned with more examples ($r = 0.5$), it starts capturing task-specific informative tokens, such as "bad", "good".

Datasets	r	Labels	Top Features
IMDB	0	Neg	we ##zog " ##men (' [SEP] capitalism lynch hell
		Pos	. [CLS] [SEP] s , t movie film plot)
	0.5	Neg	bad not no worst t off terrible nothing stupid boring
		Pos	[SEP] and great . good [CLS] love , film characters
Yelp	0	Neg	. they majestic adds state owners loud dirty priced thai
		Pos	. [CLS] [SEP] , s t for i you m
	0.5	Neg	not no bad t worst never off rude over nothing
		Pos	[SEP] great and good . [CLS] amazing love friendly experience
SNLI	0	En	a [SEP] man the woman dog sitting sits his fire
		Con	[SEP] [CLS] is the a , are in of there
		Neu	. people woman girl are playing looking [CLS] group boy
	0.5	En	[SEP] . [CLS] and is a man there woman people
		Con	the a in [SEP] at sitting with man on playing
		Neu	[SEP] are for . man [CLS] is the a girl
MNLI	0	En	the [SEP] ##ists israel ' recession ata consultants discusses attacked
		Con	[SEP] [CLS] , s to of in . the not
		Neu	. [CLS] they we you people about it really i
	0.5	En	. [CLS] and is [SEP] there are , was of
		Con	the ' . not no t [CLS] don to didn
		Neu	[SEP] [CLS] the for to all when . you it
QQP	0	NPa	? is the a ' what india does quo why
		Pa	[SEP] [CLS] ? in i , of . best s
	0.5	NPa	? what [CLS] is how , why a the .
		Pa	[SEP] quo [CLS] best trump ##ra india life your sex
TPPDB	0	NPa	trump ' the obama " we is russia a says
		Pa	[SEP] . [CLS] , s of in to ##t t
	0.5	NPa	. , [CLS] ? 'ⓐ ; - a is
		Pa	[SEP] trump [CLS] inauguration obama russia repeal ##care cia senate

Table 6.12: Top 10 important tokens for BERT predictions on different labels. Neg: negative, Pos: postive, En: entailment, Con: contradiction, Neu: neutral, NPa: nonparaphrases, Pa: paraphrases.

Quantifying model adaptation behavior

To quantify the model prediction behavior change (in Figure 6.8) during adaptation, we compute the Kullback–Leibler divergence (KLD) between the LMI distributions of the model without/with fine-tuning, i.e. $KL_y(P^0_{LMI}(w, y), P^r_{LMI}(w, y))$. The superscripts ("0" or "r") indicate the ratio of training examples used in fine-tuning. Besides, we also evaluate how much the model prediction behavior is learned from the patterns of training data. Specifically, we compute the LMI distribution of few-shot training examples via Equation 6.7 and Equation 6.8, except that E represents the set of features appearing in those examples. Then we use the LMI distribution of data as the reference and compute the KLD between it and the LMI distribution of model explanations.

Table 6.10 and Table 6.11 records the results of KLD with the LMI distribution of original pre-trained model explanations as the reference (columns of "Ori") or that of training data as the reference (columns of "Data"). Note that we do not have the results of RoBERTa on some labels (e.g. "Neg") in "Ori" columns because the pre-trained RoBERTa does not make any predictions on those labels and we do not have the reference LMI distributions.

Models adjust their prediction behaviors on different labels asynchronously. In "Ori" columns, the KLDs on minority labels are larger than those on majority labels when r is small (e.g. 0.05). The changes of KLDs are discrepant across labels with r increasing. The results show that the models focus on adjusting their prediction behavior on minority labels first rather than learning from all classes synchronously in few-shot settings.

