Imperial College London
Department of Computing

Explainable NLP for Human-AI Collaboration

Piyawat Lertvittayakumjorn

December 2021

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
and the Diploma of Imperial College London

Statement of Originality

I declare that this thesis was composed by myself and that the work it presents is my own, except where otherwise stated. I certify that all material in this thesis which is not my own work has been properly acknowledged.

Piyawat Lertvittayakumjorn

Abstract

With more data and computing resources available these days, we have seen many novel Natural Language Processing (NLP) models breaking one performance record after another. Some of them even outperform human performance in some specific tasks. Meanwhile, many researchers have revealed weaknesses and irrationality of such models, e.g., having biases against some sub-populations, producing inconsistent predictions, and failing to work effectively in the wild due to overfitting. Therefore, in real applications, especially in high-stakes domains, humans cannot rely carelessly on predictions of NLP models, but they need to work closely with the models to ensure that every final decision made is accurate and benevolent.

In this thesis, we devise and utilize explainable NLP techniques to support human-AI collaboration using text classification as a target task. Overall, our contributions can be divided into three main parts. First, we study how useful explanations are for humans according to three different purposes: revealing model behavior, justifying model predictions, and helping humans investigate uncertain predictions. Second, we propose a framework that enables humans to debug simple deep text classifiers informed by model explanations. Third, leveraging on computational argumentation, we develop a novel local explanation method for pattern-based logistic regression models that align better with human judgements and effectively assist humans to perform an unfamiliar task in real-time. Altogether, our contributions are paving the way towards the synergy of profound knowledge of human users and the tireless power of AI machines.

Copyright Declaration

The copyright of this thesis rests with the author and is made available under a Creative Commons Attribution Non-Commercial No Derivatives licence. Researchers are free to copy, distribute or transmit the thesis on the condition that they attribute it, that they do not use it for commercial purposes and that they do not alter, transform or build upon it. For any reuse or redistribution, researchers must make clear to others the licence terms of this work.

Acknowledgements

First and foremost, I would like to express my profound gratitude to my principal supervisor, Professor Francesca Toni, for her invaluable guidance and support since I started joining her research group. I do enjoy every weekly meeting we have where I update my progress and learn from her considerable expertise in the fields of explainable artificial intelligence and computational argumentation. Furthermore, whenever I face any difficult situations during my study, her advice always helps me go through such situations safe and sound. I cannot say enough how grateful I am for having Professor Francesca Toni as my principal PhD supervisor.

Secondly, I would like to thank my second supervisor, Professor Lucia Specia, for every piece of helpful suggestion she gave me from her vast and comprehensive knowledge in natural language processing. Having her as my second supervisor makes me feel more confident to tackle challenging problems in NLP. Also, having a chance to co-author a paper with her is truly my pleasure.

I would also like to thank Professor Alessandra Russo and Dr Simone Stumpf who are the examiners of my PhD study. During the viva, I received numerous interesting comments, ideas, and questions from both examiners. All of the suggestions have tremendously aided in the polishing of my thesis.

Next, I would like to thank Professor Yike Guo, who accepted me as his PhD student in the first place. During the year I worked with him, he taught me to stay focused, be ambitious, and target impactful scientific problems. These lessons are valuable to me and they are applicable beyond the PhD study as well.

Besides, many administrative staff and CSG staff at the Department of Computing and the Data Science Institute also supported me in many ways. Dr Amani El-Kholy is the person who I must say thank you to as she did save my life on many occasions. I said to her many times that the department is so fortunate to have her as the PhD administrator. Also, I would like to thank Teresa Ng, Diana O'Malley, Fay Miller, Duncan White, Geoffrey Bruce, and Lloyd Kamara for all their supports.

Additionally, I would like to take this opportunity to express my deep gratitude to Anandamahidol Foundation for awarding me a prestigious scholarship to pursue my master and PhD studies abroad. I am very honoured to be the recipient of this scholarship and I look forward to giving back to the community after I graduate.

I can survive this PhD journey partly owing to many people who listened to me, shared their views, provided me advice, and cheered me up when I was down. I would like to thank all members of CLArg (e.g., Kristijonas, Oana, Antonio, Emanuele, Andria, Neema, Guilherme, Alex, Fabrizio, Kai, Francis, etc.), many colleagues at DSI (e.g., Jingqing, Akara, Chao, Mo, Kevin), my lovely and lively London crew (e.g., Krittaboon, Preedachanok, Pasita, Nicha, Thanutcha, Peerut, etc.) as well as many seniors and friends who supported me wherever they are (e.g., Natthawut, Kritaphat, Nontawat, Kornraphop, Kritat, Vibhavee, and many more).

Finally and most importantly, I must express my profound gratitude to my parents and my family for giving me continuous support and encouragement throughout this long PhD journey. This accomplishment would not have been possible without them. Thank you.

Contents

List of Tables

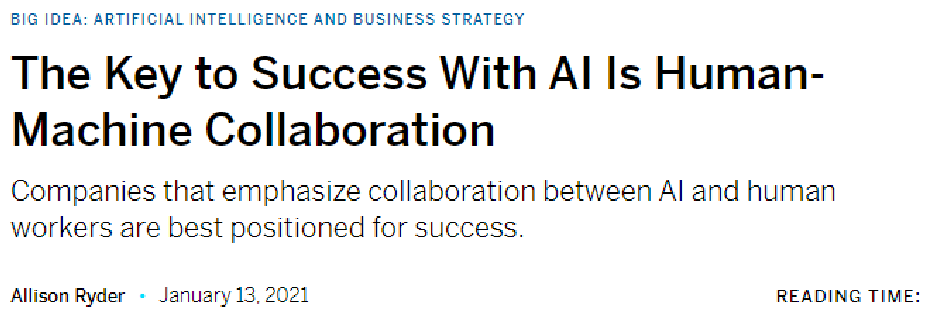
List of Figures

1. Introduction

We have seen more and more Artificial Intelligence (AI) technologies being adopted in organizations (Fountaine et al., 2019; Soni et al., 2020). The next interesting questions are "How many of them are truly successful?" and "What is the key to their success?". Ransbotham et al. (2020) surveyed 3,000 companies, 57% of which affirmed that their companies are piloting or deploying AI. However, only about 10% of the surveyed companies could obtain significant financial benefits, generating 5-10% of their overall revenue, with the AI. Figure 1.1 shows a headline of an article from MIT Sloan Management Review[1], reporting that the key to success of these companies is *human-machine collaboration*. Specifically, these organizations create systems such that they can "learn with the AI". In other words, they not only design AI systems that work for themselves but also engineer the systems so that the AI learns from humans and humans learn from the AI.

In the meanwhile, from the research side, Explainable AI (XAI) focuses on generating explanations for AI models and/or for their predictions (Adadi and Berrada, 2018). Some researchers have studied various merits of the explanations to humans, such as supporting human decision making (Lai and Tan, 2019), increasing human trust in AI (Jacovi et al., 2021) and even teaching humans to perform challenging tasks (Lai et al., 2020). On the other hand, explanations can benefit the AI systems as well, e.g., when explanations are used to promote system acceptance (Cramer et al., 2008), to verify the model reasoning (Caruana et al., 2015), and to find potential causes of errors (Han et al., 2020) and support humans fixing them (Teso and Kersting, 2019). Therefore, the two-way benefits of explanations may indeed enable the concept of "learning with the AI", mentioned above, since explanations allow both humans to learn from the AI and the AI to be improved by the humans. This also clearly highlights the importance of

[1]`https://sloanreview.mit.edu/article/the-key-to-success-with-ai-is-human-machine-collaboration/` (Accessed: 9th September 2021)


BIG IDEA: ARTIFICIAL INTELLIGENCE AND BUSINESS STRATEGY

The Key to Success With AI Is Human-Machine Collaboration

Companies that emphasize collaboration between AI and human workers are best positioned for success.

Allison Ryder • January 13, 2021 READING TIME:

Figure 1.1.: The headline of an article from MIT Sloan Management Review indicating that human-machine collaboration is the key to success for organizations deploying AI.

explainable AI towards human-AI collaboration.

Because the AI field is very broad, involving many sub-fields such as data mining, computer vision, planning, optimization, and robotics, it is difficult to generate universal explainable AI solutions for every sub-field. In this thesis, we target explainable AI for Natural Language Processing (NLP), a sub-field of AI focusing on building machine learning models to process input texts to achieve specific tasks (e.g., classification, question answering, information extraction, etc.). We focus on NLP because natural language has several characteristics which are not found in other data modalities. These become challenges for research in explainable NLP. For instance, the input space of NLP is discrete as an input text is a sequence of words or tokens. Hence, explainable AI methods that rely on continuous input spaces, such as sensitivity analysis (Dimopoulos et al., 1995) and activation maximization (Erhan et al., 2009; Nguyen et al., 2016), are not directly applicable or entirely meaningful (Lakkaraju et al., 2020). In addition, each word or token that forms the input has its own semantics which is neither complete nor definite unless it stays in a grammatically correct sequence and we consider it together with its context (Pustejovsky et al., 1996). This again makes explainable AI methods that rely on distance between inputs (Altman, 1992) and direct input perturbation (Feng et al., 2018) become less effective. Overall, these challenges make techniques in explainable NLP quite unique (meaning that the results from general XAI research cannot be directly adopted in NLP) and still far from fully solved.

As background, current research for explainability in NLP can be broadly categorized into three groups: Generating explanations, Evaluating explanations, and Applying explanations to specific tasks. We briefly overview these three groups below.

Generating Explanations. There are two types of explanations widely studied. First, *local explanations* aim to explain individual predictions made by a trained model for specific inputs. Second, *global explanations* aim to explain what the model has learned and how it works in general irrespective of any inputs. Local explanations can be presented in many forms depending on the aspect of the prediction process we want to explain. The most popular form of local explanations in NLP is input-based explanations (also known as feature importance explanations) showing which words or parts in the inputs are important or influential for the predictions (Bhatt et al., 2020b), as studied in (Li et al., 2016b; Ribeiro et al., 2016; Lundberg and Lee, 2017; Shrikumar et al., 2017) for example. Other forms of local explanations include rule-based (Ribeiro et al., 2018a), example-based (Guo et al., 2020), and counterfactual (Yang et al., 2020) explanations. Global explanations could be created by summarizing from a collection of local explanations (Ribeiro et al., 2018a; Pedreschi et al., 2019; Lundberg et al., 2020) or by creating another interpretable model to mimic the target model (Bastani et al., 2017; Tan et al., 2018; Sushil et al., 2018). Note that when the model is transparent (e.g., Naive Bayes models, logistic regression, and decision trees), we can probably use the model itself as the global explanation and extract local explanations from how the model processes the inputs (such as using the path the input follows in a decision tree as the local explanation) (Danilevsky et al., 2020). However, this does not mean that we do not need explanation methods invented for transparent models. Taking human factors into account, we realize that even machine learning experts may feel frustrated if they need to understand a decision tree with a depth of ten or more. Thus, the background knowledge, desires, and limits of the audience cannot be ignored when we design a new explanation method.

Evaluating Explanations. We can consider several desirable properties of explanations. Each of them leads to several ways to evaluate the explanations, and these may depend on the forms of the explanations as well. For example, let us consider a text classification task with feature importance scores as an explanation. These scores show how important each input word is for the classification. Early work evaluated these explanations by gradually deleting words from the input text following the order of their importance and checking how much the predicted probability drops (Arras et al., 2016). Poerner et al. (2018b) proposed two evaluation paradigms – hybrid documents and morphosyntactic agreements. Both check whether an explanation method correctly points to the (known) root cause of the prediction. These evaluation methods evaluate the *faithfulness*[2] of the explanations, checking how accurately the explanation reflects the prediction process of the model (Jacovi and Goldberg, 2020).

In contrast, Mohseni et al. (2018) proposed a benchmark which contains a list of relevant words for the actual class of each input text, identified by human experts. The high agreement between the relevant words annotated by humans and the important words shown in the explanations reflects the high *plausibility* of the explanations. Note, however, that the disagreements could be due to not only the poor explanation method but also the inaccuracy of the model or the model reasoning differently from humans.

In any case, both faithfulness and plausibility can be seen as *intrinsic properties* of explanations, i.e., properties that good explanation should have in general. In literature, there have been many works evaluating intrinsic properties across various explanation methods such as (Nguyen, 2018) for local faithfulness and plausibility and (DeYoung et al., 2020) for *comprehensiveness* and *sufficiency* (which are two other intrinsic properties).

Beyond intrinsic properties, some papers evaluate explanation methods *extrinsically* based on their performance in downstream applications. For instance, some evaluated user satisfaction with the explanations (Biran and McKeown, 2017; Narayanan et al., 2018) and how much the explanations could engender trust in the AI model (Bussone et al., 2015; Smith-Renner et al., 2020). Nevertheless, although explanations can be used to support human-AI collaboration as described earlier, only few papers have discussed the evaluation and comparison of explanation methods with respect to purposes of explanation usage by humans such as inferring model quality (Ribeiro et al., 2016) and providing supports to humans in real-time (Lai and Tan, 2019). No systematic evaluation of this sort across various explanation methods has been conducted yet.

[2]We will discuss faithfulness in detail in Section 3.2.

Applications of Explanations. As discussed above, the explanations are useful for supporting collaboration between AIs and humans in many cases (Samek et al., 2018). Firstly, if an AI outperforms humans in a certain task (e.g., AlphaGo (Silver et al., 2016)), humans can learn and distill knowledge from the given explanations. Secondly, if an AI's performance is close to human intelligence, the explanations can increase humans' confidence and trust in the AI (Symeonidis et al., 2009). Lastly, if an AI is duller than humans, the explanations can help humans verify the decisions made by the AI and also improve the AI (Biran and McKeown, 2017). Focusing on the last case, in particular, there are several existing works using explanations to support *debugging*. Actually, the term debugging in machine learning research is interpreted differently by different researchers. Some consider debugging as a process of identifying or uncovering causes of model errors (Parikh and Zitnick, 2011; Graliński et al., 2019), while others stress that debugging must not only reveal the causes of problems but also fix or mitigate them (Kulesza et al., 2015; Yousefzadeh and O'Leary, 2019). Note that, in this thesis, we adopt the latter interpretation. Previously, there have been attempts using explanations to allow humans to understand how the model works and provide feedback in response to debug the model. This process is called *explanatory debugging* and has been explored in some previous works (e.g., (Stumpf et al., 2009; Kulesza et al., 2010, 2015; Teso and Kersting, 2019)). However, most of them targeted traditional machine learning models (such as Naive Bayes and support vector machines), whereas there are only few studies exploring problems and strategies for bugs in deep learning models (Ribeiro et al., 2018b; Cho et al., 2019).

1.1. Thesis Contributions

This thesis has three main contributions which fill in the gaps existing in the three research categories of explainable NLP discussed above. Figure 1.2 illustrates where our contributions are in the context of human-AI collaboration. Note that, for all the contributions, we focus on the *text classification* task, amounting to its various forms such as topic classification, sentiment analysis, abusive language detection, spam classification, and deceptive review detection. Text classification is a fundamental task in NLP, with a variety of high-performing models and several potential applications thereof

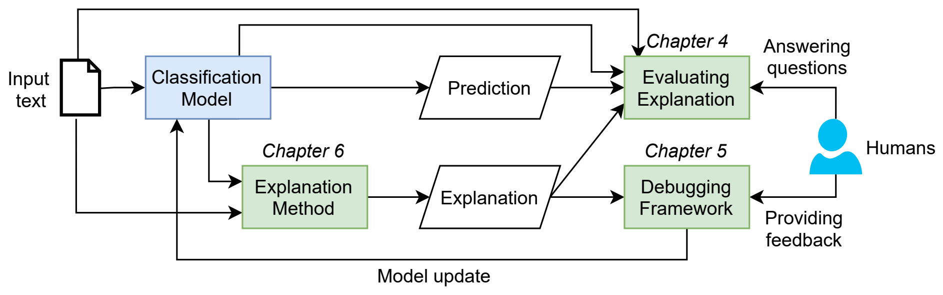


Figure 1.2.: A diagram summarizing the contributions of this thesis.

(Li et al., 2020b). There is also great scope at the intersection of explainable NLP and human-AI collaboration for targeting this task. We focus on two types of classification models – black-box models and interpretable models – at different parts of the thesis. Specifically, we choose convolutional neural network (CNN) (Kim, 2014) as representative of black-box models, because it has been found to achieve promising results in many text classification tasks (Johnson and Zhang, 2015; Gambäck and Sikdar, 2017; Zhang et al., 2019) and had several applicable explanation methods available publicly (Bach et al., 2015; Shrikumar et al., 2017) (to be evaluated in this thesis). Also, while being a black-box, CNN is leaner and does not require much human effort when it comes to debugging (which is also a part of this thesis) unlike recent complex transformer-based models (Devlin et al., 2019; Liu et al., 2019b), which would require more than the budget we have for conducting experiments with humans. Meanwhile, we choose pattern-based logistic regression as representative of interpretable models, which may still need explaining when used by humans for their benefits.

Contribution 1: Human-Grounded Evaluations of Explanation Methods. With so many local explanation methods available, it is important to evaluate and compare them in order to choose the right methods for different settings. Many existing works evaluate explanation methods by focusing on their intrinsic properties. Still, there has not been any work that evaluates various existing explanation methods systematically to demonstrate how well they can support different human-AI collaboration tasks. We close this gap by proposing three human-grounded tasks to evaluate the quality of explanation methods with respect to different purposes of explanation usage for text classification. These purposes include *(1)* revealing model behavior to human users, *(2)* justifying the predictions, and *(3)* helping humans investigate uncertain predictions. Then, using CNNs as target classification models, we apply these three tasks to evaluate nine explanation methods, including two random explanations, five configurations of existing well-known methods, and two novel methods developed in this thesis specifically for CNN text classifiers. Among the nine methods, four explain at the word level while the other five explain at the n-gram level. The results are novel, demonstrating dissimilar qualities of the tested methods and showing how well each can serve the three purposes as well as the challenges of using explanations for these purposes.

Contribution 2: Human-in-the-Loop Debugging Deep Text Classifiers. Deep learning-based text classifiers work like a black box because the way they process and reason on inputs is not interpretable to humans. This makes debugging these models become challenging, more so than explanatory debugging of traditional machine learning models (such as Naive Bayes and support vector machines), as predominantly done in the literature. In this thesis, we fill this gap by proposing a novel framework, namely **FIND**, which integrates explainable NLP and human-AI collaboration to debug deep text classifiers in the form of CNNs. FIND consists of three steps. First, it selects an intermediate layer in the target model and analyzes textual patterns that each neuron in the layer has learned to capture, by using an explanation method called LRP (Layer-wise relevance propagation) (Arras et al., 2016). These patterns are then presented to humans using word clouds. Second, the humans analyze the word clouds and identify whether the patterns of each neuron are likely to be helpful for the classification task or they are just irrelevant artifact patterns. Third, the neurons that mainly capture the irrelevant patterns are disabled in the model, and the model is re-trained, forced to use only good neurons to perform the task. Overall, FIND has two characteristics which are different from most of the existing work. First, it asks for human feedback in response to global explanations rather than local explanations of the model (as used in, for example, (Teso and Kersting, 2019; Cho et al., 2019; Smith-Renner et al., 2020)). This is beneficial because it is not susceptible from the *local decision pitfall* problem where local improvements for individual predictions could add up to inferior overall performance (Wu et al., 2019b). Second, the debugging process is a one-off improvement since the word clouds (shown as the global explanations) do not change after the model update. Hence, unlike many previous works (Kulesza et al., 2009, 2010, 2015), our work sits more in the human-in-the-loop machine learning rather than interactive machine learning. Using CNN text classifiers, we conduct three human experiments that demonstrate the effectiveness of FIND in different scenarios (i.e., coping with small training datasets, datasets with gender bias, and datasets with different train-test distributions). The results not only highlight the usefulness of our approach but also reveal interesting behaviors of CNNs for text classification.

Contribution 3: Argumentative Explanations for Text Classification. Existing explanation methods have been widely devised to address the transparency issue of black-box models or to create explanations for any kind of models (i.e., being *model-agnostic*). This leaves explanations of interpretable models underexplored by the community. On one hand, this is understandable because the standard method to extract explanations from each interpretable model is obvious and faithful (e.g., using the corresponding path in a decision tree as explanation or using k nearest neighbor examples of a kNN model as explanation). On the other hand, this leads to little attention on other desirable properties of explanations for these interpretable models. We fill this gap by focusing on explaining *pattern-based* logistic regression (PLR) for text classification. In fact, patterns implicitly emerge in CNNs. However, it is sometimes difficult to understand common characteristics of the patterns captured by specific CNN features. So, we choose to extract interpretable patterns beforehand using GrASP (Shnarch et al., 2017) and use them as features of the logistic regression model. Albeit interpretable, PLR is challenging when it comes to explanations. In particular, we found that a standard way to extract local explanations from this model does not consider relations among the features. This makes the explanations less plausible to humans. Hence, to fill this gap, we propose **AXPLR**, a novel local explanation method using computational argumentation (Čyras et al., 2021) to model agreements and disagreements among the features before generating explanations. The method consists of four steps: extracting argumentation frameworks from the model, computing strengths of arguments, post-processing the results, and generating the explanations. Finally, we conduct property analysis, empirical evaluation, and two experiments in the context of human-AI collaboration to demonstrate the advantages of AXPLR.

Summary of contributions. All in all, Table 1.1 summarizes different dimensions of our three main contributions. For models, the table shows both the models where our proposed methods / frameworks are applicable and the models we used in the experiments. The explanations used in all the contributions are friendly to end users with no machine learning experience although the model developers may perform the human roles in some of the applications (such as model debugging).

1.2. Ethical Statement

For all the human experiments conducted, unless stated otherwise, we used a crowdsourcing platform (i.e., Amazon Mechanical Turk) to recruit human participants without collecting any of their personal information. When they needed to deal with sensitive texts (e.g., investigating texts from an abusive language detection dataset), we additionally showed a clear warning message for them to consider before accepting to perform the task. Furthermore, the participants could complain about our experiments to the crowdsourcing platform if they wanted. However, we did not receive any negative feedback or complaints as of the completion of the experiments. There is no evidence that the participants received mental harm or there were any ethical issues during the experiments. For experiments in the last contribution (AXPLR), under the recommendation by the Research Governance and Integrity Team at Imperial College London, we additionally submitted an application for ethics review (which was duly approved).

1.3. Thesis Structure

This thesis is structured as follows. Chapter 2 provides essential background knowledge that we use or build upon in the thesis. Chapter 3 critically analyzes related work and points out research gaps that this thesis addresses. Chapters 4 – 6 present the main contributions in this thesis. Specifically,

Chap.	Focus	Expl.	Model	Human role(s)
4	Evaluating explanations	Local	(Any; CNN)	Task performers as evaluators
5	Debugging the model	Global	(Deep learning; CNN)	Feedback providers as debuggers
6	Explain the prediction	Local	(PLR; PLR)	End users, Learners

Table 1.1.: Dimensions of the three major contributions of this thesis. Chap. and Expl. stand for chapter and explanation scope, respectively. Models are in the form of (A; B) where A is an applicable model while B is the model used in the experiments. PLR stands for pattern-based logistic regression. Note that we target text classification for all the chapters.

Chapter 4 presents the three human-grounded evaluation tasks to assess explanation methods with respect to different purposes of usage for text classification. Chapter 5 discusses our proposed human-in-the-loop debugging framework for deep text classifiers, FIND, together with three human experiments demonstrating the effectiveness of the framework. Chapter 6 explains AXPLR, our novel local explanation method, based on computational argumentation, designed specifically for pattern-based logistic regression models. Finally, Chapter 7 concludes the thesis with possible future work.

1.4. Publications

The work presented in this thesis has resulted in two conference publications, one journal paper, and one technical report:

- (Lertvittayakumjorn and Toni, 2019) Piyawat Lertvittayakumjorn and Francesca Toni. 2019. *Human-grounded evaluations of explanation methods for text classification.* 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 5195–5205, Hong Kong, China. Association for Computational Linguistics. (Chapters 1 and 4)

- (Lertvittayakumjorn et al., 2020) Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. *FIND: Human-in-the-Loop Debugging Deep Text Classifiers.* In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 332–348, Online. Association for Computational Linguistics. (Section 2.1.3 and Chapter 5)

- (Lertvittayakumjorn and Toni, 2021) Piyawat Lertvittayakumjorn and Francesca Toni. 2021. *Explanation-Based Human Debugging of NLP Models: A Survey.* Transactions of the Association for Computational Linguistics (Accepted, forthcoming). (Chapter 1, Sections 2.2.2, 3.3, 5.6, and 7.2.3)

- (Lertvittayakumjorn et al., 2021b) Piyawat Lertvittayakumjorn, Leshem Choshen, Eyal Shnarch, and Francesca Toni. 2021b. *GrASP: A Library for Extracting and Exploring Human-Interpretable Textual Patterns.* arXiv preprint arXiv:2104.03958. [Technical report] (Section 6.1)

During my PhD study, I also co-authored the publications below, which are not incorporated in this thesis:

- (Wu et al., 2019a) Chao Wu, Guolong Wang, Jiangcheng Zhu, Piyawat Lertvittayakumjorn, Simon Hu, Chilie Tan, Hong Mi, Yadan Xu, and Jun Xiao. 2019a. *Exploratory analysis for big social data using deep network.* IEEE Access, 7:21446–21453.

- (Zhang et al., 2019) Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. *Integrating semantic knowledge to tackle zero-shot text classification.* 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 1031–1040, Minneapolis, Minnesota. Association for Computational Linguistics.

- (Albini et al., 2020) Emanuele Albini, Piyawat Lertvittayakumjorn, Antonio Rago, and Francesca Toni. 2020. *Dax: Deep argumentative explanation for neural networks.* arXiv preprint arXiv:2012.05766. [Technical report]

- (Lertvittayakumjorn et al., 2021a) Piyawat Lertvittayakumjorn, Daniele Bonadiman, and Saab Mansour. 2021a. *Knowledge-driven slot constraints for goal-oriented dialogue systems.* In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3407–3419, Online. Association for Computational Linguistics.

- (Dejl et al., 2021) Adam Dejl, Peter He, Pranav Mangal, Hasan Mohsin, Bogdan Surdu, Eduard Voinea, Emanuele Albini, Piyawat Lertvittayakumjorn, Antonio Rago, and Francesca Toni. 2021. *Argflow: A toolkit for deep argumentative explanations for neural networks.* In Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems, page 1761–1763, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

- (Lertvittayakumjorn et al., 2021c) Piyawat Lertvittayakumjorn, Ivan Petej, Yang Gao, Yamuna Krishnamurthy, Anna Van Der Gaag, Robert Jago, and Kostas Stathis. 2021c. *Supporting complaints investigation for nursing and midwifery regulatory agencies.* In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 81–91, Online. Association for Computational Linguistics.

Lastly, during my PhD study, I had the pleasure to co-supervise many undergraduate students and master students, producing the following publications (not incorporated in this thesis):

- (Kanwatchara et al., 2021) Kasidis Kanwatchara, Thanapapas Horsuwan, Piyawat Lertvittayakumjorn, Boonserm Kijsirikul, and Peerapon Vateekul. 2021. *Rational LAMOL: A rationale-based lifelong learning framework.* 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 2942–2953, Online. Association for Computational Linguistics.

- (Hongwimol et al., 2021) Pollawat Hongwimol, Peeranuth Kehasukcharoen, Pasit Laohawarutchai, Piyawat Lertvittayakumjorn, Aik Beng

Ng, Zhangsheng Lai, Timothy Liu, and Peerapon Vateekul. 2021. *ESRA: Explainable scientific research assistant.* In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 114–121, Online. Association for Computational Linguistics.

- (Zylberajch et al., 2021) Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. *HILDIF: Interactive debugging of NLI models using influence functions.* In Proceedings of the First Workshop on Interactive Learning for Natural Language Processing, pages 1–6, Online. Association for Computational Linguistics.

2. Essential Background

This chapter aims to provide background knowledge that is essential for the thesis. Section 2.1 discusses text classification, which is the target task for human-AI collaboration in this thesis. In particular, it provides the task definition, introduces relevant models, and points out some issues that make this task challenging. Next, Section 2.2 provides an overview of interpretability and explainability for NLP and also overviews two explanation methods, LRP and Grad-CAM, which play major roles in the thesis (especially in Chapters 4 and 5). Then, Section 2.3 covers the background for computational argumentation and its derived explanation which are utilized mainly in Chapter 6 of this thesis. Finally, Section 2.4 provides a summary of this chapter with a transition to the next chapter.

2.1. Text Classification

As a fundamental problem in natural language processing, text classification aims to classify a given document into one of the known classes. It also has many specific real-world applications. For example, one task for sentiment analysis is classifying whether a given text conveys positive or negative sentiment (Medhat et al., 2014). This enables businesses to do social listening from review texts in order to grasp public feelings towards their brand or products. Additionally, abusive language detection aims to predict whether a given text uses offensive/insulting language or not. This is helpful, especially in online social media, to reduce contents that might cause cyber-bullying, hate crime, and discrimination (Park et al., 2018b). Generally, although humans can manually write rules to classify texts, it is difficult to compose a complete set of high-quality rules which could effectively classify any unseen texts. This is even more difficult if the humans do not have any domain knowledge or expertise about the task. Therefore, most text classification methods rely on the *supervised learning* approach where we optimize a classifier using a set of labeled training data. Next, we will discuss this approach and some supervised learning models (used in this thesis) in Sections 2.1.1 and 2.1.2, by using chapters 4 and 5 of (Jurafsky and Martin, 2020) as the main reference unless cited otherwise. We will conclude this section with a discussion in Section 2.1.3 of issues emerging in text classification when supported by standard supervised machine learning.

2.1.1. Problem Definition

Let us consider a text classification task with $|\mathcal{C}|$ classes where $\mathcal{C}$ is the set of all classes. A training dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ is given, where x_i is the i-th document containing a sequence of L_i words, $[x_{i1}, x_{i2}, ..., x_{iL_i}]$, and $y_i \in \mathcal{C}$ is the class label of x_i. The goal of supervised learning for text classification is to train a classification model M on the dataset $\mathcal{D}$ so that it is capable of classifying a new input document x into one of the seen classes $\hat{y}$ (i.e., $\hat{y} = M(x) \in \mathcal{C}$). We call the task *binary classification* and *multiclass classification* for the settings with $|\mathcal{C}| = 2$ and $|\mathcal{C}| > 2$, respectively.

To evaluate the performance of a classifier M, we apply it to predict examples in a labeled test dataset $\mathcal{D}'$ and report the percentage of correct predictions, so called the *accuracy* or *classification rate*. However, if the test dataset is class imbalanced, i.e., having examples of one class more than others, accuracy may not be the best evaluation metric because a model can get a high accuracy only by always answering the majority class. Alternatively, we can report the model performance for each specific class $c \in \mathcal{C}$ using the class *precision*, *recall*, and *F-measure* (mostly F1). Then we aggregate these class-specific metrics to be the metrics for the overall performance by either micro-averaging or macro-averaging. More details about evaluation metrics for text classification can be found in Appendix A.1.

2.1.2. Relevant Text Classification Models

In this section, we explain some text classification models which we use or build upon in the thesis. These amount to traditional machine learning models in the form of decision trees (DTs) and logistic regression (LR), and deep learning models in the form of convolutional neural networks (CNNs). Particularly, CNNs are our target models of Chapters 4 and 5, whereas LR is the target model of Chapter 6. We also leverage DT in one of our proposed explanation methods (for CNNs) in Chapter 4.

Traditional Machine Learning Models

By nature, text documents are unstructured data, so they are inconvenient for machines to process. Hence, traditional machine learning methods perform *feature extraction*, converting each input document x into a *feature vector* $\mathbf{f} \in \mathbb{R}^d$ (for $d \geq 1$), before training a classifier to map the feature vectors of all the documents to the classes. The most widely used feature extraction method for texts is *bag-of-words* where an input text is tokenized and organized into an unordered set of words together with their frequencies in the input while the word order is ignored. Then each position in the feature vector corresponds to the number of appearances of a specific word in the input text. This feature extraction process is called *count vectorization.* Otherwise, we could use *TF-IDF vectorization* (Weiss et al., 2010, chapter 2) to re-weight values in the feature vector, taking into account not only the word frequency in the input text but also the number of documents containing each word in the whole training set. Also, we can extend the bag-of-words method to the *bag-of-n-grams* method where each element in the feature vector could correspond to not only a single word (*unigram*), but also a phrase of two words (*bigram*), three words (*trigram*), or n words (*n-gram*). Moreover, we can extract customized features we deem useful for the classification task at hand and include them in (or use them solely) as the feature vector. This process is called *feature engineering.* For example, for the fake news detection task, one may create features especially for some punctuation characters (e.g., periods, question marks, exclamation marks) and for the proportion of words that belong to psycholinguistic categories (e.g., emotional words, perceptual process) (Pérez-Rosas et al., 2018).

With feature vectors, we can train traditional machine learning classifiers to predict the classes. Examples of the classifiers include[1]

- **Decision Tree** selects a feature with its particular value to be a split point, partitioning the whole training dataset into two parts. The selected split point must optimize some metric of the dataset such

[1]Indeed, there are other types of traditional classifiers (such as Naive Bayes Classifier, k-Nearest Neighbors, Support Vector Machines, and Random Forest) though their details are omitted here as they are not relevant to the thesis.

as information gain (Quinlan, 1986) or gini impurity (Breiman et al., 1984), hoping that each partition is more specific to some classes. Splits are then performed recursively on each partition until a stopping criteria is reached (e.g., the maximum depth of the tree, the minimum number of samples per a leaf node, the minimum impurity decrease).

- **Multinomial Logistic Regression** learns the importance of each feature in the feature vectors $\mathbf{f} \in \mathbb{R}^d$ towards each class $c \in \mathcal{C}$. Formally,

$$\mathbf{p} = \text{softmax}(\mathbf{Wf} + \mathbf{b}) \tag{2.1}$$

where $\mathbf{p} \in \mathbb{R}^{|\mathcal{C}|}$ is the predicted probability of all the classes, $\mathbf{W} \in \mathbb{R}^{|\mathcal{C}| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{C}|}$ are weights and biases, respectively, which are the learned parameters of logistic regression. Thus, the predicted class $\hat{y}$ is the class with the maximum predicted probability. The softmax function is used to normalize a vector into a probability mass function. Its definition is shown below for the vector $\mathbf{z} = [z_1, ..., z_k]$.

$$\text{softmax}(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{k} exp(z_j)} \text{ for } 1 \leq i \leq k \tag{2.2}$$

$$\text{softmax}(\mathbf{z}) = [\text{softmax}(z_1), ..., \text{softmax}(z_k)] \tag{2.3}$$

We can learn the weights and biases of logistic regression by optimizing an objective function (i.e., a *loss function*) using the gradient descent algorithm (Ruder, 2016). The loss function we normally use for multiclass classification is the *categorical cross-entropy loss*:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c \in |\mathcal{C}|} y_{i,c} \log(p_{i,c}) \tag{2.4}$$

where $p_{i,c}$ is the predicted probability of class c for the example x_i, and $y_{i,c}$ equals 1 only when $y_i = c$, being 0 otherwise.

- **Binary Logistic Regression** is a specialized case of logistic regression where there are only two classes, i.e., $C = \{0, 1\}$:

$$\begin{aligned} p &= \sigma(\mathbf{w}^T\mathbf{f} + b) \\ &= \sigma(\sum_{i=1}^{d} w_i f_i + b) \\ &= \sigma(w_1 f_1 + w_2 f_2 + ... + w_d f_d + b) \end{aligned} \tag{2.5}$$

where p is the predicted probability of the main class (i.e., class 1), $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are weights and bias of the model. σ is a *sigmoid function* (so called a *logistic function*) converting any real number into a value between 0 and 1.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.6}$$

where $z = 0$ yields $\sigma(z) = 0.5$. Additionally, we use the *binary cross-entropy loss* as an objective function to optimize the model.

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \tag{2.7}$$

where $y_i \in \{0, 1\}$ is the true label of example x_i and p_i is the predicted probability of class 1 for x_i returned by the model.

Overall, traditional classifiers in general are mostly interpretable because we can easily observe the importance of each feature for the predictions (e.g., via the learned weights or the splitting features). Nevertheless, their weaknesses are that (1) the word order information is mostly ignored; (2) the feature vectors are sparse and too large to efficiently compute for some learning algorithms; and (3) the semantics of words are not modeled and exploited by the classifiers.

Deep Learning Models

Convolutional Neural Networks (CNNs) have been very successful in the computer vision domain (Krizhevsky et al., 2012; He et al., 2016) and got adapted to the NLP domain to perform many tasks including text classification (Johnson and Zhang, 2015; Gambäck and Sikdar, 2017; Zhang et al., 2019). It is a type of deep learning models, operating on multiple non-linear modules (layers) so as to learn effective representations of inputs for

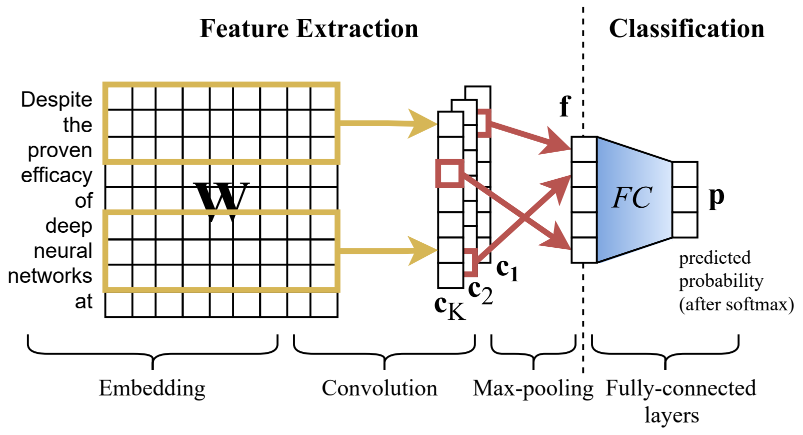


Figure 2.1.: A 1D CNN for text classification.

achieving a given target task (LeCun et al., 2015). In particular, CNNs for text classification also rely on the idea of word embeddings (Mikolov et al., 2013; Pennington et al., 2014) where each input word is represented as a vector (capturing semantics of the word) and the models then use vectors of all the input words as input to predict the output class. Additional details about deep learning and word embeddings can be found in Appendix A.2.

Figure 2.1 shows a standard one-dimensional (1D) CNN for text classification (Kim, 2014) which consists of four main steps. First, an input text $x_i = [x_{i1}, x_{i2}, ..., x_{iL}]$ is embedded into a matrix $\mathbf{W} \in \mathbb{R}^{L \times d}$ where the row j of $\mathbf{W}$ is the word embedding of x_{ij} with d dimensions. Second, K fixed-size convolution filters are applied to $\mathbf{W}$ to find n-grams that possibly discriminate one class from the others. For the filter k with window size l, the result of the convolution on $\mathbf{W}$ is a feature map $\mathbf{c}_k \in \mathbb{R}^{L-l+1}$:

$$\begin{aligned} \mathbf{c}_k &= [c_k^1, ..., c_k^{(L-l+1)}] \\ c_k^i &= g(\mathbf{F}_k \odot \mathbf{W}_{i:i+l-1} + \mathbf{b}_k) \end{aligned} \tag{2.8}$$

where $\mathbf{F}_k \in \mathbb{R}^{l \times d}$ and $\mathbf{b}_k \in \mathbb{R}$ are the learned weights of the filter k, $\mathbf{W}_{i:i+l-1}$ is a sub-matrix of $\mathbf{W}$ consisting of l consecutive rows starting from row i, $\odot$ is an element-wise multiplication operator, and g is a non-linear activation function. Note that we often use the *Rectified Linear Unit* (ReLU) as g, where

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{2.9}$$

Third, the maximum value found by each filter, corresponding to the most relevant n-gram in text, is pooled to construct a filter-based feature vector, $\mathbf{f} \in \mathbb{R}^K$, of the input:

$$\mathbf{f} = [\max(\mathbf{c}_1), ..., \max(\mathbf{c}_K)] \tag{2.10}$$

This feature vector $\mathbf{f}$, representing the input text x_i, is analogous to $\mathbf{f}$ obtained from feature extraction in Section 2.1.2. Finally, fully-connected layers (FC) are used to predict the results, and a softmax function is applied to the outputs to obtain predicted probability of the classes ($\mathbf{p}$):

$$\mathbf{p} = \text{softmax}(FC(\mathbf{f})) \tag{2.11}$$

Figure 2.1 shows the case where we use only one linear layer as FC. In fact, more hidden layers can be added to increase the model capacity for prediction. Also, more than one filter size can be used to detect n-grams with short- and long-span relations (Conneau et al., 2017).

To train the CNN model, we need to choose the loss function for the task (such as Equations 2.4 or 2.7) and use the backpropagation algorithm (Rumelhart et al., 1986) to optimize the model parameters. For the embedding layer, we usually initialize it with pre-trained word embeddings such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014). If we have a sufficient amount of training data, we may update these embeddings during model training to specialize them to the target task. Otherwise, we set these embeddings as non-trainable.

2.1.3. Issues with Supervised Text Classification

Supervised learning trains a model to map an input text to a class. Since an input text can contain both parts which are relevant and irrelevant to the classification, it is possible that the model exploits the irrelevant parts – relying on the wrong reasons – when making predictions. This usually happens when training datasets are small or contain *artifacts* (i.e., tokens or phrases which are not relevant, but strongly co-occur with one of the classes) (Wiegand et al., 2019; Gururangan et al., 2018). These can lead to suboptimal models with undesirable properties. For example, the models may have biases against some sub-populations or may not work effectively in the wild as they overfit the imperfect training data. Even though the NLP community today has advanced pre-trained language models (e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), ALBERT (Lan et al., 2019), etc.), leveraging transfer learning to improve the performance of downstream tasks, these models also suffer (more or less) from the same issue as other machine learning models (Yao et al., 2021).

To improve the suboptimal models, existing work has looked into different techniques beyond standard model fitting. If the weaknesses of the training datasets or the models are anticipated, strategies can be tailored to mitigate such weaknesses. For example, augmenting the training data with gender-swapped input texts helps reduce gender bias in the models (Park et al., 2018b; Zhao et al., 2018). Adversarial training can prevent the models from exploiting irrelevant and/or protected features (Jaiswal et al., 2020; Zhang et al., 2018). With a limited number of training examples, using human rationales or prior knowledge together with training labels can help the models perform better (Zaidan et al., 2007; Bao et al., 2018; Liu and Avci, 2019). Nonetheless, there are side-effects of sub-optimal datasets that cannot be predicted and are only found after training thanks to post-hoc error analysis. To rectify such problems, there have been attempts to enable humans to fix the trained models (i.e., to perform *model debugging*). We will review these methods in detail in Section 3.3.

2.2. Interpretability and Explainability for NLP

Explainable AI focuses on generating explanations for AI models as well as for their predictions (Adadi and Berrada, 2018). Within the NLP community, it is gaining more and more attention these days since explanations are necessary in several language-based applications, especially in high-stake domains such as healthcare (Feng et al., 2020), law (Branting et al., 2019), and finance (Yang et al., 2020). Besides explanations, the community is also interested in opening the black boxes (i.e., analyzing complicated NLP models to learn more how, when, and why they work). The insights from the analysis could help us understand the model's strengths and weaknesses (Ribeiro et al., 2020), improve the model efficiency (Dalvi et al., 2020), and contemplate how humans acquire and process languages (Linzen, 2019).

This section reviews methods for interpreting neurons in deep NLP models and methods for generating explanations which are relevant to this thesis, enabling effective human-AI collaboration.

2.2.1. Interpreting Neurons in Deep NLP Models

There has been substantial work in gaining better understanding of each neuron in complex, deep neural NLP models. In this section, we discuss two approaches (visualization and corpus analysis) along with some examples.

Visualization. To investigate how a neuron works, we could visualize its activation values with respect to input texts or input tokens. With this technique, Li et al. (2016a) observed the final representations learned by recurrent neural networks (i.e., the feature vectors) of several input texts. Then they found that some dimensions of the feature vectors capture the effect of intensification and negation in the input texts. In contrast, Karpathy et al. (2015) observed cell activation of a character-level LSTM model for language modelling after receiving each character. The results revealed the existence of some interpretable cells. For example, they found a cell acting as a line length counter and cells checking if the current letter is inside a parenthesis or a quote.

As one model may have hundreds to thousands neurons, manually looking at visualizations of every neuron is not efficient. Recent works, therefore, move towards *interactive visualizations*. For instance, LSTMVis (Strobelt et al., 2018) allows users to search for cells, in an LSTM model (Hochreiter and Schmidhuber, 1997), that are activated significantly when reading a user-specified part of an input. In addition, the system shows parts of other examples where such cells are also activated, so the users can form hypotheses about how the cells work.

Corpus Analysis. This approach aims to find n-grams or concepts which best represent the expertise of the neurons. Jacovi et al. (2018) analyzed what is captured by each filter of CNN text classifiers. They applied a threshold to activation scores of n-grams (chosen by max pooling of each filter) to separate between informative and non-informative n-grams. By analyzing informative n-grams, they found that one convolutional filter may detect more than one n-gram pattern and may also suppress negative n-grams. Meanwhile, Na et al. (2019) aimed to find top-M concepts strongly activating each CNN filter. To do so, they selected top-K training sentences which activate the filter the most and collected every node in the constituency parse trees of these sentences to be candidate concepts. For each candidate concept, they created a synthetic sentence containing the concept multiple times and checked the activation score it got from the filter. M concepts of which the synthetic sentences got the highest scores from each filter are shown to the users. Without corpus search, Poerner et al. (2018a) optimized the activation of each neuron using gradient ascend and gumbel softmax trick. This made them found an n-gram which maximally activates the neuron though it may not convey a clear meaning.

2.2.2. Categorizations of Explanation Methods

In Section 2.2.1, we consider a micro analysis of deep NLP models, i.e., investigating each neuron in the models. By contrast, in this section, we consider a macro analysis of NLP models through explanations. Explanations and explanation methods in NLP can be categorized along three main dimensions.

What to explain.

Basically, there are two things which we want to explain, corresponding to two types of explanations. First, **local explanations** explain the predictions by the model of interest for individual inputs. In other words, we want to know why the model predicts this output for a given input. This explanation type helps us verify whether a prediction was made for the right reason. Second, **global explanations** explain the model overall, independently of any specific inputs. This explanation type reveals the prediction mechanism of the model in a more comprehensible way, so it helps us verify if the model is suitable for deployment (Lakkaraju et al., 2020). Due to the complexity of NLP models nowadays, it is very challenging to obtain global explanations which are both comprehensive and comprehensible. Therefore, a majority of research in explainable NLP targets local explanations, which are more tractable, as we can see from the survey[2] by Dhanorkar et al. (2020).

[2]`https://xainlp2020.github.io/xainlp/home`

How to compute explanations.

Some ML models, e.g., Naive Bayes, logistic regression, and decision trees, are **self-explaining** (Danilevsky et al., 2020), also referred to as **transparent** (Adadi and Berrada, 2018), **inherently interpretable**[3] (Rudin, 2019), or **directly interpretable** (Arya et al., 2019). Local explanations of self-explaining models can be obtained at the same time as predictions, from the process of making those predictions, while the models themselves can usually serve directly as global explanations. In contrast, another type of explanation methods requires a separate step to analyze the model and (optionally) the input in order to generate explanations. We call them **post-hoc explanation methods**. Note that we can also apply post-hoc explanation methods to self-explaining models though this is not typically done since the self-explaining explanations are easier to obtain and guaranteed to be faithful to the model. We will discuss desirable properties of explanations in Section 3.2. Combining the two dimensions of categorizations, we get finer classes of methods, including local self-explaining methods, global self-explaining methods, local post-hoc methods, and global post-hoc methods.

How to present explanations.

Explanations can be presented in many forms. For local explanations, the main question is "Why did the model predict this output for a given input?", and there are indeed several ways to approach this question.

- **Input-based explanations** identify which parts of the input are important for the prediction. For a textual input, the explanation could be an excerpt from the input, so called a *rationale*, considered as a main reason for the prediction. Besides, it could be *attribution scores* or *relevance scores* showing the importance of words in the input text. These scores can also be visualized as *saliency maps* overlaying the input text. Note that negative relevance scores usually mean that the corresponding words contribute negatively towards the prediction.

[3]It is debatable whether *inherently interpretable* is the best term here because it sounds as if this depends on the users. However, we use this term in the thesis in the sense that we can use information emitted by the model during the prediction process as the explanation without the need to perform any extra steps.

- **Counterfactual explanations** point out which parts of the input need to be change in order to change the prediction. The smaller the change is, the better the counterfactual would be. This form of explanations is often useful when we want to change an undesirable output of the model.

- **Example-based explanations** select influential, important, or similar examples from the training set to explain why the model makes a specific prediction.

- **Rule-based explanations** provide a decision rule that approximates the prediction process. With such prediction as the rule head, the rule body is a composition of conditions that the input text satisfies. It is also desirable if the rule can generalize beyond the given example.

- **Textual explanations** verbalize the explanations in natural languages that humans (especially lay users) can easily understand. Their contents could align well with a single or a combination of the other four explanation forms discussed above. Because textual explanations have more freedom to refer to things outside the training process, they could mention, in the explanations, commonsense or external knowledge that does not appear explicitly in the input text.

Figure 2.2 illustrates how each type of the local explanations explain the same prediction. Additionally, Table 2.1 categorizes some existing works based on the form of their presented local explanations. It can be seen that the majority of explainable NLP research follow the input-based approach.

Meanwhile, the main question for global explanations is “What is the overall prediction mechanism of the model?”, and there are two popular approaches to answer this question.

- **Collections of local explanations.** Each local explanation can explain the model behavior locally (near the target example). Therefore, if we combine several local explanations from all regions of the example space, we will see the overall behavior of the model. If we want to follow this approach, there are three main questions to be answered. First, which local explanation method shall we use? Second, how shall we select a set of representative local explanations and how large is

Task: Sentiment analysis

Dataset: Amazon review polarity (Zhang et al., 2015)

Model: BiLSTM with GloVe embeddings

Input text: *Long and boring: I've read this book with much expectation, it was very boring all through out the book*

Prediction: Negative

Input-based explanations

⟶ **Rationale**: *boring*

⟶ **Saliency map**: *Long and boring: I've read this book with much expectation, it was very boring all through out the book*

Counterfactual explanations

⟶ *Long and boring: I've read this book with much expectation, it was very awesome all through out the book* (Prediction: Positive)

⟶ *Long and boring: I've read this book with much expectation, it was not boring all through out the book* (Prediction: Positive)

Example-based explanations (Influential training examples)

⟶ *Stay Away: This just plain bad. Boring..... I did not find this the least bit entertaining nor interesting. It was a waste of my time.* (Label: Negative)

⟶ *Boring: I would not reccomend this book. It was very boring and I didn't find much of a purpose. The end was the best part, but it was still bad. l did not get into this book or enjoy it. it was for school.* (Label: Negative)

Rule-based explanations

⟶ If *boring*, then prediction = Negative (Precision: 85.8%)

⟶ If *very* $\wedge$ *boring*, then prediction = Negative (Precision: 94.7%)

Textual explanations

⟶ *Because the book is long and very boring, the sentiment is negative.*

⟶ *The book did not meet the expectation, so the sentiment is negative.*

Figure 2.2.: Illustrations of different types of local explanations.

the set? Third, how shall we combine the local explanations to provide the global view of the model? Examples of this approach include Ribeiro et al. (2016, 2018a); Pedreschi et al. (2019); Lundberg et al. (2020); Setzu et al. (2021).

Input-based	Li et al. (2016b); Ribeiro et al. (2016); Arras et al. (2016, 2017); Lundberg and Lee (2017); Shrikumar et al. (2017); Sundararajan et al. (2017); Murdoch et al. (2018); De Cao et al. (2020); Kim et al. (2020)
Counterfactual	Yang et al. (2020); Ross et al. (2021)
Example-based	Bien and Tibshirani (2011); Koh and Liang (2017); Khanna et al. (2019); Han et al. (2020); Guo et al. (2020); Han and Ghosh (2020)
Rule-based	Stumpf et al. (2009); Ribeiro et al. (2018a); Tang and Surdeanu (2021); Sushil et al. (2021)
Textual	Abujabal et al. (2017); Camburu et al. (2018); Park et al. (2018a); Liu et al. (2019a); Rajani et al. (2019); Brahman et al. (2021)

Table 2.1.: Categorization of existing local explanation methods based on the form of explanations.

- **Surrogate models.** We can study a complicated model via another interpretable model which is trained to mimic the behavior of the complicated model. We call the simple model **a surrogate model** or **a mimic model**. To obtain a surrogate model M' of a model M, first, we select a dataset $\mathcal{D}$ and apply M to each example $x \in \mathcal{D}$. After we obtain the prediction $M(x)$, we pair it with x to be a training example to train M'. We can use the standard training algorithm of M' or some advanced techniques such as model distillation (Hinton et al., 2015). Good surrogate models should well replicate the predictions of the original model for even unseen examples (i.e., having high fidelity), while being simple enough to understand. Examples of research following the surrogate model approach are Bastani et al. (2017); Tan et al. (2018); Sushil et al. (2018); Lakkaraju et al. (2019).

Note that even though some classes of techniques including adversarial examples (Jia and Liang, 2017; Ribeiro et al., 2018b), adversarial attacks (Wallace et al., 2019; Li et al., 2020a), challenge sets (Nie et al., 2020; Gardner et al., 2020), and neuron interpretation (as in Section 2.2.1) reveal the overall model behavior irrespective of any specific inputs, we consider them as model analysis methods rather than global explanation methods since they do not explain the end-to-end prediction logic.

Other categorizations.

Some methods generate **interactive explanations** using, for instance, web interfaces or dialog systems, where a user can interact and explore specific aspects of interest (Kulesza et al., 2009; Tenney et al., 2020). In contrast, **static explanation** methods usually present the most important factors (for or against the prediction) to prevent the user from information overload.

In addition, some explanation methods, such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), are **model-agnostic** and do not require access to model parameters. Other methods access the model architectures and parameters to generate the explanations, such as DeepLIFT (Shrikumar et al., 2017) and LRP (layer-wise relevance propagation) (Bach et al., 2015; Arras et al., 2016). These methods are **model-specific** as they work with some architectures (such as neural networks) but not the others (such as ensemble models).

2.2.3. Relevant Explanation Methods

In this section, we review explanation methods which are used or built upon in this thesis. These methods are LRP (Arras et al., 2016, 2017) used in Chapters 4-5 and Grad-CAM (Selvaraju et al., 2017) used in Chapter 4.

LRP

Layer-wise Relevance Propagation (LRP) is a method for explaining predictions of neural networks in terms of importance scores of input features (i.e., input-based explanations) (Bach et al., 2015). Originally, it was devised to explain predictions of image classifiers by creating a heatmap on the input image highlighting pixels that are important for the classification. Then Arras et al. (2016) and Arras et al. (2017) extended LRP to work on CNNs and RNNs for text classification, respectively.

Consider a neuron k whose value is computed using n neurons in the previous layer,

$$a_k = g(\sum_{j=1}^{n} a_j w_{jk} + b_k) \tag{2.12}$$

where a_k is the value of the neuron k, g is a non-linear activation function, w_{jk} and b_k are weights and bias in the network, respectively. We can see that the contribution of a single node j to the value of the node k is

$$z_{jk} = a_j w_{jk} + \frac{b_k}{n} \tag{2.13}$$

assuming that the bias term b_k is distributed equally to the n neurons. LRP works by propagating the activation of a neuron of interest back through the previous layers in the network proportionally. We call the value each neuron receives a relevance score (R) of the neuron. To back propagate, if the relevance score of the neuron k is R_k, the relevance score that the neuron j receives from the neuron k is

$$R_{j \leftarrow k} = \frac{z_{jk}}{\sum_{j'=1}^{n} z_{j'k}} R_k \tag{2.14}$$

To make the relevance propagation more stable, we add a small positive number ϵ (as a stabilizer) to the denominator of the propagation rule:

$$R_{j \leftarrow k} = \frac{z_{jk}}{\epsilon + \sum_{j'=1}^{n} z_{j'k}} R_k \tag{2.15}$$

This propagation rule is called LRP-ϵ. For more details about other LRP propagation rules, please see (Montavon et al., 2019). Finally, we can obtain the relevance score of j in total by summing up $R_{j \leftarrow k}$ for all k in the next layer. If we back propagate until j is at the input layer, we will obtain the relevance score of the input feature j as desired.

Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) is an input-based explanation method, initially devised for 2D convolutional neural networks (CNNs) used in computer vision tasks. These CNNs can be divided into two parts, as explained in Section 2.1.2, which are the feature extraction part (convolutional and pooling layers) and the classification part (dense layers). However, the output of the feature extraction part is not a feature vector but a list of feature maps A^k (i.e., two-dimensional matrices) each of which corresponds to a CNN filter k in the last convolutional layer. A^k is smaller than the original input image; however, each value in A^k can roughly represent an area in the input image at the similar relative position.

Grad-CAM calculates the relevance scores by relying on the gradient-based approach. In other words, the importance of the input feature x_i towards the output y can be defined as $\frac{\partial y}{\partial x_i}$, showing how a small change in x_i would affect the value of y. Adapting this idea to CNNs, first, Grad-CAM computes the gradient of y^c (i.e., the predicted score before the softmax of the class c) with respect to every pixel of feature maps A^k of the last convolutional layer. To find the relevance score of filter k towards class c (α_k^c), it performs global average pooling over the gradient of every pixel in A^k. Given Z is the number of pixels in A^k,

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{2.16}$$

Next, Grad-CAM performs a weighted combination of the feature maps where the weights are α_k^c.

$$L^c = \mathrm{ReLU}\left(\sum_k \alpha_k^c A^k\right) \tag{2.17}$$

The ReLU function is applied to keep only relevance scores for the target class c. Note that L^c is smaller than the input image. So, Grad-CAM up-samples L^c to the input image size using bi-linear interpolation. This results in the relevance score of every pixel in the original input image as desired.

2.3. Computational Argumentation

With incomplete or inconsistent information, humans use argumentation, both internally and externally, to form a conclusion and/or to make a decision (Atkinson et al., 2017). We can see argumentation practices in many real-world situations such as political debates, legal procedures, and scientific knowledge development and discussions. As a prominent aspect of human intelligence, argumentation has also been studied in a computational way under symbolic AI. This research field is called *Computational Argumentation*, concerning several tasks which include identifying arguments, analyzing their dialectical relationships, evaluating their strengths, and presenting the argumentation to achieve specific purposes (Walton, 2009; Atkinson et al., 2017). Apart from its applications in several domains (e.g., collaborative decision making (Aurisicchio et al., 2015), clinical decision support (Chapman et al., 2019), and legal reasoning (Bench-Capon et al., 2009)), computational argumentation has also been used in many explainable AI methods (see (Čyras et al., 2021; Vassiliades et al., 2021) for recent overviews of argumentative XAI).

In Chapter 6 of this thesis, we use quantitative bipolar argumentation frameworks (Baroni et al., 2019) to enable argumentative explanations for text classification. So, this section provides relevant background in computational argumentation needed for that chapter.

2.3.1. Abstract Argumentation

There are two main approaches to represent an argument. One of them, i.e., *abstract argumentation*, treats an argument as a single abstract entity. The other one, i.e., *structural argumentation* or *assumption-based argumentation*, treats an argument as a deduction from assumptions to a conclusion, so we can see components inside an argument. However, in this thesis, we follow the former approach. Definition 1 defines an abstract argumentation framework (AF), as introduced in (Dung, 1995).

Definition 1. *An abstract argumentation framework $\mathcal{F}$ is a pair $\langle \mathcal{A}, \mathcal{R}^- \rangle$, where $\mathcal{A}$ is a set of arguments and $\mathcal{R}^-$ is a binary relation of attack on $\mathcal{A}$, i.e., $\mathcal{R}^- \subseteq \mathcal{A} \times \mathcal{A}$.*

According to Dung (1995), anything can be an argument as long as it is in dialectical relation with other arguments (e.g., attacking or to be attacked). To visualize an AF, we use nodes and edges as arguments and relations between them, respectively. Figure 2.3 is an example of AF with $\mathcal{A} = \{a, b, c, d\}$ and $\mathcal{R}^- = \{(c, b), (d, b), (b, a)\}$ (c attacks b, d attacks b, and b attacks a).

Definition 2. *Given an abstract argumentation framework $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}^- \rangle$ and an argument $a \in \mathcal{A}$, $\mathcal{R}^-(a)$ is a set of arguments in $\mathcal{A}$ which attack a. In other words, $\mathcal{R}^-(a) = \{b \in \mathcal{A} | (b, a) \in \mathcal{R}^-\}$.*

For the AF in Figure 2.3, $\mathcal{R}^-(a) = \{b\}$, whereas $\mathcal{R}^-(b) = \{c, d\}$. Indeed, $\mathcal{R}^-(c) = \mathcal{R}^-(d) = \varnothing$. Moreover, we can define the notion of attack involving sets of arguments as follows.

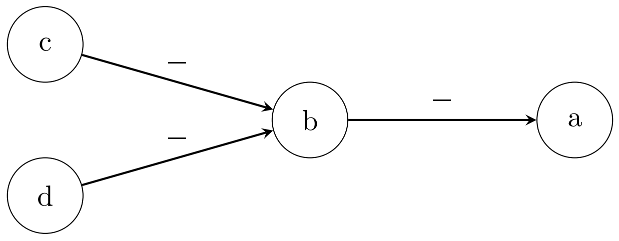


Figure 2.3.: An example of abstract argumentation framework.

- For $P \subseteq \mathcal{A}$ and $Q \subseteq \mathcal{A}$, P attacks Q if and only if (iff) there are $p \in P$ and $q \in Q$ such that p attacks q.
- For $P \subseteq \mathcal{A}$ and $q \in \mathcal{A}$, P attacks q iff there exists $p \in P$ that attacks q.

Given an AF in general, we can evaluate the arguments using an *extension-based semantics*, which is a method to identify some subsets of $\mathcal{A}$ that satisfy some meaningful dialectical constraints. Such "good" subsets of arguments are called *extensions*. Definition 3 lists some examples of extension-based semantics discussed in (Dung, 1995).

Definition 3. *Given an AF $\mathcal{F} = \langle \mathcal{A}, \mathcal{R}^- \rangle$, we say that the subset $\mathcal{E} \in \mathcal{A}$ is*

- ***conflict-free*** *iff $\mathcal{E}$ does not attack itself.*
- ***admissible*** *iff $\mathcal{E}$ is conflict-free and attacks any argument attacking a member in $\mathcal{E}$.*
- ***preferred*** *iff $\mathcal{E}$ is maximal (with respect to $\subseteq$) admissible set of $\mathcal{F}$.*
- ***stable*** *iff $\mathcal{E}$ is conflict-free and attacks every argument in $\mathcal{A} - \mathcal{E}$.*

As the AF in Figure 2.3 has four arguments, there are $2^4 = 16$ subsets of $\mathcal{A}$. Nine of them are conflict-free, including $\varnothing$, $\{a\}$, $\{b\}$, $\{c\}$, $\{d\}$, $\{a, c\}$, $\{a, d\}$, $\{c, d\}$, and $\{a, c, d\}$. All of them are also admissible except $\{a\}$ and $\{b\}$. However, only $\{a, c, d\}$ is stable and preferred.

2.3.2. Quantitative Bipolar Argumentation Frameworks

Quantitative Bipolar Argumentation Frameworks (QBAF) which we will use in Chapter 6 are different from the basic argumentation framework in Section 2.3.1 in two aspects, i.e., *bipolarity* and *gradual semantics.*

Definition 4. *A bipolar argumentation framework (BAF) $\mathcal{F}$ is a triplet $\langle \mathcal{A}, \mathcal{R}^-, \mathcal{R}^+ \rangle$, where $\mathcal{A}$ is a set of arguments whereas $\mathcal{R}^-$ and $\mathcal{R}^+$ are binary relations of attack and support on $\mathcal{A}$, respectively. In other words, $\mathcal{R}^- \subseteq \mathcal{A} \times \mathcal{A}$ and $\mathcal{R}^+ \subseteq \mathcal{A} \times \mathcal{A}$.*

Bipolar argumentation adds a new relation type between arguments which is *support* via $\mathcal{R}^+$ (Cayrol and Lagasquie-Schiex, 2005). For an argument $a \in \mathcal{A}$, the definition of $\mathcal{R}^+(a)$ is analogous to $\mathcal{R}^-(a)$ in Definition 2 (i.e., $\mathcal{R}^+(a) = \{b \in \mathcal{A} | (b, a) \in \mathcal{R}^+\}$). Also, different interpretations of support in the literature lead to many extension-based semantics proposed for BAF (Cohen et al., 2014).

Another type of semantics used for BAF is *gradual semantics* where each argument is coupled with a base score (showing the argument's internal strength) and then the dialectical strength of each argument will be computed by aggregating the strengths of its attackers and the strengths of its supporters. The higher the dialectical strength, the more acceptable the argument is. So, we do not have "good" sets of arguments out of the gradual semantics unless we set a threshold to separate the good from the bad. A BAF using gradual semantics with a base score equipped with each argument can be called a quantitative BAF (QBAF) of which the definition is shown below (as formalized in (Baroni et al., 2019)).

Definition 5. *A quantitative bipolar argumentation framework (QBAF) $\mathcal{F}$ is a quadruple $\langle \mathcal{A}, \mathcal{R}^-, \mathcal{R}^+, \tau \rangle$, where $\mathcal{A}$, $\mathcal{R}^-$, and $\mathcal{R}^+$ are defined in the same way as BAF (see Definition 4), and a total function $\tau : \mathcal{A} \to \mathbb{I}$ indicates the base score of each argument in $\mathcal{A}$.*

In other words, $\tau(a)$ is the base score of $a \in \mathcal{A}$. Additionally, we use σ to represent the (dialectical) strength function for arguments in $\mathcal{A}$, defined below (Baroni et al., 2019).

Definition 6. *For any $a \in \mathcal{A}$, the strength of a is given by $\sigma(a)$ where $\sigma : \mathcal{A} \to \mathbb{I}$ is a total strength function. For any set of arguments $B \subseteq \mathcal{A}$, $\sigma(B)$ refers to the multiset $\{\sigma(b) | b \in B\}$.*

$\mathbb{I}$ in Definitions 5 and 6 is the range of the base scores and the strengths. Figure 2.4 shows an example of QBAF together with the base scores and the strengths of the arguments noted as value pairs. So far, there have been different strength functions σ proposed (e.g., DF-QuAD (Rago et al., 2016)

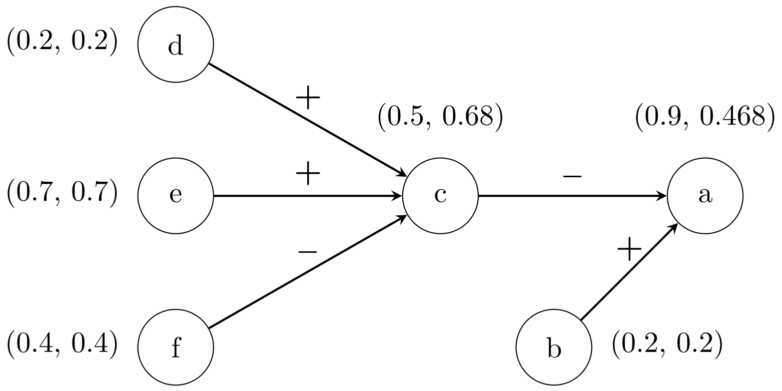


Figure 2.4.: An example of quantitative bipolar argumentation framework (QBAF). With each argument, there is a value pair (x, y) where x and y represent the base score and the strength (based on DF-QuAD semantics (Rago et al., 2016)) of the argument, respectively.

and Euler-based (Amgoud and Ben-Naim, 2018)) amounting to different gradual semantics (with different desirable properties, discussed next).

2.3.3. Desirable Properties of Gradual Semantics

Due to a variety of gradual semantics proposed, many research papers introduce and study desirable properties that the semantics should satisfy to make the strength calculation process consistent with how humans evaluate arguments (Amgoud and Ben-Naim, 2018; Bonzon et al., 2016). Recently, Baroni et al. (2019) unified 29 literature properties into 11 group properties (GP) for generic QBAFs. Given a QBAF $\langle \mathcal{A}, \mathcal{R}^-, \mathcal{R}^+, \tau \rangle$ and $\alpha, \beta \in \mathcal{A}$,

- **GP1.** If $\mathcal{R}^-(\alpha) = \varnothing$ and $\mathcal{R}^+(\alpha) = \varnothing$, then $\sigma(\alpha) = \tau(\alpha)$.
- **GP2.** If $\mathcal{R}^-(\alpha) \neq \varnothing$ and $\mathcal{R}^+(\alpha) = \varnothing$, then $\sigma(\alpha) < \tau(\alpha)$.
- **GP3.** If $\mathcal{R}^-(\alpha) = \varnothing$ and $\mathcal{R}^+(\alpha) \neq \varnothing$, then $\sigma(\alpha) > \tau(\alpha)$.
- **GP4.** If $\sigma(\alpha) < \tau(\alpha)$, then $\mathcal{R}^-(\alpha) \neq \varnothing$.
- **GP5.** If $\sigma(\alpha) > \tau(\alpha)$, then $\mathcal{R}^+(\alpha) \neq \varnothing$.
- **GP6.** If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) = \sigma(\beta)$.

- **GP7.** If $\mathcal{R}^-(\alpha) \subset \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) > \sigma(\beta)$.
- **GP8.** If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) \subset \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) < \sigma(\beta)$.
- **GP9.** If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) < \tau(\beta)$, then $\sigma(\alpha) < \sigma(\beta)$.
- **GP10.** If $\mathcal{R}^-(\alpha) < \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) > \sigma(\beta)$.
- **GP11.** If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) < \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) < \sigma(\beta)$.

The definition of < between two sets used in GP10 and GP11 is defined as follows. Given P and Q are subsets of $\mathcal{A}$, $P \leq Q$ iff there exists an injective mapping f from P to Q such that $\forall \alpha \in P, \sigma(\alpha) \leq \sigma(f(\alpha))$. Furthermore, $P < Q$ iff $P \leq Q$ but $Q \nleq P$. Actually, there are other properties defined for QBAF such as balanced, monotonicity, anonymity, independence, etc. (Baroni et al., 2019; Potyka, 2021); however, we do not review them here since they are outside the scope of this thesis.

2.3.4. From Argumentation Frameworks to Explanations

Mapping arguments into an argumentation framework is a standard way for enabling machines to evaluate the arguments. However, argumentation frameworks generally contain more than necessary information for the end users. Also, the notions used could be unfamiliar to lay users. Many explanation methods relying on computational argumentation, hence, do not present the argumentation framework as the explanation as it is, but convert the framework into another form which is easily consumable and suitable for the target task. Atkinson et al. (2017) refers to this step as the *rhetorical layer* of argumentation. Examples of the form include *extensions* showing arguments which are reasons for the acceptability of a target argument (Fan and Toni, 2015), *dispute trees* visualizing a debate between a proponent and an opponent (Modgil and Caminada, 2009), and *dialogues* allowing the users to engage with for more details (amounting to traversing the underlying argumentation framework) (Cocarascu et al., 2019). Besides, counterfactual explanations can be generated from an argumentation framework by suggesting a modification to the input (which in turn modulates the argumentation framework) that would change the target prediction (Albini et al., 2021).

2.4. Summary

We presented, in this chapter, the essential background for the thesis. Specifically, we started from introducing the text classification task, which we will focus on, as well as explaining classification models used in the thesis. Then we reviewed and categorized advances in interpretability and explainability for NLP before closing with the background about computational argumentation and its connection with explainability.

In the next chapter, we will explain in more detail current state-of-the-art in specific topic areas of this PhD thesis. Also, we will provide critical analyses of these works and summarize research gaps we particularly tackle in this thesis.

3. Related Work

This chapter describes and critically analyzes the current state-of-the-art in three specific areas which are relevant to this thesis. Section 3.1 discusses related works concerning local and global explanations for text classifiers. Section 3.2 focuses on evaluation of explanation methods (including both intrinsic and extrinsic evaluations). Section 3.3 explains the general framework for human debugging of NLP models via explanations and analyzes state-of-the-art approaches for doing so. Within these three sections, we also emphasize where our contributions are specifically as well as the research gaps they address to highlight the significance of our work.

3.1. Explanations for Text Classifiers

Text classification is a fundamental task in NLP, so there exist several explanation methods which are applicable to this task. Concerning *local explanation* methods, which provide explanations for classifications for specific inputs, many of them are input-based and model-agnostic. In other words, given an input text, these methods explain the model's predicted class by identifying relevant parts in the input. Since they aim to work with any type of models, these methods try to generate the explanations while treating the model as a black box. One prominent way to do so is to perturb the original input and observe how the model responses. For example, the leave-one-out method (Li et al., 2016b) defines a relevance score of an input word (or token) by the drop in the predicted probability after the word (or token) being removed from the input. LIME (Ribeiro et al., 2016) generates neighbor examples around the original input and feeds them to the model to obtain the corresponding predictions. Then it fits a linear classifier to mimic the behavior of the model locally around the original input. During training, neighbor examples which are more similar (defined by a specific distance function) to the original input have higher weights in the objective function. Finally, the weight of each feature (i.e., word or token) in the linear model is treated as the relevance score of the feature. Similarly, SHAP (Lundberg and Lee, 2017) relies on input perturbation. Unlike LIME, it has a more systematic way to perturb the input and a different distance function in the objective function. Nonetheless, the perturbed inputs of these perturbation-based methods are usually out of the distribution that the target model was trained on since they could be ungrammatical due to some words being dropped. Hence, the model behavior for these perturbed inputs may not reflect its behavior when being applied to normal inputs. To remedy this issue, Kim et al. (2020) perturbed each input token by trying all possible tokens in that position (according to the likelihood specified by a language model) and then marginalized the effect to be the relevance score of the token.

In contrast, model-specific methods are devised to generate explanations for specific types of models. Thereby, they can get access to and leverage the model architecture and parameters. For instance, because computing SHAP is time-consuming in general, Lundberg and Lee (2017) also proposed TreeSHAP which is an efficient variant of SHAP designed particularly for tree-based machine learning models (e.g., decision trees and random forests). As discussed in Section 2.2.2, DeepLIFT (Shrikumar et al., 2017) and LRP (layer-wise relevance propagation) (Bach et al., 2015; Arras et al., 2016) are methods applicable to neural networks since they rely on relevance propagation from the target output node to the input nodes, taking into account the network connections and weights. Gradient-based analysis is another group of model-specific methods, applying to models where derivative of the output with respect to the input feature is computable (including neural networks). The broad idea is that the derivative reflects the sensitivity of the output when there is a small change in the input. Methods of this group are such as Saliency (Simonyan et al., 2013), Integrated Gradient (Sundararajan et al., 2017), and SmoothGrad (Smilkov et al., 2017). They are also applied to models beyond NLP as well. In the computer vision domain, there are methods designed specifically for a class of neural networks, namely convolutional neural networks (CNNs), such as Class Activation Mapping (CAM) (Zhou et al., 2016) and Gradient-weighted Class Activation Mapping (Grad-CAM) (Selvaraju et al., 2017). Although the NLP community is using CNNs in many existing works, there is still no explanation method which is applicable specifically to CNN models under the NLP setting. In Chapter 4 of this thesis, we fill the gap by proposing two model-specific local explanation methods for 1D CNN text classifiers. One of them adapts Grad-CAM to make it suitable for the one-dimensional (1D) structure. The other one combines a surrogate decision tree with n-grams selected by max-pooling of the CNNs to generate local explanations.

When it comes to *interpretable models*, there has been very little attention to explain them since their inherent explanations are already obvious and faithful to the models. This in turn makes other desirable properties of explanations[1] for these models underexplored. For example, the product of a feature value and the corresponding weight in a binary logistic regression model can be seen as the contribution of the feature towards the positive class. However, if the model contains features which are not independent from one another, the individual contribution, explained above, may not accurately reflect how the model utilizes the corresponding feature since the contribution was trained to interact with other contributions from dependent features. This results in the explanations being less plausible to humans. In Chapter 6 of this thesis, we define a novel local explanation method for pattern-based logistic regressions to deal with this problem, thus filling the research gap.

To address dependencies between features, we choose to use computational argumentation in our proposed explanation method. In fact, there have been some previous works leveraging computational argumentation to generate explanations for classification tasks. For example, Cocarascu et al. (2020) worked on an argumentation-based classification model, called AA-CBR (Cyras et al., 2016), using training examples as arguments. These arguments argue with each other to classify the target example. Hence, their argumentative explanation can be directly obtained from the argumentative classification model. Dejl et al. (2021) generated argumentative explanation by extracting an argumentation framework from the target neural network where arguments come from neurons and relations come from connections between the neurons. Sendi et al. (2019) explained a deep ensemble model by extracting interpretable rules from the base models of the ensemble and using the rules as arguments arguing to find the final output. However, we can see that none of the existing works focuses exclusively on

[1]We will explain desirable properties of explanations in detail in Section 3.2.

logistic regression. Therefore, our work in Chapter 6 contributes to the field in this aspect. Note that our work is somewhat related to *argument mining* where arguments and their relations are extracted from natural language texts (Lawrence and Reed, 2019). Nonetheless, the arguments and relations of our work are not directly from the input text but from the pattern-based logistic regression model to be explained instead (i.e., using pattern-based features as arguments with specificity relations between the patterns defining dialectical relations).

In contrast to local explanations, there are fewer works studying *global explanations* for text classifiers. This is because interpretable text classifiers can be the global explanations themselves while black-box classifiers are usually too involved to be captured faithfully in global explanations at the complexity level where humans can understand. As introduced in Section 2.2.2, there are two main classes of global explanations for NLP – one using collections of local explanations and the other one using surrogate models. Some of them are model-agnostic such as (Ribeiro et al., 2016, 2018a; Bastani et al., 2017; Setzu et al., 2021). The rest are model-specific such as (Lundberg et al., 2020) for tree-based models, (Sushil et al., 2018) for neural networks, and (Sushil et al., 2021) for recurrent neural networks. These global explanations may sometimes reveal weaknesses of the model (i.e., something that the model has wrongly learned from the training data); however, they did not show how to leverage the global explanations for mitigating the problems. Actually, these methods were intentionally designed just for explaining but not for supporting model debugging and improvement. To fill this gap, in Chapter 5, we provide a new method based on neuron visualization which globally explains deep text classifiers (in particular 1D CNNs) where the explanations enable us to easily remove what the model has wrongly learned.

3.2. Evaluation of Explanation Methods

With so many explanation methods available, the next challenge is how to evaluate them so as to choose the right methods for different settings. Existing evaluation methods can be grouped into two main types with regards to what they evaluate, as shown in Figure 3.1. The first type of evaluation assesses the quality of the generated explanations, while the second type

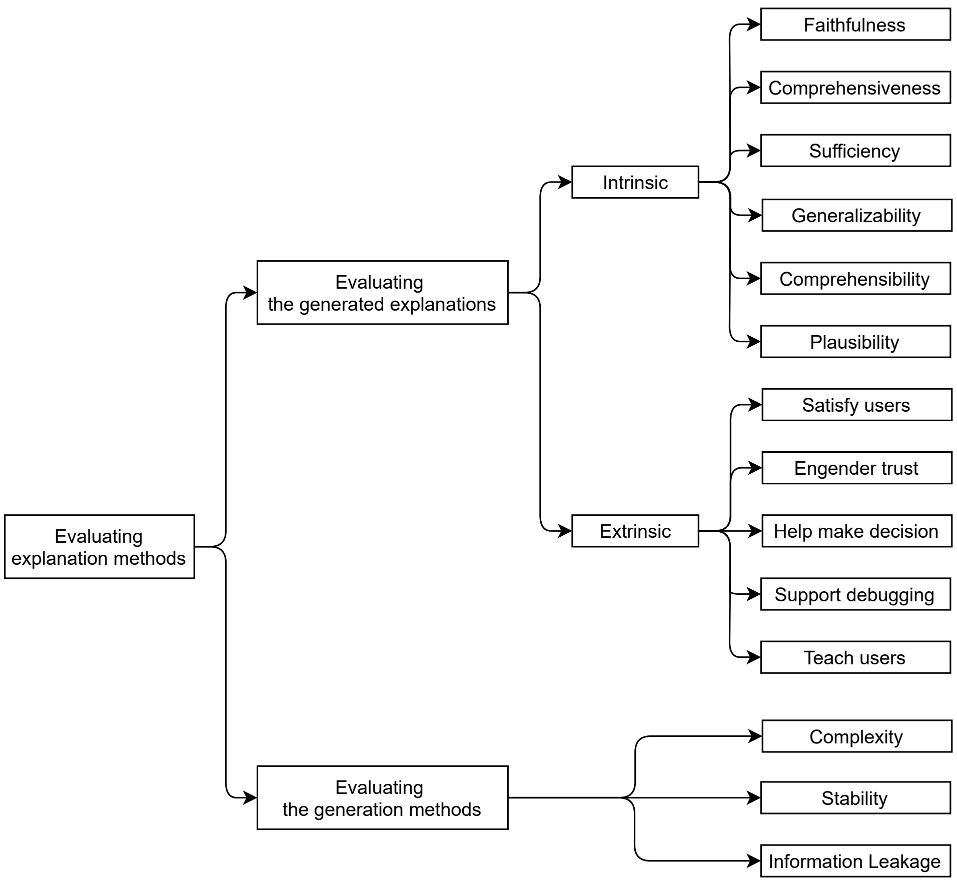


Figure 3.1.: A taxonomy of evaluation methods for explanations.

evaluates the generation methods or the algorithms themselves. In this section, we discuss only the first type since the second type is out of the scope of this thesis.

When evaluating the generated explanations, we need to decide which aspect of the explanations we want to evaluate. This can then be categorized into two groups including **intrinsic evaluation** (focusing on desirable properties of explanations) and **extrinsic evaluation** (focusing on how useful the explanations are in downstream tasks). We explain both of them here.

3.2.1. Intrinsic Evaluation of Explanations

Intrinsic evaluation aims to quantify desirable properties. One of early works in explainable AI by Kulesza et al. (2013) studied two properties of explanations, namely **soundness** and **completeness**, and how they impact end users' mental models and trust. In their work, soundness amounts to the extent to which each component of an explanation's content is truthful in describing the underlying system. In contrast, completeness amounts to the extent to which all of the underlying system is described by the explanation. However, Kulesza et al. (2013) did not propose any methods to evaluate the degree of soundness or completeness of a given explanation in their work.

For the last five years, when explainable AI has been studied and applied to deep NLP models, **faithfulness** is another desirable property of explanations discussed in many works. However, most of them mention it without defining what exactly faithful or faithfulness means (Ribeiro et al., 2018a; Gilpin et al., 2018; Lakkaraju et al., 2019; Setzu et al., 2021). This makes evaluation methods for faithfulness being different across different papers. Recently, Jacovi and Goldberg (2020) published a paper to discuss this issue in particular. They defined that faithfulness refers to how accurately the interpretation reflects the true reasoning process of the model. Later, Jacovi and Goldberg (2021) said more specifically that a faithful interpretation is the accurate representation of the causal chain of decision making in the model, and explanation is a process of conveying causal information about the model's decision to a person. According to these definitions, faithfulness is a more general concept, with soundness and completeness, as defined by Kulesza et al. (2013), being aspects of faithfulness.

However, the definitions of faithfulness by Jacovi and Goldberg (2020, 2021) emphasize a significant issue concerning evaluating faithfulness of explanations. For deep learning models which are black boxes, humans do not know the models' true reasoning process, leading to the lack of ground truths when evaluating faithfulness. Therefore, based on our observations, many existing works consider faithfulness in a looser meaning instead: *an explanation is faithful if the model demonstrates the same behavior as the explanation says.* For example, some previous works evaluated faithfulness of relevance scores (for text classification) by word deletion – gradually deleting words from the input text in the order of their relevance and checking how the prediction confidence drops (Arras et al., 2016; Nguyen, 2018). If we remove a word with a high relevance score (according to the explanation) from the input and then the predicted probability given by the model drops drastically, we can say that the model demonstrates the same behavior as the explanation says, i.e., treating that word as very relevant to the prediction[2]. Hence, the explanation – the relevance score – is faithful.

Nevertheless, evaluating explanations by word deletion relies on the linearity assumption, i.e., the contributions of different parts of the input are independent from each other (Jacovi and Goldberg, 2020), but this assumption does not hold in practice because the model might rely on a combination of input words. So, word deletion is not a perfect method to evaluate faithfulness even with respect to the looser meaning of faithfulness. This problem could be mitigated by deleting a set of words instead of word-by-word and checking the correlation between the probability change and the sum of the relevance scores of the deleted words (Bhatt et al., 2020a).

Note that the idea of deletion can also be applied to evaluate example-based explanations. Particularly, one can remove influential examples from the training data and see how the prediction performance changes (Han et al., 2020). If the change is significant, it means that the explanation method faithfully points out influential examples (according to the looser meaning of faithfulness).

Apart from relevance scores, some previous works use a surrogate model, i.e., a simpler model mimicking the behavior of the target complex model, as an explanation. Because the causal chain of the surrogate model is very likely different from the causal chain of the target model (e.g., a decision tree versus a convolutional neural network), we can hardly say that the surrogate model is faithful to the target model according to the stricter definition of faithfulness by Jacovi and Goldberg (2021). To evaluate the quality of the surrogate model, nonetheless, some previous works consider that the surrogate model is faithful if its predictions are the same as the predictions of the models it explains. We call the percentage of the same predictions **fidelity** (Lakkaraju et al., 2019). Because high fidelity shows that the model demonstrates the same behavior as the explanation says, people can use fidelity to measure faithfulness (with respect to the looser meaning of faithfulness we noted above). In any case, regardless of which definition of faithfulness we use, we must not involve humans when evaluating faithfulness because faithfulness is the property with respect to the underlying model only (Jacovi and Goldberg, 2020).