Models can capture the shallow patterns of training data. In "Data" columns, the KLDs on SNLI and QQP are overall smaller than those on IMDB, illustrating that it is easier for models to learn the patterns of datasets on sentence-pair classification tasks. With r increasing, the KLDs on the entailment label of SNLI are smaller than those on other labels, which validates the observations in previous work [111, 252] that models can capture lexical overlaps to predict the entailment label. Another interesting observation is the KLDs on Yelp in "Data" columns are mostly smaller than those on IMDB. This indicates that models may rely on the shallow patterns of in-domain datasets to make predictions on out-of-domain datasets.

6.2.3 Conclusion

In this work, we take a closer look into the adaptation behavior of pre-trained language models in few-shot fine-tuning via post-hoc explanations. We discover many pathologies in model prediction behavior. The insight drawn from our observations is that promising model performance gain in few-shot learning could be misleading. Future research on few-shot fine-tuning or learning requires sanity check on model prediction behavior and some careful design in model evaluation and analysis.

Chapter 7

Conclusion

My research is centered around the interpretability of neural networks with an ultimate goal of developing trustworthy NLP systems. My work approaches this goal along three directions: (1) building interpretable neural language models by transparentizing their decision-making; (2) explaining neural language models and evaluating their explanations for model understanding; (3) diagnosing and debugging models via explanations for trustworthy NLP.

Building interpretable neural language models by transparentizing their decision-making. Designing inherently interpretable neural networks is challenging for NLP, requiring much engineering effort. I propose to improve the interpretability of existing neural language models via data augmentations in Section 3.1 and variational word masks (VMASK) in Section 3.2. The basic idea is to teach models to make predictions based on important features, hence improving their interpretability. The proposed methods are model-agnostic and can be applied to any neural text classifiers.

Explaining neural language models and evaluating their explanations for model understanding. The main challenge of explaining blackbox models is to produce explanations that are faithful to model predictions and comprehensible to humans. I propose a hierarchical explanation method based on feature interaction detection in Section 4.1. The hierarchical explanation visualizes how words and phrases are interacted and combined at different levels of the hierarchy, providing a comprehensive picture of the model decision-making to users. To efficiently explain sentence-pair modeling tasks, I propose the Group Masks method, which implicitly detects word correlations by grouping correlated words from input texts together and measures their contributions to model predictions in Section 4.2. This method is computationally efficient and provides faithful explanations for sentence-pair modeling tasks (e.g., NLI, paraphrase identification). In addition to explaining prediction labels, I also propose to explain predictive uncertainty to understand model prediction behavior in Section 4.3. In terms of explanation evaluation, I focus on the open research question—to what extent an explanation explains a label, or how much new information (e.g., background knowledge, reasoning) an explanation supplies to explain a label beyond the original input. To quantify the new information in explanations, I propose an information-theoretic metric called REV in Section 5.1. REV demonstrates consistent evaluations with human judgements and offers deeper insights into a model's reasoning and prediction processes.

Diagnosing and debugging models via explanations for trustworthy NLP. I propose to improve model robustness through the lens of explanations in Section 6.1. The idea is to make models behave consistently on original/adversarial examples, hence producing the same predictions (***what***) based on the same reasons (***how***). Experiments show the effectiveness of the proposed method in improving the robustness with respect to both predictions and explanations of four neural network models (LSTM, CNN, BERT, and DeBERTa) to two adversarial attacks on four text classification tasks. I also utilize explanations to find the pathologies of pre-trained language models in few-shot fine-tuning Section 6.2. The observations provide insights for future research on building more reliable and trustworthy models with limited data.

This dissertation is expected to benefit NLP and AI developers, providing them with a better understanding of neural network models and helping them build trustworthy and reliable intelligent systems.

Outlook

Looking forward, my future research is committed to formalizing a holistic trustworthy AI framework that bridges intelligent machines, system developers, and end users.

- **Developing the next generation of explanations.** In the short term, I plan to extend my expertise on explanation generation and evaluation and move towards breaking down the barriers in explaining and understanding neural network models. To achieve trustworthy AI, there are additional requirements of model explanations beyond current considerations (e.g., faithfulness, robustness), which I have investigated in this dissertation. For example, model explanations are expected to be informative (to model reasoning), diverse (to model behavior), comprehensive (to data distribution), and controlled (to task). Existing explanations (e.g., saliency maps) barely show the reasoning process of a model. This poses a challenge for humans to visualize the model's reasoning process, which may cause potential issues due to the gap between model reasoning and human understanding. Emerging natural language explanations are promising in explaining models' reasoning. However, my work in Section 5.1 shows that they can be uninformative, providing no new information to fill the reasoning gap. I plan to incorporate additional knowledge (e.g., common sense, knowledge base) into models' reasoning and prediction processes, as well as explanation generation, hence producing more reasonable predictions and informative explanations. Moreover, I am excited about developing diverse explanations that explain model behavior from a variety of perspectives. My preliminary work in Section 4.3 has pushed this direction along the dimension of explaining model predictive uncertainty. My future work will further investigate this dimension and explore other dimensions, such as explaining model predictive dynamics. Additionally, I will extend my research from sample-wise explanations to corpus-wise, in order to gain a comprehensive understanding on model prediction behavior over the whole data distribution. In addition, I would like to explore the direction of controlled explanation generation, which is vital while less explored. For example, an explanation may explain a model prediction from unwanted perspectives which cannot be accepted by end users. Even worse, a generated explanation could be harmful, containing bias (e.g., racism, sexism) or misinformation, hence leading to ethical and social risks. I also want to collaborate with people from social science, cognitive science, and the humanities, to formalize a holistic methodology for model explanation generation and evaluation.

- **Unifying machines, developers, and users for trustworthy AI.** In the long term, I believe fostering model interpretability is indispensable to trustworthy AI. It builds connections between intelligent machines, system developers, and end users. My future research will focus on machine-developer and machine-user communication and interaction.

  Interpretability for system development. Interpretations provide system developers the access to understand neural network models, which in turn allows them to develop better models. My prior work in Chapter 6 has taken steps towards diagnosing and debugging models via explanations. Looking forward, I would like to contribute to developing trustworthy systems by incorporating model interpretability with other properties, including robustness, fairness, and privacy. In practice, coordinating those properties could be challenging and there may be some trade-offs between them. For example, interpretability encourages models to be transparent, which may lead to leaking sensitive information the model learns from data, causing privacy issues. I look forward to expanding my collaborations on robustness and fairness, security and privacy, as well as cyber-physical systems, to gain a better understanding on the necessities of different system developers, and develop solutions for specific deployments.

Interpretability for human-centric AI. The overarching goal of AI development is to serve humanity, improving our daily lives (e.g., autonomous driving). Thus, my future research on model interpretability will surround human-centric AI. As we are stepping into an era where intelligent systems have reasoning abilities, interpretability is facing the challenge of neural models growing ever more sophisticated and even going beyond human understanding. I plan to look deeper into the interplay between model reasoning and human understanding, and collaborate with people from AI ethics, cognitive science, and psychology, developing a formalism to understand and regularize model behaviors. In addition, I am eager to explore human–computer interaction (HCI) and believe interpretability is an important step towards this direction, bridging the gap between machines and end users. I would like to develop a responsive and interactive AI framework, where human feedback can guide and improve intelligent machines, and machine predictions can assist human decision-makers.

These directions are challenging and fascinating. I believe their significance to the evolution of AI and broader impacts on the society. In the future, I plan to dive into these directions and look for interdisciplinary collaborations to conduct research that will generate far-reaching positive effects.

References

[1] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

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

[3] Chandan Singh, W James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. *arXiv preprint arXiv:1806.05337*, 2018.

[4] John Hewitt, Kawin Ethayarajh, Percy Liang, and Christopher Manning. Conditional probing: measuring usable information beyond a baseline. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1639, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.122. URL `https://aclanthology.org/2021.emnlp-main.122`.

[5] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3): 55–75, 2018.

[6] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[7] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3): 3713–3744, 2023.

[8] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

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

[10] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

[11] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[12] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

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

Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

[14] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[16] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[17] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[18] Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International conference on machine learning*, pages 8116–8126. PMLR, 2020.