[2]Note that the leave-one-out explanation method (Li et al., 2016b) will obtain the perfect score from the word deletion evaluation method as they are equivalent (relying on the same principle).

Additionally, when using rationale extraction (i.e., using some parts of the input text as explanation), two properties which are related to faithfulness (in the looser meaning) are **comprehensiveness** and **sufficiency**, proposed by DeYoung et al. (2020). An explanation has high comprehensiveness when it covers every reason that the model could use to make the same prediction. This concept is similar to recall in standard classification. However, we do not have any ground truth to evaluate comprehensiveness directly. So, DeYoung et al. (2020) proposed that, if a model M predicts class j for an input x, the comprehensiveness of a rationale r (extracted from x) can be reflected by $M(x)_j - M(x_{\setminus r})_j$ where $M(x)_j$ is the probability of class j the model M predicts for the input x and $x_{\setminus r}$ is the input x with the rationale r removed. We say that r is more comprehensive when $M(x)_j - M(x_{\setminus r})_j$ is high. In contrast, according to DeYoung et al. (2020), sufficiency concerns whether only the explanation (i.e., the extracted rationale) is sufficient to lead to the prediction. In other words, the sufficiency s of the rationale r equals $M(x)_j - M(r)_j$. If s is close to 0, it means the rationale r is sufficient to contribute (at least) almost equally to $M(x)_j$. However, we need to be cautious when using comprehensiveness and sufficiency proposed by DeYoung et al. (2020). First, if an extracted rationale is comprehensive, it does not mean that the model uses every part of the rationale for the prediction. In fact, it may rely only on a few parts when making the prediction, and the rest will not be exploited unless the few important parts are missing. Second, according to the definition above, an explanation r is sufficient if and only if $M(x)_j \approx M(r)_j$. Nonetheless, r is actually a part in x, and there could be other parts in x that are evidence against class j. Therefore, it may need more than r in x to make the model predicts class j for x even though r is sufficient to make the model predict class j in isolation.

Another desirable property which is applicable to local explanations is **generalizability**. It amounts to how well the explanation generalizes (i.e., still holds) beyond the target example. This property is desirable because some may want to exploit insights learned from the explanation to similar examples (Sokol and Flach, 2020). It could be measured via a support set, i.e., the percentage of examples that the explanation applies. Note that Sokol and Flach (2020) referred to this property as **completeness** which is different from completeness defined by Kulesza et al. (2013).

Besides, there are also properties of explanations that involve humans (i.e., the explainees). One of them is **comprehensibility** concerning whether the explanations are understandable to humans. Other names for this property include **understandability**, **human-interpretability**, and **readability**. This could be assessed via some proxy metrics, based on the idea that simplicity enables comprehensibility. For example, using decision sets as explanations, Lakkaraju et al. (2017) considered, for instance, the number of rules, the number of predicates, and the maximum width of the conditions to be the inverse indicators of comprehensibility. Using relevance scores of input features as an explanation, Bhatt et al. (2020a) considered sparseness of the scores as a proxy of comprehensibility due to an assumption that humans are capable of understanding explanations with a few features than too many features.

Besides, we can measure comprehensibility by involving humans into the evaluation process. In the case where the model is complex but still understandable to machine learning experts (e.g., a system of multiple interpretable models), one may present the explanation of the system to lay users and elicit their understandings about the system after seeing the explanation. If their understandings match the answers from machine learning experts, we can consider the explanation high-quality (understandable) (Hoffman et al., 2018). Furthermore, Hase and Bansal (2020) used simulatability to evaluate explanation methods. Specifically, they let human participants learn from model predictions and the associated explanations and asked them to predict the model behavior on a new input. Correct answers imply comprehensibility of the explanations. It must be careful, however, that simulatability here is not only affected by comprehensibility but also by faithfulness of the explanation. In other words, the explanations must be faithful and comprehensible so that humans can simulate the prediction results correctly.

Last but not least, **plausibility** is another property of explanations which take humans into account. It amounts to how convincing the explanations are to humans (Jacovi and Goldberg, 2020; Wiegreffe and Pinter, 2019). Hence, we somehow need humans to evaluate this plausibility. Mohseni et al. (2018) proposed a benchmark which contains a list of relevant words for the actual class of each input text, identified by human experts. Then they compared machine explanation to human-identified words. Wiegreffe and Marasović (2021) compiled a list of datasets with human explanations that could be used to assess plausibility. It is noteworthy that less plausible explanations could be due to not only the poor explanation method but also the inaccuracy of the model or the model reasoning differently from humans. Thus, when we compare plausibility of different explanation methods, we need to make sure that they are applied to the same model.

Even though there are many properties we can evaluate intrinsically, we do not always require all of them. It depends on why and how we use the explanations. For example, if our user is a machine learning engineer who wants to verify the model via the explanations, faithfulness is the most important property (as in Chapter 5). Meanwhile, if we want to use explanations to persuade lay users, plausibility is of our interest (as in Chapters 4 and 6) whereas comprehensiveness is not that important because humans do not explain using all the possible reasons either (Miller, 2019). All in all, faithfulness, comprehensiveness, sufficiency, and generalizability are evaluated with respect mainly to the underlying model, while comprehensibility and plausibility are evaluated with respect mainly to the users.

In the following chapters, we will refer to faithfulness, fidelity, sufficiency, and plausibility. Because some of these terms have several definitions in the literature as explained above, we provide their definitions considered in this thesis as follows.

Definition 7. *Given a text classification model M, an input text x, a local explanation E which explains the output class $M(x)$, and a human user H (i.e., the explainee),*

- *E is **faithful** if M demonstrates the same behavior as indicated in E with respect to the process of M predicting the output for x.*
- *Let x consist of x_+, x_-, and x_0 which are parts or features of x that contribute positively, negatively, and nothing to $M(x)$, respectively. For $E \in x_+$, E is **sufficient** if $M(x - x_+ + E) = M(x)$ where $x - x_+ + E$ is the input x without any parts or features supporting $M(x)$ except E.*
- *E is **plausible** if it is convincing to H that E is a reason for $M(x)$.*

Definition 8. *Given a text classification model M, a surrogate model E which is a global explanation of M, and a test dataset $\mathcal{D}$, the fidelity of E is the percentage of $x \in \mathcal{D}$ such that $M(x) = E(x)$.*

It can be seen that the definitions of fidelity and plausibility we use are the same as in related work. We adopt the looser meaning of faithfulness here to make it consistent with most of the related work (although we do not fully endorse this definition). Finally, our definition of sufficiency concerns not only the explanation E but also the other context in x to avoid the weakness of what DeYoung et al. (2020) defined, as discussed above.

3.2.2. Extrinsic Evaluation of Explanations

Extrinsic evaluation focuses on evaluating explanations according to a target downstream task. One of the most basic tasks of explanations is to **satisfy users**, covering many factors as perceived by the users (e.g., usefulness, understandability, sufficiency, actionability, and accuracy). Hoffman et al. (2018) proposed that both researchers who generate the explanations and end users who consume the explanations can perform the evaluation. Specifically, according to Hoffman et al. (2018), the researchers evaluate goodness of the explanations a priori whereas the end users evaluate the satisfaction a posteriori. It is important particularly in applications aiming to satisfy users such as recommendation systems and personal assistants. Existing works assess user satisfaction mainly by using interviews or questionnaires (Gedikli et al., 2014; Biran and McKeown, 2017; Narayanan et al., 2018).

In addition, it has been discussed in literature that explanations can **engender human trust** in the AI (Symeonidis et al., 2009; Mercado et al., 2016; Miller, 2019). One standard way to measure the trust is to ask humans explicitly through questionnaires how much they trust the model (before and after seeing the explanations) and let them answer using rating scales (Bussone et al., 2015; Smith-Renner et al., 2020). The words used in the questions could actually be different. For instance, Pu and Chen (2006) measured user trust via perceived competence and intention to return to use the AI agent after the users saw explanations. Recent work, however, found that self-reported trust may not be reliable (Schaffer et al., 2019). So, Zhang et al. (2020) considered, as a measure of trust, the percentage of the times that the user decided to use the AI's prediction as their final prediction even though the user's initial prediction disagreed with the AI's. Jacovi et al. (2021) further suggested that it is important, when evaluating the user trust in the AI, to set up the situation where the user considers some actions of the AI unfavorable to them and both the favorable and unfavorable outcomes could happen from the AI actions.

In addition to predictions and confidence scores computed by the AI, explanations could also **help humans make more accurate decisions**. One way to evaluate explanation methods is, therefore, to measure how well humans perform when being assisted by explanations from such methods. For example, Lai and Tan (2019) compared AI performance (full automation) to human performance when they are shown, as supporting information by the AI, nothing, only explanations, only predicted labels, both explanations and predicted labels, and both plus whether the AI is confident. Lai and Tan (2019) also compared input-based explanations (corresponding to weights of their linear SVM classifier) and example-based explanations (based on their nearest neighbors classifier) in this experiment.

Another downstream application is using explanations to **support model inspection and debugging**. When a model does not work properly, good explanation methods should be able to highlight the misbehavior to the users. Ribeiro et al. (2016) asked human participants to choose, from two given models, the one which can generalize better by considering their local explanations (either LIME or a greedy baseline). The more answers the participants got correct, the better the explanation method they read was. Besides, example-based explanations of incorrect predictions can help identify potentially wrong labels in the training data. Koh and Liang (2017) and Khanna et al. (2019) compared their explanation methods with others by checking how many mislabeled examples they found out of all the suspicious training examples reported by the explanation methods. As one of our contributions in the thesis is directly related to using explanations to enable human debugging of NLP models, we discuss this topic in detail in Section 3.3.

Last but not least, for some tasks, machine learning models are more capable than humans such as authorship attribution (Juola, 2007) and deceptive review detection (Lai et al., 2020). Thus, the models may learn something that lay humans have never known or noticed. Thereby, an explanation method is useful if it can reveal such insights learned by the models to

teach humans to perform the tasks better (Mac Aodha et al., 2018). The higher human performance after learning from the explanations can be a measure of the quality of the explanation method in this regards (Lai et al., 2020).

3.2.3. Comparison across Explanation Methods

When a new explanation method is proposed, it must be evaluated against some baselines to demonstrate the effectiveness. However, datasets, properties, and baselines used in the experiments could be different across different papers. Hence, it is also important to conduct research comparing several existing explanation methods using the same setup so as to make reliable comparison and conclusion. Most of the existing works which compare many explanation methods focus on intrinsic evaluation. For instance, Poerner et al. (2018b) experimented on eight different explanation methods (some of which have several configurations) to check if each method correctly *identifies the known root cause* of the prediction. In other words, they evaluated plausibility of the explanations. If the underlying model works nearly perfectly and the task is not too complex, it can be assumed that the model relies on the correct known root cause. So, the match between the explanation and the root cause could also reflect the faithfulness of the explanations under this assumption. Additionally, using seven datasets, DeYoung et al. (2020) evaluated *comprehensiveness* and *sufficiency* of the explanations returned from four explanation methods by modifying the input according to the explanation and investigating the change in the predicted probability as discussed in Section 3.2.1. Similarly, Nguyen (2018) compared four explanation methods to evaluate their local faithfulness by checking the number of words in the explanations that needs to be deleted from the input before the prediction switches to another class. These experiments assess qualities of the explanations with regards to the underlying model without human involvement. In fact, Nguyen (2018) also conducted human evaluation with the four methods to check if humans can correctly guess the model prediction by seeing the input and the explanation or not. This evaluated plausibility of the explanation. However, the results might be inaccurate as humans could biased by the input shown. One of the evaluation methods we propose in Chapter 4, therefore, mitigates this issue by hiding the input from the human participants.

In contrast, the works conducting evaluation of explanations for NLP tasks with respect to downstream applications (i.e., extrinsic evaluation) are still very limited. Existing works compare only few explanation methods. For example, Ribeiro et al. (2016) compared greedy baselines and LIME for identifying better text classifiers and compared two different variants of LIME for model debugging. As explained above, Lai and Tan (2019) compared explanations from two inherently interpretable models (linear SVM and nearest neighbors) for helping humans make decisions. After that, Lai et al. (2020) compared three explanations – weights of linear SVM, attention scores of BERT (Devlin et al., 2019), and LIME scores of BERT – for training humans to perform a difficult task. All in all, the community still lacks research comparing various post-hoc explanation methods in NLP with respect to various downstream applications. This is the major research gap we aim to address in Chapter 4 – formalizing the evaluation methods and conducting experiments with different types of post-hoc explanation methods.

3.3. Explanation-Based Human Debugging of NLP Models

This section reviews advances specifically on how explanations have been used in the literature to enable humans to fix bugs in NLP models. We refer to this research area as *explanation-based human debugging (EBHD)*, as a general umbrella term encompassing *explanatory debugging* and *human-in-the-loop debugging*.

As noted in Chapter 1, the term *debugging* is interpreted differently by different researchers. It may refer to a process of identifying or uncovering causes of model errors according to some researchers (Parikh and Zitnick, 2011; Graliński et al., 2019). Meanwhile, others stress that debugging must not only reveal the causes of problems but also fix or mitigate them (Kulesza et al., 2015; Yousefzadeh and O'Leary, 2019). In this thesis, we adopt the latter interpretation.

Assuming that there is no implementation mistake in the source code, the causes of model errors usually arise from imperfect training data. As

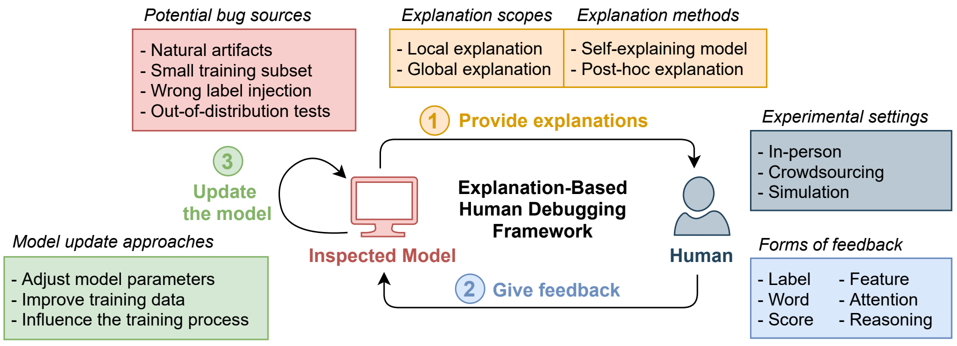
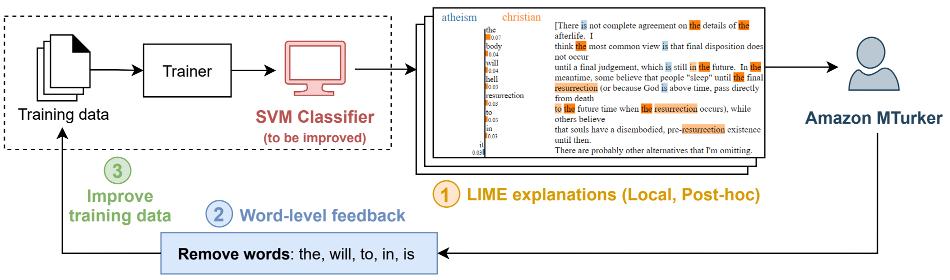


Figure 3.2.: A general framework for explanation-based human debugging (EBHD) of NLP models.



[There is not complete agreement on the details of the afterlife. I think the most common view is that final disposition does not occur until a final judgement, which is still in the future. In the meantime, some believe that people "sleep" until the final resurrection (or because God is above time, pass directly from death to the future time when the resurrection occurs), while others believe that souls have a disembodied, pre-resurrection existence until then. There are probably other alternatives that I'm omitting.

Figure 3.3.: The proposal by Ribeiro et al. (2016) as an instance of the general EBHD framework.

introduced in Section 2.1.3, there are some cases when the causes can be anticipated and fixed beforehand (Park et al., 2018b; Liu and Avci, 2019; Jaiswal et al., 2020). However, sometimes the causes or even the problems cannot be predicted and are only found after training using post-hoc error analysis. In these cases, human knowledge is needed to verify and improve the trained models. To enable humans to do so effectively and efficiently, explanations come to play a role here, leading to the process of explanation-based human debugging (EBHD).

Overall, existing EBHD approaches can be seen as instances of our proposed general framework in Figure 3.2. To demonstrate the debugging process, existing works need to set up the bug situation they aim to fix, amounting to the target NLP task, the machine learning model used to carry out that task, and the source of the bug to be addressed. Then the EBHD process can be done generally in three main steps. First, the model provides interpretable insights about the bugs to human debuggers via explanations. After that, the humans inspect the explanations and give feedback in response. In the experiments, this step could be done in-person, by crowd sourcing, or by simulation. Finally, the feedback is used to update and improve the model by directly adjusting model parameters, improving the training data, or influencing the (re)training process. Note that these three steps can be carried out once, as a one-off improvement, or iteratively, depending on how the debugging framework is designed.

As a concrete example, Figure 3.3 illustrates how Ribeiro et al. (2016) improved an SVM text classifier (Joachims, 1998) trained on the 20Newsgroups dataset (Lang, 1995)[3]. This dataset has many artifacts which could make the model rely on wrong words when making predictions, reducing its generalizability. To perform EBHD, Ribeiro et al. (2016) recruited humans from a crowdsourcing platform (i.e., Amazon Mechanical Turk) and asked them to inspect LIME explanations (word attribution scores) for model predictions of ten examples. After that, the humans gave feedback by identifying words in the explanations that should have not got high attribution scores (supposed to be the artifacts). These words were then removed from the training data, and the model was retrained. The process was repeated for three iterations, and the results show that the model generalizes better after every iteration. Using the general EBHD framework in Figure 3.2, we can break the framework of Ribeiro et al. (2016) into components as depicted in Figure 3.3.

Concerning the big picture of the field, early EBHD work comes from the human-computer interaction community. Kulesza et al. (2009) aimed to let humans improve a Naive Bayes email classification model. As the classifier is self-explaining, its explanations can be easily derived from the model. They allowed the users to ask questions (e.g., "Why will this message be filed to folder A?") and presented the explanations as answers using template-based sentences and some bar plots. The users then can adjust the bar plots, corresponding to the parameters of the underlying model to fix the bugs. Stumpf et al. (2009) studied the types of feedback humans usually give in response to machine-generated predictions and explanations. The most prominent ones include removing-adding features (words), tuning weights, and leveraging feature combinations. Also, some of the feedback collected (i.e., important words of each category) was used to improve their

[3]`http://qwone.com/~jason/20Newsgroups/`

Naive Bayes classifier via two techniques – constraint optimization and user co-training. Kulesza et al. (2010) used auto-coding of transcripts as the target task and asked humans to check local explanations generated by the system (i.e., important words in the input) so as to confirm or correct the explanations. Words identified as unimportant were removed from the model, whereas the identified important words became features with high weights for the user-specified category. Later, in 2015, Kulesza et al. proposed, as desirable principles, that the presented explanations for explanatory debugging should be sound and complete, but not overwhelming. They also presented an EBHD system which satisfies these principles. The system was made *interactive* to reveal incremental changes in real time after each user action and allow the user to undo the action for which the results are not desirable. The interactiveness is possible as their model update step does not require re-training the model but changing the model parameters directly. Nonetheless, all these early works targeted simple machine learning classifiers (i.e., Naive Bayes classifiers with bag-of-words). So, it is not obvious how to apply their proposed approaches to deep learning-based text classifiers where we do not clearly see how the models use the input words. Also, the principles suggested by Kulesza et al. (2015) are indeed challenging when working on non-interpretable complex models.

Recently, thanks to several post-hoc explanation methods proposed, there have been new attempts to use explanations and human feedback to debug classifiers in general. Some of them were tested on traditional text classifiers. One of them is (Ribeiro et al., 2016) as described earlier (cf. Figure 3.3). In addition, Teso and Kersting (2019) proposed CAIPI, which is an explanatory interactive learning framework. At each iteration, it selects an unlabeled example to predict and explain to users using LIME, and the users respond by identifying irrelevant features from the explanation. CAIPI then uses this feedback to generate augmented data that reduce the importance of the identified irrelevant features and retrains the model using both the original training data and the augmented data. It is noteworthy that both (Ribeiro et al., 2016) and (Teso and Kersting, 2019) present local explanations to elicit feedback from the users and the explanations change after model update. Therefore, their processes can be done iteratively (but not interactively though as retraining the model takes time). Iterative debugging is beneficial because humans can fix vital bugs first and finer bugs in later iterations, as we can see from the performance plots in (Ribeiro et al., 2016). However, it could also be susceptible to the phenomenon called *local decision pitfalls* where local improvements for individual predictions could add up to inferior overall performance (Wu et al., 2019b). Meanwhile, to the best of our knowledge, there is still no work using global explanations to debug deep NLP models in the EBHD setting. Chapter 5 in the thesis, therefore, aims to fill this gap, using global explanations to enable human debugging of deep text classifiers (with a focus on 1D CNNs).

So far, all the existing works discussed exploit input-based explanations in their framework. Another line of research of EBHD focuses on using example-based explanations to debug the model by improving the quality of the training data. As briefly introduced in Section 3.2.2, Khanna et al. (2019) randomly flipped the labels of a random 20% of the training examples so as to set up the bug situation. To debug the trained model, they identified training examples which were responsible for misclassifications of the validation data using their proposed (explanation) method. Next, they simulated the manual inspection of the returned examples and corrected their labels if needed before retraining the model. To correct the labels, they cross-checked the current labels with the true labels (before the random flipping process). This step is equivalent to humans providing feedback in the EBHD framework, while the retraining step is equivalent to the model update step. This process could also be used to evaluate example-based explanation methods extrinsically, as conducted in (Koh and Liang, 2017; Khanna et al., 2019; Han and Ghosh, 2020). Particularly, with the same amount of training data inspected, an example-based explanation method which leads to higher accuracy after model update is considered to be a better explanation method. However, these methods are effective when there are wrong labels in the training data, but they do not work when spurious correlations emerge despite correct labels. In contrast, our debugging work in Chapter 5 focuses more on the latter case and, hence, targets a different bug situation from these methods.

3.4. Summary

This thesis contributes in three main research areas: explanations for text classifiers, evaluation of explanation methods, and explanation-based human debugging of NLP models. Below is the summary of research gaps, with respect to existing works, identified earlier in this chapter and addressed within this thesis in later chapters.

Explanations for text classifiers

- The community still lacks explanation methods which are applicable specifically to 1D CNN models used in many NLP applications. Hence, we propose two novel local explanation methods – adapted from Grad-CAM and decision trees – in Chapter 4.
- Explanations for interpretable models do not always satisfy every desirable property of explanations while their dedicated explanation methods are still underexplored. In Chapter 6, we propose a novel local explanation method (based on computational argumentation) aiming to improve plausibility of the explanations for pattern-based logistic regression to fill this research gap.
- Existing global explanation methods were devised intentionally for explaining the model but not for supporting model debugging and improvement. In Chapter 5, we fill this gap by proposing a new form of global explanation based on neuron analysis allowing us to easily remove what the model has wrongly learned.

Evaluation of explanation methods

- The notion of sufficiency of explanations as proposed by DeYoung et al. (2020) does not consider the interaction between the extracted rationale and the rest of the input. So, we propose a new definition of sufficiency in Section 3.2.1 to address this issue. We also adapt this definition to evaluate explanations in Chapter 6.
- The community still lacks research comparing various post-hoc explanation methods in NLP with respect to various downstream applications. Chapter 4 in this thesis mainly targets this issue by formalizing human evaluation methods for three downstream tasks and conducting the experiments with different types of post-hoc explanation methods.

Explanation-based human debugging of NLP models

- Most of the existing approaches use local explanations in their EBHD frameworks which could cause local decision pitfall, leading to sub-optimal models. There is still no work using global explanations to debug deep NLP models in the EBHD setting. Hence, we fill this gap by proposing a novel framework using global explanations to enable human debugging of deep text classifiers (with a focus on 1D CNNs), to be discussed in Chapter 5.

4. Human-Grounded Evaluations of Explanation Methods

As introduced in the previous chapter, it is important to compare different explanation methods to select the most suitable one for a use case of interest. Still, research on evaluating explanation methods for specific explanation usage is scarce. Some purposes of explanation usage (i.e., model inspection and improvement and human decision support) have been studied with only few explanation methods (Ribeiro et al., 2016; Lai and Tan, 2019). There are other purposes of usage which have not been experimented with at all. Also, the community still lacks comparisons of various post-hoc explanation methods in NLP for potential human-AI collaboration tasks. To fill this gap, in this chapter, we propose three evaluation tasks which target different purposes of explanations for text classification – *(1)* revealing model behavior to human users, *(2)* justifying the predictions (i.e., providing sensible reasons for the predicted classes), and *(3)* helping humans investigate uncertain predictions. We then use these tasks to evaluate nine input-based explanation methods working on 1D CNNs for text classification. These explanation methods are different in several aspects. For example, regarding granularity, four explanation methods select words from the input text as explanations, whereas the other five select n-grams as explanations. In terms of generality, one of the explanation methods is model-agnostic, two are random (i.e., worst case) methods, another two (newly proposed in this chapter) are specific to 1D CNNs for text classification, and the rest are applicable to neural networks in general. So, our work is more comprehensive than previous related works in terms of the various human-grounded evaluation tasks proposed and the number and dimensions of explanation methods being evaluated.

The rest of this chapter is organized as follows. Section 4.1 proposes the three novel human-grounded evaluation tasks for three purposes of explanation usage for text classification. Section 4.2 describes the experimental setup where we evaluate the nine explanation methods with the proposed evaluation tasks. The two newly proposed explanation methods for 1D CNNs are also presented in this section. Then Section 4.3 discusses experimental results, highlighting dissimilar qualities of the explanation methods and pointing out the most suitable explanation method for each purpose. Finally, Section 4.4 summarizes our contributions and important findings of this chapter.

4.1. Proposed Evaluation Methods

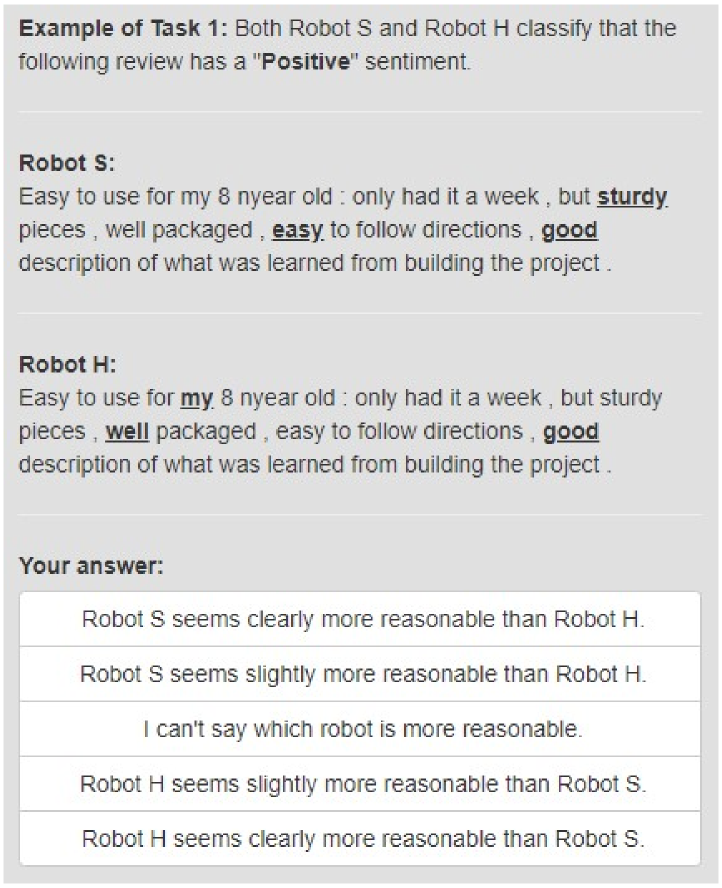
We propose three human tasks to evaluate explanation methods for text classification as summarized in Table 4.1. Following the notations in Section 2.1.1, let M and x be the target model and an input text, respectively. Also, let $\hat{y} = M(x) \in \mathcal{C}$ be the predicted class. An explanation method may return rationales or relevance scores of input words as explanations. To normalize these forms, we represent each *explanation* ζ as an ordered list of text fragments (words or n-grams) in x which are most relevant to a prediction $\hat{y}$. In other words, an explanation ζ amounts to a sequence $\langle z_1, \ldots, z_k \rangle$ where z_i is a word or an n-gram from the input text x ordered by its relevance. Here, we have $k \geq 0$ where $k = 0$ means that the explanation is empty. Explanations for and against the predicted class $\hat{y}$ are called *evidence* ($\zeta^+ = \langle z_1^+, \ldots, z_{k^+}^+ \rangle$) and *counter-evidence* ($\zeta^- = \langle z_1^-, \ldots, z_{k^-}^- \rangle$), respectively. Generally, having evidence text fragments in the input makes the model more certain about its predicted class, while having counter-evidence text fragments in the input makes the model less certain about its predicted class. Given $\zeta = \langle z_1, \ldots, z_k \rangle$, we can also specify the top-m (evidence or counter-evidence) text fragments as ζ_m which equals $\langle z_1, \ldots, z_m \rangle$ where $m \leq k$.

4.1.1. Task 1: Revealing the Model Behavior

Task 1 evaluates whether explanations can expose irrational behavior of a poor model. This property of explanation methods is very useful when we do not have a labelled dataset to evaluate the model quantitatively. To set up the task, firstly, we train two models to make them have different

	Task 1 (Section 4.1.1)	**Task 2** (Section 4.1.2)	**Task 3** (Section 4.1.3)
Assumption	Good explanations can reveal model behavior	Good explanations justify the predictions	Good explanations help humans investigate uncertain predictions
Model(s)	Two classifiers with different performance on a test dataset	One well-trained classifier	One well-trained classifier
Input text	A test example for which both classifiers predict the same class	A test example which the classifier predicts with high confidence ($\mathbf{p}_{\hat{y}} > \tau_h$)	A test example which the classifier predicts with low confidence ($\mathbf{p}_{\hat{y}} < \tau_l$)
Information displayed	1. The input text 2. The predicted class 3. (Highlighted) top-m evidence texts of each model	1. Top-m evidence texts	1. The predicted class 2. The predicted probability $\mathbf{p}$ 3. Top-m evidence and top-m counter-evidence texts
Human task	Humans select the more reasonable model and state if they are confident or not	Humans select the most likely class of the document which contains the evidence texts and state if they are confident or not	Humans select the most likely class of the input text and state if they are confident or not
Scores to the explanation method	(-)1.0: (In)correct, confident (-)0.5: (In)correct, unconfident 0.0: No preference	(-)1.0: (In)correct, certain (-)0.5: (In)correct, uncertain 0.0: No preference	(-)1.0: (In)correct, certain (-)0.5: (In)correct, uncertain

Table 4.1.: A summary of the proposed human-grounded evaluation tasks where τ_h and τ_l are thresholds for high confidence and low confidence predictions, respectively.


Example of Task 1: Both Robot S and Robot H classify that the following review has a "**Positive**" sentiment.

Robot S:
Easy to use for my 8 nyear old : only had it a week , but **sturdy** pieces , well packaged , **easy** to follow directions , **good** description of what was learned from building the project .

Robot H:
Easy to use for **my** 8 nyear old : only had it a week , but sturdy pieces , **well** packaged , easy to follow directions , **good** description of what was learned from building the project .

Your answer:

Robot S seems clearly more reasonable than Robot H.

Robot S seems slightly more reasonable than Robot H.

I can't say which robot is more reasonable.

Robot H seems slightly more reasonable than Robot S.

Robot H seems clearly more reasonable than Robot S.

Figure 4.1.: Example question for Task 1 (revealing the model behavior) and the user interface for the experiment.

performance on classifying testing examples (i.e., different capability to generalize to unseen data). Then we use these models to classify an input text and apply the explanation method of interest to explain the predictions – highlighting top-m evidence text fragments on the text for each model. Next, we ask humans, based on the highlighted texts from the two models, which model is more reasonable. If the performance of the two models is clearly different, good explanation methods should enable humans to notice the poor model, which is more likely to decide based on non-discriminative (i.e., irrelevant) words, even though both models predict the same class for an input text. Normally, humans will answer that the model with a more plausible explanation is more reasonable. Therefore, an explanation method will score well in this task if the plausibility of the generated explanation positively correlates with the performance of the underlying model.

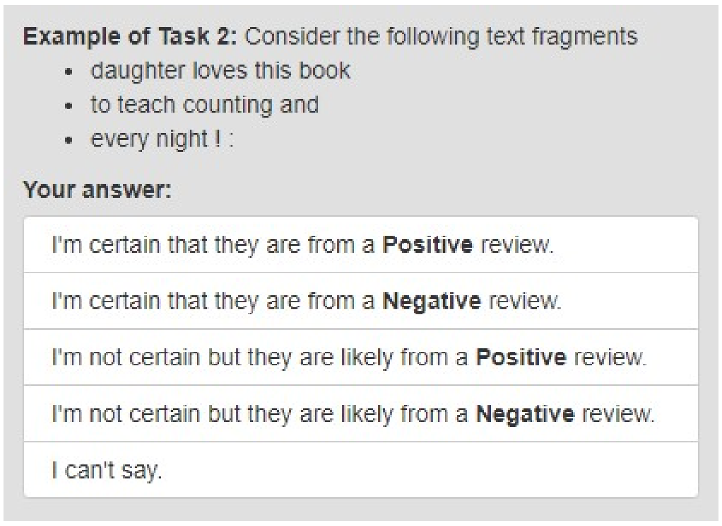
We formalize this task as follows.

Task 1. *To evaluate the performance of a target explanation method, we need*

- *Two text classification models M_A and M_B where M_A outperforms M_B on a test dataset with F1 difference Δ and*
- *An input text x such that $M_A(x) = M_B(x)$.*

We generate an evaluation question by applying the target explanation method to compute the explanations for x, $(\zeta_m^+)_A$ and $(\zeta_m^+)_B$ (i.e., top-m evidence text fragments for the predictions $M_A(x)$ and $M_B(x)$, respectively). Then we show the humans (1) *the input text x twice, one highlighted with $(\zeta_m^+)_A$ and the other highlighted with $(\zeta_m^+)_B$ and* (2) *the predicted class. Next, we ask the humans which model they believe is more reasonable. Five options are provided to the humans: M_A seems clearly more reasonable than M_B, M_A seems slightly more reasonable than M_B, I can't say which model is more reasonable, M_B seems slightly more reasonable than M_A, M_B seems clearly more reasonable than M_A. The scores that the target explanation method will obtain when the humans select these options are +1.0, +0.5, 0.0, -0.5, and -1.0, respectively.*

An example question for this task in the real user interface used in our experiment is shown in Figure 4.1. There are some important points to elaborate and note for this task. First, as stated in the formalization above, the chosen input texts must be classified into the same class by both models so the humans make decisions based only on the different explanations. However, it is worth to consider both the cases where both models correctly classify and where they misclassify. Second, to make the questions understandable to lay humans in practice, we choose to use the word *Robot* instead of *model* as shown in Figure 4.1. Also, we randomly select the order of the good and the poor models shown for every question and sample English characters (without replacement) to be the robot names (e.g., Robot S and Robot H in Figure 4.1). These aims to prevent the humans from associating the model quality with the order shown. Third, we provide the answer options for the two models along with confidence levels for the humans to select. In Figure 4.1, options with the word *clearly* indicate that


Example of Task 2: Consider the following text fragments

- daughter loves this book
- to teach counting and
- every night ! :

Your answer:

I'm certain that they are from a **Positive** review.

I'm certain that they are from a **Negative** review.

I'm not certain but they are likely from a **Positive** review.

I'm not certain but they are likely from a **Negative** review.

I can't say.

Figure 4.2.: Example question for Task 2 (justifying the predictions) and the user interface for the experiment.

the humans are confident, whereas options with the word *slightly* indicate that they are unconfident. If they select the right model with high confidence, the explanation method will get a higher positive score (+1.0). In contrast, a confident but incorrect answer results in a large negative score (-1.0). Also, the humans have the option to state no preference (i.e., *I can't say which robot is more reasonable* in Figure 4.1), for which the explanation method will get a zero score.

4.1.2. Task 2: Justifying the Predictions

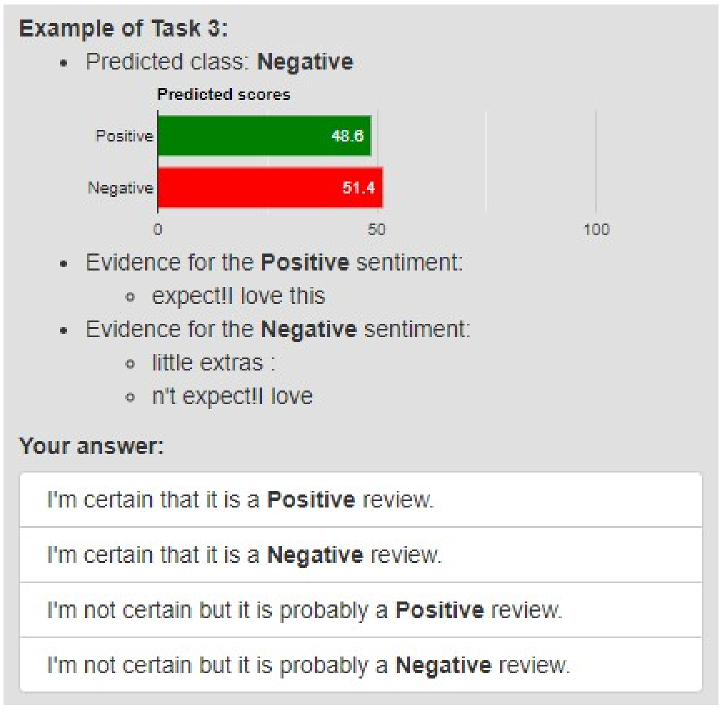
Explanations are sometimes used by humans as the reasons for the predicted class. Task 2 tests whether the evidence texts are truly related to the predicted class and can distinguish it from the other classes, so called *class-discriminative* (Selvaraju et al., 2017). To set up the task, we use a well-trained model and select an input example classified by this model with high confidence ($\mathbf{p}_{\hat{y}} > \tau_h$ where τ_h is a threshold parameter), so as to reduce the cases of unclear explanations due to low model accuracy or text ambiguity. (Note that we look at low-confidence predictions in Task 3.) Then we show only the top-m evidence text fragments generated by the method of interest to humans and ask them to guess the class of the document containing the evidence. The explanation method which makes the humans surely guess the class predicted by the model will get a high positive score. Formally, a question of Task 2 can be generated as follows.

Task 2. *To evaluate the performance of a target explanation method, we need*

- *A well-trained text classification model M and*
- *An input text x such that $\hat{y} = M(x)$ and $\mathbf{p}_{\hat{y}} > \tau_h$ where τ_h is a threshold parameter and $\frac{1}{|\mathcal{C}|} < \tau_h < 1$.*

We generate an evaluation question by applying the target explanation method to compute the explanation ζ^+_m (i.e., top-m evidence text fragments for the prediction $\hat{y}$). Then we show ζ^+_m to the humans and ask them, based on the explanation, which class of the input text these evidence text fragments are from. Regarding answer options, we generate two options for each possible class in $\mathcal{C}$, one saying that the humans are certain of their answer and the other one saying that they are uncertain. For the certain options selected by the humans, the target explanation method will obtain +1.0 if the selected class is $\hat{y}$ and -1.0 if the selected class is not $\hat{y}$. For the uncertain options selected by the humans, the target explanation method will obtain +0.5 if the selected class is $\hat{y}$ and -0.5 if the selected class is not $\hat{y}$. We additionally provide the "I can't say" option where the target explanation method will obtain 0.0 if this option is selected.

An example question for this task in the real user interface used in our experiment is shown in Figure 4.2. Concerning τ_h, it must be less than 1 because it is a probability threshold. Also, there is no point to go lower than $\frac{1}{|\mathcal{C}|}$ since the predicted probability of the winning class is guaranteed to be greater than $\frac{1}{|\mathcal{C}|}$ in the context of multi-class classification. As in the previous task, this task considers both the correct and incorrect predictions with high confidence to see how well the explanations justify each of the cases. For incorrect predictions, an explanation method gets a positive score when a human guesses the same incorrect class after seeing the explanation. In real applications, convincing explanations for incorrect classes can help humans understand the model's weakness and suggest additional examples to retrain and improve the model. Note that this task is somewhat related to plausibility of the explanations; however, we do not directly measure plausibility as we do not match machine explanations to human explanations. We


Example of Task 3:

- Predicted class: **Negative**

- Evidence for the **Positive** sentiment:
  - expect!! love this
- Evidence for the **Negative** sentiment:
  - little extras :
  - n't expect!! love

Your answer:

I'm certain that it is a **Positive** review.

I'm certain that it is a **Negative** review.

I'm not certain but it is probably a **Positive** review.

I'm not certain but it is probably a **Negative** review.

Figure 4.3.: Example question for Task 3 (investigating uncertain predictions) and the user interface for the experiment.

instead check whether the explanations can make humans associate them with the predicted class or not.

4.1.3. Task 3: Investigating Uncertain Predictions

If an AI system makes a prediction with low confidence, it may need to raise the case with humans and let them decide, but with the analyzed results as additional information. Task 3 aims to check if the explanations can help humans comprehend the situation and correctly classify the input text or not. To set up, we use a well-trained model and an input text classified by this model with low confidence ($\mathbf{p}_{\hat{y}} < \tau_l$ where τ_l is a threshold parameter). Then we apply the explanation method of interest to find top-m evidence and top-m counter-evidence texts of the predicted class. We present both types of evidence to humans[1] together with the predicted class and probability $\mathbf{p}$ and ask the humans to use all the information to

[1]We present counter-evidence as evidence for the other classes to simplify the task questions.

guess the actual class of the input text, without seeing the input text itself. To sum up, we can formalize Task 3 as follows.

Task 3. *To evaluate the performance of a target explanation method, we need*

- *A well-trained text classification model M and*
- *An input text x such that $\hat{y} = M(x)$ and $\mathbf{p}_{\hat{y}} < \tau_l$ where τ_l is a threshold parameter and $\frac{1}{|C|} < \tau_l < 1$.*

We generate an evaluation question by applying the target explanation method to compute the explanations ζ_m^+ and ζ_m^- (i.e., top-m evidence and counter-evidence text fragments for the prediction $\hat{y}$). Then we show the humans (1) the predicted class $\hat{y}$, (2) the predicted probability $\mathbf{p}$ of all the classes, and (3) ζ_m^+ and ζ_m^-. Next, we ask the humans to classify the input text x. Regarding answer options, we generate two options for each possible class in $\mathcal{C}$, one saying that the humans are certain of their answer and the other saying that they are uncertain. For the certain options selected by the humans, the target explanation method will obtain +1.0 if the selected class is actually the correct class of x and -1.0 if the selected class is not correct. For the uncertain options selected by the humans, the target explanation method will obtain +0.5 if the selected class is actually the correct class of x and -0.5 if the selected class is not correct.