[19] Piyawat Lertvittayakumjorn and Francesca Toni. Explanation-based human debugging of nlp models: A survey. *Transactions of the Association for Computational Linguistics*, 9:1508–1528, 2021.

[20] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.

[21] Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 624–635, 2021.

[22] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai—explainable artificial intelligence. *Science Robotics*, 4(37):eaay7120, 2019.

[23] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16:1–85, 2022.

[24] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[25] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[26] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[27] Sofia Serrano and Noah A Smith. Is attention interpretable? *arXiv preprint arXiv:1906.03731*, 2019.

[28] Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[29] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. *arXiv preprint arXiv:2001.00396*, 2020.

[30] Bhargavi Paranjape, Mandar Joshi, John Thickstun, Hannaneh Hajishirzi, and Luke 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)*, pages 1938–1952, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.153. URL `https://aclanthology.org/2020.emnlp-main.153`.

[31] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18:1–78, 2018.

[32] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

[33] Dheeraj Rajagopal, Vidhisha Balachandran, Eduard Hovy, and Yulia Tsvetkov. Selfexplain: A self-explaining architecture for neural text classifiers. *arXiv preprint arXiv:2103.12279*, 2021.

[34] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? *arXiv preprint arXiv:2004.03685*, 2020.

[35] Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. *arXiv preprint arXiv:1906.08286*, 2019.

[36] Tianlu Wang, Diyi Yang, and Xuezhi Wang. Identifying and mitigating spurious correlations for improving robustness in nlp models. *arXiv preprint arXiv:2110.07736*, 2021.

[37] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.

[38] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[39] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[40] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

[41] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL `https://aclanthology.org/D15-1075`.

[42] Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-SNLI: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pages 9539–9549, 2018.

[43] Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. Learning credible deep neural networks with rationale regularization. *arXiv preprint arXiv:1908.05601*, 2019.

[44] Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.

[45] Gregory Plumb, Maruan Al-Shedivat, Ángel Alexander Cabrera, Adam Perer, Eric Xing, and Ameet Talwalkar. Regularizing black-box models for improved interpretability. *Advances in Neural Information Processing Systems*, 33:10526–10536, 2020.

[46] Sawan Kumar and Partha Talukdar. Nile: Natural language inference with faithful natural language explanations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8730–8742, 2020.

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

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

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

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

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

[52] Xisen Jin, Zhongyu Wei, Junyi Du, Xiangyang Xue, and Xiang Ren. Towards hierarchical importance attribution: Explaining compositional semantics for neural sequence models. *arXiv preprint arXiv:1911.06194*, 2019.

[53] Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126, 2017.

[54] Adrien Bibal, Rémi Cardon, David Alfter, Rodrigo Wilkens, Xiaoou Wang, Thomas François, and Patrick Watrin. Is attention explanation? an introduction to the debate. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3889–3900, 2022.

[55] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*, 2019.

[56] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[57] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[58] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[59] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

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

[61] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[62] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[63] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

[64] Seojin Bang, Pengtao Xie, Heewook Lee, Wei Wu, and Eric Xing. Explaining a black-box using deep variational information bottleneck approach. *arXiv preprint arXiv:1902.06918*, 2019.

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

[66] Yang Zhang, Ashkan Khakzar, Yawei Li, Azade Farshad, Seong Tae Kim, and Nassir Navab. Fine-grained neural network explanation by identifying input features with predictive information. *Advances in Neural Information Processing Systems*, 34:20040–20051, 2021.

[67] Michael Tsang, Youbang Sun, Dongxu Ren, and Yan Liu. Can i trust you more? model-agnostic hierarchical explanations. *arXiv preprint arXiv:1812.04801*, 2018.

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

[69] Jianbo Chen and Michael Jordan. Ls-tree: Model interpretation when the data are linguistic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3454–3461, 2020.

[70] Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.21. URL `https://aclanthology.org/2020.emnlp-main.21`.