An example question for this task in the real user interface used in our experiment is shown in Figure 4.3. The scoring criteria of this task are similar to the previous tasks except that we do not provide the "no preference" option as the humans can still rely on the predicted scores when all the explanations are unhelpful. Note that our questions in the experiment come from two scenarios equally, where $\hat{y}$ is correct and where $\hat{y}$ is incorrect.

4.2. Experimental Setup

4.2.1. Datasets

We used two English textual datasets for all the three tasks.
(1) **Amazon** Review Polarity is a sentiment analysis dataset for product reviews with positive and negative classes (Zhang et al., 2015). We chose

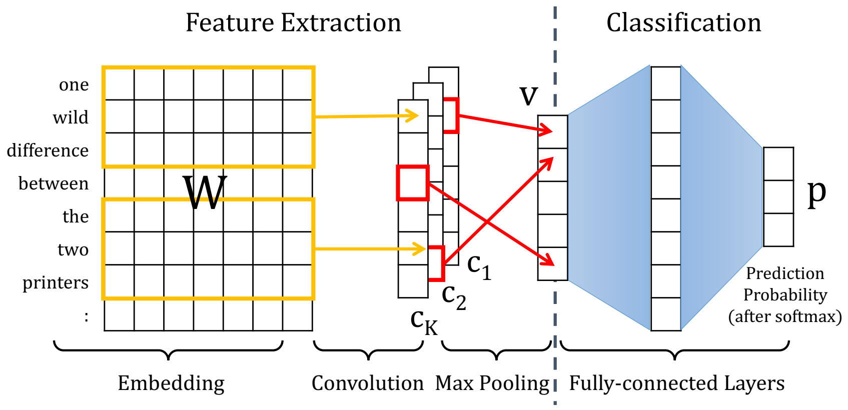


Figure 4.4.: A 1D CNN for text classification used in this chapter.

this dataset as representative of binary classification problems which do not require any domain-specific knowledge to perform. We randomly selected 100K, 50K, and 100K examples for training, validating, and testing the CNN models, respectively.
(2) **ArXiv** Abstract is a text classification dataset we created by collecting abstracts of scientific articles publicly available on ArXiv[2]. Particularly, we collected abstracts from the "Computer Science (CS)", "Mathematics (MA)", and "Physics (PH)" categories, which are the three main categories on ArXiv. We then created a dataset with three disjoint classes, removing the abstracts which belong to more than one of the three categories. We chose this dataset as representative of multi-class classification problems which require domain-specific knowledge (i.e., science and mathematics) to perform. In the experiments, we randomly selected 6K, 1.5K, and 10K examples for training, validating, and testing the CNN model, respectively.

4.2.2. Classification Models: 1D CNNs

As for the classifiers, we used 1D CNNs (explained in Section 2.1.2) with the same structure (shown in Figure 4.4) for all the tasks and datasets. Specifically, we used 200-dim GloVe vectors as non-trainable weights in the embedding layer (Pennington et al., 2014). The convolution layer had three filter sizes [2, 3, 4] with 50 filters for each size, while the intermediate fully-connected layer had 150 units. The activation functions of the filters and the

[2]https://arxiv.org

fully-connected layers are ReLU (except the softmax at the output layer). The models were implemented using Keras and trained with Adam optimizer[3] using categorical cross entropy as the loss function (batch size = 2048). Furthermore, to prevent overfitting, we used the validation split to perform early stopping. Specifically, if the validation loss does not improve for three epochs consecutively, the training process is stopped. Finally, we selected the model with the lowest validation loss as the (well-trained) model in the experiments. As a result, the macro-average F1 of the models on the test sets are 0.90 and 0.94 for the Amazon and the ArXiv datasets, respectively. Overall, classification with ArXiv appears to be an easier task as it is likely solvable by looking at individual keywords. In contrast, classification with the Amazon sentiment analysis is not quite easy: many reviews mention both pros and cons of the products, so a classifier needs to analyze several parts of the input to reach a conclusion. However, this is still manageable by the CNN architecture we used.

For task 1, we need another model which performs worse than the well-trained model. In this experiment, we trained the second CNNs (i.e., the worse models) for the two datasets in different ways to examine the capability of explanation methods in two different scenarios. For the Amazon dataset, while the first (well-trained) CNN needed eight epochs until the validation loss converged, we trained the second CNN for only one epoch to make it underfitting. For the ArXiv dataset, we trained the second CNN using the same number of examples as the first model but with more specific topics. To explain, we randomly selected only examples from the subclass 'Computation and Language', 'Dynamical Systems', and 'Quantum Physics' as training and validation examples for the class 'Computer Science', 'Mathematics', and 'Physics', respectively. In other words, the training and testing data of the worse CNN came from different distributions. The other settings for training these two worse models (including the loss function, the optimizer, and the batch size) are the same as used for the two well-trained models. As a result, the macro-average F1 of the worse CNNs are 0.81 and 0.85 for the Amazon and the ArXiv datasets, respectively, resulting in the Δ (i.e., the F1 difference between the better and the worse models) of 0.09 for both datasets. The full results of all the CNN

[3]We use the default parameters of Adam optimizer in Keras (see `https://keras.io/api/optimizers/adam/`).

1st CNN (better)	Prec.	Recall	F1	No. of examples
Negative	0.92	0.89	0.90	50039
Positive	0.89	0.92	0.90	49961
micro avg	0.90	0.90	0.90	100000
macro avg	0.90	0.90	0.90	100000
2nd CNN (worse)	Prec.	Recall	F1	No. of examples
Negative	0.82	0.81	0.81	50039
Positive	0.81	0.82	0.81	49961
micro avg	0.81	0.81	0.81	100000
macro avg	0.81	0.81	0.81	100000

Table 4.2.: Precision, Recall, and F1 scores of both CNNs for the Amazon dataset.

1st CNN (better)	Prec.	Recall	F1	No. of examples
Computer science	0.94	0.93	0.93	10000
Mathematics	0.92	0.93	0.92	10000
Physics	0.96	0.94	0.95	10000
micro avg	0.94	0.94	0.94	30000
macro avg	0.94	0.94	0.94	30000
2nd CNN (worse)	Prec.	Recall	F1	No. of examples
Computer science	0.96	0.74	0.84	10000
Mathematics	0.75	0.94	0.83	10000
Physics	0.89	0.88	0.89	10000
micro avg	0.85	0.85	0.85	30000
macro avg	0.87	0.85	0.85	30000

Table 4.3.: Precision, Recall, and F1 scores of both CNNs for the ArXiv dataset

models of the Amazon and the ArXiv datasets are reported in Tables 4.2 and 4.3, respectively.

4.2.3. Explanation Methods

We evaluated nine explanation methods as summarized in Table 4.4. First, we used Random (W) and Random (N) as two random (pseudo-)explanation methods serving as the worst case explanations. They select, respectively, words and non-overlapping n-grams randomly from the input text as evidence and counter-evidence. For the n-gram random method (and other n-gram based explanation methods in this chapter), n is one of the CNN filter sizes (2, 3, 4).

Second, we selected LIME which is a well-known model-agnostic perturbation-

Method Name	**Approach**	**Granularity**
Random (W)	Random Explanations	Words
Random (N)		N-grams
LIME	Perturbation	Words
LRP (W)	Relevance Propagation	Words
LRP (N)		N-grams
DeepLIFT (W)		Words
DeepLIFT (N)		N-grams
Grad-CAM-Text †	Gradient	N-grams
Decision Trees (DTs) †	Model Extraction	N-grams

Table 4.4.: Nine explanation methods evaluated. Examples of the generated explanations can be found in Table 4.5 and Appendix B.1. † denotes methods newly proposed in this thesis.

based method (Ribeiro et al., 2016). It trains a linear model using samples (5,000 samples in this chapter) around the input text to explain the importance of each word towards the prediction. The importance scores can be either positive (for the predicted class) or negative (against the predicted class).

Third, we selected layer-wise relevance propagation (LRP), specifically ϵ-LRP (Bach et al., 2015), and DeepLIFT (Shrikumar et al., 2017) which are applicable to neural networks in general and perform very well in several evaluations without humans (Xiong et al., 2018; Poerner et al., 2018b). LRP propagates the output of the target class (before softmax) back through layers to find attributing words, while DeepLIFT does the same but propagates the difference between the output and the predicted output of the reference input (i.e., all-zero embeddings in this chapter). These two methods assign relevance scores to every word in the input text. Words with the highest and the lowest scores are selected as evidence for and counter-evidence against the predicted class, respectively. Also, we extended LRP and DeepLIFT to generate explanations at n-gram level: we considered all possible n-grams in the input text where n is one of the CNN filter sizes; then the explanations are generated based on the relevance score of each n-gram, i.e., the sum of relevance scores of all words in the n-gram.

For more details about LIME, LRP, and DeepLIFT, please see Section 2.2.3.

Next, we searched for model-specific explanation methods which target specifically 1D CNNs for text classification. We found that Jacovi et al.

(2018) proposed one: listing only n-grams corresponding to feature values in $\mathbf{v}$ that pass thresholds for their filters. Each of the thresholds is set subject to sufficient purity of the classification results above it. However, their method is applicable to CNNs with only one linear layer as FC (as shown in Figure 2.1), while our CNNs have an additional hidden layer (with ReLU activation). So, we could not compare with their method in this work. To increase diversity in the experiments with model-specific explanation methods, we therefore propose two new methods applicable to 1D CNNs with multiple layers in FC, presented next.

Grad-CAM-Text

Because we could not find any explanation method developed specifically for 1D CNNs when we conduct this experiment, we search for model-specific methods applied to a similar architecture, i.e., 2D CNNs used in computer vision (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014). Generally, as in 1D CNNs, 2D CNNs consist of two main parts: the first performing convolution over the input to find the image representation and the second using the representation to make a prediction via feed-forward layers. Two major differences between 1D and 2D CNNs are: *(i)* 2D CNNs usually have more than one convolution layer unlike our 1D CNNs, which have exactly one convolution layer. *(ii)* each filter of the last convolution layer of a 2D CNN produces an activation map which is a matrix representing the input image. So, the whole representation is a tensor of K activation maps where K is the number of filters in the last convolution layer. In contrast, each filter in a 1D CNN produces only a scalar value. Hence, the whole input representation is a vector of K elements where K is the number of filters in the only convolution layer.

Among several explanation methods for 2D CNNs specifically, we find Grad-CAM (Selvaraju et al., 2017) being a very promising method for adapting to 1D CNNs. For more details about the original Grad-CAM, please see Section 2.2.3. Here, to make it applicable to 1D CNNs, we propose Grad-CAM-Text, adapting Grad-CAM to find the most relevant n-grams for text classification. Since each value in the feature vector $\mathbf{v}$ of the target 1D CNN corresponds to an n-gram selected by a filter, we use $E^{+}_{j,k}$ to show the effect of an n-gram selected by the k^{th} filter towards the prediction of class j:

$$E^{+}_{j,k} = |\max(\frac{\partial FC(\mathbf{v})_j}{\partial \mathbf{v}_k}, 0)| \times \mathbf{v}_k. \tag{4.1}$$

The partial derivative term shows how much the prediction of class j changes if the value from the k^{th} filter slightly changes. (The intuition is similar to Equation 2.16 of the original Grad-CAM, but our method does not need to average the derivatives over an activation map matrix since each 1D CNN filter produces only a scalar value.) As we are finding the evidence for the target class j, we consider only the positive value of the derivative. Then $E^{+}_{j,k}$ combines this term with the strength of $\mathbf{v}_k$ to show the overall effect of the k^{th} filter for the input text. Next, using $E^{+}_{j,k}$ for all the filters k, we calculate the effect of each word w_i in the input text towards class j by aggregating the effects of all the n-grams containing w_i.

$$E^{+}_{j,w_i} = \sum_k (E^{+}_{j,k} \times \mathbb{I}[w_i \in N_k]) \tag{4.2}$$

where N_k is an n-gram detected by the k^{th} filter. (The intuition is similar to Equation 2.17 of the original Grad-CAM. However, the original one needs to attribute the computed effect $E^{+}_{j,k}$ to the input image by bi-linear interpolation, whereas our Grad-CAM-Text can identify an n-gram in the input text that corresponds to each filter k precisely thanks to the max-pooling mechanism.) Lastly, to produce the final explanation, we select, as the evidence, non-overlapping n-grams which are detected by at least one of the filters and have the highest sums of the effects of all the words they contain. For example, to decide whether we will select the n-gram N_k as an evidence text or not, we consider $\sum_{w_i \in N_k} E^{+}_{j,w_i}$.

In contrast, to find counter-evidence of class j, we change max in Equation 4.1 to min because we are interested in the case where $FC(\mathbf{v})_j$ is lower when $\mathbf{v}_k$ is higher. Formally,

$$E^{-}_{j,k} = |\min(\frac{\partial FC(\mathbf{v})_j}{\partial \mathbf{v}_k}, 0)| \times \mathbf{v}_k. \tag{4.3}$$

$$E^{-}_{j,w_i} = \sum_k (E^{-}_{j,k} \times \mathbb{I}[w_i \in N_k]) \tag{4.4}$$

where $E^{-}_{j,k}$ and E^{-}_{j,w_i} are the effect of an n-gram selected by the k^{th} filter and the input word w_i *against* the prediction of class j. To decide whether we will select the n-gram N_k as an counter-evidence text of class j or not, we consider $\sum_{w_i \in N_k} E^{-}_{j,w_i}$.

Decision Trees

This explanation method is based on model extraction (Bastani et al., 2017). To generate the explanations, we create a decision tree (DT) which mimics the behavior of the classification part (fully-connected layers FC) of the trained CNN. This is different from what Bastani et al. (2017) proposed which mimics the behavior of the whole model. That cannot be applicable here since the numerical input of 1D CNNs is the embedding matrix, the values of which make no sense to humans and thus cannot be used in the explanations. Therefore, we propose a way to adapt it to 1D CNNs.

Given a filter-based feature vector $\mathbf{v}$, the DT needs to predict the same class as predicted by the CNN. Formally, we want

$$DT(\mathbf{v}) = \underset{j}{\operatorname{argmax}}\, \mathbf{p}_j = \underset{j}{\operatorname{argmax}}\, FC(\mathbf{v})_j. \tag{4.5}$$

For multi-class classification, we construct one DT for each class (one vs. rest classification). We employ CART with Gini index for learning DTs (Leo et al., 1984). All the training examples are generated by the trained CNN using a training dataset, whereas a validation dataset is used to prune the DTs to prevent overfitting.

Also, for each feature v_j in $\mathbf{v}$, we calculate the Pearson's correlation between v_j and the output of each class (before softmax) in $FC(\mathbf{v})$ using a training dataset, so we know which class is usually predicted given a high score of v_j (i.e., correlated most to this feature). We use c_j denoting the most correlated class of the feature v_j.

We can consider the DTs as a global explanation of the model as it explains the CNN in general. To create a local explanation, we use the DT of the predicted class to classify the input. At each decision node, we collect associated n-grams passing the nodes' thresholds to be evidence for (or counter-evidence against) the predicted class (depending on the most correlated class of each splitting feature). For example, an input text x is classified to class a, so we use the DT of class a to predict with the input

x. If a decision node checks whether feature v_j of this input is greater than 0.25 and assume it is true for this input, the n-gram corresponding to v_j will be evidence if the most correlated class of v_j is class a (i.e., $c_j = a$). Otherwise, it will be counter-evidence if $c_j \neq a$.

Finally, we show examples of the generated explanations for all the nine explanation methods in Table 4.5 and Appendix B.1.

4.2.4. Human Evaluation Details

Given the three evaluation tasks (as in Section 4.1) and the two datasets (i.e., Amazon and ArXiv), we had $3 \times 2 = 6$ experiments in total. For each experiment, we asked humans to perform the task to collect scores for the nine explanation methods. Then we averaged out the scores and compared the results.

Participants and Procedure. For the Amazon dataset, we recruited participants by using a crowdsourcing platform, namely Amazon Mechanical Turk (MTurk). One unit of work on Amazon MTurk is called a *HIT* (which stands for a *Human Intelligence Task*). Here, we posted on MTurk the three evaluation tasks separately with short descriptions. An MTurk worker could select to do one or more tasks. For each task, s/he could perform one or more HITs each of which amounted to a set of ten questions to be answered. After a worker accepted to perform a HIT, s/he was given a link to our website which provided the full instructions of the task and two sample questions (with answers and reasons for the answers) to familiarize her/himself with the task. Below the instructions and the sample questions, there were a list of ten questions for them to answer. Once the worker answered all the ten questions, s/he was then allowed to click submit to finish the HIT. The worker who wanted to do another HIT could request the new URL from the MTurk website. To ensure the quality of crowdsourcing, we allowed only workers whose history had at least 50 approved HITs and more than 95% HITS approval rate to do the tasks. Moreover, each question we generated was answered by three workers and the scores were averaged.

However, for the ArXiv dataset, we could not use crowdsourcing because the tasks required background knowledge of the related subjects. Therefore, we recruited graduates and post-graduate students in Computer Sci-

An example from the Amazon dataset, Actual: Pos, Predicted: Pos, (Predicted scores: Pos 0.514, Neg 0.486): *"OK but not what I wanted: These would be ok but I didn't realize just how big they are. I wanted something I could actually cook with. They are a full 12" long. The handles didn't fit comfortably in my hand and the silicon tips are hard, not rubbery texture like I'd imagined. The tips open to about 6" between them. Hope this helps someone else know ..."*

Method	**Top-3 evidence texts**
Random (W)	not / wanted / 'd
Random (N)	did n't / be ok / could actually cook
LIME (W)	comfortably / wanted / helps
LRP (W)	are / not / 6
LRP (N)	are hard , not / about 6" between / not what I wanted
DeepLIFT (W)	are / not / 6
DeepLIFT (N)	are hard , not / about 6 " between / not what I wanted
Grad-CAM-T (N)	comfortably in my hand / I wanted : These / . The tips open
DTs (N)	imagined . The tips

Method	**Top-3 counter-evidence texts**
Random (W)	texture / . / what
Random (N)	this helps someone else / , not / wanted something I
LIME (W)	not / else / someone
LRP (W)	: / tips / open
LRP (N)	. The tips open / : These would / in my hand and
DeepLIFT (W)	: / tips / open
DeepLIFT (N)	. The tips open / : These would / in my hand and
Grad-CAM-T (N)	not what I wanted / not rubbery texture like / Hope this helps someone
DTs (N)	'd imagined . / are . I wanted / would be ok

Table 4.5.: Examples of evidence and counter-evidence texts, generated by the tested explanation methods and separated by '/'.

ence, Mathematics, Physics, and Engineering to perform the tasks, both via direct invitations and via the college mailing lists. Because the recruited participants here were likely more reliable than crowdsourced participants, we collected only one answer per generated question. We used the same user interface as in the Amazon tasks, to provide instructions, sample questions, and real questions to the participants. Similarly, one unit of work contained ten questions, and they could do as many tasks or units as they wanted. (The system would randomly show a new set of questions when they wanted to do more.) After completing the tasks, some invited participants reached back to us with some qualitative feedback about the experiments, but this is truly optional.

In total, we have 367 and 121 participants for the Amazon and the ArXiv datasets, respectively.

Prototype and Materials. For each task and dataset, we selected 100 input texts (which satisfy the task condition) from the test split to generate evaluation questions. Half of the selected input texts were classified correctly by the model(s) and the rest were misclassified. So, with nine explanation methods being evaluated, each task had 900 questions per dataset for human participants to answer.

To generate the explanations, we used public libraries of LIME[4], LRP (Alber et al., 2018), and DeepLIFT[5] in our experiments. Besides, the code for computing Grad-CAM-Text was adapted from keras-vis[6], whereas we used scikit-learn (Pedregosa et al., 2011) for decision tree construction[7]. Tables 4.6 and 4.7 report the decision trees' performance in mimicking the CNNs' predictions (i.e., fidelity) on the test sets. All the DTs achieved over 80% macro-F1 in mimicking the CNNs' predictions. As the F1 scores say, it is easier for the DTs to mimic the behavior of the well-trained CNNs than the poor CNNs.

For the task parameters (see Section 4.1), we set $m = 3$, $\tau_h = 0.9$, and $\tau_l = 0.7$. To clarify, for each explanation, we showed top three (counter-) evidence text fragments. When an explanation method returned less than three text fragments in an explanation, we just showed all of them. This could happen when the input text was too short or the corresponding path

[4]`https://github.com/marcotcr/lime`
[5]`https://github.com/kundajelab/deeplift`
[6]`https://github.com/raghakot/keras-vis`
[7]The code and datasets are available at `https://github.com/plkumjorn/CNNAnalysis`.

1st CNN (better)	Prec.	Recall	F1	No. of examples
Negative	0.84	0.84	0.84	48333
Positive	0.85	0.85	0.85	51667
micro avg	0.85	0.85	0.85	100000
macro avg	0.85	0.85	0.85	100000
2nd CNN (worse)	Prec.	Recall	F1	No. of examples
Negative	0.81	0.82	0.82	49482
Positive	0.82	0.82	0.82	50518
micro avg	0.82	0.82	0.82	100000
macro avg	0.82	0.82	0.82	100000

Table 4.6.: Performance of the decision trees in mimicking the CNNs' predictions for the Amazon dataset

1st CNN (better)	Prec.	Recall	F1	No. of examples
Computer science	0.89	0.91	0.90	9971
Mathematics	0.89	0.87	0.88	10203
Physics	0.90	0.91	0.90	9826
micro avg	0.89	0.89	0.89	30000
macro avg	0.89	0.89	0.89	30000
2nd CNN (worse)	Prec.	Recall	F1	No. of examples
Computer science	0.83	0.81	0.82	7653
Mathematics	0.82	0.88	0.85	12506
Physics	0.88	0.81	0.84	9841
micro avg	0.84	0.84	0.84	30000
macro avg	0.84	0.83	0.84	30000

Table 4.7.: Performance of the decision trees in mimicking the CNNs' predictions for the ArXiv dataset

from the DTs method had less than three (counter-)evidence text fragments. Also, as specified by τ_h, and τ_l above, we consider a model prediction to have high confidence (in Task 2) and low confidence (in Task 3) when the predicted probability of the output class is greater than 0.9 and lower than 0.7, respectively.

The website we created to collect human answers was implemented using a Python-based web framework, called Flask[8], and hosted on PythonAnywhere[9], while the user interface was mainly implemented using the Bootstrap 3.3.5 front-end framework[10]and Google Charts[11]. Examples of user

[8]`https://flask.palletsprojects.com/`
[9]`https://www.pythonanywhere.com/`
[10]`https://getbootstrap.com/docs/3.3/`
[11]`https://developers.google.com/chart`

interfaces for questions have been shown in Figures 4.1, 4.2, and 4.3.

Data Collection and Analysis. Every answer from the participants had a score associated to it as summarized in the last row of Table 4.1. A higher score implies a better explanation method with respect to the associated task. Hence, we calculated and compared the average scores the nine explanation methods obtained from all the questions in that task ($\mathcal{A}$). To better understand the tested methods, we further calculated the average scores separately for correctly classified examples (✔) and misclassified examples (✘). Moreover, we checked whether the scores of other methods were significantly lower than the scores of the best method in each setting or not. For the Amazon dataset in particular, we also calculated the inter-rater agreement measures (Fleiss' kappa) (Fleiss, 1971) to show to what extent the three answers for each question agreed with one another. In contrast, we do not have the agreement measures for the ArXiv dataset as one question in this dataset was answered by only one participants. Please note that we will not directly compare the numeric scores across the two datasets as they were from different pools of participants, but we can still observe the rankings of the nine explanation methods within the same dataset and discuss their rankings across datasets to some degree.

4.3. Experimental Results

The results of our experiments for task 1, 2, and 3 are reported in Tables 4.8, 4.9, and 4.10, respectively. Each table shows the average scores of each explanation method obtained from human answers for both datasets (Amazon and ArXiv). The score range is [-1,1] where 1 is the best possible score. $\mathcal{A}$, ✔, and ✘ in the tables indicate whether the average scores were computed based on all the 100 input texts, or only 50 correctly classified input texts, or 50 misclassified input texts, respectively. Boldface numbers are the highest average scores in the columns. A number is underlined when there is no statistically significant difference between the scores of the corresponding method and the best method in the same column (at a significance level of 0.05). In addition, the last row reports inter-rater agreement measures (Fleiss' kappa) (Fleiss, 1971) in the format of α / β where α considers answers with human confidence/certainty levels (5 answer options for task 1-2

Explanation Method	**Task 1**: Revealing the model behavior					
	Amazon			ArXiv		
	$\mathcal{A}$	✔	✘	$\mathcal{A}$	✔	✘
Random (W)	0.02	0.00	0.04	-0.11	-0.05	-0.17
Random (N)	0.02	0.02	0.02	-0.12	-0.16	-0.07
LIME (W)	-0.02	0.02	-0.06	0.03	0.02	**0.03**
LRP (W)	0.00	-0.01	0.02	-0.03	-0.01	-0.05
LRP (N)	-0.07	-0.04	-0.09	**0.12**	**0.24**	-0.01
DeepLIFT (W)	0.04	0.03	0.04	0.07	0.13	0.00
DeepLIFT (N)	0.06	0.06	**0.05**	0.06	0.22	-0.10
Grad-CAM-Text (N)	**0.07**	**0.11**	0.03	-0.03	-0.04	-0.01
DTs (N)	-0.05	-0.02	-0.08	-0.13	-0.22	-0.03
Fleiss κ (Amazon)	0.050 / 0.054			N/A		

Table 4.8.: The average scores and the inter-annotator agreement measure for Task 1 – revealing the model behavior. $\mathcal{A}$, ✔, and ✘ refer to all, correctly classified, and misclassified input texts, respectively.

and 4 answer options for task 3) and β considers answers regardless of the human confidence/certainty levels (5 answer options reduced to 3 categories for task 1-2 and 4 answer options reduced to 2 categories for task 3). Furthermore, Figure 4.5 displays the distributions of individual scores (from all the input examples) for all three tasks. We show the score distribution plots for correctly classified examples only and misclassified examples only in Appendix B.2. To help us notice methods which behaved differently from the others more easily, we decided to use line graphs for these plots.

4.3.1. Results for Task 1: Revealing the Model Behavior

For the Amazon dataset, though Grad-CAM-Text achieved the highest overall score, the performance was not significantly different from other methods, including the random explanations. Also, the inter-rater agreement for this task was quite poor. It suggests that existing explanation methods cannot apparently reveal irrational behavior of the underfitting CNN to lay human users. Hence, as Figure 4.5(a) shows, the scores of most explanation methods distribute symmetrically around zero. Moreover, only 20-30% of the answers have high confidence (i.e., selecting the "clearly more reasonable" options), whereas the rest have low confidence (i.e., selecting the "slightly more reasonable" options) or no preference (i.e., the "I can't say" option).

For the ArXiv dataset, LRP (N) and DeepLIFT (N) got the highest scores

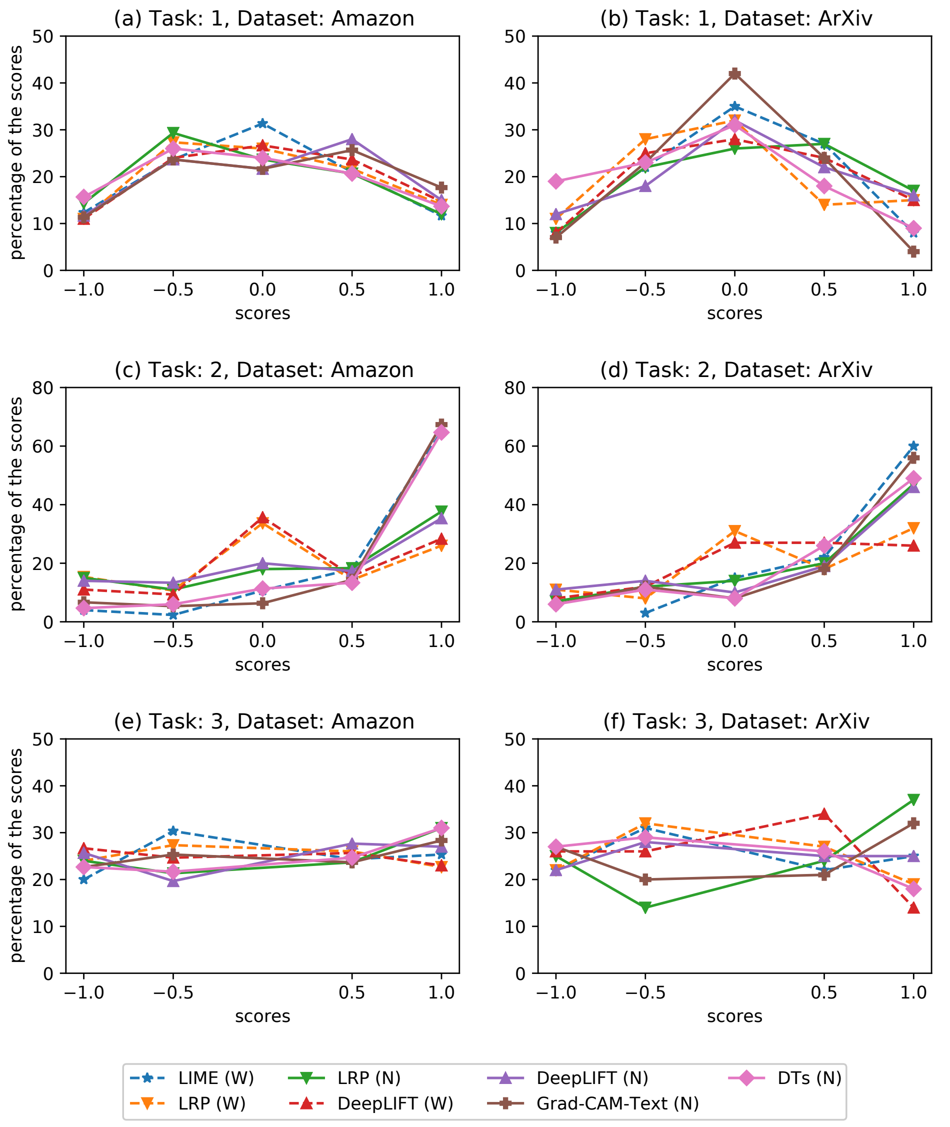


Figure 4.5.: Score distributions of the Amazon dataset and the ArXiv dataset for all the three evaluation tasks.

when both CNNs predicted correctly. So, they can help humans identify the poor model to some extent. However, there was no clear winner when both CNNs predicted wrongly. One plausible reason is that evidence for an incorrect class is usually not convincing when we do not set any condition on the confidence of the predictions (unlike task 2 in which all predictions have high confidence only).

Additionally, based on our manual observation of the human answers, when there were two explanations with comparable semantic quality (i.e., covering the same sets of concepts), humans usually selected the explanation with more evidence text fragments as more reasonable. This is consistent with the findings by Zemla et al. (2017). Conversely, the DTs method performed in the opposite way. The DT of the better model usually focuses on a few most relevant texts in the input and outputs fewer evidence texts. This possibly causes the low performance of the DTs method in this task. Also, we got qualitative feedback from the participants that they sometimes penalized an evidence text which is highlighted without syntax integrity, such as "... greedy algorithm. In this paper, we ...". Hence, in real applications, the syntax integrity issue should be taken into account to generate explanations.

Overall, this task is challenging for all the tested explanation methods. Seeing that the ideal score is 1 and the best average scores overall were only 0.07 (Amazon) and 0.12 (ArXiv), there is still large room for improvement in this task. On the other hand, we may reduce the task difficulty by increasing the Δ parameter so that the difference between the two models are clearer and more noticeable to humans via the explanations. Hopefully, this will better differentiate between effective and ineffective explanation methods. Another way that may help adjust the task difficulty is to set some thresholds for selecting input texts to be questions as we did in Tasks 2 and 3. Besides, in this experiment, we trained the poor models by stopping the training too early (as in the Amazon task) and training on a too specific dataset (as in the ArXiv task). Other ways to generate the poor models could be explored such as training on a noisy dataset with many artifacts (Ribeiro et al., 2016). We leave these ideas for further improving the evaluation task to be studied in future work.

4.3.2. Results for Task 2: Justifying the Predictions

Table 4.9 shows that LIME clearly achieved the best results in task 2 followed by Grad-CAM-Text and DTs. These methods are class discriminative, being able to find good evidence for the predicted class regardless of whether the prediction is correct.

We believe that LIME performed well because it tests that the absence

Explanation Method	**Task 2**: Justifying the predictions					
	Amazon			ArXiv		
	$\mathcal{A}$	✔	✘	$\mathcal{A}$	✔	✘
Random (W)	0.06	0.10	0.02	0.07	0.09	0.04
Random (N)	0.12	0.13	0.12	0.29	0.32	0.25
LIME (W)	**0.69**	**0.74**	0.64	**0.70**	**0.75**	**0.64**
LRP (W)	0.13	0.26	-0.01	0.26	0.36	0.16
LRP (N)	0.26	0.45	0.08	0.44	0.49	0.39
DeepLIFT (W)	0.21	0.37	0.04	0.26	0.35	0.16
DeepLIFT (N)	0.23	0.47	-0.01	0.38	0.47	0.28
Grad-CAM-Text (N)	0.65	0.64	**0.66**	0.53	0.65	0.41
DTs (N)	0.64	0.68	0.59	0.51	0.69	0.32
Fleiss κ (Amazon)	0.274 / 0.371			N/A		

Table 4.9.: The average scores and the inter-annotator agreement measure for Task 2 – justifying the predictions. $\mathcal{A}$, ✔, and ✘ refer to all, correctly classified, and misclassified input texts, respectively.

of evidence words from the input text greatly reduces the probability of the predicted class, so these words are semantically related to the predicted class (given that the model is accurate). Meanwhile, the DTs method selects evidence based on the most correlated class of the splitting features. So, the evidence n-grams are more likely related to the predicted class than the other classes. However, they may be less relevant than LIME's as the evidence is generated from a global explanation of the model (DTs). Besides, Grad-CAM-Text worked relatively well here probably because it preserves the class discriminative property of Grad-CAM (Selvaraju et al., 2017). However, it is important to note that these three methods (LIME, DTs, and Grad-CAM-Text) can justify even the incorrect predictions. On one hand, this helps the model looks rational as the explanations supports the predictions. On the other hand, this could be dangerous as the explanations might not alarm humans that there may be something wrong with its predictions.

By contrast, LRP and DeepLIFT generated acceptable evidence only for the correct predictions with their scores under the ✔ columns not as high as LIME, DTs, and Grad-CAM-Text. Meanwhile, when it comes to misclassifications (✘), the evidence of LRP and DeepLIFT could not really justify their predictions. This undermined their overall scores in this task, yet it could be useful in other situations (e.g., in Task 3, discussed next). Also, we can notice that the n-gram version of an explanation method is signifi-

Explanation Method	**Task 3**: Investigating uncertain predictions					
	Amazon			ArXiv		
	$\mathcal{A}$	✔	✘	$\mathcal{A}$	✔	✘
Random (W)	0.05	0.53	-0.43	0.01	0.32	-0.30
Random (N)	-0.01	0.54	-0.55	0.02	0.29	**-0.25**
LIME (W)	0.02	0.50	-0.45	-0.02	0.31	-0.34
LRP (W)	-0.02	0.50	-0.54	-0.06	0.33	-0.44
LRP (N)	0.08	0.60	-0.43	**0.17**	**0.60**	-0.26
DeepLIFT (W)	-0.03	0.47	-0.53	-0.08	0.28	-0.44
DeepLIFT (N)	0.05	0.59	-0.49	0.02	0.33	-0.30
Grad-CAM-Text (N)	0.05	0.51	-0.42	0.06	0.56	-0.45
DTs (N)	**0.10**	**0.60**	**-0.40**	-0.11	0.29	-0.50
Fleiss κ (Amazon)	0.212 / 0.499			N/A		

Table 4.10.: The average scores and the inter-annotator agreement measure for Task 3 – investigating uncertain predictions. $\mathcal{A}$, ✔, and ✘ refer to all, correctly classified, and misclassified input texts, respectively.

cantly better than the word version of the same method. For instance, LRP (N) and DeepLIFT (N) performed better than LRP (W) and DeepLIFT (W) in both datasets. This might be because one evidence n-gram contains more information than one evidence word. Nevertheless, even the Random (N) method surpasses the LRP (W) and the DeepLIFT (W) for the ArXiv dataset. Thereby, whenever we use LRP and DeepLIFT (as well as other explanation methods), we should present to humans the most relevant words together with their contexts in order to well justify the model predictions.

4.3.3. Results for Task 3: Investigating Uncertain Predictions

The goal of this task is to provide important information for the humans to correctly identify the correct class of the input texts. Particularly, when the predictions from the model are incorrect (✘), the counter-evidence explanation should be able to alarm the humans and make them distrust the incorrect predictions. However, according to Table 4.10, the negative scores under the ✘ columns show that using explanations to help humans rectify the predictions is not easy. Many humans still trusted the model predictions even when they were incorrect. Hence, the overall average scores of many explanation methods stay close to zero.

DTs performed well on the Amazon dataset but not the ArXiv dataset.

For DTs, the average numbers of n-grams per explanation are 2.00 and 1.77 for the Amazon and ArXiv datasets, respectively. Also, the reported n-grams could be repetitive and overlapping because many features in the decision nodes might pick the same or overlapping n-grams during the max-pooling step. For example, a PH input text was classified as a CS text, and the DT showed ⟨"data and theoretical models", "experimental data", "and theoretical models"⟩ as evidence (ζ_3^+) for CS but showed nothing as the counter-evidence (i.e., $\zeta_3^- = \langle\rangle$). We can see that the text fragments in ζ_3^+ are highly overlapping with each other, reducing the amount of useful information displayed. Moreover, the decision path of this particular input led to no counter-evidence being reported. The insufficient information by DTs like this could make it difficult for humans to choose one of the CS, MA, and PH categories, which are more similar to one another than the positive and negative sentiments.

Meanwhile, LRP (N) performed consistently well on both datasets. This is reasonable considering our discussions in task 2. To explain, on one hand, LRP (N) generates good evidence for correct predictions, so it can gain high scores in the ✔ columns. On the other hand, the evidence for incorrect predictions (✘) is usually not convincing, so the counter-evidence (which is likely to be the evidence of the correct class) can attract humans' attention. Furthermore, the fact that LRP is not always class discriminative does not harm it in this task as humans can recognize an evidence text even if it is selected by the LRP (N) as counter-evidence (and vice versa). For instance, in the ArXiv dataset, we found a case in which the predicted scores are 0.07, 0.45, and 0.48 for the CS, MA, and PH categories, respectively, but the actual class is CS. LRP (N) selected the following n-grams as evidence for the class PH: 'armed bandit settings with', 'the Wasserstein distance', and 'derive policy gradients'. However, these n-grams are not truly related to the predicted class (PH). Rather, they revealed the true class of this text (CS) and made a human choose the CS option with high confidence despite the disputing predicted scores. Overall, these properties make LRP (N) suitable for helping humans investigate uncertain predictions.

Regarding LIME, the situation is reversed as LIME effectively find both good evidence and counter-evidence. These could make humans be indecisive. When the predictions were already correct, the counter-evidence looked rather convincing. When the predictions were incorrect, the evidence could also sound reasonable. Furthermore, LIME explanation was presented at a word level (without any contexts). These factors did not help humans select the right option in this task.

4.3.4. Model Complexity

Apart from the results of the three tasks, it is worth to discuss the size of the DTs which mimic the four CNNs in our experiments. As shown in Table 4.11, the size of the DTs can reflect the complexity of the CNNs. Although the well-trained CNN of the Amazon dataset got 0.9 F1 score, the DTs of this CNN needed more than 5,500 nodes to achieve 85% fidelity (compared to only hundreds of nodes required for the ArXiv dataset). This illustrates the high complexity of the Amazon task compared to the ArXiv task even though both tasks can be managed effectively by the same CNN architecture.

For the ArXiv dataset, the DTs of the poor CNN are smaller than the ones of the well-trained CNN. This is likely because the poor CNN was trained on a specific fragment of the dataset (i.e., selected subtopics of the main categories), so it had to deal with fewer discriminative patterns in texts compared to the first CNN trained using texts from all subtopics.

This quality of the DTs method can be potentially useful for measuring model complexity (Bianchini and Scarselli, 2014) and for model compression (Cheng et al., 2018). These are worth studying in the future although they are out of the scope of this thesis.

4.4. Summary

We proposed three human tasks to evaluate local explanation methods for text classification. Each of the tasks targets a specific downstream application of explanations including revealing the model behavior, justifying the predictions, and investigating uncertain predictions. Using these tasks, we conducted experiments on 1D CNNs and nine forms of explanations, the results of which show that different explanation methods are suitable for different human tasks. Specifically, we found that *(i)* LIME is the most class discriminative method, justifying predictions with relevant evidence; *(ii)* LRP (N) works fairly well in helping humans investigate uncertain pre-

	#Nodes	**Depth**	**#Leaves**
Amazon: 1st CNN (well-trained) – $F = 0.85$			
Negative	5535	38	2768
Positive	5537	45	2769
Amazon: 2nd CNN (underfitting) – $F = 0.82$			
Negative	6405	40	3203
Positive	6369	40	3185
ArXiv: 1st CNN (well-trained) – $F = 0.89$			
Computer Science	363	25	182
Mathematics	565	24	283
Physics	325	24	163
ArXiv: 2nd CNN (specific data) – $F = 0.84$			
Computer Science	107	17	54
Mathematics	263	28	132
Physics	237	29	119

Table 4.11.: Metadata of the DTs in the experiments. F refers to fidelity of the DTs (macro-average F1).

dictions; *(iii)* whenever using LRP and DeepLIFT, we should present to humans the most relevant words together with their contexts; *(iv)* the size of the DTs can also reflect the model complexity and *(v)* using explanations to reveal model behavior is challenging and needs more research (i.e., improving the explanation methods or adjusting the difficulty of the evaluation task). Lastly, we consider evaluating on other datasets and other advanced architectures beneficial future work as it may reveal further interesting qualities of the explanation methods.

All the three purposes of explanation usage in this chapter finish at the user side (e.g., the user choosing the better model or making a decision). Even though we saw some flaws in the model via the explanations, we have not yet improved the model accordingly. In the next chapter, we will focus on another application of explanations which is model debugging, where we provide explanations to humans and leverage on their feedback to improve the underlying model. Basically, we will use LRP to help us understand the patterns detected by each learned feature, enabling human investigation and debugging. We choose LRP for this purpose, not other explanation methods, because LRP can be easily adapted to explain individual neurons in the model apart from explaining the model output. Taking the result *(iii)* above to the next chapter, although we will not explain the predictions directly, we will show the explanations from LRP as word clouds of n-grams to provide more contexts and information about the learned features to human debuggers.

5. Human-in-the-Loop Debugging Deep Text Classifiers

As discussed in Section 2.1.3, supervised text classification can suffer from imperfect training data and, therefore, produce suboptimal models. Often times, we are not aware of the problems in the training dataset until the explanations reveal them to us. For instance, attribution scores may show that the model does not use the right reasons for its predictions, and this undermines the model generalizability. Some existing works let humans debug the suboptimal models by inspecting explanations for their predictions and providing feedback in response, as explained in Section 3.3 under the topic *Explanation-based human debugging (EBHD)*. However, these existing works mainly use local explanations and iterative improvement, preventing the humans from seeing the big picture of the model when giving feedback (Ribeiro et al., 2016; Teso and Kersting, 2019; Wu et al., 2019b). Furthermore, there is still no work using global explanations to perform EBHD for deep NLP models. To narrow this gap, in this chapter, we propose a framework which allows humans to debug simple deep text classifiers (with limited number of features) by investigating hidden features learned by the model and disabling the features that could undermine the prediction accuracy during testing. We name this framework **F**eature **I**nvestigation a**N**d **D**isabling (**FIND**). As a part of the framework, we also propose a new form of global explanation based on neuron analysis and word clouds to help humans comprehend behaviors of the learned features and give accurate feedback to improve the model. The main differences between our work and existing work are: (i) first, FIND leverages human feedback on the model components, not the individual predictions, to perform debugging; (ii) second, FIND targets deep text classifiers which are more convoluted than traditional classifiers used in existing work (such as Naive Bayes classifiers and Support Vector Machines).

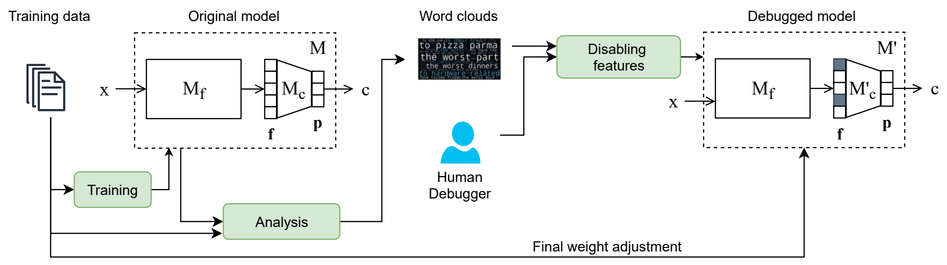


Figure 5.1.: Overview of the proposed debugging framework, FIND.

The rest of this chapter is organized as follows. Section 5.1 presents our debugging framework, FIND. To demonstrate its effectiveness, we conduct three human experiments with CNN text classifiers where the general setup is explained in Section 5.2. The first experiment in Section 5.3 is a feasibility study confirming that, with our proposed word clouds as explanations, humans can accurately assess the quality of learned features in a model. Then the second and the third experiments (Sections 5.4-5.5) show that the overall framework, by exploiting human feedback, can help mitigate gender biases and deal with the dataset shift problem, respectively. After that, Section 5.6 provides general discussions about the proposed framework as well as its limitations. Finally, we summarize key messages of this chapter in Section 5.7.

5.1. FIND: Feature Investigation and Disabling

Generally, deep text classifiers can be divided into two parts. The first part performs *feature extraction*, transforming an input text into a dense vector (i.e., a *feature vector*) which represents the input. There are several alternatives to implement this part such as using convolutional layers, recurrent layers, and transformer layers. The second part performs *classification* passing the feature vector through a dense layer with softmax activation to get predicted probability of the classes. These deep classifiers are not transparent, as humans cannot interpret the meaning of either the intermediate vectors or the model parameters used for feature extraction. This prevents humans from applying their knowledge to modify or debug the classifiers.

In contrast, if we understand which patterns or qualities of the input are captured in each feature, we can comprehend the overall reasoning mechanism of the model as the dense layer in the classification part then becomes interpretable. In this chapter, we make this possible using layer-wise relevance propagation (LRP)[1] which can be adapted to find parts of the input which are important for individual features in the model. By understanding the model, humans can check whether the input patterns detected by each feature are relevant for classification. Also, the features should be used by the subsequent dense layer to support the right classes. If these are not the case, debugging can be done by disabling the features which may be harmful if they exist in the model. Figure 5.1 shows the overview of our proposed debugging framework, FIND.

5.1.1. Model and Training

As a recap from Section 2.1.1, let us consider a text classification task with $|\mathcal{C}|$ classes where $\mathcal{C}$ is the set of all classes and let $\mathcal{V}$ be a set of unique words in the corpus (the vocabulary). A training dataset $\mathcal{D} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ is given, where x_i is the i-th document containing a sequence of L words[2], $[x_{i1}, x_{i2}, ..., x_{iL}]$, and $y_i \in \mathcal{C}$ is the class label of x_i. A deep text classifier M trained on dataset $\mathcal{D}$ classifies a new input document x into one of the classes (i.e., $M(x) \in \mathcal{C}$). In addition, M can be divided into two parts – a feature extraction part M_f and a classification part M_c. Formally, $M(x) = (M_c \circ M_f)(x)$; $M_f(x) = \mathbf{f}$; $M(x) = M_c(\mathbf{f}) = \text{softmax}(\mathbf{Wf} + \mathbf{b}) = \mathbf{p}$ where $\mathbf{f} = [f_1, f_2, \dots, f_d] \in \mathbb{R}^d$ is the feature vector of x, while $\mathbf{W} \in \mathbb{R}^{|\mathcal{C}| \times d}$ and $\mathbf{b} \in \mathbb{R}^{|\mathcal{C}|}$ are parameters of the dense layer of M_c. The final output is the predicted probability vector $\mathbf{p} \in [0,1]^{|\mathcal{C}|}$, which in turn implies the predicted class $c = \text{argmax}_c \mathbf{p}_c$.

5.1.2. Understanding the Model

To understand how the model M works, we analyze the patterns or characteristics of the input that activate each feature f_i. Specifically, using LRP, for each f_i of an example x_j in the training dataset, we calculate a relevance vector $\mathbf{r}_{ij} \in \mathbb{R}^L$ showing the relevance scores (the contributions) of each word in x_j for the value of f_i. This is different from how we used LRP in the

[1]See Section 2.2.3 for more details on how LRP works.

[2]For x_i that have more than L or less than L words, we trim or pad it to have exactly L words, respectively.

previous chapter where we explained *individual predictions* of a CNN. To recap, in the previous chapter, we propagated an activation value of the target output node back to the word embedding matrix. After that, the relevance score of an input word equals the sum of relevance scores each dimension of its word vector received. By contrast, in this chapter, we want to analyze the hidden features rather than the output, so we start back propagating from the hidden features instead to capture patterns of input words which highly activate the features. After doing this for all d features of all training examples, we can produce word clouds to help the users better understand the model M.

Word clouds – For each feature f_i, we create (one or more) word clouds to visualize the patterns in the input texts which highly activate f_i. This can be done by analyzing $\mathbf{r}_{ij}$ for all x_j in the training data and displaying, in the word clouds, words or n-grams which get high relevance scores. Note that different model architectures may have different ways to generate the word clouds so as to effectively reveal the behavior of the features.

For CNNs, the classifiers we experiment with in this chapter, each feature has one word cloud containing the n-grams, from the training examples, which were selected by the max-pooling of the CNNs. For instance, Figure 5.2, corresponding to a feature of filter size 2, shows bi-grams (e.g., "love love", "love my", "loves his", etc.) whose font size corresponds to the feature values of the bi-grams. This is similar to how previous works analyze CNN features (Jacovi et al., 2018), and it is equivalent to back-propagating the feature values to the input using LRP and cropping the consecutive input words with non-zero LRP scores to show in the word clouds. Hence, it follows what we learned from Chapter 4 that presenting n-grams as explanations would provide more contexts and information to the users.

5.1.3. Disabling Features

As explained earlier, we want to know whether the learned features are valid and relevant to the classification task and whether or not they get appropriate weights from the next layer. This is possible by letting humans consider the word cloud(s) of each feature and tell us which class the feature is relevant to. A word cloud receiving human answers that are different from the class it should support (as indicated by $\mathbf{W}$) exhibits a flaw in the model.

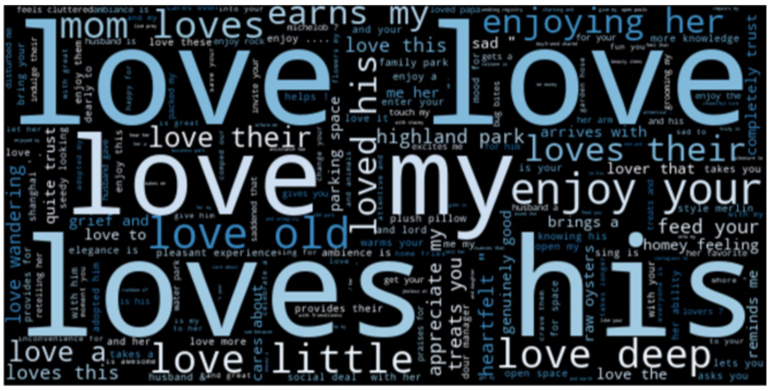


Figure 5.2.: A word cloud (or, literally, an n-gram cloud) of a feature from a CNN.

For example, if the word cloud in Figure 5.2 represents the feature f_i in a sentiment analysis task but the i^{th} column of $\mathbf{W}$ implies that f_i supports the negative sentiment class, we know the model is not correct here. If this word cloud appears in a product categorization task, this is also problematic because the phrases in the word cloud are not discriminative of any product category. Hence, we provide options for the users to disable the features which correspond to any problematic word clouds so that the features do not play a role in the classification. To enable this to happen, we modify M_c to be M_c' where $\mathbf{p} = M_c'(\mathbf{f}) = \text{softmax}((\mathbf{W} \odot \mathbf{Q})\mathbf{f} + \mathbf{b})$ and $\mathbf{Q} \in \mathbb{R}^{|\mathcal{C}| \times d}$ is a masking matrix with $\odot$ being element-wise multiplication. Initially, all elements in $\mathbf{Q}$ are ones which enable all the connections between the features and the output. To disable feature f_i, we set the i^{th} column of $\mathbf{Q}$ to be a zero vector. After disabling features, we then freeze the parameters of M_f and fine-tune the parameters of M_c' (except the masking matrix $\mathbf{Q}$) with the original training dataset $\mathcal{D}$ in the final step. Unlike other existing works, debugging using FIND is a one-off improvement. We do not update the model iteratively because the explanations (i.e., the word clouds) do not change after the model update due to the frozen M_f. As a result, no new feedback is needed for further improving the model.

It is important for the FIND framework that *all* the problematic features in $\mathbf{f}$ must be disabled. It is inefficient to disable only problematic features which contribute largely to the output (e.g., features with the high corresponding weights in $\mathbf{W}$) because the model can switch to extensively exploit

Exp	Dataset	$\lvert\mathcal{C}\rvert$	Train / Dev / Test
1	Yelp	2	500 / 100 / 38000
	Amazon Products	4	100 / 100 / 20000
2	Biosbias	2	3832 / 1277 / 1278
	Waseem	2	10144 / 3381 / 3382
	Wikitoxic	2	- / - / 18965
3	20Newsgroups	2	863 / 216 / 717
	Religion	2	- / - / 1819
	Amazon Clothes	2	3000 / 300 / 10000
	Amazon Music	2	- / - / 8302
	Amazon Mixed	2	- / - / 100000

Table 5.1.: Datasets used in the experiments.

the remaining problematic features after retraining, leading to insignificant model improvement. Therefore, FIND is not suitable for complex deep learning models where the number of features in $\mathbf{f}$ (d) is too large. A model with 500 features requires the investigation of 500 word clouds, and this may be too expensive and impractical. We will discuss this issue in detail in Section 5.6.

5.2. General Setup for the Experiments

The next three sections demonstrates how effective FIND was in three bug scenarios including small training sets, biased training sets, and dataset shift. All datasets and their splits used in the experiments are listed in Table 5.1. We will explain each of them in the following sections. For this section, we explain the general setup which was shared among the three experiments. First, for each classification task, we ran and improved three models, using different random seeds, independently of one another, and the reported results are the average of the three runs.

Regarding the models, we used 1D CNNs with the same structures for all the tasks and datasets. The convolution layer of the CNNs had three filter sizes [2, 3, 4] with 10 filters for each size (i.e., $d = 10 \times 3 = 30$). All the activation functions were ReLU except the softmax at the output layer. The input documents were padded or trimmed to have 150 words ($L = 150$). We used pre-trained 300-dim GloVe vectors (Pennington et al., 2014) as non-trainable weights in the embedding layers. The total number of parameters for binary classification tasks, therefore, equalled 120,000,600 (for the fixed word embeddings) + 27,030 (for the convolutional layers) + 62 (for the final dense layer) + 60 (for the masked matrix $\mathbf{Q}$). This also applied to a 4-class classification task in experiment 1 except that the last two numbers of the 4-class task were + 124 (for the final dense layer) + 120 (for the masked matrix $\mathbf{Q}$). All the models were implemented using Keras and trained with Adam optimizer. We used iNNvestigate (Alber et al., 2018) to run LRP on CNN features. In particular, we used the LRP-ϵ propagation rule to stabilize the relevance scores ($\epsilon = 10^{-7}$). All the word clouds generated in these experiments can be found at `https://plkumjorn.github.io/FIND/`. In terms of the computing infrastructure, our machine (used for training, explaining, and re-training the models) had an Intel Core i9-9900X (3.5GHz) as its CPU, an 11GB NVIDIA GeForce RTX 2080 Ti as a GPU, and its 32GB Corsair Vengeance DDR4 as its RAM.

Finally, we used Amazon Mechanical Turk (MTurk) to collect crowd-sourced responses for selecting features to disable. Each question was answered by ten workers and the answers were aggregated using majority votes or average scores depending on the question type (as explained next).

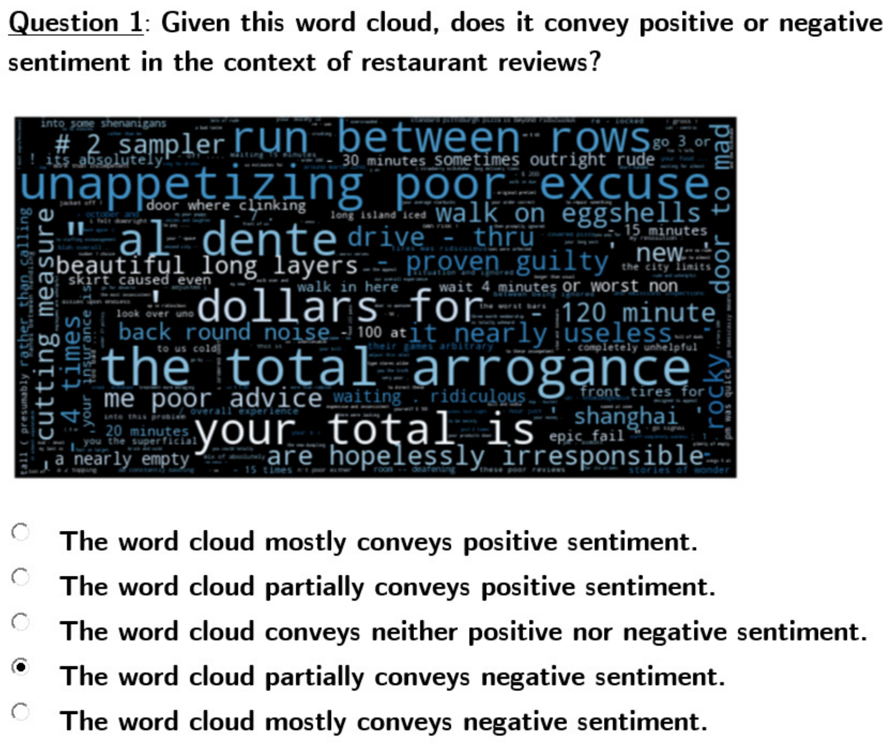
5.3. Experiment 1: Feasibility Study

In this feasibility study, we assessed the effectiveness of word clouds as visual explanations to reveal the behavior of CNN features. Basically, if the quality of word clouds as seen by human participants correlates positively with the drop in the model predictive performance when the corresponding features of the word clouds are removed, it meant that humans could understand and accurately assess CNN features using word clouds.

5.3.1. Debugging Context and Materials

We used two datasets in this experiment. First, the **Yelp** dataset aims to predict sentiments of restaurant reviews (positive or negative). We sampled examples from the dataset provided by Zhang et al. (2015) here[3]. Second, the **Amazon Products** dataset aims to classify product reviews into one of four categories (Clothing Shoes and Jewelry, Digital Music, Office Products,

[3]`https://github.com/zhangxiangxiao/Crepe`



Question 1: **Given this word cloud, does it convey positive or negative sentiment in the context of restaurant reviews?**

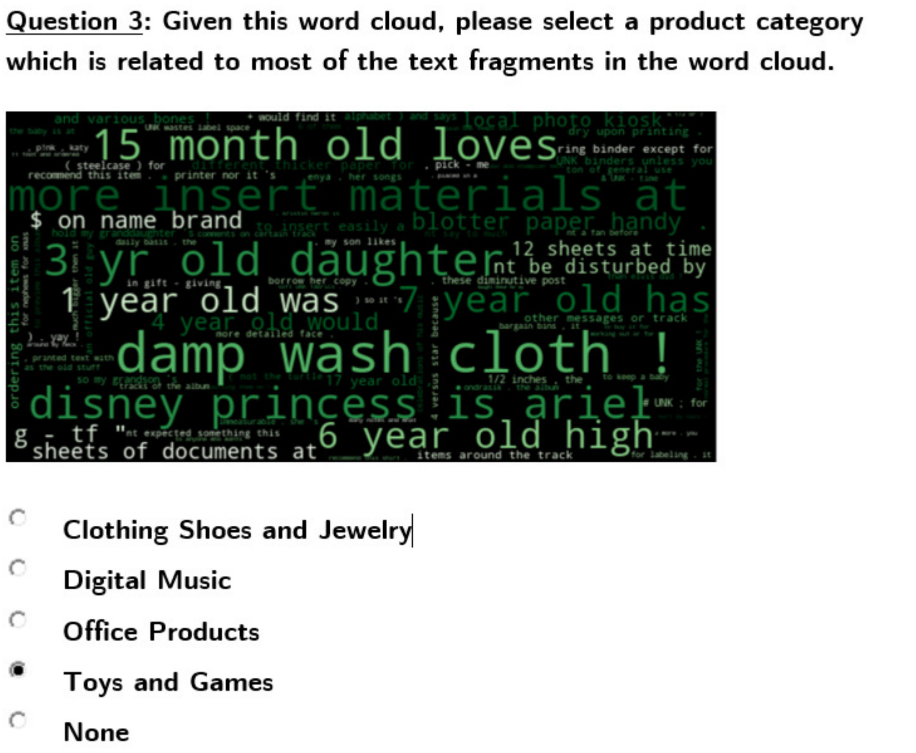
- The word cloud mostly conveys positive sentiment.
- The word cloud partially conveys positive sentiment.
- The word cloud conveys neither positive nor negative sentiment.
- [✓ The word cloud partially conveys negative sentiment.]
- The word cloud mostly conveys negative sentiment.

Figure 5.3.: Example of user interface in Experiment 1 (Yelp).

or Toys and Games). We sampled examples from the datasets provided by He and McAuley (2016) here[4].

We then trained three CNN models for each dataset. To make the models suitable for debugging, we used only a small number of training examples so that the trained CNNs had features with different levels of quality. Some features detected useful patterns, while others overfitted the training data. Specifically, we sampled 500 and 100 examples to be the training data for Yelp and Amazon Products, respectively. After that, using our method described in Section 5.1.2, we generated word clouds for every feature f_i of every model. Doing the math, we had 2 datasets $\times$ 3 models / dataset $\times$ 30 features (i.e., word clouds) / model = 180 word clouds to be assessed by human participants. For each word cloud, we needed answers from ten participants, so we generated $180 \times 10 = 1800$ questions altogether in this experiment.

[4]`http://jmcauley.ucsd.edu/data/amazon/`



Question 3: Given this word cloud, please select a product category which is related to most of the text fragments in the word cloud.

- Clothing Shoes and Jewelry
- Digital Music
- Office Products
- [✓ Toys and Games]
- None

Figure 5.4.: Example of user interface in Experiment 1 (Amazon Products).

Since we wanted to check whether each word cloud should be used to support the class selected by **W**, we asked the participants which class corresponded to the word cloud. In particular, for the Yelp dataset (which is a binary sentiment analysis task), each question asked whether the word cloud (mostly or partially) conveyed positive sentiment, negative sentiment, or neither, as shown in Figure 5.3. For the Amazon Products dataset (which is a multiclass classification task), we used a slightly different user interface as shown in Figure 5.4. We asked the participants to select a class which was most related to text fragments in the word cloud. However, we did not provide the options for mostly and partly relatedness; otherwise, there would have been nine options per question which are difficult for the participants to answer especially when the word cloud is related to more than one class but not all the classes.

5.3.2. Participants and Procedure

We recruited human participants via Amazon Mechanical Turk (MTurk). As with Chapter 4, we posted on MTurk the tasks for two datasets separately with short descriptions. An MTurk worker could select to do one or more tasks. For each task, s/he could perform one or more HITs each of which amounted to a set of seven questions to be answered in the MTurk system.[5] After a worker accepted to perform a HIT, s/he would saw a set of instructions followed by three sample questions (with answers and reasons for the answers) to familiarize her/himself with the task. Below the instructions and the sample questions, we provided a list of seven questions to be answered. At the end of the page, there was a feedback text box where s/he could (optionally) leave any feedback or comment. The submit button became clickable only after all the seven questions were answered. After clicking submit to finish the HIT, the worker who wanted to do another HIT could accept the new HIT immediately using the interface of MTurk. To ensure the quality of crowdsourcing, we allowed only workers whose history had at least 50 approved HITs and more than 97% HITS approval rate to do the tasks. Additionally, we required that the location of the workers must be one of the following English-speaking countries: the United States, the United Kingdom, Australia, and New Zealand to make certain that they were proficient in English.

In total, we had 99 and 77 MTurk participants for the Yelp and the Amazon Products datasets, respectively.

5.3.3. Data Collection and Analysis

We aimed to assign a score to gauge the quality of each feature based on the answers its corresponding word cloud received from the MTurk participants. After that, as each classifier had 30 original features ($d = 30$), we divided them into three ranks (A, B, and C) each of which with 10 features. We expected that features in rank A were most relevant and useful for the prediction task, features in rank C were the least relevant, potentially undermining the performance of the model, and features in rank B

[5]Unlike Chapter 4, we did not create a separate website to collect the participants' answers here because, in this chapter, all the participants were recruited via MTurk whereas, in Chapter 4, we needed to collect answers from both crowdsourced participants and invited participants.

were something between rank A and C.

For the Yelp dataset, if the answer matches how the model really uses the feature (as indicated by $\mathbf{W}$), the feature gets a positive score from this human response. To illustrate, if the CNN feature of the word cloud in Figure 5.3 is used by the model for the negative sentiment class, the scores of the five options in the figure are -2, -1, 0, 1, 2, respectively. We collected ten responses for each word cloud and used the average score to represent the quality of the feature. Then we sorted all the 30 features in each CNN descendingly. After sorting, the 1^{st}-10^{th} features, 11^{th}-20^{th} features, and 21^{st}-30^{th} features are considered as rank A, B, and C, respectively.

For the Amazon Products dataset with the user interface in Figure 5.4, we gave a score to the feature f_i based on the participant answer. To explain, we re-scaled values in the i^{th} column of $\mathbf{W}$ to be in the range [0,1] using min-max normalization and gave the normalized value of the chosen class as a score to the feature f_i. If the participant selects None, this feature gets a zero score. After computing the average score (from ten responses) of each feature, we divided all the 30 features into rank A, B, and C by sorting and splitting in the same way as we did for the Yelp dataset.

To show the effects of feature disabling, we compared the original model M with the modified model M' with features in rank X disabled where X $\in$ {A, B, C, A and B, A and C, B and C}. Specifically, we observed the macro-F1 of the original model and each modified model (averaged from all the three random seeds).

5.3.4. Results and Discussions

Figure 5.5 shows the distribution of average feature scores from one of the three CNN instances for the Yelp dataset. Examples of the word clouds from each rank are displayed in Figure 5.6. We can clearly see dissimilar qualities of the three features. The rank A feature in Figure 5.6 clearly captures positive adjectives and got the perfect score from the participants. Some participants answered that the rank B feature in Figure 5.6 was relevant to the positive class (probably due to the words 'delicious' and 'yum'), and the weights of this feature in $\mathbf{W}$ agreed (Positive:Negative = 0.137:-0.135). Interestingly, the rank C feature in Figure 5.6 got a negative score because some participants believed that this word cloud was relevant to the positive

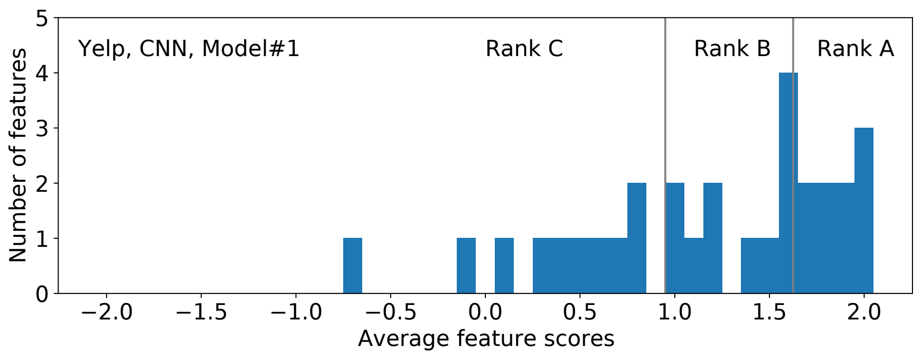


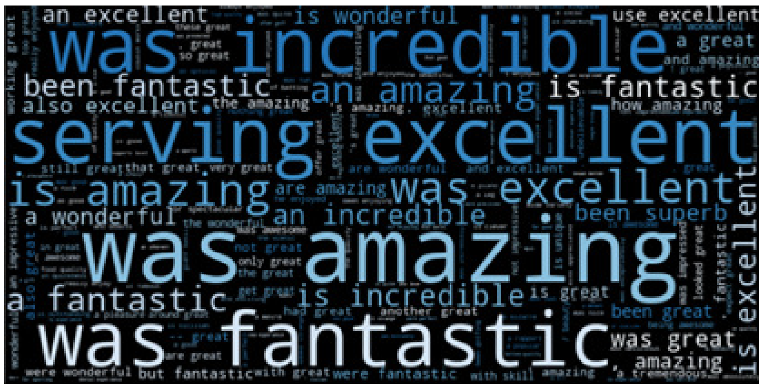
Figure 5.5.: The distribution of average feature scores in a CNN model trained on the Yelp dataset.

class, but actually the model used this feature as evidence for the negative class (Positive:Negative = 0.209:0.385).

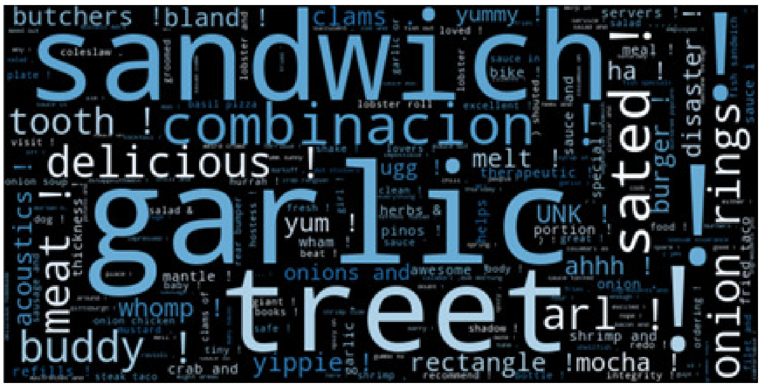
Considering all the three runs, Figure 5.7 (top) shows the average macro F1 score of the original model (the blue line) and of each modified model. The order of the performance drops is AB > A > AC > BC > B > Original > C. This makes sense because disabling important features (rank A and/or B) caused larger performance drops, and the overall results are consistent with the average feature scores given by the participants (as in Figure 5.5). It confirms that using word clouds is an effective way to assess CNN features. Also, it is worth noting that the macro F1 of the model slightly increased when we disabled the low-quality features (rank C). This shows that humans can improve the model by disabling irrelevant features.

For the Amazon Products dataset, Figure 5.8 shows the distribution of average feature scores from one of the three CNN instances. Considering all the three runs, Figure 5.7 (bottom) shows that the CNNs of Amazon Products also behaved in a similar way as we saw in the Yelp dataset, except that disabling rank C features slightly undermined, not increased, performance. This implies that even the rank C features contain a certain amount of useful knowledge for this classifier.[6]

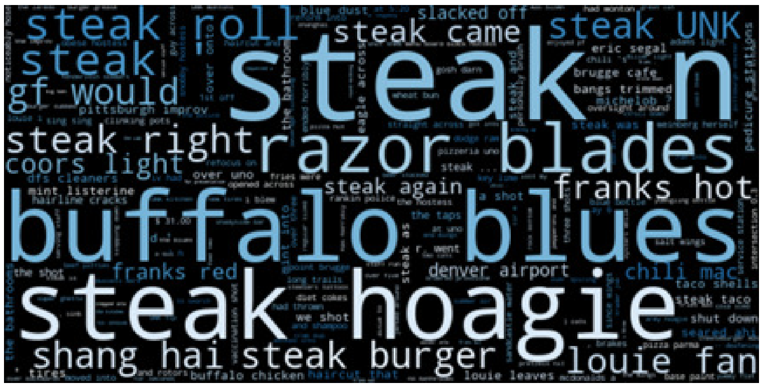
[6] We also conducted the same experiments here with bidirectional LSTM networks (BiLSTMs) which required a different way to generate the word clouds (see Appendix C.1).



Rank A - Average score = 2.0

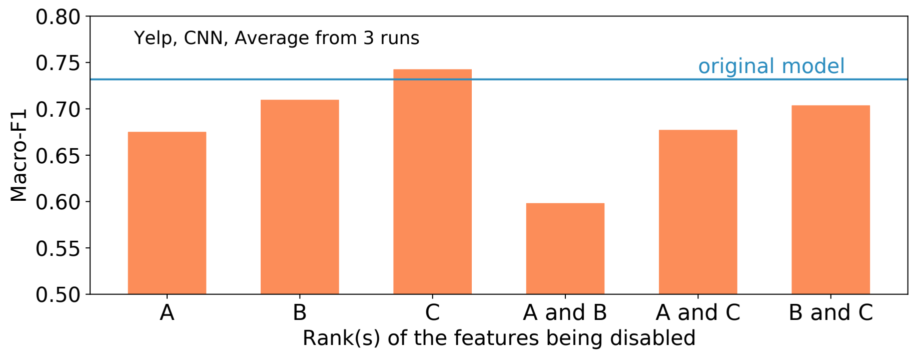


Rank B - Average score = 1.2



Rank C - Average score = -0.7

Figure 5.6.: Examples of word clouds of CNN features in ranks A, B, and C (Experiment 1, Yelp – sentiment).



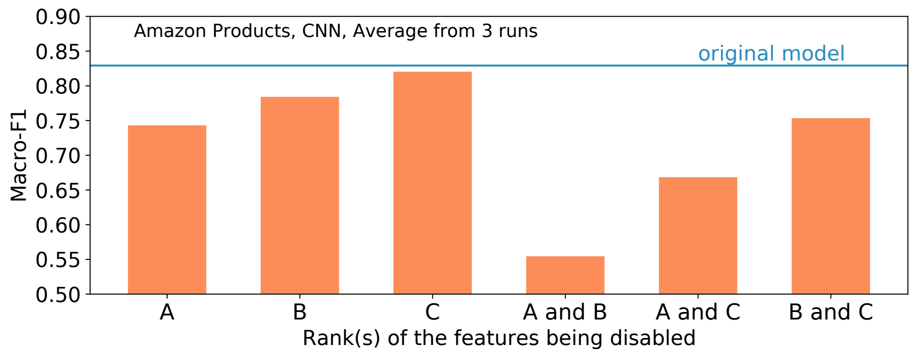


Figure 5.7.: The average macro F1, from the three runs, of all the CNN models for the Yelp dataset (top) and the Amazon Products dataset (bottom). It shows that disabling both features in rank A and rank B resulted in the worst macro F1, while disabling features in rank C only affected the macro F1 slightly.

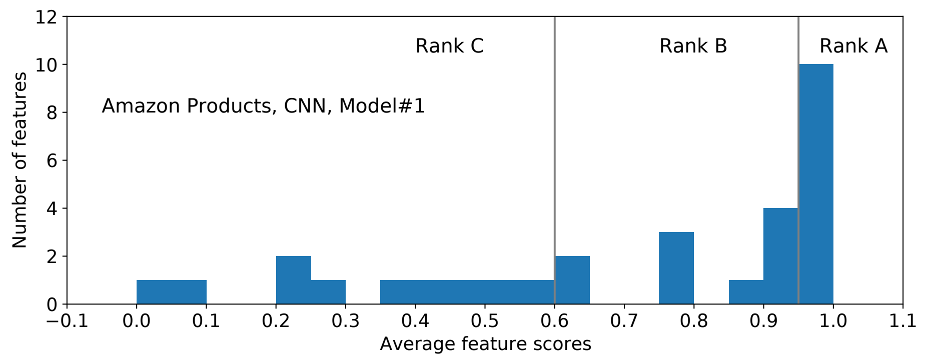


Figure 5.8.: The distribution of average feature scores in a CNN model trained on the Amazon Products dataset.

5.4. Experiment 2: Training Data with Biases

Given a biased training dataset, a text classifier may absorb the biases and produce biased predictions against some sub-populations. We hypothesize that if the biases are captured by some of the learned features, we can apply FIND to disable such features and reduce the model biases. Ultimately, this experiment does not aim to remove bias existing in the data. Instead, it aims to prevent the model from exploiting the existing bias which could be unfair to some sub-populations during prediction.

5.4.1. Debugging Context and Materials

We focus on reducing gender bias of CNN models trained on two datasets – **Biosbias** (De-Arteaga et al., 2019) and **Waseem** (Waseem and Hovy, 2016). For Biosbias, the task is predicting the occupation of a given bio paragraph, i.e., whether the person is ‘a surgeon’ (class 0) or ‘a nurse’ (class 1). We created the training dataset using the code provided by De-Arteaga et al. (2019) here[7], and all the bios were from Common Crawl August 2018 Index. Due to the gender imbalance in each occupation, a classifier usually exploits gender information when making predictions. As a result, bios of female surgeons and male nurses are often misclassified. For Waseem, the task is abusive language detection – assessing if a given text is abusive (class 1) or not abusive (class 0). The authors of (Waseem and Hovy, 2016) kindly provided the dataset to us by email. We considered “racism” and “sexism” examples in the original dataset as “Abusive” and “neither” examples as “Non-abusive”. Previous work found that this dataset contains a strong negative bias against females (Park et al., 2018b). In other words, texts related to females are usually classified as abusive although the texts themselves are not abusive at all. Hence, we wanted to address this issue in the experiment. In addition, we tested the models, trained on the Waseem dataset, using another abusive language detection dataset, **Wikitoxic** (Thain et al., 2017), to assess generalizability of the models. The Wikitoxic dataset can be downloaded here[8]. We used only examples which were given the same label by all the annotators.

As with Experiment 1, we trained three CNN models with different ran-

[7]`https://github.com/Microsoft/biosbias`
[8]`https://figshare.com/articles/Wikipedia_Talk_Labels_Toxicity/4563973`

dom seeds for each of the two training datasets (Biosbias and Waseem). Thereby, we had 180 word clouds to be assessed by human participants. This resulted in 1800 questions in total for this experiment. Unlike the interface in Figure 5.3, for each word cloud, we asked the participants to select the relevant class from three options. Specifically, for Biosbias, we asked "Given this wordcloud, are the text fragments more relevant to *Surgeon* or *Nurse*?" and provided three options – Surgeon, Nurse, and It could be either. Similarly, for Waseem, we asked "Given this wordcloud, is it relevant to *Abusive* or *Non-abusive* texts?" with three options – Abusive, Non-abusive, and It could be either.

5.4.2. Participants and Procedure

We also used Amazon MTurk to recruit human participants here, similar to Experiment 1. We posted on MTurk the tasks for two datasets separately with short descriptions. For the Waseem dataset, we additionally warned the workers in the description that "This HIT may contain offensive content. Worker discretion is advised.". The three qualifications were still required for a worker to participate in our experiment: (1) currently living in the US, the UK, Australia, or New Zealand, (2) having at least 50 approved HITs and (3) having more than 97% HITS approval rate so far. Again, an MTurk worker who passed all the requirements could do as many tasks or HITs as s/he wanted. The structure of a HIT is the same as in Experiment 1, consisting of the instructions, three sample questions (with answers and reasons for the answers), seven questions to be answered, a feedback text box, and a submit button. In the instructions of this experiment particularly, we firmly stressed that gender-related terms should not be used as an indicator for one or the other class to ensure that the participants do not use their biases while answering our questions. After completing the seven questions and clicking submit to finish the HIT, the worker could accept to do the next HIT if they prefer.

In total, we had 39 and 63 MTurk participants for the Biosbias and the Waseem datasets, respectively.

5.4.3. Data Collection and Analysis

We aimed to use the human answers to decide whether we should disable a feature or not. The features of which majority of the human answers disagreed with the class suggested by the weight matrix **W** were disabled before the model was re-trained[9]. We expected that the disabled features included the ones causing gender bias in the model, and therefore the model bias should be reduced after re-training.

To quantify gender biases, we adopted two metrics – false positive equality difference (FPED) and false negative equality difference (FNED) (Dixon et al., 2018). Formally,

$$FPED = \sum_{t \in T} |FPR - FPR_t| \tag{5.1}$$

$$FNED = \sum_{t \in T} |FNR - FNR_t| \tag{5.2}$$

where T is a set of all sub-populations we consider (i.e., $T = \{\text{male}, \text{female}\}$). FPR and FNR stand for false positive rate and false negative rate, respectively. The subscript t means that we calculate the metrics using data examples mentioning the sub-population t only. We used the following keywords to identify examples which are related to or mentioning the sub-populations.

Male gender terms:

"male", "males", "boy", "boys", "man", "men", "gentleman", "gentlemen", "he", "him", "his", "himself", "brother", "son", "husband", "father", "uncle", "dad", "boyfriend"

Female gender terms:

"female", "females", "girl", "girls", "woman", "women", "lady", "ladies", "she", "her", "herself", "sister", "daughter", "wife", "mother", "aunt", "mom", "girlfriend"

According to Equation 5.1, FPED will be zero when the false positive rate of input texts mentioning gender terms equals the false positive rate of the whole dataset. FNED is also defined in an analogous way to FPED. Therefore, the lower these metrics are, the less biases the model has. To demonstrate the effectiveness of our debugging framework, we compared the bias metrics before and after debugging. Also, we observed the macro F1

[9]We disabled features with tie scores (having more than one majority answer) as well.

changes to see how much bias removal affected the model performance on the original test set. Note that all the reported results are the average of the three runs.

5.4.4. Results and Discussions

The results of this experiment are displayed in Figure 5.9. For Biosbias, on average, the participants' responses suggested us to disable 11.33 out of 30 CNN features. By doing so, the FPED of the models decreased from 0.250 to 0.163, and the FNED decreased from 0.338 to 0.149. After investigating the word clouds of the CNN features, we found that some of them detected patterns containing both gender-related terms and occupation-related terms such as "his surgical expertise" and "she supervises nursing students". Most of the MTurk participants answered that these word clouds were relevant to the occupations, and thus the corresponding features were not disabled. However, we believe that these features might contain gender biases. So, we considered all the word clouds again and manually disabled every feature for which the prominent n-gram patterns contained any gender-related terms, no matter whether the patterns detect occupation-related terms. With this new *brutal* disabling policy, 12 out of 30 features were disabled on average, and the model biases further decreased, as shown in Figure 5.9 (Debugged (One)). The side-effect of disabling 33% of all the features here was only a slight drop in the macro F1 from 0.950 to 0.933. Hence, our framework was successful in reducing gender biases without severe negative effects in classification performance.

Concerning the abusive language detection task, on average, the MTurk participants' responses suggested us to disable 12 out of 30 CNN features. Unlike Biosbias, disabling features based on MTurk responses unexpectedly increased the gender bias for both Waseem and Wikitoxic datasets. However, we found one similar finding to Biosbias, that many of the CNN features captured n-grams which were both abusive and related to a gender such as 'these girls are terrible' and 'of raping slave girls'. The participants considered these word clouds as abusive, agreeing with the weight matrix $\mathbf{W}$. So, these features were not disabled. So, we tried applying the brutal policy manually to this dataset – disabling all features which involved gender words even though some of them also detected abusive words. By

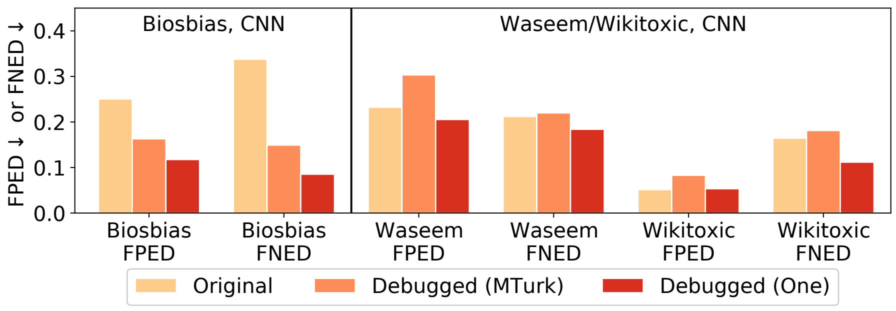


Figure 5.9.: The average FPED and FNED of the CNN models in Experiment 2 (the lower, the better).

disabling 18 out of 30 features on average, the gender biases were reduced for both datasets (except FPED on Wikitoxic which stayed close to the original value). Another consequence was that we sacrificed 4% and 1% macro F1 on the Waseem and Wikitoxic datasets, respectively. The macro F1 drop for the Waseem dataset was larger because, in this dataset, some gender terms have strong correlation with abusive or non-abusive texts. Disabling the features related to genders also turned down the chance to exploit superficial clues to the correct answers. This finding is consistent with (Park et al., 2018b) that reducing the bias and maintaining the classification performance at the same time is very challenging.

5.5. Experiment 3: Dataset Shift

Dataset shift is a problem where the joint distribution of inputs and outputs differs between training and test stage (Quionero-Candela et al., 2009). Many classifiers perform poorly under dataset shift because some of the learned features are inapplicable (or sometimes even harmful) to classify test documents. We hypothesize that FIND is useful for investigating the learned features and disabling the overfitting ones to increase the generalizability of the model.