[71] Jiacheng Xu, Shrey Desai, and Greg Durrett. Understanding neural abstractive summarization models via uncertainty. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6275–6281, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.508. URL `https://aclanthology.org/2020.emnlp-main.508`.

[72] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. Eraser: A benchmark to evaluate rationalized nlp models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, 2020.

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

[74] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. A unified view of gradient-based attribution methods for deep neural networks. In *NIPS Workshop on Interpreting, Explaining and Visualizing Deep Learning*, 2017.

[75] Peter Hase and Mohit Bansal. Evaluating explainable ai: Which algorithmic explanations help users predict model behavior? *arXiv preprint arXiv:2005.01831*, 2020.

[76] Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Samuel J Gershman, and Finale Doshi-Velez. Human evaluation of models built for interpretability. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, pages 59–67, 2019.

[77] Siddhant Arora, Danish Pruthi, Norman Sadeh, William W Cohen, Zachary C Lipton, and Graham Neubig. Explain, edit, and understand: Rethinking user study design for evaluating model explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5277–5285, 2022.

[78] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *AAAI*, 2019.

[79] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. *Advances in neural information processing systems*, 32, 2019.

[80] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. How can we fool lime and shap? adversarial attacks on post hoc explanation methods. *arXiv preprint arXiv:1911.02508*, 1, 2019.

[81] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. *Advances in Neural Information Processing Systems*, 32, 2019.

[82] Junlin Wang, Jens Tuyls, Eric Wallace, and Sameer Singh. Gradient-based analysis of nlp models is manipulable. *arXiv preprint arXiv:2010.05419*, 2020.

[83] Sanchit Sinha, Hanjie Chen, Arshdeep Sekhon, Yangfeng Ji, and Yanjun Qi. Perturbing inputs for fragile interpretations in deep natural language processing. *arXiv preprint arXiv:2108.04990*, 2021.

[84] Ruixuan Tang, Hanjie Chen, and Yangfeng Ji. Identifying the source of vulnerability in explanation discrepancy: A case study in neural text classification. *arXiv preprint arXiv:2212.05327*, 2022.

[85] Danish Pruthi, Rachit Bansal, Bhuwan Dhingra, Livio Baldini Soares, Michael Collins, Zachary C Lipton, Graham Neubig, and William W Cohen. Evaluating explanations: How much do explanations from the teacher aid students? *Transactions of the Association for Computational Linguistics*, 10:359–375, 2022.

[86] Marcos V Treviso and André FT Martins. The explanation game: Towards prediction explainability through sparse communication. *arXiv preprint arXiv:2004.13876*, 2020.

[87] Sharan Narang, Colin Raffel, Katherine Lee, Adam Roberts, Noah Fiedel, and Karishma Malkan. Wt5?! training text-to-text models to explain their predictions. *arXiv preprint arXiv:2004.14546*, 2020.

[88] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, 2019.

[89] Faeze Brahman, Vered Shwartz, Rachel Rudinger, and Yejin Choi. Learning to rationalize for nonmonotonic reasoning with distant supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12592–12601, 2021.

[90] Peter Hase, Shiyue Zhang, Harry Xie, and Mohit Bansal. Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4351–4367,

Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.390. URL `https://aclanthology.org/2020.findings-emnlp.390`.

[91] Sarah Wiegreffe, Ana Marasović, and Noah A. Smith. Measuring association between labels and free-text rationales. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10266–10284, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.804. URL `https://aclanthology.org/2021.emnlp-main.804`.

[92] Aaron Chan, Shaoliang Nie, Liang Tan, Xiaochang Peng, Hamed Firooz, Maziar Sanjabi, and Xiang Ren. Frame: Evaluating simulatability metrics for free-text rationales. *arXiv preprint arXiv:2207.00779*, 2022.

[93] Jiao Sun, Swabha Swayamdipta, Jonathan May, and Xuezhe Ma. Investigating the benefits of free-form rationales. *arXiv preprint arXiv:2206.11083*, 2022.