5.5.1. Debugging Context and Materials

We considered two tasks in this experiment. For each task, we had one dataset for training and at least one (slightly) different dataset for testing. The first task aims to classify “Christianity” vs “Atheism” documents from the **20Newsgroups** dataset (Lang, 1995). We downloaded the standard splits of the dataset using scikit-learn[10]. The header and the footer of each text were removed. This dataset is special because it contains a lot of artifacts – tokens (e.g., person names, punctuation marks) which are not relevant, but strongly co-occur with one of the classes. This prevents the models from generalizing effectively to the target dataset with the absence of these artifacts. For evaluation, we used the **Religion** dataset by Ribeiro et al. (2016), downloaded from here[11], as a target dataset. It contains “Christianity” and “Atheism” web pages to be classified; however, the artifacts found in the 20Newsgroups dataset do not exist in the Religion dataset. The second task is sentiment analysis. We used, as a training dataset, **Amazon Clothes**, with reviews of clothing, shoes, and jewelry products, and as test sets three out-of-distribution datasets – **Amazon Music**, **Amazon Mixed**, and the **Yelp** dataset (which was used in Experiment 1). Amazon Music contains only reviews from the “Digital Music” product category which was found to have an extreme distribution shift from the clothes category (Hendrycks et al., 2020). Amazon Mixed compiles the reviews from various kinds of products, while Yelp focuses on restaurant reviews. The **Amazon Clothes** and **Amazon Music** datasets we used were sampled from the datasets provided by He and McAuley (2016)[12], whereas the **Amazon Mixed** dataset was sampled from the datasets provided by Zhang et al. (2015).

We trained three CNN models with different random seeds for each of the two training datasets (20Newsgroups and Amazon Clothes). In total, we had 180 word clouds (i.e., 1800 questions) in this experiment. As with Experiments 2, for each word cloud, we asked the participants to select the relevant class from three options. In particular, for 20Newsgroups, we asked “Given this wordcloud, is it more relevant to *Atheism* or *Christianity*?” and provided three options – Atheism, Christianity, and None of the two options.

[10]`https://scikit-learn.org/`

[11]`https://github.com/marcotcr/lime-experiments`

[12]`http://jmcauley.ucsd.edu/data/amazon/`

Similarly, for Amazon Clothes, we asked "Given this wordcloud, does it convey negative or positive sentiment in the context of product reviews?" with three options – Negative, Positive, and Neither.

5.5.2. Participants and Procedure

Most of the procedure in this experiment is the same as in Experiment 2. In particular, we used Amazon MTurk to recruit human participants The three qualifications were still required: (1) currently living in the US, the UK, Australia, or New Zealand, (2) having at least 50 approved HITs and (3) having more than 97% HITS approval rate so far. A single HIT consists of the instructions, three sample questions (with answers and reasons for the answers), seven questions to be answered, a feedback text box, and a submit button. The worker could submit a HIT after s/he answered all the seven questions while the feedback was completely optional. After that, s/he could stop, accept to do the next HIT, or switch to another task as they wish. In total, we had 48 and 35 MTurk participants for the 20Newsgroups and the Amazon Clothes datasets, respectively.

5.5.3. Data Collection and Analysis

Again, we used the human answers to decide which features to disable. When the majority vote from the ten human answers disagreed with the weight matrix $\mathbf{W}$, we disabled that corresponding feature. Following the FIND framework, we then re-trained the model and observed the performance (mainly the macro F1) on both the original test set and the out-of-domain test set(s) before and after re-training. We hypothesized that the disabled features were specific to the training set; therefore, the model should generalize better after the model update. The reported results were averaged over three runs.

5.5.4. Results and Discussions

For the first task, on average, 14.33 out of 30 features were disabled and the macro F1 scores of the 20Newsgroups before and after debugging are 0.853 and 0.828, respectively. The same metrics of the Religion dataset are 0.731 and 0.799. This shows that disabling irrelevant features mildly undermined the predictive performance on the in-distribution dataset, but

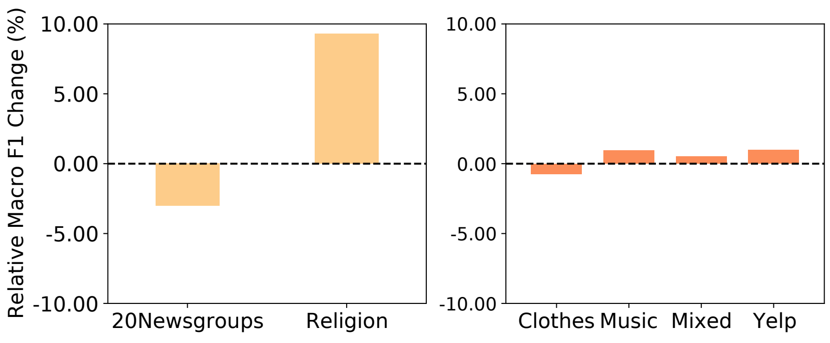


Figure 5.10.: The relative Macro F1 changes (in %) of the CNN models for both tasks in Experiment 3.

clearly enhanced the performance on the out-of-distribution dataset (see Figure 5.10, left). This is especially evident for the Atheism class for which the F1 score increased around 15% absolute. We noticed from the word clouds that many prominent words for the Atheism class learned by the models are person names (e.g., Keith, Gregg, Schneider) and these are not applicable to the Religion dataset. Forcing the models to use only relevant features (detecting terms like 'atheists' and 'science'), therefore, increased the macro F1 on the Religion dataset.

Unlike 20Newsgroups, Amazon Clothes does not seem to have obvious artifacts. Still, the responses from crowd workers suggested that we disable 6 features. The disabled features were correlated to, but not the reason for, the associated class. For instance, one of the disabled features was highly activated by the pattern "my .... year old" which often appeared in positive reviews such as "my 3 year old son loves this.". However, these correlated features are not very useful for the three out-of-distribution datasets (Music, Mixed, and Yelp). Disabling them made the model focus more on the right evidence and increased the average macro F1 for the three datasets, as shown in Figure 5.10 (right). Nonetheless, the performance improvement here was not as apparent as in the previous task because, even without feature disabling, the majority of the features are relevant to the task and can lead the model to the correct predictions in most cases.[13]

[13]See Appendix C.2 for the full results from all experiments.

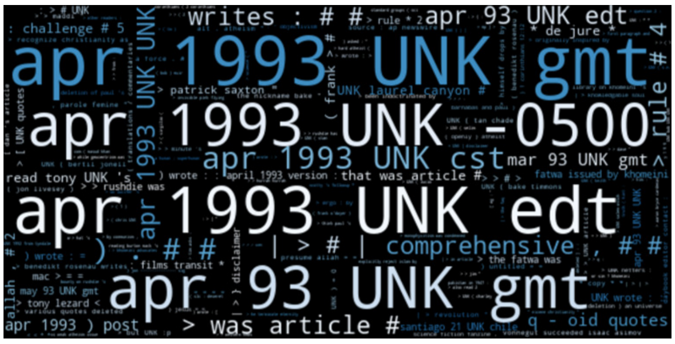


Figure 5.11.: A word cloud representing a CNN feature for the 20Newsgroup dataset. It mainly detects April 1993 (followed by an unknown token and a time zone).

5.6. General Discussions

5.6.1. Ensuring Model Improvement

Obviously, the effectiveness of the FIND framework depends largely on the quality of human feedback. In the experiments, we tried to ensure the quality by specifying qualifications for MTurk workers who were allowed to participate in our experiments. Moreover, for each word cloud, we made the enabling/disabling decision based on the average scores or the majority of ten answers so as to dilute the effect of low-quality individual answers. However, this could work only under the assumption that majority of the MTurk workers are trustworthy, and we had nothing to guarantee this assumption. This could be a reason why the second experiment on bias mitigation was not very successful when we used only answers from MTurk workers.

Even though the humans who provide feedback perform the task at their best, it does not necessarily lead to the best model. Das et al. (2013) found that users assess importance of features based on the concept relatedness between the features and the target classes (e.g., according to a commonsense knowledge base (Liu and Singh, 2004)). However, this may not correlate perfectly with discriminative powers of the features when it comes to classification. Moreover, the model may find some patterns which are truly useful but humans do not understand why and, hence, disable them. For example, Figure 5.11 displays a word cloud of a CNN feature for the 20Newsgroups dataset. It mainly detects April 1993 followed by an unknown token and a time zone which seems to be artifacts in the training data. Among the ten human participants who checked this word cloud, nine answered that it is relevant to neither Atheism nor Christianity, whereas the other one answered that it is relevant to Atheism. As a result, this CNN feature was disabled in our experiment. Nonetheless, we found out later that April is an important month of Atheism for two reasons. First, some sources claim that April 1st is National Atheist Day[14]. Second, some also claim that National Ask An Atheist Day has always been observed annually in April[15]. Therefore, having a feature detecting April in the CNN might actually be more helpful than harmful. Indeed, the weight the CNN gave to this feature supported the Atheism class (Atheism:Christianity = 0.475:0.179).

In practice, to ensure before deployment that the overall human feedback is more helpful than harmful, we may check whether the performance of the updated model is better than that of the original model using a small set of labeled examples from the test distribution (if we have it). In Experiment 1 and 2, we could in fact use a development set and measure the performance using macro-F1 and FPED/FNED, respectively. If possible, in Experiment 3, we could use a labeled development set of each test dataset (not shown in Table 5.1) and measure the macro-F1. This data availability condition is similar to the setting of semi-supervised domain adaptation where we are allowed to access a small set of labeled data as well as a set of unlabeled data from the target domain (Daumé III et al., 2010). If we have only unlabeled data of the target domain (which is equivalent to unsupervised domain adaptation (Ganin and Lempitsky, 2015)), we may generate word clouds using the unlabeled data instead of the original training data. This helps us make enabling/disabling decisions more accurately since we can see patterns the CNN features would likely detect when running on the target distribution.

5.6.2. Integration to an AI Development Process

The FIND framework is helpful for model verification and improvement. Therefore, it could be used after the training process to let humans verify what the model has learned and improve it if necessary. In practice, we do

[14] `https://www.holidayinsights.com/moreholidays/April/atheist-day.htm` and `https://happydays365.org/atheist-day/national-atheist-day-april-1/` (Accessed on 28 November 2021)

[15] `https://www.worldnationaldays.com/national-ask-an-atheist-day/` and `shorturl.at/xAGQZ` (Accessed on 28 November 2021)

not need ten people and aggregate their answers as we did in our experiments to reduce the effect of bad annotators. Only one person who knows the domain well would be enough to provide feedback. This person could be a model developer (who is usually a machine learning expert) or a domain expert in the development team. In the former case, we may also show the weights (in $\mathbf{W}$) of each feature as additional information which is especially helpful for multi-class classification tasks where a feature may contribute to more than one class.

5.6.3. Generalization to Other Models

Although the ultimate goal of FIND is to tackle deep text classifiers in general, dealing with more complex deep models would be inevitably an ambitious attempt. As discussed in Section 5.1.3, the more hidden features the model has, the more human effort FIND needs for debugging. For instance, applying FIND to BERT-base (Devlin et al., 2019), which has 768 features (before the final dense layer), would require lots of human effort to perform investigation. In this case, it would be more efficient to use FIND to disable attention heads rather than individual features (Voita et al., 2019). Still, the best way to visualize attention heads for debugging needs to be researched (Vig, 2019; DeRose et al., 2020). We further discuss this in the future work part (Section 7.2).

Another issue is that it is not clear how to divide M_f and M_c in complex models in order to yield good results. One factor which makes our framework effective in the experiments is the fact that M_c is just a linear layer, showing clearly how the features contribute to the output nodes. For some deep models (such as BERT) and some classification tasks, however, features in a lower layer are more suitable to be investigated because semantic properties which are relevant to the task emerge at that layer (Tenney et al., 2019). If we wait until a higher layer, these properties may already be mixed up, making feature disabling more difficult. However, we need to answer two additional questions if we want to divide M_f and M_c at a lower layer. First, what is the best dividing point? We may need to consult recent research analyzing knowledge emerging at different layers of such complex models (Rogers et al., 2020). Second, how does each feature contribute to output nodes given that there is more than a linear layer between the two?

A possible solution is to compute the partial derivative of the output nodes with respect to the feature, similar to what we did for Grad-CAM-Text in Chapter 4. Whether there is a better solution in the context of model debugging is an open research question.

5.6.4. Generalization to Other Tasks

FIND is suitable for any single-input text classification tasks where a model might learn irrelevant or harmful features during training. It is also convenient to use since only the trained model and the training data are required as input (if we do not evaluate the updated model before deployment to ensure performance improvement as discussed in Section 5.6.1). Moreover, it can address many problems simultaneously such as removing religious and racial bias together with gender bias even if we might not be aware of such problems before using FIND.

It would be interesting to extend FIND to other NLP tasks, e.g., question answering and natural language inference. However, this will require some modifications to understand how the features capture relationships between two input texts. Meanwhile, applying FIND beyond textual domain is quite challenging because we are not sure whether humans can truly understand visualization of each feature. For example, in computer vision, we may utilize activation maximization (Erhan et al., 2009) to find an artificial input image that would maximize the value of each neuron, telling us which kind of pattern the neuron is looking for. However, because the image is synthesized to optimize the neuron value, it could be difficult for humans to recognize and decide whether it is relevant to the classification task or not. Another way to visualize a neuron is to use a collection of real input images with the relevant part highlighted such as network dissection (Bau et al., 2017). This is more similar to our word cloud but the users may need to look at multiple images to grasp the semantics of the node unless we have an effective way to collate them as we did for texts using word clouds.

5.6.5. Other Limitations

There are a few additional limitations of FIND worth discussing here. First, the word clouds may reveal sensitive contents in the training data to human debuggers. So, this framework may not be suitable if we do not want the humans who provide feedback to partially see the content in the training dataset unless we use another dataset to generate the word clouds instead. However, if the dataset used is unrelated to the distribution of the target domain, the resulting word clouds may be less useful since they do not reveal the model behavior under the scenario of interest.

Besides, it is possible that one feature detects several patterns (Jacovi et al., 2018), and it will be difficult to disable the feature if some of the detected patterns are useful while the others are harmful. Hence, it would be interesting to apply FIND to models with disentangled text representations (Cheng et al., 2020) so that it is easier to make enabling/disabling decisions.

5.7. Summary

We proposed FIND, a framework which enables humans to debug simple deep text classifiers by disabling irrelevant or harmful features. Using the proposed framework on CNN text classifiers, we found that *(i)* word clouds generated by running LRP on the training data accurately revealed the behaviors of CNN features, *(ii)* some of the learned features might be more useful to the task than the others and *(iii)* disabling the irrelevant or harmful features could improve the model predictive performance and reduce unintended biases in the model.

In this chapter, we focused on the learned intermediate features and verified whether they are relevant for the classification. In the next chapter, in contrast, we will study relations among features of a more interpretable model, using computational argumentation, in order to generate local explanations that align better with human perceptions.

6. Argumentative Explanations for Text Classification

In Chapter 5, we learned that hidden features in a CNN captures some patterns that are useful for classification, as shown in Figure 5.6 for examples. Each of the feature may detect more than one pattern, and some of the patterns are not easy for humans to interpret. This is not a problem for a machine learning model aiming to get high scores for a given classification task. However, this may not be convenient for humans to learn or to extract knowledge from the trained model. This inspires us to study a more transparent model in which we can precisely see the patterns the model exploits for classification. Specifically, this chapter focuses on pattern-based (binary) logistic regression models, whose patterns are drawn by means of the GrASP algorithm (Shnarch et al., 2017), explained in Section 6.1. As discussed in Chapter 3 that explanation methods for interpretable models have been scarcely studied in the literature, this chapter also aims to fill the gap by devising a local explanation method which is likely to generate more plausible explanations for the pattern-based logistic regression.

Given that the learned patterns interact by disagreeing or agreeing with classifications for input text, we choose to extract argumentation frameworks to explain to humans this type of classifier in particular. These argumentation frameworks may be non-flat, unearthing chains of arguments in explanations (see Section 6.2). We call the resulting explanatory framework **AXPLR**. After that, the experimental setup for AXPLR is described in Section 6.3, followed by one empirical experiment in Section 6.4 and two human experiments (to assess the amenability of the argumentation underpinning AXPLR specifically) in Sections 6.5 and 6.6 to show the strengths of AXPLR. Lastly, we discuss the method complexity and other possible uses of AXPLR in Section 6.7 and summarize the chapter in Section 6.8.

6.1. Pattern-Based Logistic Regression

As explained in Section 2.1.2, logistic regression (LR) is a traditional machine learning model that can be used for text classification. Before we can use LR, we need to perform the feature extraction step. The easiest way to do is using n-gram features together with TF-IDF vectorization. However, we will use patterns as features in this chapter because patterns are more generalizable than n-grams and easier for humans to learn from.

Logistic Regression. Given an input text x, we extract the feature vector $\mathbf{f} = [f_1, f_2, \ldots, f_d] \in \{0,1\}^d$ where f_i is a binary feature and d is the number of textual patterns used. f_i equals 1 if the input x contains the pattern p_i; otherwise, f_i equals 0. Then we use the standard binary logistic regression with the binary cross-entropy loss to learn from the training data. To recap from Section 2.1.2,

$$\begin{aligned} P(y=1|x) &= \text{sigmoid}(\mathbf{w}^T\mathbf{f} + b) \\ &= \text{sigmoid}(\sum_{i=1}^{d} w_i f_i + b) \\ &= \text{sigmoid}(w_1 f_1 + w_2 f_2 + \ldots + w_d f_d + b) \end{aligned} \tag{6.1}$$

The next questions are “How do the d patterns look like?” and “How can we obtain them?”. We envisage that a pattern should be able to indicate high-level characteristics of words in the pattern in addition to specifying exact words or lemmas. These high-level characteristics include both syntactic attributes (such as part-of-speech tags) and semantic attributes (such as synonyms and hypernyms). Thereby, we choose **GrASP** for this purpose.

GrASP: GReedy Augmented Sequential Patterns. GrASP is a supervised algorithm which learns expressive patterns characterizing linguistic phenomena (Shnarch et al., 2017). To illustrate, examples 1–3 in Table 6.1 are all SMS spam messages from the dataset by (Almeida et al., 2011). While there is little word overlap between them, their commonality is apparent, even if hard to name. The GrASP algorithm can reveal an underlying structure which generalizes these three realizations of spams: a positive-sentiment word, closely followed by a determiner, and then by a proper noun.

Sentences matched	You are **awarded a SiPix** Digital Camera... ...to **WIN a** FREE **Bluetooth** Headset... ...for **Free** ! Call **The Mobile** Update...
GrASP pattern	[[`SENTIMENT:pos`], [`POS:DET`], [`POS:PROPN`]] (A positive-sentiment word, closely followed by a determiner, and then by a proper noun)

Table 6.1.: Example of GrASP pattern capturing the common structure in a variety of surface forms – the sentences matched. Matched words are in bold. The description of the pattern is provided below it, in parenthesis.

The input for the algorithm amounts to two sets of texts: in one (the positive set) the target phenomenon appears in all examples, and in the other (the negative set) it does not. GrASP looks for commonalities prominent within the texts of one set but not shared across the sets. To be able to recognize common aspects of texts, beyond their surface form realizations, all input tokens are augmented with a variety of linguistic attributes such as part-of-speech tags, named entity information, or pertinence to a lexicon (e.g., of sentiment words). Attributes are selected to maximize a score (by default their information gain about the label). Then, they are combined by a greedy algorithm to generate patterns which are most indicative either to the positive or the negative set.

Since the patterns are a combination of readable attributes, they are human interpretable. So, they can be used to provide insights about the data and contribute to explainability. For instance, the first sentence in Table 6.1 is a spam message because it contains the phrase "awarded a SiPix". With this strength, we decide to use GrASP patterns as our features in the binary logistic regression model.

The Overall Process. During training, given a dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $y_i \in \{0, 1\}$, we first split $\mathcal{D}$ into $\mathcal{D}^+$ and $\mathcal{D}^-$, containing positive examples ($y_i = 1$) and negative ($y_i = 0$) examples, respectively. Next, we feed $\mathcal{D}^+$ and $\mathcal{D}^-$ to the GrASP algorithm along with some hyperparameters (e.g., the number of patterns d, the number of gaps allowed in a pattern, and the set of linguistic attributes for augmenting input tokens). After obtaining the d patterns, we extract the binary feature vector $\mathbf{f}$ for each training example x_i and use it to train the logistic regression model

Input: *There is nothing better than hot sausages of this restaurant.*
Matched patterns:

$p_1 = [[\texttt{TEXT:nothing}], [\texttt{SENTIMENT:pos}]]$ $\quad w_1 f_1 = -0.9$
$p_2 = [[\texttt{TEXT:nothing}]]$ $\quad w_2 f_2 = -0.4$
$p_3 = [[\texttt{SENTIMENT:pos}]]$ $\quad w_3 f_3 = 1.2$
$p_4 = [[\texttt{TEXT:hot}], [\texttt{TEXT:sausages}]]$ $\quad w_4 f_4 = 0.5$
$b = -0.1$

Predicted class: 1 (Positive)
Prob.: sigmoid(-0.9-0.4+1.2+0.5-0.1) = sigmoid(0.3) = 0.5744.

Figure 6.1.: An illustrative example of using pattern-based logistic regression for sentiment analysis.

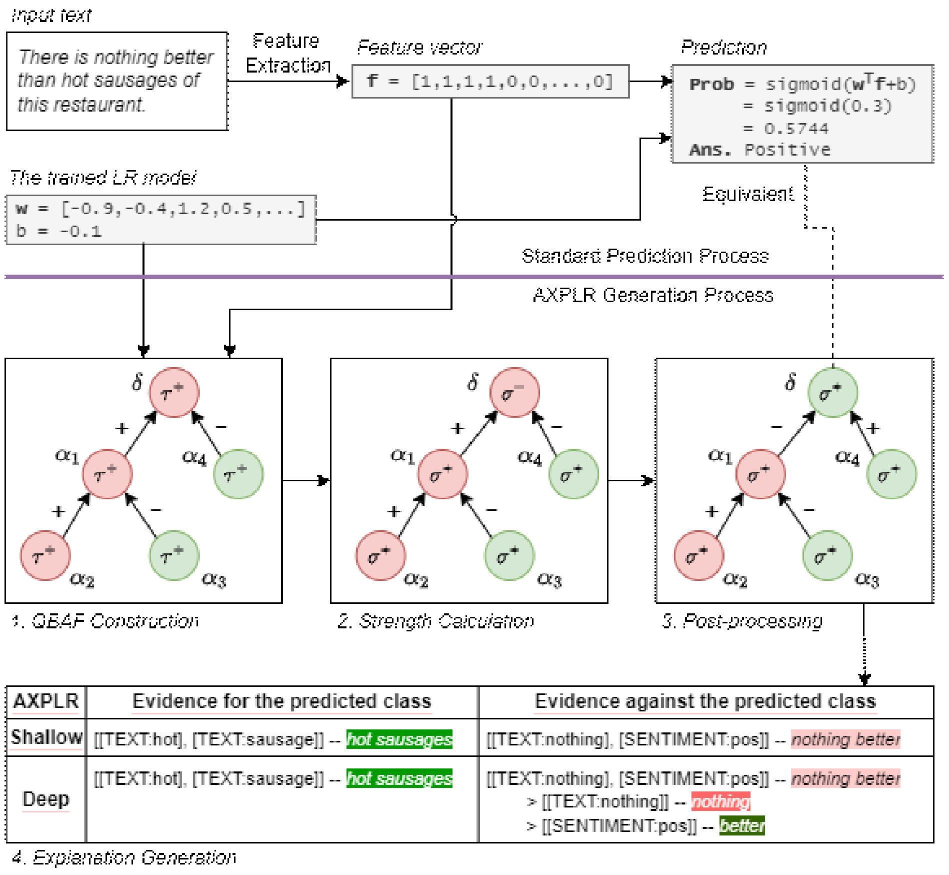
together with the class label y_i.

During testing, given an unseen document x, we get the prediction by extracting the feature vector $\mathbf{f}$ using the d GrASP patterns and running the logistic regression model on $\mathbf{f}$. Since logistic regression is inherently transparent, we can generate a local explanation by reporting parts of x that match top-k patterns of this example. Formally, let c_j be the contribution of the pattern p_j for the prediction $\hat{y}$. According to Equation 6.1, we can see that when $\hat{y} = 1$, $c_j = w_j f_j$. When $\hat{y} = 0$, $c_j = -w_j f_j$. So, we can combine both cases to be $c_j = (-1)^{\hat{y}+1} w_j f_j$. Then we return, as the local explanation for $\hat{y}$, a list of $(p_{j'}, \pi(p_{j'}, x), c_j)$ triplets where $c_{j'}$ is one of the k highest c_j, $c_{j'} \neq 0$, and $\pi(p_{j'}, x)$ is a part of x that matches the pattern $p_{j'}$. We call this explanation the **flat logistic regression explanation (FLX)**.

However, one problem of FLX is that it does not take into account relationships among the d patterns. Consider the example in Figure 6.1, the input text matches four patterns, and the model predicts positive. If we use FLX with top-1 pattern, it will return p_3 (having a positive word in the input) as the reason for predicting positive (due to the highest contribution of 1.2). Nevertheless, the model has actually weakened the effect of p_3 by p_1 because the positive word in this case follows the word "nothing" and the model no longer considers it strongly positive. What really makes the model answer positively is rather p_4, which is not selected by FLX. Although the contribution of p_4 (0.5) is lower than that of p_3, it is not overridden by other patterns. We could see that these four patterns are arguing to make the prediction, in that each pattern is an argument for or against the prediction. Some patterns have dialectical relations with one another (such as the disagreement between p_1 and p_3). Hence, to improve plausibility of the explanations, it is likely promising to apply computational argumentation (as introduced in Section 2.3) to generate local explanations for this pattern-based LR model.

6.2. AXPLR: Argumentative Explanations for Pattern-Based Logistic Regression

In this section, we introduce our AXPLR explanation where its overall generation process is shown in Figure 6.2 with the illustrative example from Figure 6.1. The part above the bold purple line is the standard prediction process which has been shown in Figure 6.1, starting from extracting the feature vector from the input text and then computing the predicted probability using the model weights ($\mathbf{w}$ and b). Below the purple line, it shows the four main steps for generating AXPLR. Using the feature vector and the model weights, the first step constructs a quantitative bipolar argumentation framework (QBAF) to represent relationships between four pattern features found in the input text (as well as the bias term of the model). The second step computes the dialectical strength of each argument considering its attacker(s) and supporter(s). The dialectical strengths of some arguments might be less than zero which are difficult to interpret, so we do post-processing in the third step, making all the strength values to be positive and adjusting the arguments and the relations accordingly in a way that preserves the meaning of the argumentation graph. Finally, using the post-processed QBAF, the fourth step generates the AXPLR explanation which could be a shallow AXPLR (using only top-level arguments in the explanation) or a deep AXPLR (using arguments at the top-level and deeper in the explanation). The background color of text fragments matching the patterns reflects the post-processed strengths of the corresponding arguments. We then can present the AXPLR explanations to assist humans to perform a human-AI collaboration task (such as learning to detect deceptive



AXPLR	Evidence for the predicted class	Evidence against the predicted class
Shallow	[[TEXT:hot], [TEXT:sausage]] -- hot sausages	[[TEXT:nothing], [SENTIMENT:pos]] -- nothing better
Deep	[[TEXT:hot], [TEXT:sausage]] -- hot sausages	[[TEXT:nothing], [SENTIMENT:pos]] -- nothing better > [[TEXT:nothing]] -- nothing > [[SENTIMENT:pos]] -- better

Figure 6.2.: Overview of the AXPLR generation process. Above the bold purple line, it shows the standard prediction process of pattern-based logistic regression. Below the bold purple line, it shows the four main steps to generate AXPLR explanations. τ^+ and τ^- indicate that the base scores of the arguments are positive and negative, respectively. Similarly, σ^+ and σ^- indicate that the dialectical strengths of the arguments are positive and negative, respectively. Note that here we use a bottom-up QBAF (BQBAF). Other details will be explained throughout this section.

reviews in Section 6.6).

This section provides details for each step of the AXPLR generation process. Sections 6.2.1, 6.2.2, 6.2.3, and 6.2.5 explain steps 1, 2, 3, and 4, respectively. In Section 6.2.4, before step 4, we prove formal properties of (original and post-processed) QBAFs, providing formal guarantees about their suitability to give rise to explanations.

6.2.1. Argumentation Framework

We aim to use a quantitative bipolar argumentation framework (QBAF) to simulate how the pattern-based LR model works on an input text. Then we will derive the local explanation, called AXPLR, from the QBAF. To begin with, we define how two patterns can be related using the concept of specificity. Particularly, Definition 9 defines *more specific than or equivalent* and *more specific than* in line with other definitions in argumentation literature, e.g., (Baroni et al., 2019).

Definition 9. *A pattern p_1 is more specific than or equivalent to another pattern p_2 (written as $p_1 \succeq p_2$) if and only if for every text t matched by p_1, t is also matched by p_2. In addition, p_1 is more specific than p_2 (written as $p_1 \succ p_2$) if and only if $p_1 \succeq p_2$ but $p_2 \not\succeq p_1$.*

For instance, we can say from Figure 6.1 that $p_1 \succeq p_3$ because every text matched by p_1 is guaranteed to have a positive sentiment word which makes it matched by p_3. However, $p_3 \not\succeq p_1$ because a text matched by p_3 is guaranteed to have a positive word but it may not have the word "nothing" followed by a positive word. These two facts also imply $p_1 \succ p_3$. Similarly, $p_1 \succ p_2$.

Lemma 1. *The relation $\succ$ is not reflexive, not symmetric, but transitive.*

Proof. Let us consider each of the properties.

- *Not reflexive*: Proof by contradiction. Assume that $\succ$ is reflexive. Thus, $p \succ p$. According to Definition 9, it implies that $p \succeq p$ and $p \not\succeq p$, resulting in contradiction. Hence, $\succ$ is not reflexive.

- *Not symmetric*: Assume $p_1 \succ p_2$. By Definition 9, we obtain that $p_1 \succeq p_2$ and $p_2 \not\succeq p_1$. So, $p_2 \not\succ p_1$, implying that $\succ$ is not symmetric.

- *Transitive*: Before proving that $\succ$ is transitive, we will prove that $\succeq$ is transitive first. Assume that $p_1 \succeq p_2$ and $p_2 \succeq p_3$. Because $p_1 \succeq p_2$, by Definition 9, every text t matched by p_1 is also matched by p_2. Similarly, because $p_2 \succeq p_3$, every text t matched by p_2, including those matched by p_1, is also matched by p_3. Therefore, $p_1 \succeq p_3$, implying that $\succeq$ is transitive.

Next, we will prove that $\succ$ is transitive using proof by contradiction. Assume that $p_1 \succ p_2$, $p_2 \succ p_3$, but $p_1 \nsucc p_3$. Because $p_1 \succ p_2$, by Definition 9, we get that $p_1 \succeq p_2$ and $p_2 \nsucceq p_1$. Similarly, because $p_2 \succ p_3$, we get that $p_2 \succeq p_3$ and $p_3 \nsucceq p_2$. Due to the transitivity of $\succeq$, we know that $p_1 \succeq p_3$. Now, let us consider $p_1 \nsucc p_3$. It is true iff $p_1 \nsucceq p_3$ or $p_3 \succeq p_1$. Still, we know that $p_1 \nsucceq p_3$ cannot be true, so it must be the case that $p_3 \succeq p_1$. Due to the transitivity of $\succeq$, $p_3 \succeq p_1$ and $p_1 \succeq p_2$ imply that $p_3 \succeq p_2$. However, this contradicts with the result of $p_2 \succ p_3$. With this contradiction, $p_1 \nsucc p_3$ cannot be true. In other words, $p_1 \succ p_3$, implying that $\succ$ is transitive.

□

Next, we extract a QBAF for a trained pattern-based LR model and an input text x. This QBAF will be the core underlying our local explanations. We know that arguments for two patterns that have the "more specific than" relation should have a dialectical relation between them. However, we are uncertain whether the more specific one should be the attacker/supporter or should be attacked/supported.[1] So, we propose two variations of the extracted QBAF which are the top-down QBAF and the bottom-up QBAF.

Definition 10. *Given a trained binary logistic regression model based on the feature patterns $p_1, \dots, p_d$ with the weights $\langle w_1, \dots, w_d, b\rangle$ and an input text x with the binary feature vector $\boldsymbol{f} = [f_1, \dots, f_d]$, the extracted top-down QBAF (TQBAF) and the extracted bottom-up QBAF (BQBAF) are 5-tuples $\langle \mathcal{A}, \mathcal{R}_T^-, \mathcal{R}_T^+, \tau, c\rangle$ and $\langle \mathcal{A}, \mathcal{R}_B^-, \mathcal{R}_B^+, \tau, c\rangle$, respectively, such that:*

- *$\mathcal{A} = \{\alpha_i | f_i = 1\} \cup \{\delta\}$ is the set of arguments where α_i is the argument for the pattern p_i that appears in x, whereas δ is the* default argument, *corresponding to the bias term in the trained model.*
- *$\tau : \mathcal{A} \to [0, \infty)$ is the base score function where $\tau(\alpha_i) = |w_i|$ and $\tau(\delta) = |b|$.*
- *$c : \mathcal{A} \to \{0, 1\}$ is the function mapping an argument to the class it supports. Here, $c(\alpha_i) = 1$ if $w_i \geq 0$; otherwise, $c(\alpha_i) = 0$. Similarly,*

[1] In this chapter, we abuse terminology and use the term 'support' with two meanings: an argument may support a class (by means of the function c in Definition 10 below) or an argument may support another argument in a dialectical sense (relations $\mathcal{R}_T^-$ and $\mathcal{R}_B^-$ in Definition 10 below).

$c(\delta) = 1$ if $b \geq 0$; otherwise, $c(\delta) = 0$.

- *$\mathcal{R}_T^- \subseteq \mathcal{A} \times \mathcal{A}$ is the set of all attacks for TQBAF where*

$$\begin{aligned}\mathcal{R}_T^- = &\{(\alpha_i, \delta) | c(\alpha_i) \neq c(\delta) \wedge \nexists j[\alpha_j \in \mathcal{A} \wedge p_i \succ p_j]\} \cup \\ &\{(\alpha_i, \alpha_j) | c(\alpha_i) \neq c(\alpha_j) \wedge p_i \succ p_j \wedge \nexists k[\alpha_k \in \mathcal{A} \wedge p_i \succ p_k \succ p_j]\}.\end{aligned}$$

- *$\mathcal{R}_T^+ \subseteq \mathcal{A} \times \mathcal{A}$ is the set of all supports for TQBAF where*

$$\begin{aligned}\mathcal{R}_T^+ = &\{(\alpha_i, \delta) | c(\alpha_i) = c(\delta) \wedge \nexists j[\alpha_j \in \mathcal{A} \wedge p_i \succ p_j]\} \cup \\ &\{(\alpha_i, \alpha_j) | c(\alpha_i) = c(\alpha_j) \wedge p_i \succ p_j \wedge \nexists k[\alpha_k \in \mathcal{A} \wedge p_i \succ p_k \succ p_j]\}.\end{aligned}$$

- *$\mathcal{R}_B^- \subseteq \mathcal{A} \times \mathcal{A}$ is the set of all attacks for BQBAF where*

$$\begin{aligned}\mathcal{R}_B^- = &\{(\alpha_i, \delta) | c(\alpha_i) \neq c(\delta) \wedge \nexists j[\alpha_j \in \mathcal{A} \wedge p_j \succ p_i]\} \cup \\ &\{(\alpha_j, \alpha_i) | c(\alpha_i) \neq c(\alpha_j) \wedge p_i \succ p_j \wedge \nexists k[\alpha_k \in \mathcal{A} \wedge p_i \succ p_k \succ p_j]\}.\end{aligned}$$

- *$\mathcal{R}_B^+ \subseteq \mathcal{A} \times \mathcal{A}$ is the set of all supports for BQBAF where*

$$\begin{aligned}\mathcal{R}_B^+ = &\{(\alpha_i, \delta) | c(\alpha_i) = c(\delta) \wedge \nexists j[\alpha_j \in \mathcal{A} \wedge p_j \succ p_i]\} \cup \\ &\{(\alpha_j, \alpha_i) | c(\alpha_i) = c(\alpha_j) \wedge p_i \succ p_j \wedge \nexists k[\alpha_k \in \mathcal{A} \wedge p_i \succ p_k \succ p_j]\}.\end{aligned}$$

To explain, both TQBAF and BQBAF use the same $\mathcal{A}$, τ, and c. If the input text x matches n patterns, $\mathcal{A}$ will have $n + 1$ arguments. Amongst them, n arguments (those of the form α_i) are for the n matched patterns, while the other one is for the default argument (δ) corresponding to the bias term b in the LR model. Therefore, the QBAFs always have at least one argument, which is the default. The supported class (c) of each argument depends on whether the corresponding weight in the LR model is positive or negative. If w_i is positive, it means that the existence of the pattern p_i contributes to the positive class. So, the supported class of α_i should be positive (1). For the default argument, we consider the sign of the bias term b instead. Because the supported class has represented the sign, the base score (τ) of the argument will be only the absolute value of the corresponding weight.

The differences between TQBAF and BQBAF are the $\mathcal{R}^-$ and $\mathcal{R}^+$ com-

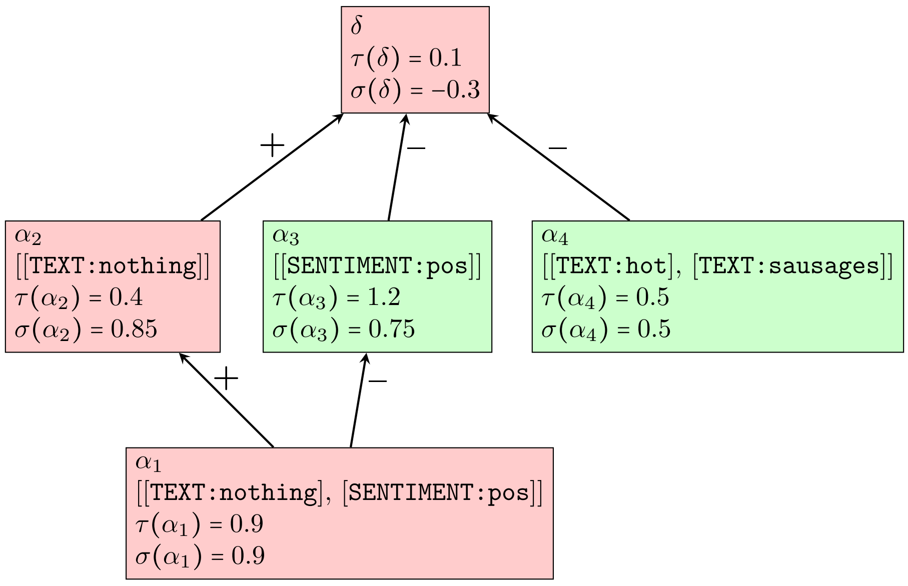


Figure 6.3.: The extracted top-down QBAF for the example in Figure 6.1. Here and everywhere in this chapter we show QBAFs as graphs, with nodes representing the arguments and labelled edges representing attack (-) or support (+). The color of the nodes represents the supported class (i.e., green for positive (1) and red for negative (0)). (The meaning of the equalities of the form $\tau(x) = v$ and $\sigma(x) = v$ will be explained later.)

ponents. For TQBAF, (arguments for) more specific patterns attack or support (arguments for) more general patterns. The most general patterns, in turn, attack or support the default argument. Hence, the more general patterns will stay closer to the default. That is why we call it *top-down*. Conversely, for BQBAF, (arguments for) more general patterns attack or support (arguments for) more specific patterns. The most specific patterns, in turn, attack or support the default argument. Therefore, the more specific patterns will stay closer to the default argument, so we call it *bottom-up*. To decide whether a relation is attack or support, we check the classes both arguments support. If they support the same class, the relation is support; otherwise, it is attack. The extracted TQBAF and BQBAF for the example in Figure 6.1 are shown in Figures 6.3 and 6.4, respectively. We believe that both top-down and bottom-up arrangements may be legitimate, but in different situations. Later, we will show that, in TQBAF, we explain to users with general patterns first and provide more specific patterns as

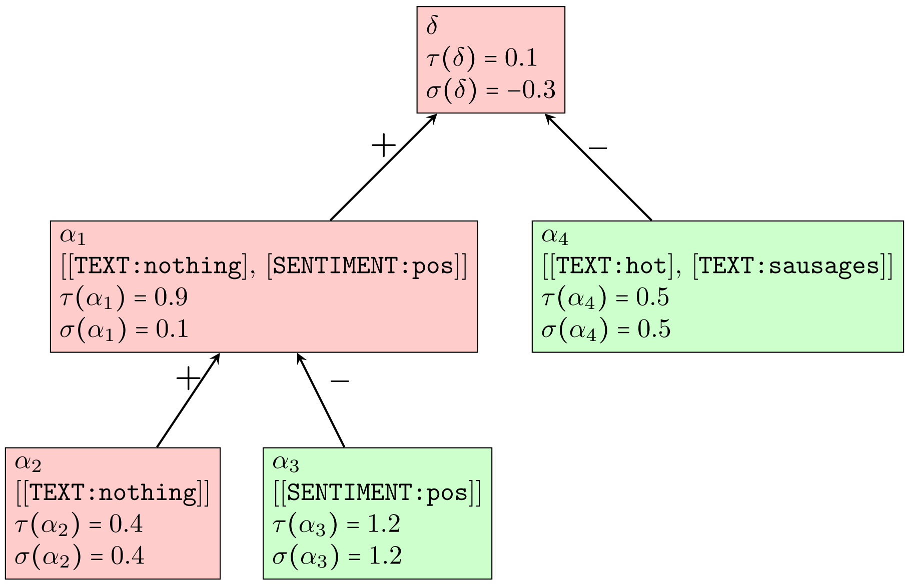


Figure 6.4.: The extracted bottom-up QBAF for the example in Figure 6.1. The color represents the supported class (i.e., green for positive (1) and red for negative (0)). (The meaning of the equalities of the form $\tau(x) = v$ and $\sigma(x) = v$ will be explained later.)

details when requested. In BQBAF, by contrast, we explain to users with specific patterns first (as they contain more information) and mention general patterns as supporting or opposing reasons.

After this point, when we mention a QBAF in this chapter, we mean that it could be either TQBAF or BQBAF, unless otherwise stated.

Lemma 2. *Given a QBAF $\langle \mathcal{A}, \mathcal{R}^-, \mathcal{R}^+, \tau, c \rangle$, then $\delta \notin \mathcal{R}^-(a)$ and $\delta \notin \mathcal{R}^+(a)$ for all $a \in \mathcal{A}$. So, the out-degree of δ is 0.*

Indeed, we can see from $\mathcal{R}_T^-$, $\mathcal{R}_T^+$, $\mathcal{R}_B^-$, and $\mathcal{R}_B^+$ in Definition 10 that δ never attacks or supports any other argument. So, its out-degree equals 0, and therefore we usually put it at the top of the argumentation framework figures (as shown in Figures 6.3 and 6.4).

Next, we show that the graph structures underlying TQBAFs and BQBAFs are directed acyclic graphs (DAGs).

Theorem 1. *The graph structure of TQBAFs (i.e., $\langle \mathcal{A}, \mathcal{R}_T^-, \mathcal{R}_T^+ \rangle$) is a directed acyclic graph (DAG). So is for BQBAFs.*

Proof. Let us consider the TQBAF first. From the graph theory perspective, our graph $\langle\mathcal{A},\mathcal{R}_T^-,\mathcal{R}_T^+\rangle$ is equivalent to $G=\langle V,E\rangle$ where $V=\mathcal{A}$ is a set of vertices and $E=\mathcal{R}_T^-\cup\mathcal{R}_T^+$ is a set of edges. According to Definition 10, we can write E explicitly as

$$\begin{aligned}
E =&\mathcal{R}_T^- \cup \mathcal{R}_T^+\\
E =&\{(\alpha_i,\delta)|c(\alpha_i)\neq c(\delta)\wedge\nexists j[\alpha_j\in\mathcal{A}\wedge p_i\succ p_j]\}\cup\\
&\{(\alpha_i,\alpha_j)|c(\alpha_i)\neq c(\alpha_j)\wedge p_i\succ p_j\wedge\nexists k[\alpha_k\in\mathcal{A}\wedge p_i\succ p_k\succ p_j]\}\cup\\
&\{(\alpha_i,\delta)|c(\alpha_i)=c(\delta)\wedge\nexists j[\alpha_j\in\mathcal{A}\wedge p_i\succ p_j]\}\cup\\
&\{(\alpha_i,\alpha_j)|c(\alpha_i)=c(\alpha_j)\wedge p_i\succ p_j\wedge\nexists k[\alpha_k\in\mathcal{A}\wedge p_i\succ p_k\succ p_j]\}\\
E =&\{(\alpha_i,\delta)|\nexists j[\alpha_j\in\mathcal{A}\wedge p_i\succ p_j]\}\cup\\
&\{(\alpha_i,\alpha_j)|p_i\succ p_j\wedge\nexists k[\alpha_k\in\mathcal{A}\wedge p_i\succ p_k\succ p_j]\}
\end{aligned}$$

We will prove the result by contradiction. Assume that the graph G is not a DAG. Hence, there must be a non-trivial path which forms a cycle in G. Assume that the path is $\alpha_{i_0},\alpha_{i_1},\ldots,\alpha_{i_k},\alpha_{i_0}$ with $k\geq 1$. In this path, there must not be δ since the out-degree of δ is 0 (from Lemma 2). So, every edge in this path must be in the second set of the union above. Hence, we obtain that $p_{i_0}\succ p_{i_1}, p_{i_1}\succ p_{i_2},\ldots,p_{i_k}\succ p_{i_0}$. Because $\succ$ is transitive (from Lemma 1), $p_{i_0}\succ p_{i_0}$, but this is impossible since $\succ$ is not reflexive (also from Lemma 1). Here is the contradiction. Thus, the graph structure of the TQBAF is a DAG.

The proof for BQBAFs is similar to the proof for TQBAFs. We will obtain that

$$\begin{aligned}
E =&\mathcal{R}_B^- \cup \mathcal{R}_B^+\\
E =&\{(\alpha_i,\delta)|\nexists j[\alpha_j\in\mathcal{A}\wedge p_j\succ p_i]\}\cup\\
&\{(\alpha_j,\alpha_i)|p_i\succ p_j\wedge\nexists k[\alpha_k\in\mathcal{A}\wedge p_i\succ p_k\succ p_j]\}
\end{aligned}$$

Assume the directed cyclic path in G is $\alpha_{i_0},\alpha_{i_1},\ldots,\alpha_{i_k},\alpha_{i_0}$ with $k\geq 1$. Hence, $p_{i_0}\succ p_{i_k}$ (from the last edge in the path), $p_{i_k}\succ p_{i_{k-1}}$ (from the second last edge), $\ldots, p_{i_1}\succ p_{i_0}$ (from the first edge). Because $\succ$ is transitive, $p_{i_0}\succ p_{i_0}$, but this is impossible since $\succ$ is not reflexive. Here is the contradiction. Thus, the graph structure of BQBAFs is also a DAG. $\square$

6.2.2. Semantics

After we obtain the QBAFs, the next step is to calculate the dialectical strength of each argument. To make this strength faithful to the underlying LR model, we propose the *logistic regression semantics* with the strength function σ, defined next.

Definition 11. *The strength according to the* logistic regression semantics *is defined as $\sigma : \mathcal{A} \rightarrow \mathbb{R}$, where for any $a \in \mathcal{A}$*

$$\sigma(a) = \tau(a) + \sum_{b \in \mathcal{R}^+(a)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(a)} \frac{\sigma(b)}{\nu(b)} \tag{6.2}$$

where $\tau(a)$ is the base score of a and $\nu(b)$ is the out-degree of b.

This semantics can be applied to both TQBAFs and BQBAFs. According to Equation 6.2, the strength of an argument starts from its base score, and it is increased and decreased by the strengths of its supporters and its attackers, respectively. However, the strength of each supporter/attacker must be divided by its out-degree (i.e., $\nu(b)$) before being combined with the base score. Note that $\nu(b)$ in Equation 6.2 is always greater than or equal to 1 because $b \in \mathcal{R}^-(a)$ or $b \in \mathcal{R}^+(a)$, meaning that b attacks or supports at least one argument (which is a). So, the divide-by-zero never happens with this equation. Additionally, any argument a with no attackers or supporters (i.e., $\mathcal{R}^-(a) = \mathcal{R}^+(a) = \varnothing$) will have the strength equal to its base score, by Definition 11.

Because the QBAFs are DAGs (thanks to Theorem 1), we can use topological sorting to define the order to compute the strengths. As a result, the computational complexity of this semantics applied to the extracted TQBAFs and BQBAFs is $O(|\mathcal{A}|)$, and the worst case happens when $|\mathcal{A}| = d + 1$ (i.e., the input text matches all the patterns). Considering the TQBAF in Figure 6.3, for example, α_1 and α_4 do not have any attacker or supporter, so their strengths equal their base scores. Next, we can calculate the strengths of α_2 and α_3, and then δ.

$$\begin{aligned}
\sigma(\alpha_2) &= \tau(\alpha_2) + \sum_{b \in \mathcal{R}^+(\alpha_2)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha_2)} \frac{\sigma(b)}{\nu(b)} \\
&= 0.4 + \sum_{b \in \{\alpha_1\}} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \varnothing} \frac{\sigma(b)}{\nu(b)} = 0.4 + \frac{0.9}{2} = 0.85 \\
\sigma(\alpha_3) &= \tau(\alpha_3) + \sum_{b \in \mathcal{R}^+(\alpha_3)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha_3)} \frac{\sigma(b)}{\nu(b)} \\
&= 1.2 + \sum_{b \in \varnothing} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \{\alpha_1\}} \frac{\sigma(b)}{\nu(b)} = 1.2 - \frac{0.9}{2} = 0.75 \\
\sigma(\delta) &= \tau(\delta) + \sum_{b \in \mathcal{R}^+(\delta)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\delta)} \frac{\sigma(b)}{\nu(b)} \\
&= 0.1 + \sum_{b \in \{\alpha_2\}} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \{\alpha_3, \alpha_4\}} \frac{\sigma(b)}{\nu(b)} \\
&= 0.1 + \frac{0.85}{1} - \frac{0.75}{1} - \frac{0.5}{1} = -0.3
\end{aligned}$$

All the results are displayed in Figure 6.3. Similarly, the strengths are computed for the BQBAF and shown in Figure 6.4. We can see that the strength of the default arguments δ of both TQBAF and BQBAF is equal to the absolute of the logit $\sum_{i=1}^{d} w_i f_i + b$ of the LR model.

Theorem 2. *For a given QBAF, the prediction of the underlying LR model can be inferred from the strength of the default argument:*

1. *The predicted probability for the class $c(\delta)$ equals $sigmoid(\sigma(\delta))$.*
2. *Hence, if $\sigma(\delta) > 0$, the LR model predicts class $c(\delta)$. Otherwise, it predicts the opposite class (i.e., $1 - c(\delta)$).*

Proof. First, we will prove that the predicted probability for the class $c(\delta)$ equals $sigmoid(\sigma(\delta))$. In other words, we need to prove that, for $c(\delta) = 1$, $sigmoid(\sigma(\delta)) = sigmoid(\sum_{i=1}^{d} w_i f_i + b)$, i.e., $\sigma(\delta) = \sum_{i=1}^{d} w_i f_i + b$. Also, we need to prove that, for $c(\delta) = 0$, $sigmoid(\sigma(\delta)) = 1 - sigmoid(\sum_{i=1}^{d} w_i f_i + b) = sigmoid(-\sum_{i=1}^{d} w_i f_i - b)$, i.e., $\sigma(\delta) = -\sum_{i=1}^{d} w_i f_i - b$.[2]

Case 1: $c(\delta) = 1$ – The class supported by δ is Positive.

[2] $1 - sigmoid(x) = sigmoid(-x)$.

Each argument $\alpha_i \in \mathcal{A} - \{\delta\}$ supports class $c(\alpha_i) \in \{0,1\}$. We partition $\mathcal{A} - \{\delta\}$ into two sets – one with Positive (1) as the supported class and the other with Negative (0) as the supported class. We use $\mathcal{A}^+$ and $\mathcal{A}^-$ to represent the two sets, respectively.

Applying Definition 11 to δ, we obtain

$$\sigma(\delta) = \tau(\delta) + \sum_{g \in \mathcal{R}^+(\delta)} \frac{\sigma(g)}{\nu(g)} - \sum_{g \in \mathcal{R}^-(\delta)} \frac{\sigma(g)}{\nu(g)}$$

Because $c(\delta) = 1$, we know from Definition 10 that the bias term of the underlying LR model is $b \geq 0$. Hence, $\tau(\delta) = |b| = b$. Furthermore, $\mathcal{R}^+(\delta) \subseteq \mathcal{A}^+$ and $\mathcal{R}^-(\delta) \subseteq \mathcal{A}^-$. So, we obtain that

$$b = \sigma(\delta) - \sum_{g \in \mathcal{R}^+(\delta)} \frac{\sigma(g)}{\nu(g)} + \sum_{g \in \mathcal{R}^-(\delta)} \frac{\sigma(g)}{\nu(g)} \tag{6.3}$$

Next, for α_i in $\mathcal{A}^+$, we know that $c(\alpha_i) = 1$, so the corresponding weight in the LR model is $w_i \geq 0$. Hence, $\tau(\alpha_i) = |w_i| = w_i$. Again, $\mathcal{R}^+(\alpha_i) \subseteq \mathcal{A}^+$ and $\mathcal{R}^-(\alpha_i) \subseteq \mathcal{A}^-$. By Definition 11,

$$w_i = \tau(\alpha_i) = \sigma(\alpha_i) - \sum_{g \in \mathcal{R}^+(\alpha_i)} \frac{\sigma(g)}{\nu(g)} + \sum_{g \in \mathcal{R}^-(\alpha_i)} \frac{\sigma(g)}{\nu(g)} \tag{6.4}$$

For $\alpha_j \in \mathcal{A}^-$, in contrast, we know that $c(\alpha_j) = 0$, so the corresponding weight in the LR model is $w_j < 0$. Hence, $\tau(\alpha_j) = |w_j| = -w_j$. By Definition 10, all the supporters must support the same class whereas all the attackers must support the opposite class. So, $\mathcal{R}^+(\alpha_j) \subseteq \mathcal{A}^-$ and $\mathcal{R}^-(\alpha_j) \subseteq \mathcal{A}^+$. By Definition 11,

$$w_j = -\tau(\alpha_j) = -\sigma(\alpha_j) + \sum_{g \in \mathcal{R}^+(\alpha_j)} \frac{\sigma(g)}{\nu(g)} - \sum_{g \in \mathcal{R}^-(\alpha_j)} \frac{\sigma(g)}{\nu(g)} \tag{6.5}$$

By Definition 10, $\mathcal{A} - \{\delta\} = \mathcal{A}^+ \cup \mathcal{A}^- = \{\alpha_i | f_i = 1\}$, so $\sum_{i=1}^{d} w_i f_i + b = \sum_{\alpha_i \in \mathcal{A}^+} w_i + \sum_{\alpha_j \in \mathcal{A}^-} w_j + b$. By summing up Equations 6.3, 6.4 (for all $\alpha_i \in \mathcal{A}^+$), and 6.5

(for all $\alpha_i \in \mathcal{A}^-$), we obtain that

$$\begin{aligned}\sum_{i=1}^{d} w_i f_i + b = \sigma(\delta) &+ \sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i) - \sum_{\alpha_j \in \mathcal{A}^-} \sigma(\alpha_j) \\ &- \sum_{g \in \mathcal{R}^+(\delta)} \frac{\sigma(g)}{\nu(g)} + \sum_{g \in \mathcal{R}^-(\delta)} \frac{\sigma(g)}{\nu(g)} \\ &+ \sum_{\alpha_i \in \mathcal{A}^+} \left(- \sum_{g \in \mathcal{R}^+(\alpha_i)} \frac{\sigma(g)}{\nu(g)} + \sum_{g \in \mathcal{R}^-(\alpha_i)} \frac{\sigma(g)}{\nu(g)} \right) \\ &+ \sum_{\alpha_j \in \mathcal{A}^-} \left(\sum_{g \in \mathcal{R}^+(\alpha_j)} \frac{\sigma(g)}{\nu(g)} - \sum_{g \in \mathcal{R}^-(\alpha_j)} \frac{\sigma(g)}{\nu(g)} \right)\end{aligned}$$

$$\begin{aligned}\sum_{i=1}^{d} w_i f_i + b = \sigma(\delta) &+ \sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i) - \sum_{\alpha_j \in \mathcal{A}^-} \sigma(\alpha_j) \\ &- \left(\sum_{g \in \mathcal{R}^+(\delta)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_i \in \mathcal{A}^+} \sum_{g \in \mathcal{R}^+(\alpha_i)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_j \in \mathcal{A}^-} \sum_{g \in \mathcal{R}^-(\alpha_j)} \frac{\sigma(g)}{\nu(g)} \right) \\ &+ \left(\sum_{g \in \mathcal{R}^-(\delta)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_i \in \mathcal{A}^+} \sum_{g \in \mathcal{R}^-(\alpha_i)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_j \in \mathcal{A}^-} \sum_{g \in \mathcal{R}^+(\alpha_j)} \frac{\sigma(g)}{\nu(g)} \right)\end{aligned}$$

Next, we will show that $\sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i)$ and the blue part above are equal. As $\alpha_i \in \mathcal{A}^+$, it can appear as g only in the blue part. In other words, α_i can either support another argument in $\mathcal{A}^+$ or δ or attack another argument in $\mathcal{A}^-$. If α_i supports or attacks $\nu(\alpha_i)$ arguments in total (where $\nu(\alpha_i)$ is the out-degree of α_i), we will find exactly $\nu(\alpha_i)$ terms of $\frac{\sigma(\alpha_i)}{\nu(\alpha_i)}$ in the blue part, and they sum up to $\sigma(\alpha_i)$. Hence, for every $\sigma(\alpha_i)$ in $\sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i)$, we can find the equivalent amount in the blue part. Meanwhile, g in the blue part must come from $\mathcal{A}^+$ only (not δ or $\mathcal{A}^-$ according to Definition 10). Thereby, $\sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i)$ and the blue part are equal and cancelling each other.

Similarly, $\alpha_j \in \mathcal{A}^-$ can either support another argument in $\mathcal{A}^-$ or attack another argument in $\mathcal{A}^+$ or δ. With the same logic as for the blue part, we obtain that $\sum_{\alpha_j \in \mathcal{A}^-} \sigma(\alpha_j)$ and the magenta part are equal and cancelling each other.

Finally, we obtain $\sum_{i=1}^{d} w_i f_i + b = \sigma(\delta)$ as required.

Case 2: $c(\delta) = 0$ – The class supported by δ is Negative.

The proof of this case is similar to the previous case, so we will highlight only the differences here. First, because $c(\delta) = 0$, the bias term of the LR model is $b < 0$. Hence, $\tau(\delta) = |b| = -b$. Equation 6.3 then becomes

$$b = -\sigma(\delta) + \sum_{g \in \mathcal{R}^+(\delta)} \frac{\sigma(g)}{\nu(g)} - \sum_{g \in \mathcal{R}^-(\delta)} \frac{\sigma(g)}{\nu(g)} \tag{6.6}$$

However, Equations 6.4 and 6.5 remain the same. By summing up Equations 6.6, 6.4 (for all $\alpha_i \in \mathcal{A}^+$), and 6.5 (for all $\alpha_i \in \mathcal{A}^-$), we obtain that

$$\begin{aligned}\sum_{i=1}^{d} w_i f_i + b = -\sigma(\delta) + \sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i) - \sum_{\alpha_j \in \mathcal{A}^-} \sigma(\alpha_j) \\ - \left(\sum_{g \in \mathcal{R}^-(\delta)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_i \in \mathcal{A}^+} \sum_{g \in \mathcal{R}^+(\alpha_i)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_j \in \mathcal{A}^-} \sum_{g \in \mathcal{R}^-(\alpha_j)} \frac{\sigma(g)}{\nu(g)} \right) \\ + \left(\sum_{g \in \mathcal{R}^+(\delta)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_i \in \mathcal{A}^+} \sum_{g \in \mathcal{R}^-(\alpha_i)} \frac{\sigma(g)}{\nu(g)} + \sum_{\alpha_j \in \mathcal{A}^-} \sum_{g \in \mathcal{R}^+(\alpha_j)} \frac{\sigma(g)}{\nu(g)} \right)\end{aligned}$$

Because $\alpha_i \in \mathcal{A}^+$ can either support another argument in $\mathcal{A}^+$ or attack another argument in $\mathcal{A}^-$ or δ, with the same logic as in the previous case, we obtain that $\sum_{\alpha_i \in \mathcal{A}^+} \sigma(\alpha_i)$ and the blue part are equal and cancelling each other. Similarly, $\sum_{\alpha_j \in \mathcal{A}^-} \sigma(\alpha_j)$ and the magenta part are equal and cancelling each other. What remains is $\sum_{i=1}^{d} w_i f_i + b = -\sigma(\delta)$. So, $\sigma(\delta) = -\sum_{i=1}^{d} w_i f_i - b$ as required.

Finally, the second point of the theorem is pretty obvious. Using the result from the first point, the predicted probability of class $c(\delta)$ equals $sigmoid(\sigma(\delta))$ which is greater than 0.5 if $\sigma(\delta) > 0$. So, it predicts $c(\delta)$. Otherwise, $sigmoid(\sigma(\delta)) < 0.5$, and it predicts the other class which is $1 - c(\delta)$. $\square$

In other words, we can read the prediction from the default argument δ, and this proves the faithfulness[3] of our QBAF and the logistic regression semantics to the underlying LR model. The negative strength of δ implies that the argument can no longer support its supported class; therefore, the prediction must be the opposite class. Since $\sigma(\delta)$ is computed from $\tau(\delta)$ and the strengths of the attackers and the supporters of δ, we can use these attackers and supporters as explanation for the prediction. Furthermore, we may generalize the results of Theorem 2 to other arguments $\alpha_i \in \mathcal{A}$.

[3]See the definition of faithfulness in Section 3.2.1, Definition 7.

For instance, in Figure 6.4, we could say that the pattern [[`TEXT:nothing`], [`SENTIMENT:pos`]] of α_1 (weakly) supports the negative class with a strength of 0.1, but it is not sufficient to make the prediction become negative.

6.2.3. Post-Processing

We know from the previous section that we can extract an explanation from the QBAF and the dialectical strengths. Nevertheless, the negative final strengths may render the human interpretation of these explanations difficult. Using Figure 6.3 as an example, we can see that argument α_2 ([[`TEXT:nothing`]]), supporting the negative class, *supports* argument δ, which represents the final prediction. However, the prediction now is the positive class due to the negative $\sigma(\delta)$. So, it is counterintuitive to say that a pattern for the negative class supports the prediction of the positive class. Hence, we propose a post-processing step for QBAF to make the explanations (to be generated) align better with human interpretation.

Definition 12. *Given a QBAF $\langle \mathcal{A}, \mathcal{R}^-, \mathcal{R}^+, \tau, c \rangle$ with the calculated strength $\sigma(a)$ for every $a \in \mathcal{A}$, the corresponding post-processed QBAF (QBAF') is defined as $\langle \mathcal{A}', \mathcal{R}^{-\prime}, \mathcal{R}^{+\prime}, \tau', c' \rangle$ where*

- $\mathcal{A}' = \mathcal{A}$.
- $\tau' : \mathcal{A} \to \mathbb{R}$ *and* $c' : \mathcal{A} \to \{0, 1\}$ *are defined such that, for each* $a \in \mathcal{A}$,
  - *If* $\sigma(a) \geq 0$, *then* $\tau'(a) = \tau(a)$ *and* $c'(a) = c(a)$.
  - *If* $\sigma(a) < 0$, $\tau'(a) = -\tau(a)$ *and* $c'(a) = 1 - c(a)$.
- $\mathcal{R}^{-\prime} = \{(a, b) \in \mathcal{R}^- \cup \mathcal{R}^+ | c'(a) \neq c'(b) \wedge \sigma(a) \neq 0\}$.
- $\mathcal{R}^{+\prime} = \{(a, b) \in \mathcal{R}^- \cup \mathcal{R}^+ | c'(a) = c'(b) \wedge \sigma(a) \neq 0\}$.

Theorem 3. *Given a QBAF $\langle \mathcal{A}, \mathcal{R}^-, \mathcal{R}^+, \tau, c \rangle$ and the corresponding QBAF' $\langle \mathcal{A}', \mathcal{R}^{-\prime}, \mathcal{R}^{+\prime}, \tau', c' \rangle$, using the logistic regression semantics, we use $\sigma(a)$ and $\sigma(a)'$ to represent the strengths of $a \in \mathcal{A} = \mathcal{A}'$ in QBAF and QBAF', respectively. The following statements are true for $a \in \mathcal{A} = \mathcal{A}'$.*

- *If* $\sigma(a) \geq 0$, *then* $\sigma(a)' = \sigma(a)$.
- *If* $\sigma(a) < 0$, $\sigma(a)' = -\sigma(a)$.

Proof. According to Theorem 1, $G = \langle V, E\rangle$ for the QBAF is a DAG where $V = \mathcal{A}$ and $E = \mathcal{R}^- \cup \mathcal{R}^+$. Let $G' = \langle V', E'\rangle$ be the graph structure of QBAF' where $V' = \mathcal{A}'$ and $E' = \mathcal{R}^{-\prime} \cup \mathcal{R}^{+\prime}$. By Definition 12, we know that $V' = \mathcal{A}' = \mathcal{A} = V$ and $E' = \mathcal{R}^{-\prime} \cup \mathcal{R}^{+\prime} \subseteq \mathcal{R}^- \cup \mathcal{R}^+ = E$. Hence, G' is a subgraph of G and also a DAG[4].

Then we can obtain the topological ordering t of arguments in V' which is the ordering of strength computation for QBAF'. Assume that the order t is $a_1, a_2, \dots, a_k$. Obviously, a_1 must be an argument in V' that has no attack or support. Furthermore, we can divide the vertices in t into two groups (corresponding to the two bullet points of this theorem), one with $\sigma(a_i) \geq 0$ and the other with $\sigma(a_i) < 0$. We name arguments in the former group and the latter group as $g_1, \dots, g_r$ and $l_1, \dots, l_s$, respectively, where g_i must be before g_{i+1} in t, l_i must be before l_{i+1} in t, and $r + s = k$. So, t could be written as, for example, $g_1, g_2, l_1, g_3, l_2, l_3, \dots, g_r, l_{s-1}, l_s$. In any case, g_1 must be a_1 since a_1 has neither attacker nor supporter and, therefore, $\sigma(a_1) = \tau(a_1) \geq 0$. In general, the first part of t must be $g_1, \dots, g_{r^*}, l_1, \dots$ where $1 \leq r^* \leq r$, and arguments after l_1 could be from either g or l. We will use mathematical induction on this topological ordering t to prove the two bullet points of this theorem.

- For g_1 (the base case): Because it has neither attacker nor supporter, $\sigma(g_1) = \tau(g_1)$ and $\sigma(g_1)' = \tau'(g_1)$. Since $\sigma(g_1) \geq 0$, by Definition 12, $\tau'(g_1) = \tau(g_1)$. Hence, $\sigma(g_1)' = \sigma(g_1)$ as required.

- For g_i with $i \leq r^*$ (using strong induction): We have shown that $\sigma(g_1)' = \sigma(g_1)$. Next, assuming that $\sigma(g_h)' = \sigma(g_h)$ for $1 \leq h \leq i < r^*$, we need to show that $\sigma(g_{i+1})' = \sigma(g_{i+1})$ where $i + 1 \leq r^*$.

  Due to the topological ordering, all the original attackers and supporters of g_{i+1} must be in $\{g_h | 1 \leq h \leq i\}$. As a result, for $a \in \mathcal{R}^-(g_{i+1}) \cup \mathcal{R}^+(g_{i+1})$, $\tau'(a) = \tau(a)$, $c'(a) = c(a)$, and $\sigma(a)' = \sigma(a)$. Also, $\tau'(g_{i+1}) = \tau(g_{i+1})$ and $c'(g_{i+1}) = c(g_{i+1})$ because $\sigma(g_{i+1}) \geq 0$ by the definition of g. Since the classes of both g_{i+1} and its original attackers and supporters do not change, $\mathcal{R}^{-\prime}(g_{i+1}) = \mathcal{R}^-(g_{i+1})$ and $\mathcal{R}^{+\prime}(g_{i+1}) = \mathcal{R}^+(g_{i+1})$[5].

[4]A subgraph of a DAG must be a DAG, since it cannot contain a cycle that does not exist in the supergraph.

[5]For simplicity, we include the attackers and supporters a where $\sigma(a) = 0$ in $\mathcal{R}^{-\prime}(g_{i+1})$

Applying Definition 11 to g_{i+1} in QBAF', we obtain

$$\begin{aligned}\sigma(g_{i+1})' &= \tau'(g_{i+1}) + \sum_{b\in\mathcal{R}^{+\prime}(g_{i+1})} \frac{\sigma(b)'}{\nu(b)} - \sum_{b\in\mathcal{R}^{-\prime}(g_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\ &= \tau(g_{i+1}) + \sum_{b\in\mathcal{R}^{+}(g_{i+1})} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^{-}(g_{i+1})} \frac{\sigma(b)}{\nu(b)} \\ &= \sigma(g_{i+1})\end{aligned}$$

Hence, $\sigma(g_{i+1})' = \sigma(g_{i+1})$ where $i+1 \le r^*$ as required.

- For l_1: Because $\sigma(l_1) < 0$ by the definition of l, we need to show that $\sigma(l_1)' = -\sigma(l_1)$.

  According to the ordering t, all the original attackers and supporters of l_1 must be in $\{g_h | 1 \le h \le r^*\}$. As a result, for $a \in \mathcal{R}^-(l_1) \cup \mathcal{R}^+(l_1)$, $\tau'(a) = \tau(a)$, $c'(a) = c(a)$, and $\sigma(a)' = \sigma(a)$ (as proven above). In contrast, since $\sigma(l_1) < 0$, by Definition 12, $\tau'(l_1) = -\tau(l_1)$ and $c'(l_1) = 1 - c(l_1)$. In other words, the base score and the supported class of l_1 are flipped after post-processing. Because the classes of the attackers and supporters are not change whereas the class of l_1 is flipped, $\mathcal{R}^{-\prime}(l_1) = \mathcal{R}^+(l_1)$ and $\mathcal{R}^{+\prime}(l_1) = \mathcal{R}^-(l_1)$.

  Applying Definition 11 to l_1 in QBAF', we obtain

$$\begin{aligned}\sigma(l_1)' &= \tau'(l_1) + \sum_{b\in\mathcal{R}^{+\prime}(l_1)} \frac{\sigma(b)'}{\nu(b)} - \sum_{b\in\mathcal{R}^{-\prime}(l_1)} \frac{\sigma(b)'}{\nu(b)} \\ &= -\tau(l_1) + \sum_{b\in\mathcal{R}^{-}(l_1)} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^{+}(l_1)} \frac{\sigma(b)}{\nu(b)} \\ &= -\left(\tau(l_1) + \sum_{b\in\mathcal{R}^{+}(l_1)} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^{-}(l_1)} \frac{\sigma(b)}{\nu(b)}\right) \\ &= -\sigma(l_1)\end{aligned}$$

  Hence, $\sigma(l_1)' = -\sigma(l_1)$ as required.

- For any argument a_i in t after l_1 (using strong induction): So far, we have shown that this theorem is true for $g_1, \ldots, g_{r^*}$ and l_1. Next, assuming that the theorem is true for any argument $a_1, \ldots, a_i$, we need

and $\mathcal{R}^{+\prime}(g_{i+1})$, respectively, as they play no role when computing $\sigma(g_{i+1})'$ anyway. The same logic also applies to the next cases in this proof.

to show that the theorem is also true for a_{i+1} regardless of the group it belongs to.

If a_{i+1} belongs to the g group (i.e., $\sigma(a_{i+1}) \geq 0$), we obtain that $\tau'(a_{i+1}) = \tau(a_{i+1})$ and $c'(a_{i+1}) = c(a_{i+1})$. The supported class of a_{i+1} is not flipped after post-processing. The original attackers of a_{i+1} could belong to either the g group or the l group. We use $\mathcal{R}_g^-(a_{i+1})$ and $\mathcal{R}_l^-(a_{i+1})$ to represent those sets of original attackers, respectively. After post-processing, the attackers in $\mathcal{R}_g^-(a_{i+1})$ will still be attackers (due to the unchanged supported classes on both sides of the relations) whereas the ones in $\mathcal{R}_l^-(a_{i+1})$ will become supporters (due to the supported class flipped only on one side). Similarly, the original supporters of a_{i+1} could be split into $\mathcal{R}_g^+(a_{i+1})$ and $\mathcal{R}_l^+(a_{i+1})$. After post-processing, those in $\mathcal{R}_g^+(a_{i+1})$ will still be supporters while those in $\mathcal{R}_l^+(a_{i+1})$ will become attackers. To sum up, $\mathcal{R}^{+\prime}(a_{i+1})$ is the union of two disjoint sets – $\mathcal{R}_g^+(a_{i+1})$ and $\mathcal{R}_l^-(a_{i+1})$. Meanwhile, $\mathcal{R}^{-\prime}(a_{i+1})$ is the union of two disjoint sets – $\mathcal{R}_g^-(a_{i+1})$ and $\mathcal{R}_l^+(a_{i+1})$.

Applying Definition 11 to a_{i+1} in QBAF', we obtain

$$\begin{aligned}
\sigma(a_{i+1})' &= \tau'(a_{i+1}) + \sum_{b\in\mathcal{R}^{+\prime}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} - \sum_{b\in\mathcal{R}^{-\prime}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\
&= \tau(a_{i+1}) + \sum_{b\in\mathcal{R}_g^{+}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} + \sum_{b\in\mathcal{R}_l^{-}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\
&\quad - \sum_{b\in\mathcal{R}_g^{-}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} - \sum_{b\in\mathcal{R}_l^{+}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\
&= \tau(a_{i+1}) + \sum_{b\in\mathcal{R}_g^{+}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}_l^{-}(a_{i+1})} \frac{-\sigma(b)}{\nu(b)} \\
&\quad - \sum_{b\in\mathcal{R}_g^{-}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}_l^{+}(a_{i+1})} \frac{-\sigma(b)}{\nu(b)} \\
&= \tau(a_{i+1}) + \left(\sum_{b\in\mathcal{R}_g^{+}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}_l^{+}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \right) \\
&\quad - \left(\sum_{b\in\mathcal{R}_g^{-}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}_l^{-}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \right) \\
&= \tau(a_{i+1}) + \sum_{b\in\mathcal{R}^{+}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^{-}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \\
&= \sigma(a_{i+1})
\end{aligned}$$

Hence, for a_{i+1} where $\sigma(a_{i+1}) \geq 0$, $\sigma(a_{i+1})' = \sigma(a_{i+1})$ as the theorem stated.

Analogously, if a_{i+1} belongs to the l group (i.e., $\sigma(a_{i+1}) < 0$), we obtain that $\tau'(a_{i+1}) = -\tau(a_{i+1})$ and $c'(a_{i+1}) = 1 - c(a_{i+1})$. After post-processing, the supported class of a_{i+1} is flipped. Also, the attackers in $\mathcal{R}_g^{-}(a_{i+1})$ will become supporters (due to the supported class flipped only on one side of the relations at a_{i+1}) whereas the ones in $\mathcal{R}_l^{-}(a_{i+1})$ will still be attackers (due to the supported class flipped on both sides). Similarly, the supporters in $\mathcal{R}_g^{+}(a_{i+1})$ will become attackers, while those in $\mathcal{R}_l^{+}(a_{i+1})$ will still be supporters. To sum up, $\mathcal{R}^{+\prime}(a_{i+1})$ is the union of two disjoint sets – $\mathcal{R}_l^{+}(a_{i+1})$ and $\mathcal{R}_g^{-}(a_{i+1})$. Meanwhile, $\mathcal{R}^{-\prime}(a_{i+1})$ is the union of two disjoint sets – $\mathcal{R}_l^{-}(a_{i+1})$ and $\mathcal{R}_g^{+}(a_{i+1})$.

Applying Definition 11 to a_{i+1} in QBAF', we obtain

$$\begin{aligned}
\sigma(a_{i+1})' &= \tau'(a_{i+1}) + \sum_{b\in\mathcal{R}^{+\prime}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} - \sum_{b\in\mathcal{R}^{-\prime}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\
&= -\tau(a_{i+1}) + \sum_{b\in\mathcal{R}^{+}_{l}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} + \sum_{b\in\mathcal{R}^{-}_{g}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\
&\quad - \sum_{b\in\mathcal{R}^{-}_{l}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} - \sum_{b\in\mathcal{R}^{+}_{g}(a_{i+1})} \frac{\sigma(b)'}{\nu(b)} \\
&= -\tau(a_{i+1}) + \sum_{b\in\mathcal{R}^{+}_{l}(a_{i+1})} \frac{-\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}^{-}_{g}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \\
&\quad - \sum_{b\in\mathcal{R}^{-}_{l}(a_{i+1})} \frac{-\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^{+}_{g}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \\
&= -\tau(a_{i+1}) - \left(\sum_{b\in\mathcal{R}^{+}_{g}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}^{+}_{l}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \right) \\
&\quad + \left(\sum_{b\in\mathcal{R}^{-}_{g}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}^{-}_{l}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \right) \\
&= -\tau(a_{i+1}) - \sum_{b\in\mathcal{R}^{+}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} + \sum_{b\in\mathcal{R}^{-}(a_{i+1})} \frac{\sigma(b)}{\nu(b)} \\
&= -\sigma(a_{i+1})
\end{aligned}$$

Hence, for a_{i+1} where $\sigma(a_{i+1}) < 0$, $\sigma(a_{i+1})' = -\sigma(a_{i+1})$ as the theorem stated.

From both cases, the induction step is completed.

Our proof has covered all the arguments in the topological ordering t. Thus, the theorem is true for $a \in \mathcal{A} = \mathcal{A}'$. □

Corollary 1. *Given a QBAF and the corresponding QBAF', $\sigma(a)' = |\sigma(a)|$ for all $a \in \mathcal{A} = \mathcal{A}'$.*

Corollary 2. *Theorem 2 also applies to QBAF'.*

To explain, the goal of the post-processing step is to flip all the negative strengths to be positive, so we adjust the QBAF accordingly, while preserving the interpretations of the arguments. For instance, if the original argument a has $\tau(a) = 0.3$, $c(a) = 1$ and $\sigma(a) = -0.5$, the meaning is that

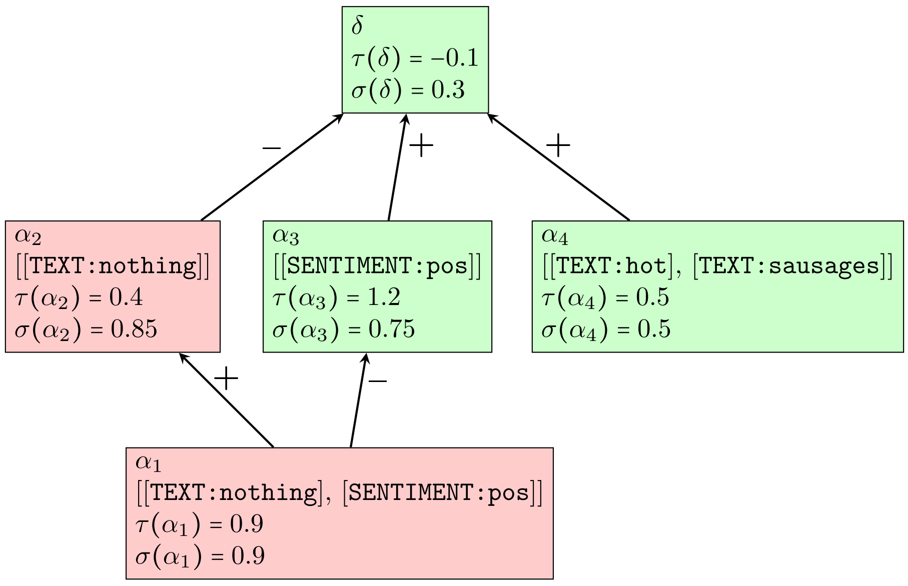


Figure 6.5.: The extracted top-down QBAF in Figure 6.3 after being post-processed.

the argument initially supports the positive class with the base score of 0.3, but after taking into account dialectical relations, it supports the negative class instead with the strength of 0.5. After post-processing, we will obtain $\tau'(a) = -0.3$, $c'(a) = 0$ and $\sigma(a)' = 0.5$ with the meaning that the argument supports the negative class with the strength of 0.5 though previously it supported the opposite class with the base score of 0.3. We can see that the two meanings are equivalent. In addition, with Corollary 2, the QBAF' with the logistic regression semantics is faithful to the underlying LR model since we can read the model prediction directly from $\sigma(\delta)'$.

Note that, during the post-processing step, we remove any edges where the strengths of the attackers or the supporters equal 0 since they no longer attack or support. Then we re-label attacks and supports to the remaining relations according to the new supported classes c' while keeping the direction of the edges intact. In conclusion, the original QBAF started from the non-negative base scores, whereas the post-processed QBAF enforces the non-negative strengths and the rest is adjusted accordingly.

Figures 6.5 and 6.6 show the post-processed QBAFs of Figures 6.3 and 6.4, respectively. We can see that all the strengths become positive now.

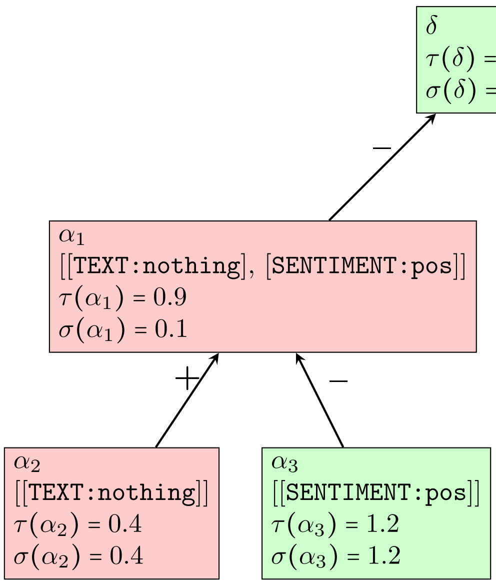


Figure 6.6.: The extracted bottom-up QBAF in Figure 6.4 after being post-processed.

GP	1	2	3	4	5	6	7	8	9	10	11
$\langle QBAF, \sigma \rangle$	✔	✘	✘	✘	✘	✔	✘	✘	✔	✘	✘
$\langle QBAF', \sigma \rangle$	✔	✔	✔	✔	✔	✔	✔	✔	✔	✘	✘

Table 6.2.: Summary of the group properties for gradual semantics (Baroni et al., 2019) satisfied or unsatisfied by the logistic regression semantics σ when applied on QBAF and QBAF'.

6.2.4. Analyzing Properties

In this section, we will analyze the properties of the logistic regression semantics σ when applied to QBAF and QBAF' according to 11 group properties of gradual semantics proposed by Baroni et al. (2019) (as introduced earlier in Section 2.3.3). These properties have been used to evaluate many argumentation frameworks and semantics in the literature (Albini et al., 2020; Potyka, 2021; Sukpanichnant et al., 2021) in order to ensure that the resulting frameworks and dialectical strengths will lead to explanations that are consistent with general human reasoning and debate. Table 6.2 summarizes the results of our analysis. When conducting the proofs, we may use the QBAF and the corresponding QBAF' in Figures 6.7 and 6.8, respectively, as a counterexample.

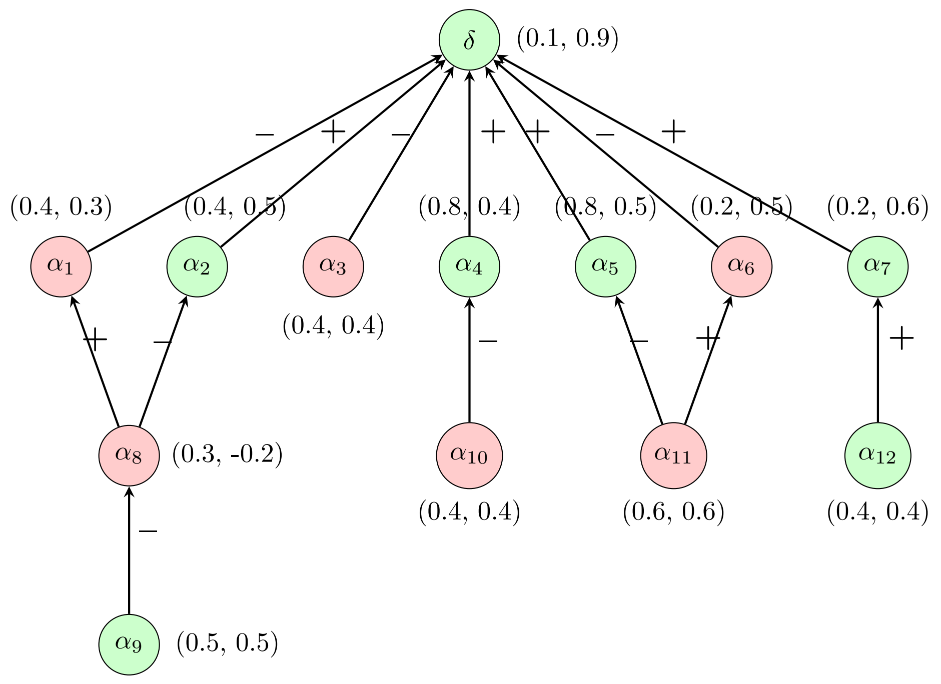


Figure 6.7.: Example of QBAF used as a counterexample in Section 6.2.4. With each argument, there is a value pair (x, y) where x and y represent the base score and the strength (based on the logistic regression semantics σ) of the argument, respectively. The color represents the argument's supported class (i.e., green for positive (1) and red for negative (0)).