[94] Ye Zhang, Iain Marshall, and Byron C Wallace. Rationale-augmented convolutional neural networks for text classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 795. NIH Public Access, 2016.

[95] Akhilan Boopathy, Sijia Liu, Gaoyuan Zhang, Cynthia Liu, Pin-Yu Chen, Shiyu Chang, and Luca Daniel. Proper network interpretability helps adversarial robustness in classification. In *International Conference on Machine Learning*, pages 1014–1023. PMLR, 2020.

[96] Christopher J Anders, Leander Weber, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Finding and removing clever hans: Using explanation methods to debug and improve deep models. *Information Fusion*, 77:261–295, 2022.

[97] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[98] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[99] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.

[100] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[101] Jiefeng Chen, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. Robust attribution regularization. *Advances in Neural Information Processing Systems*, 32, 2019.

[102] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *EMNLP*, 2018.

[103] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[104] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. Contextualized perturbation for textual adversarial attack. In *NAACL-HLT*, 2021.

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

[106] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`.

[107] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL `https://aclanthology.org/2020.acl-main.740`.

[108] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL `https://aclanthology.org/2021.eacl-main.20`.

[109] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. 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)*, pages 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL `https://aclanthology.org/2021.acl-long.295`.

[110] Yiming Chen, Yan Zhang, Chen Zhang, Grandee Lee, Ran Cheng, and Haizhou Li. Revisiting self-training for few-shot learning of language model. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9125–9135, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.718. URL `https://aclanthology.org/2021.emnlp-main.718`.

[111] Prasetya Ajie Utama, Nafise Sadat Moosavi, Victor Sanh, and Iryna Gurevych. Avoiding inference heuristics in few-shot prompt-based finetuning. *arXiv preprint arXiv:2109.04144*, 2021.

[112] Tingting Ma, Jin-ge Yao, Chin-Yew Lin, and Tiejun Zhao. Issues with entailment-based zero-shot text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 786–796, 2021.

[113] Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*, 2021.

[114] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[115] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

[116] Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. Towards interpreting and mitigating shortcut learning behavior of nlu models. *arXiv preprint arXiv:2103.06922*, 2021.

[117] Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. Generating data to mitigate spurious correlations in natural language inference datasets. *arXiv preprint arXiv:2203.12942*, 2022.

[118] Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. Hypothesis only baselines in natural language inference. *arXiv preprint arXiv:1805.01042*, 2018.

[119] Ronen Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

[120] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[121] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[122] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[123] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570, 2017.

[124] Rie Johnson and Tong Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373*, 2016.

[125] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

[126] Bin Yu and Karl Kumbier. Three principles of data science: predictability, computability, and stability (pcs). *arXiv preprint arXiv:1901.08152*, 2019.

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

[128] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

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

[130] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

[131] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[132] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. 2010.

[133] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[134] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[135] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

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

[137] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

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

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

[140] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[141] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/E17-2068`.

[142] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[143] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pages 7775–7784, 2018.

[144] Gregory Plumb, Maruan Al-Shedivat, Eric Xing, and Ameet Talwalkar. Regularizing black-box models for improved interpretability. *arXiv preprint arXiv:1902.06787*, 2019.

[145] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[146] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

[147] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[148] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[149] Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

[150] Yoon Kim, Sam Wiseman, and Alexander M Rush. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*, 2018.

[151] Xiang Lisa Li and Jason Eisner. Specializing word embeddings (for parsing) by information bottleneck. *arXiv preprint arXiv:1910.00163*, 2019.

[152] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027, 2016.

[153] Jasmijn Bastings, Wilker Aziz, and Ivan Titov. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1284. URL `https://www.aclweb.org/anthology/P19-1284`.

[154] Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.262. URL `https://www.aclweb.org/anthology/2020.emnlp-main.262`.

[155] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[156] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[157] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[158] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[159] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[160] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural language object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564, 2016.

[161] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

[162] Guillermo Owen. Multilinear extensions of games. *Management Science*, 18(5-part-2):64–79, 1972.

[163] Michel Grabisch. K-order additive discrete fuzzy measures and their representation. *Fuzzy sets and systems*, 92(2):167–189, 1997.

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

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

[166] Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

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

[168] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[169] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[170] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Generating token-level explanations for natural language inference. *arXiv preprint arXiv:1904.10717*, 2019.

[171] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.494. URL `https://www.aclweb.org/anthology/2020.acl-main.494`.

[172] Michael Tsang, Dehua Cheng, Hanpeng Liu, Xue Feng, Eric Zhou, and Yan Liu. Feature interaction interpretability: A case for explaining ad-recommendation systems via neural interaction detection. *arXiv preprint arXiv:2006.10966*, 2020.

[173] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *EMNLP*, 2015.

[174] Wenpeng Yin and Hinrich Schütze. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911, 2015.

[175] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

[176] Hanjie Chen and Yangfeng Ji. Learning variational word masks to improve the interpretability of neural text classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4236–4251, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.347. URL `https://www.aclweb.org/anthology/2020.emnlp-main.347`.

[177] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.

[178] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[179] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[180] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[181] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[182] Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. Calibrated language model fine-tuning for in- and out-of-distribution data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1326–1340, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.102. URL `https://aclanthology.org/2020.emnlp-main.102`.

[183] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021.

[184] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. *arXiv preprint arXiv:2006.06848*, 2020.

[185] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5578–5593, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.494. URL `https://aclanthology.org/2020.acl-main.494`.

[186] Hanjie Chen, Song Feng, Jatin Ganhotra, Hui Wan, Chulaka Gunasekara, Sachindra Joshi, and Yangfeng Ji. Explaining neural network predictions on sentence pairs via learning word-group masks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3917–3930, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.306. URL `https://aclanthology.org/2021.naacl-main.306`.

[187] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[188] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399, 2017.

[189] Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. Towards debiasing fact verification models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3419–3425, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1341. URL `https://aclanthology.org/D19-1341`.

[190] Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. Towards interpreting and mitigating shortcut learning behavior of NLU models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.71. URL `https://aclanthology.org/2021.naacl-main.71`.

[191] Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. *Advances in Neural Information Processing Systems*, 28:3474–3482, 2015.

[192] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[193] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[194] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pages 2805–2814. PMLR, 2018.

[195] Jeremiah Zhe Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax-Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108*, 2020.

[196] Iker Perez, Piotr Skalski, Alec Barns-Graham, Jason Wong, and David Sutton. Attribution of predictive uncertainties in classification models. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

[197] Dan Ley, Umang Bhatt, and Adrian Weller. Diverse, global and amortised counterfactual explanations for uncertainty estimates. *arXiv preprint arXiv:2112.02646*, 2021.

[198] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1407. URL `https://aclanthology.org/D18-1407`.

[199] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL `https://aclanthology.org/2020.acl-main.442`.

[200] Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. A theory of usable information under computational constraints. In *International Conference on Learning Representations*, 2020.

[201] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[202] Sarah Wiegreffe and Ana Marasović. Teach me to explain: A review of datasets for explainable natural language processing, 2021.

[203] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[204] Thomas M Cover and Joy A Thomas. *Elements of information theory.* Wiley, 2nd edition, 2006.

[205] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with v-usable information. In *International Conference on Machine Learning*, pages 5988–6008. PMLR, 2022.

[206] Jifan Chen, Eunsol Choi, and Greg Durrett. Can nli models verify qa systems' predictions? In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3841–3854, 2021.

[207] Dorottya Demszky, Kelvin Guu, and Percy Liang. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*, 2018.

[208] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[209] Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. Explanations for CommonsenseQA: New Dataset and 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)*, pages 3050–3065, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.238. URL `https://aclanthology.org/2021.acl-long.238`.

[210] Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. Quartz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946, 2019.

[211] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[212] Sumithra Bhakthavatsalam, Chloe Anastasiades, and Peter Clark. Genericskb: A knowledge base of generic statements. *arXiv preprint arXiv:2005.00660*, 2020.

[213] Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. Reframing human-AI collaboration for generating free-text explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 632–658, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.47. URL `https://aclanthology.org/2022.naacl-main.47`.

[214] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[215] Sarthak Jain, Sarah Wiegreffe, Yuval Pinter, and Byron C Wallace. Learning to faithfully rationalize by construction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4459–4473, 2020.

[216] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2019.

[217] Aaron Chan, Maziar Sanjabi, Lambert Mathias, Liang Tan, Shaoliang Nie, Xiaochang Peng, Xiang Ren, and Hamed Firooz. Unirex: A unified learning framework for language model rationale extraction. In *International Conference on Machine Learning*, pages 2867–2889. PMLR, 2022.

[218] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

[219] Suranjana Samanta and Sameep Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

[220] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE SPW*, 2018.

[221] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[222] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *ACL*, 2019.

[223] Siddhant Garg and Goutham Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *EMNLP*, 2020.

[224] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging NLP models. In *ACL*, 2018.

[225] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL-HLT*, 2018.

[226] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*, 2019.

[227] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41, 2020.

[228] Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *EMNLP-IJCNLP*, 2019.

[229] Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. Frequency-guided word substitutions for detecting textual adversarial examples. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 171–186, Online, April 2021. Association for Computational Linguistics. URL `https://aclanthology.org/2021.eacl-main.13`.

[230] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In *EMNLP-IJCNLP*, 2019.

[231] Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. Achieving verified robustness to symbol substitutions via interval bound propagation. In *EMNLP-IJCNLP*, 2019.

[232] Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. *arXiv preprint arXiv:2002.06622*, 2020.

[233] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*, 2021.

[234] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, 2020.

[235] Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. Robust encodings: A framework for combating adversarial typos. In *ACL*, 2020.

[236] Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. Defense against synonym substitution-based adversarial attacks via Dirichlet neighborhood ensemble. In *ACL*, 2021.

[237] Xiaosen Wang, Hao Jin, and Kun He. Natural language adversarial attacks and defenses in word level. *arXiv preprint arXiv:1909.06723*, 2019.

[238] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. In *ACL*, 2020.

[239] George A Miller. *WordNet: An electronic lexical database.* MIT press, 1998.

[240] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.346. URL `https://aclanthology.org/2020.emnlp-main.346`.

[241] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.185. URL `https://aclanthology.org/2021.naacl-main.185`.

[242] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL `https://aclanthology.org/D19-1250`.

[243] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020. doi: 10.1162/tacl_a_00324. URL `https://aclanthology.org/2020.tacl-1.28`.

[244] Chengyu Wang, Jianing Wang, Minghui Qiu, Jun Huang, and Ming Gao. TransPrompt: Towards an automatic transferable prompting framework for few-shot text classification. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2792–2802, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.221. URL `https://aclanthology.org/2021.emnlp-main.221`.

[245] Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. Meta label correction for noisy label learning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[246] Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. Meta self-training for few-shot neural sequence labeling. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1737–1747, 2021.

[247] Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Veselin Stoyanov, and Alexis Conneau. Self-training improves pre-training for natural language understanding. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5408–5418, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.426. URL `https://aclanthology.org/2021.naacl-main.426`.

[248] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL `https://aclanthology.org/N18-1101`.

[249] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. Quora question pairs. 2017.

[250] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1126. URL `https://aclanthology.org/D17-1126`.

[251] James Mullenbach, Sarah Wiegreffe, Jon Duke, Jimeng Sun, and Jacob Eisenstein. Explainable prediction of medical codes from clinical text. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1101–1111, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1100. URL `https://aclanthology.org/N18-1100`.

[252] Yixin Nie, Yicheng Wang, and Mohit Bansal. Analyzing compositionality-sensitivity of nli models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6867–6874, Jul. 2019. doi: 10.1609/aaai.v33i01.33016867. URL `https://ojs.aaai.org/index.php/AAAI/article/view/4663`.