GP1. If $\mathcal{R}^-(\alpha) = \varnothing$ and $\mathcal{R}^+(\alpha) = \varnothing$, then $\sigma(\alpha) = \tau(\alpha)$.

Proof. According to Definition 11, with $\mathcal{R}^-(\alpha) = \varnothing$ and $\mathcal{R}^+(\alpha) = \varnothing$, we have $\sigma(\alpha) = \tau(\alpha) + \sum_{b\in\varnothing} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\varnothing} \frac{\sigma(b)}{\nu(b)} = \tau(\alpha)$ as required.

Hence, both $\langle QBAF, \sigma\rangle$ and $\langle QBAF', \sigma\rangle$ satisfy GP1. $\square$

GP2. If $\mathcal{R}^-(\alpha) \neq \varnothing$ and $\mathcal{R}^+(\alpha) = \varnothing$, then $\sigma(\alpha) < \tau(\alpha)$.

Proof. According to Definition 11, with $\mathcal{R}^+(\alpha) = \varnothing$, we have

$$\sigma(\alpha) = \tau(\alpha) + \sum_{b\in\varnothing} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} = \tau(\alpha) - \sum_{b\in\mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)}.$$

In QBAF, $\sigma(b)$ could be either positive or negative, so it is possible that $\sum_{b\in\mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} < 0$, making $\sigma(\alpha) > \tau(\alpha)$. α_2 in Figure 6.7 is one counterexample of this GP. By contrast, in QBAF', we have removed any attacker and supporter of which the strength is zero. According to this and Corollary 1, $\sigma(b) > 0$ for $b \in \mathcal{R}^-(\alpha)$. Therefore, $\sum_{b\in\mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} > 0$, resulting in $\sigma(\alpha) < \tau(\alpha)$ as required.

Hence, $\langle QBAF, \sigma\rangle$ does not satisfy GP2, but $\langle QBAF', \sigma\rangle$ does. □

GP3. If $\mathcal{R}^-(\alpha) = \varnothing$ and $\mathcal{R}^+(\alpha) \neq \varnothing$, then $\sigma(\alpha) > \tau(\alpha)$.

Proof. According to Definition 11, with $\mathcal{R}^-(\alpha) = \varnothing$, we have

$$\sigma(\alpha) = \tau(\alpha) + \sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\varnothing} \frac{\sigma(b)}{\nu(b)} = \tau(\alpha) + \sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)}.$$

As in the proof of GP2, for QBAF, $\sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)}$ could be negative, rendering $\sigma(\alpha) < \tau(\alpha)$. α_1 in Figure 6.7 is one counterexample of this GP. Meanwhile, in QBAF', $\sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} > 0$. Therefore, $\sigma(\alpha) > \tau(\alpha)$ as required.

Hence, $\langle QBAF, \sigma\rangle$ does not satisfy GP3, but $\langle QBAF', \sigma\rangle$ does. □

GP4. If $\sigma(\alpha) < \tau(\alpha)$, then $\mathcal{R}^-(\alpha) \neq \varnothing$.

Proof. From $\sigma(\alpha) < \tau(\alpha)$ with Definition 11, we obtain that

$$\sigma(\alpha) = \tau(\alpha) + \sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} < \tau(\alpha).$$

Therefore,

$$\sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b\in\mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} < 0.$$

For QBAF, it is possible that $\mathcal{R}^-(\alpha) = \varnothing$ because $\sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)}$ could be negative, satisfying the inequality. So, the property is not satisfied. α_1 in Figure 6.7 is one counterexample of this GP. In contrast, for QBAF', let us proof by contradiction. Assume that $\mathcal{R}^-(\alpha) = \varnothing$. We will obtain $\sum_{b\in\mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} < 0$ which is impossible under QBAF' where $\sigma(b) > 0$. Therefore, it must be the case that $\mathcal{R}^-(\alpha) \neq \varnothing$.

Hence, $\langle QBAF, \sigma\rangle$ does not satisfy GP4, but $\langle QBAF', \sigma\rangle$ does. □

GP5. If $\sigma(\alpha) > \tau(\alpha)$, then $\mathcal{R}^+(\alpha) \neq \varnothing$.

Proof. From $\sigma(\alpha) > \tau(\alpha)$ with Definition 11, we obtain that

$$\sigma(\alpha) = \tau(\alpha) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} > \tau(\alpha).$$

Therefore,

$$\sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} > 0.$$

As in the proof of GP4, for QBAF, it is possible that $\mathcal{R}^+(\alpha) = \varnothing$ because $\sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)}$ could be negative, satisfying the inequality. So, the property is not satisfied. α_2 in Figure 6.7 is one counterexample of this GP. In contrast, for QBAF', let us proof by contradiction. Assume that $\mathcal{R}^+(\alpha) = \varnothing$. We will obtain $-\sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} > 0$. Thus, $\sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} < 0$ which is impossible under QBAF' where $\sigma(b) > 0$. Therefore, it must be the case that $\mathcal{R}^+(\alpha) \neq \varnothing$.

Hence, $\langle QBAF, \sigma \rangle$ does not satisfy GP5, but $\langle QBAF', \sigma \rangle$ does. □

GP6. If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) = \sigma(\beta)$.

Proof. According to Definition 11, we have

$$\sigma(\alpha) = \tau(\alpha) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)}.$$

Because $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, we can replace $\tau(\alpha)$, $\mathcal{R}^+(\alpha)$, and $\mathcal{R}^-(\alpha)$ with $\tau(\beta)$, $\mathcal{R}^+(\beta)$, and $\mathcal{R}^-(\beta)$, respectively. Therefore,

$$\sigma(\alpha) = \tau(\beta) + \sum_{b \in \mathcal{R}^+(\beta)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\beta)} \frac{\sigma(b)}{\nu(b)} = \sigma(\beta).$$

Hence, both $\langle QBAF, \sigma \rangle$ and $\langle QBAF', \sigma \rangle$ satisfy GP6. □

GP7. If $\mathcal{R}^-(\alpha) \subset \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) > \sigma(\beta)$.

Proof. Since $\mathcal{R}^-(\alpha) \subset \mathcal{R}^-(\beta)$, we can partition $\mathcal{R}^-(\beta)$ into two disjoint sets which are $\mathcal{R}^-(\alpha)$ and a non-empty set of arguments $X = \mathcal{R}^-(\beta) - \mathcal{R}^-(\alpha)$.

According to Definition 11, we have

$$\begin{aligned}\sigma(\beta) &= \tau(\beta) + \sum_{b \in \mathcal{R}^+(\beta)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\beta)} \frac{\sigma(b)}{\nu(b)} \\ &= \tau(\beta) + \sum_{b \in \mathcal{R}^+(\beta)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in X} \frac{\sigma(b)}{\nu(b)}\end{aligned}$$

Because $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$ and $\tau(\alpha) = \tau(\beta)$, we obtain that

$$\begin{aligned}\sigma(\beta) &= \tau(\alpha) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in X} \frac{\sigma(b)}{\nu(b)} \\ &= \sigma(\alpha) - \sum_{b \in X} \frac{\sigma(b)}{\nu(b)}\end{aligned}$$

As in the previous proofs, for QBAF, it is possible that $\sum_{b \in X} \frac{\sigma(b)}{\nu(b)} < 0$, rendering $\sigma(\alpha) < \sigma(\beta)$ and unsatisfying GP7. We can find a counterexample in Figure 6.7 with $\alpha = \alpha_3$ and $\beta = \alpha_2$. In contrast, for QBAF', $\sigma(b) > 0$. Therefore, $\sum_{b \in X} \frac{\sigma(b)}{\nu(b)}$ is always greater than 0. As a result, $\sigma(\alpha) > \sigma(\beta)$ as required.

Hence, $\langle QBAF, \sigma \rangle$ does not satisfy GP7, but $\langle QBAF', \sigma \rangle$ does. □

GP8. If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) \subset \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) < \sigma(\beta)$.

Proof. Since $\mathcal{R}^+(\alpha) \subset \mathcal{R}^+(\beta)$, we can partition $\mathcal{R}^+(\beta)$ into two disjoint sets which are $\mathcal{R}^+(\alpha)$ and a non-empty set of arguments $X = \mathcal{R}^+(\beta) - \mathcal{R}^+(\alpha)$. According to Definition 11, we have

$$\begin{aligned}\sigma(\beta) &= \tau(\beta) + \sum_{b \in \mathcal{R}^+(\beta)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\beta)} \frac{\sigma(b)}{\nu(b)} \\ &= \tau(\beta) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} + \sum_{b \in X} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\beta)} \frac{\sigma(b)}{\nu(b)}\end{aligned}$$

Because $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$ and $\tau(\alpha) = \tau(\beta)$, we obtain that

$$\begin{aligned}\sigma(\beta) &= \tau(\alpha) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} + \sum_{b \in X} \frac{\sigma(b)}{\nu(b)} \\ &= \sigma(\alpha) + \sum_{b \in X} \frac{\sigma(b)}{\nu(b)}\end{aligned}$$

As in the previous proofs, for QBAF, it is possible that $\sum_{b \in X} \frac{\sigma(b)}{\nu(b)} < 0$, rendering $\sigma(\alpha) > \sigma(\beta)$ and unsatisfying GP8. We can find a counterexample in Figure 6.7 with $\alpha = \alpha_3$ and $\beta = \alpha_1$. In contrast, for QBAF', $\sigma(b) > 0$. Therefore, $\sum_{b \in X} \frac{\sigma(b)}{\nu(b)}$ is always greater than 0. As a result, $\sigma(\alpha) < \sigma(\beta)$ as required.

Hence, $\langle QBAF, \sigma \rangle$ does not satisfy GP8, but $\langle QBAF', \sigma \rangle$ does. □

GP9. If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) < \tau(\beta)$, then $\sigma(\alpha) < \sigma(\beta)$.

Proof. Let us proof by contradiction. Assume that $\sigma(\alpha) \geq \sigma(\beta)$. Applying Definition 11, we obtain

$$\tau(\alpha) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} \geq \tau(\beta) + \sum_{b \in \mathcal{R}^+(\beta)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\beta)} \frac{\sigma(b)}{\nu(b)}$$

Because $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$ and $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$,

$$\tau(\alpha) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)} \geq \tau(\beta) + \sum_{b \in \mathcal{R}^+(\alpha)} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^-(\alpha)} \frac{\sigma(b)}{\nu(b)}$$
$$\tau(\alpha) \geq \tau(\beta)$$

However, this contradicts the given statement that $\tau(\alpha) < \tau(\beta)$, so $\sigma(\alpha) \geq \sigma(\beta)$ cannot be true. Thus, $\sigma(\alpha) < \sigma(\beta)$ as required.

Hence, both $\langle QBAF, \sigma \rangle$ and $\langle QBAF', \sigma \rangle$ satisfy GP9. □

As a recap from Section 2.3, the definition of $<$ between two sets used in GP10 and GP11 is defined as follows. Given P and Q are subsets of $\mathcal{A}$, $P \leq Q$ iff there exists an injective mapping f from P to Q such that $\forall \alpha \in P, \sigma(\alpha) \leq \sigma(f(\alpha))$. Furthermore, $P < Q$ iff $P \leq Q$ but $Q \nleq P$.

GP10. If $\mathcal{R}^-(\alpha) < \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) > \sigma(\beta)$.

Proof. Counterexamples of this GP for QBAF and QBAF' are in Figures 6.7 and 6.8, respectively, where $\alpha = \alpha_4$ and $\beta = \alpha_5$. We can see that $\mathcal{R}^-(\alpha) \leq \mathcal{R}^-(\beta)$ as we have a mapping from $\mathcal{R}^-(\alpha)$ to $\mathcal{R}^-(\beta)$, $f = \{(\alpha_{10}, \alpha_{11})\}$, where $\forall a \in \mathcal{R}^-(\alpha), \sigma(a) \leq \sigma(f(a))$. In this case, $\sigma(\alpha_{10}) \leq \sigma(\alpha_{11})$. However, $\mathcal{R}^-(\beta) \nleq \mathcal{R}^-(\alpha)$. So, $\mathcal{R}^-(\alpha) < \mathcal{R}^-(\beta)$. In addition, $\mathcal{R}^+(\alpha) = \mathcal{R}^+(\beta) = \varnothing$ and

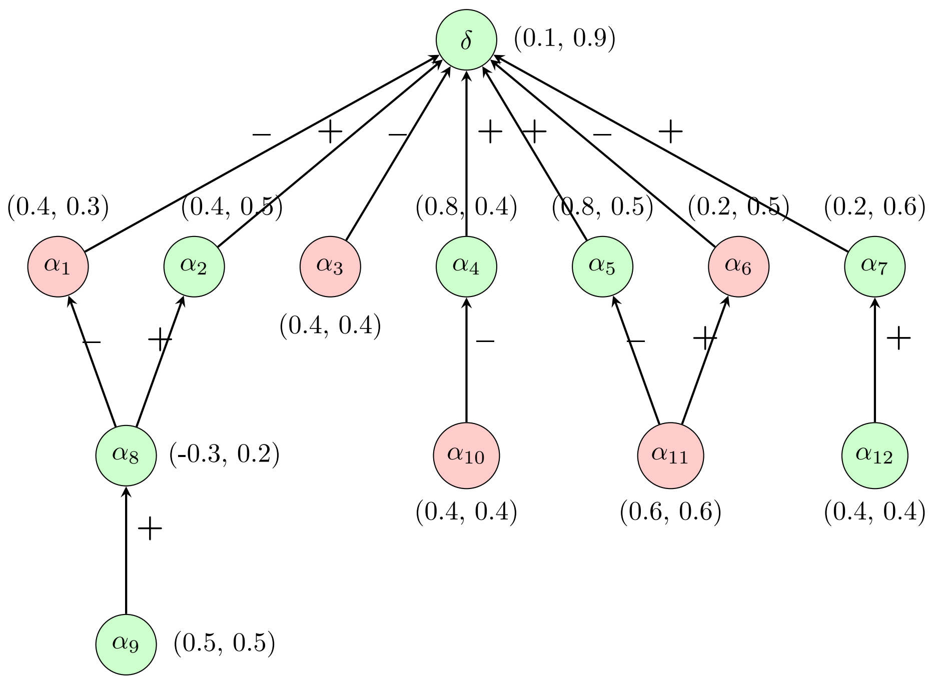


Figure 6.8.: The corresponding QBAF' of the QBAF in Figure 6.7, used as a counterexample in Section 6.2.4. With each argument, there is a value pair (x, y) where x and y represent the base score and the strength (based on the logistic regression semantics σ) of the argument, respectively. The color represents the argument's supported class (i.e., green for positive (1) and red for negative (0)).

$\tau(\alpha) = \tau(\beta) = 0.4$. Nevertheless, $\sigma(\alpha) = 0.4$ and $\sigma(\beta) = 0.5$, so $\sigma(\alpha) > \sigma(\beta)$ is not true.

Hence, both $\langle QBAF, \sigma\rangle$ and $\langle QBAF', \sigma\rangle$ do not satisfy GP10. (This is because σ considers not only the strengths of the attackers and the supporters but also their out-degrees.) $\square$

GP11. If $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta)$, $\mathcal{R}^+(\alpha) < \mathcal{R}^+(\beta)$, and $\tau(\alpha) = \tau(\beta)$, then $\sigma(\alpha) < \sigma(\beta)$.

Proof. Counterexamples of this GP for QBAF and QBAF' are in Figures 6.7 and 6.8, respectively, where $\alpha = \alpha_7$ and $\beta = \alpha_6$. We can see that $\mathcal{R}^+(\alpha) \leq \mathcal{R}^+(\beta)$ as we have a mapping from $\mathcal{R}^+(\alpha)$ to $\mathcal{R}^+(\beta)$, $f = \{(\alpha_{12}, \alpha_{11})\}$, where $\forall a \in \mathcal{R}^+(\alpha), \sigma(a) \leq \sigma(f(a))$. In this case, $\sigma(\alpha_{12}) \leq \sigma(\alpha_{11})$. However, $\mathcal{R}^+(\beta) \nleq \mathcal{R}^+(\alpha)$. So, $\mathcal{R}^+(\alpha) < \mathcal{R}^+(\beta)$. In addition, $\mathcal{R}^-(\alpha) = \mathcal{R}^-(\beta) = \varnothing$ and

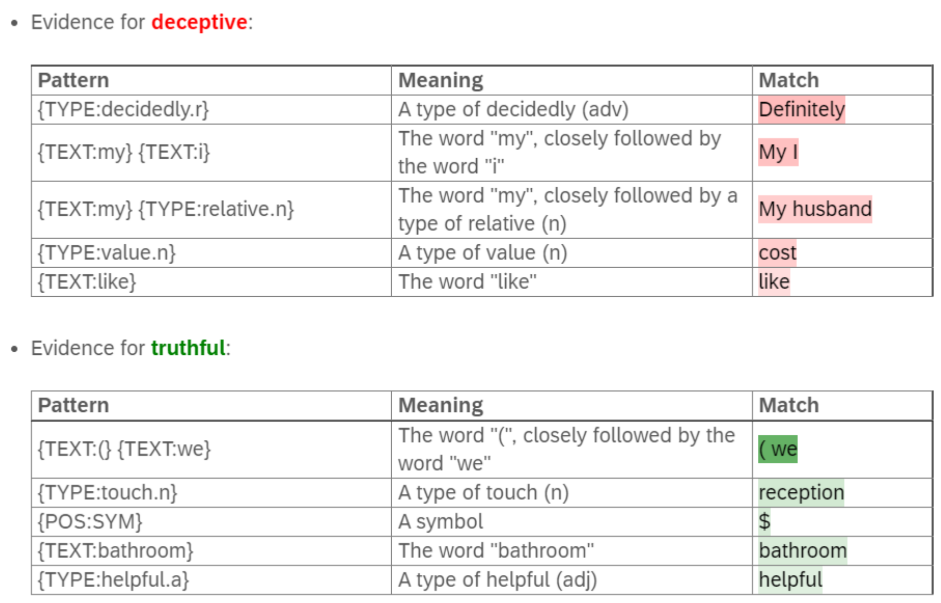
$\tau(\alpha) = \tau(\beta) = 0.2$. Nevertheless, $\sigma(\alpha) = 0.6$ and $\sigma(\beta) = 0.5$, so $\sigma(\alpha) < \sigma(\beta)$ is not true.

Hence, both $\langle QBAF, \sigma\rangle$ and $\langle QBAF', \sigma\rangle$ do not satisfy GP11. (This is because σ considers not only the strengths of the attackers and the supporters but also their out-degrees.) □

In conclusion, $\langle QBAF', \sigma\rangle$ satisfies nine out of the eleven group properties, while $\langle QBAF, \sigma\rangle$ satisfies only three. It means that our post-processing step is important to make the argumentation framework align better with human interpretations and become more suitable for generating local explanations.

6.2.5. Generating Explanations

Presenting the whole QBAF' as a local explanation to lay users is probably not a good idea since the graph could be very complicated (in terms of the number of arguments, relations, and the depth). Also, the notions of attacks and supports may not be familiar to the users. So, the last step of AXPLR is extracting the explanation from the QBAF'. We know from Theorem 2 and Corollary 2 that the prediction of the LR model is associated to the strength of δ. Hence, we can explain the prediction based on how $\sigma(\delta)'$ was calculated. The value of $\sigma(\delta)'$ depends on $\tau'(\delta)$ (corresponding to the bias term in LR) and the strength σ of all the attackers and supporters of δ. Therefore, we return, as the local explanation for $c'(\delta)$, a list of $(p_j, \pi(p_j, x), \sigma(\alpha_j)')$ triplets where x is the input text, α_j (representing the pattern p_j) is one of the k strongest supporters of δ, and $\pi(p_j, x)$ is a part of x that matches the pattern p_j. If we want both evidence for and counter-evidence against the prediction, we can show $(p_j, \pi(p_j, x), \sigma(\alpha_j)')$ for the top supporters and top attackers with the highest $\sigma(\alpha_j)'$. We call this explanation the **shallow AXPLR**. Figure 6.9 shows an example of shallow AXPLR (extracted from a BQBAF') for the deceptive review detection task where the color intensity represents the strengths of the arguments. Shallow AXPLR is similar to the flat logistic regression explanation (FLX) introduced in Section 6.1. The only differences are that *(i)* FLX selects top k patterns based on the size of $w_j f_j$ (which is equivalent to $\tau(\alpha_j)$) while the shallow AXPLR selects top k arguments based on the dialectical strength $\sigma(\alpha_j)$ and *(ii)* any patterns matched in x can be in the FLX whereas only



- Evidence for **deceptive**:

Pattern	Meaning	Match
{TYPE:decidedly.r}	A type of decidedly (adv)	Definitely
{TEXT:my} {TEXT:i}	The word "my", closely followed by the word "i"	My I
{TEXT:my} {TYPE:relative.n}	The word "my", closely followed by a type of relative (n)	My husband
{TYPE:value.n}	A type of value (n)	cost
{TEXT:like}	The word "like"	like

- Evidence for **truthful**:

Pattern	Meaning	Match
{TEXT:(} {TEXT:we}	The word "(", closely followed by the word "we"	(we
{TYPE:touch.n}	A type of touch (n)	reception
{POS:SYM}	A symbol	$
{TEXT:bathroom}	The word "bathroom"	bathroom
{TYPE:helpful.a}	A type of helpful (adj)	helpful

Figure 6.9.: Example of shallow AXPLR for deceptive review detection. The meaning of each pattern is also provided. The color and its intensity represent the supported class and the strengths of the arguments, respectively.

attackers and supporters of δ can be in the shallow AXPLR.

The shallow AXPLR leverages only the attackers and supporters of δ although we have more information. Therefore, we propose another variation of AXPLR, called **deep AXPLR**, which also uses other arguments in the QBAF'. Basically, it shows what the shallow AXPLR shows but additionally allows the users to expand the arguments shown to see their attackers and supporters (if any). They can expand further to see deeper arguments in the QBAF' until there is no attacker or supporter for that argument. Figure 6.10 is a deep AXPLR, explaining the same example and using the same BQBAF' as the shallow AXPLR in Figure 6.9 does.

Actually, QBAF' has potential to be used for generating other forms of explanations such as conversational explanations (Cocarascu et al., 2019) and counterfactual explanations (Albini et al., 2021). These interesting directions, however, are left for future work.

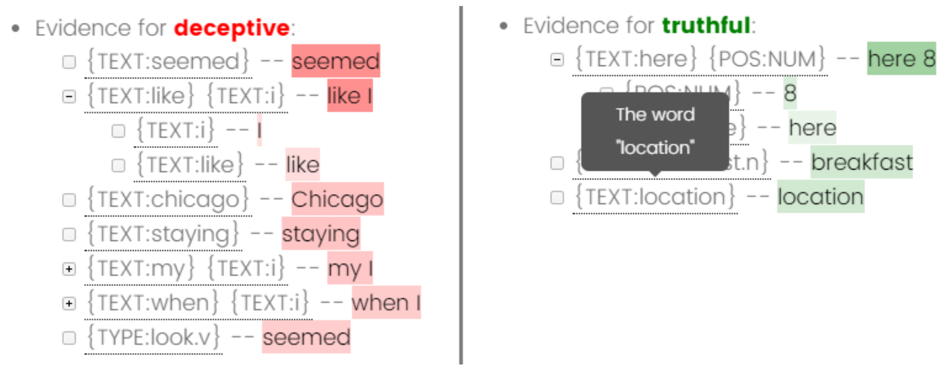

Evidence for deceptive:
{TEXT:seemed} -- seemed
{TEXT:like} {TEXT:i} -- like I
{TEXT:i} -- I
{TEXT:like} -- like
{TEXT:chicago} -- Chicago
{TEXT:staying} -- staying
{TEXT:my} {TEXT:i} -- my I
{TEXT:when} {TEXT:i} -- when I
{TYPE:look.v} -- seemed
Evidence for truthful:
{TEXT:here} {POS:NUM} -- here 8
} -- 8
} -- here
t.n} -- breakfast
{TEXT:location} -- location
The word "location"

Figure 6.10.: Example of deep AXPLR for deceptive review detection. A user can expand some patterns to see their sub-patterns (i.e., their attackers and/or supporters). The meaning of each pattern is provided as a tooltip. The color and its intensity represent the supported class and the strengths of the arguments, respectively.

6.3. Experimental Setup

To evaluate AXPLR, we conducted both empirical and human evaluations. For the empirical evaluation, we calculated some statistics for the QBAFs extracted for target examples and performed analyses concerning *sufficiency* of the generated explanations. For the human evaluation, we *(i)* assessed plausibility of the explanations (i.e., how well the explanations from AXPLR align with human explanations compared to a standard method for explaining logistic regression results) and *(ii)* assessed how well AXPLR can teach and support humans to perform a new task.

In the experiments, we targeted binary text classification using three English datasets as shown in Table 6.3. The table also shows the classes we consider as positive and negative classes when running GrASP and AXPLR.

- **SMS Spam Collection** (Almeida et al., 2011) focusing on detecting spams in a collection of SMS (short message service) messages. The dataset is imbalanced, containing 13.40% spam messages and 86.60% ham messages (i.e., non-spams).
- **Amazon Clothes** (He and McAuley, 2016), the same dataset as in Chapter 5, focusing on classifying whether a review (of clothing, shoes, and jewelry products) has positive or negative sentiment. The overall

Dataset	Positive	Negative	Train / Dev / Test
SMS Spam Collection	Spam	Not spam	3567 / 892 / 1115
Amazon Clothes	Positive	Negative	3000 / 300 / 10000
Deceptive Hotel Reviews	Deceptive	Truthful	1024 / 256 / 320

Table 6.3.: Datasets used in the experiments.

Dataset	Positive F1	Negative F1	Macro F1	Accuracy
SMS Spam Collection	0.891	0.986	0.939	0.975
Amazon Clothes	0.836	0.836	0.836	0.836
Deceptive Hotel Reviews	0.847	0.859	0.853	0.853

Table 6.4.: Performance of the pattern-based LR models on the test sets.

dataset is balanced.

- **Deceptive Hotel Reviews** (Ott et al., 2011, 2013) focusing on identifying whether a given hotel review is truthful (genuine) or deceptive (fake). There are 1600 reviews in total for 20 hotels. For each hotel, there are 20 truthful positive, 20 truthful negative, 20 deceptive positive, and 20 deceptive negative reviews. (Positive and negative here refer to the review sentiment.)

For the LR classifiers of the first two datasets, the GrASP patterns were constructed with lemma, part-of-speech tags (POS), wordnet hypernyms, and sentiment attributes. We used alphabet size of 200, allowed two gaps in the patterns, and generated 100 patterns in total. For the last dataset (Deceptive Hotel Reviews), the settings were the same except that we used text attributes (capturing the whole word) instead of the lemma attributes and we generated 200 patterns in total. The performance of the LR classifiers of the three datasets are reported in Table 6.4.

6.4. Experiment 1: Empirical Evaluation

We divide the empirical evaluation into two parts. The first part discusses the statistics for QBAF and QBAF' we generated from the test sets. This helps us understand what the graphs look like on average. The second part focuses on *sufficiency*, aiming to answer "How many supporting arguments are needed on average so as to sufficiently make the model predicts what it predicts?". This helps us decide how many arguments we should show in AXPLR generally.

6.4.1. Statistics for QBAF and QBAF'

Tables 6.5-6.7 show the statistics of the QBAF and QBAF' for the SMS Spam Collection, Amazon Clothes, and Deceptive Hotel Reviews, respectively. Due to space limitation, we introduce a few new symbols for using in the tables.

$$\begin{aligned} \mathcal{A}^{+,\delta} &= \{a \in \mathcal{A} | c(a) = 1\} \\ \mathcal{A}^{-,\delta} &= \{a \in \mathcal{A} | c(a) = 1\} \\ \mathcal{R} &= \mathcal{R}^{-} \cup \mathcal{R}^{+} \\ \mathcal{R}_{\backslash\delta} &= \{(a,b) \in \mathcal{R} | b \neq \delta\} \end{aligned}$$

Basically, $\mathcal{A}^{+,\delta}$ and $\mathcal{A}^{-,\delta}$ are sets of arguments (possibly including δ) that support the positive class and the negative class, respectively. $\mathcal{R}$ is the set of all relations in the QBAF or QBAF'. $\mathcal{R}_{\backslash\delta}$ is the set of all relations that are not connected to δ. If $\mathcal{R}_{\backslash\delta} = \varnothing$, the generated AXPLR will look like FLX where we do not consider relationships between features.

Number of arguments. According to the result tables, the spam dataset had the minimum average number of arguments (~ 10 arguments per example as shown by $|\mathcal{A}|$ in Table 6.5). However, if we look at examples of which the prediction is positive (i.e., both TP and FP), we can see that they had 36 arguments per example on average. Looking at the underlying PLR model, we found that the default argument δ before post processing supported the negative class with $\tau(\delta) = 5.800$, which was very high compared to the base scores of other arguments. It means that the classifier answered "Not spam" by default unless there is sufficient evidence to answer "Spam". Even true negative examples (TN) had around three arguments for the negative class on average, including δ. Interestingly, false negative examples (FN) had relatively higher arguments than true negatives, but still less than those of true positives. It implies that the false negatives usually had some, but insufficient, evidence for the positive class, compared to the true negatives

Measurement	TQBAF	TQBAF'	BQBAF	BQBAF'
# Examples	1115 (TP: 115, TN: 972, FP: 5, FN: 23)			
$\lvert\mathcal{A}\rvert$	10.08±11.55	10.08±11.55	10.08±11.55	10.08±11.55
- TP	35.98±12.26	35.98±12.26	35.98±12.26	35.98±12.26
- TN	6.66±6.01	6.66±6.01	6.66±6.01	6.66±6.01
- FP	36.00±11.98	36.00±11.98	36.00±11.98	36.00±11.98
- FN	19.30±9.66	19.30±9.66	19.30±9.66	19.30±9.66
$\lvert\mathcal{A}^{+,\delta}\rvert$	5.86±7.12	6.25±7.77	5.86±7.12	6.47±8.07
- TP	22.36±7.17	24.68±7.70	22.36±7.17	25.42±7.52
- TN	3.70±3.57	3.85±3.69	3.70±3.57	4.00±4.01
- FP	20.40±6.31	22.20±6.87	20.40±6.31	24.20±6.02
- FN	11.26±4.73	11.74±4.83	11.26±4.73	12.48±5.69
$\lvert\mathcal{A}^{-,\delta}\rvert$	4.22±4.63	3.83±4.14	4.22±4.63	3.61±3.81
- TP	13.63±5.60	11.30±5.37	13.63±5.60	10.57±5.41
- TN	2.96±2.66	2.81±2.65	2.96±2.66	2.66±2.33
- FP	15.60±5.77	13.80±5.50	15.60±5.77	11.80±6.06
- FN	8.04±5.17	7.57±5.29	8.04±5.17	6.83±4.25
$\lvert\mathcal{R}\rvert$	11.64±16.77	11.64±16.77	13.82±21.40	13.82±21.40
- TP	49.31±19.63	49.31±19.63	61.43±26.11	61.43±26.11
- TN	6.69±8.17	6.69±8.17	7.57±10.37	7.57±10.37
- FP	50.60±21.14	50.60±21.14	64.20±28.35	64.20±28.35
- FN	23.96±14.27	23.96±14.27	29.17±19.09	29.17±19.09
$\lvert\mathcal{R}_{\setminus\delta}\rvert$	8.32±14.76	8.32±14.76	8.32±14.76	8.32±14.76
- TP	40.81±18.81	40.81±18.81	40.81±18.81	40.81±18.81
- TN	4.06±7.09	4.06±7.09	4.06±7.09	4.06±7.09
- FP	43.40±21.41	43.40±21.41	43.40±21.41	43.40±21.41
- FN	18.43±13.93	18.43±13.93	18.43±13.93	18.43±13.93
$\lvert\mathcal{R}^{-}\rvert$	6.39±8.48	5.31±6.33	7.51±10.90	5.73±7.93
- TP	24.92±9.91	16.50±8.10	31.50±13.48	20.22±11.75
- TN	3.95±4.43	3.74±4.15	4.36±5.43	3.75±4.71
- FP	26.60±10.83	20.60±10.36	32.40±14.84	22.40±12.95
- FN	12.78±7.11	12.13±6.59	15.30±9.18	13.35±7.79
$\lvert\mathcal{R}^{+}\rvert$	5.24±8.49	6.33±10.88	6.31±10.72	8.10±13.88
- TP	24.39±10.38	32.81±12.21	29.93±13.40	41.22±15.19
- TN	2.74±3.99	2.95±4.28	3.21±5.20	3.82±5.85
- FP	24.00±10.89	30.00±11.07	31.80±14.04	41.80±15.69
- FN	11.17±7.51	11.83±7.95	13.87±10.33	15.83±11.62

Table 6.5.: Statistics (Average ± SD) of QBAF and QBAF' for the SMS Spam Collection dataset. TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively. Please find the meanings of $\mathcal{A}^{+,\delta}, \mathcal{A}^{-,\delta}, \mathcal{R}$ and $\mathcal{R}_{\setminus\delta}$ in Section 6.4

Measurement	TQBAF	TQBAF'	BQBAF	BQBAF'
# Examples	10000 (TP: 4176, TN: 4186, FP: 848, FN: 790)			
$\|\mathcal{A}\|$	16.09±8.13	16.09±8.13	16.09±8.13	16.09±8.13
- TP	14.85±7.57	14.85±7.57	14.85±7.57	14.85±7.57
- TN	17.45±8.21	17.45±8.21	17.45±8.21	17.45±8.21
- FP	14.99±8.14	14.99±8.14	14.99±8.14	14.99±8.14
- FN	16.57±9.31	16.57±9.31	16.57±9.31	16.57±9.31
$\|\mathcal{A}^{+,\delta}\|$	7.39±4.36	7.45±4.50	7.39±4.36	7.43±4.88
- TP	8.91±4.18	9.74±4.07	8.91±4.18	10.22±4.51
- TN	5.93±3.98	5.20±3.76	5.93±3.98	4.75±3.68
- FP	7.80±4.27	8.47±4.12	7.80±4.27	8.49±4.34
- FN	6.65±4.55	6.10±4.29	6.65±4.55	5.84±4.37
$\|\mathcal{A}^{-,\delta}\|$	8.70±5.23	8.64±5.89	8.70±5.23	8.65±6.24
- TP	5.94±3.99	5.10±4.18	5.94±3.99	4.63±4.27
- TN	11.52±4.96	12.25±5.36	11.52±4.96	12.71±5.53
- FP	7.19±4.18	6.52±4.38	7.19±4.18	6.50±4.44
- FN	9.92±5.14	10.47±5.49	9.92±5.14	10.73±5.57
$\|\mathcal{R}\|$	17.64±10.25	17.64±10.25	21.68±13.29	21.68±13.29
- TP	16.33±9.78	16.33±9.78	20.55±12.49	20.55±12.49
- TN	19.10±10.20	19.10±10.20	23.07±13.47	23.07±13.47
- FP	16.38±10.47	16.38±10.47	20.06±13.57	20.06±13.57
- FN	18.18±11.57	18.18±11.57	22.03±15.32	22.03±15.32
$\|\mathcal{R}_{\setminus\delta}\|$	12.71±8.73	12.71±8.73	12.71±8.73	12.71±8.73
- TP	12.34±8.42	12.34±8.42	12.34±8.42	12.34±8.42
- TN	13.30±8.74	13.30±8.74	13.30±8.74	13.30±8.74
- FP	11.68±9.01	11.68±9.01	11.68±9.01	11.68±9.01
- FN	12.65±9.77	12.65±9.77	12.65±9.77	12.65±9.77
$\|\mathcal{R}^{-}\|$	5.27±3.79	5.11±3.96	8.10±5.54	4.08±3.47
- TP	4.74±3.55	5.50±4.38	8.20±5.18	3.72±3.43
- TN	5.73±3.86	4.56±3.27	8.00±5.70	4.12±3.31
- FP	5.20±3.82	6.30±4.66	8.09±5.73	5.15±3.75
- FN	5.61±4.25	4.68±3.67	8.14±6.31	4.72±3.84
$\|\mathcal{R}^{+}\|$	12.38±7.04	12.53±7.20	13.58±8.44	17.59±10.52
- TP	11.59±6.76	10.83±5.97	12.35±7.72	16.83±9.79
- TN	13.37±6.98	14.54±7.64	15.06±8.65	18.95±10.82
- FP	11.18±7.16	10.08±6.30	11.97±8.35	14.91±10.26
- FN	12.56±7.92	13.49±8.52	13.89±9.65	17.32±11.89

Table 6.6.: Statistics (Average ± SD) of QBAF and QBAF' for the Amazon Clothes dataset. TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively. Please find the meanings of $\mathcal{A}^{+,\delta}, \mathcal{A}^{-,\delta}, \mathcal{R}$ and $\mathcal{R}_{\setminus\delta}$ in Section 6.4

Measurement	TQBAF	TQBAF'	BQBAF	BQBAF'
# Examples	320 (TP: 130, TN: 143, FP: 26, FN: 21)			
$\lvert\mathcal{A}\rvert$	19.44±6.82	19.44±6.82	19.44±6.82	19.44±6.82
- TP	19.54±6.54	19.54±6.54	19.54±6.54	19.54±6.54
- TN	20.24±6.86	20.24±6.86	20.24±6.86	20.24±6.86
- FP	17.77±6.48	17.77±6.48	17.77±6.48	17.77±6.48
- FN	15.48±7.34	15.48±7.34	15.48±7.34	15.48±7.34
$\lvert\mathcal{A}^{+,\delta}\rvert$	9.14±4.46	9.89±5.02	9.14±4.46	10.34±5.00
- TP	12.06±4.27	13.58±4.53	12.06±4.27	13.85±4.50
- TN	6.85±3.11	6.92±3.21	6.85±3.11	7.43±3.32
- FP	9.50±3.56	10.77±3.74	9.50±3.56	11.42±4.09
- FN	6.19±3.60	6.29±3.61	6.19±3.60	7.10±3.96
$\lvert\mathcal{A}^{-,\delta}\rvert$	10.30±4.91	9.55±5.33	10.30±4.91	9.10±5.13
- TP	7.48±3.16	5.96±3.14	7.48±3.16	5.69±2.93
- TN	13.39±4.77	13.32±4.81	13.39±4.77	12.81±4.68
- FP	8.27±3.34	7.00±3.27	8.27±3.34	6.35±2.86
- FN	9.29±4.23	9.19±4.19	9.29±4.23	8.38±3.77
$\lvert\mathcal{R}\rvert$	22.03±9.43	22.03±9.43	21.48±9.62	21.48±9.62
- TP	23.96±9.93	23.96±9.93	23.12±10.32	23.12±10.32
- TN	21.41±8.69	21.41±8.69	21.01±8.67	21.01±8.67
- FP	20.04±8.88	20.04±8.88	20.00±9.75	20.00±9.75
- FN	16.81±9.35	16.81±9.35	16.29±9.37	16.29±9.37
$\lvert\mathcal{R}_{\setminus\delta}\rvert$	8.19±6.35	8.19±6.35	8.19±6.35	8.19±6.35
- TP	10.69±7.09	10.69±7.09	10.69±7.09	10.69±7.09
- TN	6.47±5.13	6.47±5.13	6.47±5.13	6.47±5.13
- FP	7.35±5.64	7.35±5.64	7.35±5.64	7.35±5.64
- FN	5.52±4.59	5.52±4.59	5.52±4.59	5.52±4.59
$\lvert\mathcal{R}^{-}\rvert$	9.30±4.35	7.00±3.43	8.43±4.25	6.13±3.24
- TP	11.41±4.43	6.45±3.34	10.43±4.50	5.98±3.51
- TN	7.75±3.48	7.55±3.34	6.96±3.29	6.32±2.96
- FP	9.58±4.03	7.12±3.84	8.81±4.04	6.38±3.65
- FN	6.52±3.75	6.52±3.75	5.52±3.14	5.48±2.91
$\lvert\mathcal{R}^{+}\rvert$	12.73±6.18	15.03±7.09	13.05±6.45	15.34±7.35
- TP	12.55±6.27	17.52±7.66	12.69±6.54	17.14±7.66
- TN	13.66±6.03	13.86±6.11	14.06±6.25	14.69±6.81
- FP	10.46±5.58	12.92±5.59	11.19±6.15	13.62±6.66
- FN	10.29±6.29	10.29±6.29	10.76±6.74	10.81±7.10

Table 6.7.: Statistics (Average ± SD) of QBAF and QBAF' for the Deceptive Hotel Review dataset. TP, TN, FP, and FN stand for true positives, true negatives, false positives, and false negatives, respectively. Please find the meanings of $\mathcal{A}^{+,\delta}, \mathcal{A}^{-,\delta}, \mathcal{R}$ and $\mathcal{R}_{\setminus\delta}$ in Section 6.4

which almost have nothing.

Unlike the SMS Spam Collection dataset, the base scores of δ for the Amazon Clothes and the Deceptive Hotel Reviews datasets were 0.2597 and 0.6932 supporting the negative class, respectively. In order to push the prediction to either positive or negative, we needed evidence. Hence, the average number of arguments for these two datasets were similar for both classes (as shown by $|\mathcal{A}|$ of TP, TN, FP, FN in Tables 6.6-6.7). Examples predicted as positive, therefore, had higher number of arguments for the positive class ($|\mathcal{A}^{+,\delta}|$) than those predicted as negative, and vice versa.

Number of relations. The number of relations, $|\mathcal{R}|$, had the similar trend as the number of arguments. Texts predicted as spams had significantly higher attacks and supports than those predicted as non-spams (see $|\mathcal{R}^{-}|$ and $|\mathcal{R}^{+}|$ in Table 6.5). For the other two datasets, they usually had more supports than attacks, especially after post-processing, to provide sufficient evidence for the predictions. In any case, all the three datasets had $|\mathcal{R}_{\setminus\delta}|$ from 8 to 12, on average, this sure made the explanations extracted from QBAF' (i.e., AXPLR) different from the standard explanations for logistic regressions (i.e., FLX) due to many relations between features.

Other remarks. Finally, we could make a few interesting remarks from these result tables. First, the number of arguments $|\mathcal{A}|$ for TQBAF, TQBAF', BQBAF, and BQBAF' for the same example are always equal. This is expected from Definition 10 and 12. Second, $|\mathcal{R}|$ of TQBAF and BQBAF are different but their $|\mathcal{R}_{\setminus\delta}|$ are the same. This is because TQBAF and BQBAF have the same relations between two non-default arguments except that the directions are reversed. For the relations with the default argument, TQBAF connects the arguments of the most general patterns to the default whereas BQBAF connects the most specific patterns to the default. That is why $|\mathcal{R}|$ was different between TQBAF and BQBAF. Lastly, post-processing does not change the number of total relations in the experiments as you can see from $|\mathcal{R}|$ of TQBAF and TQBAF' and $|\mathcal{R}|$ of BQBAF and BQBAF'. In theory, it could possibly change as the relations (a, b) with $\sigma(a) = 0$ are removed. However, because all the base scores in QBAF are from the weights of the trained LR model, each of which has around 15 decimal points, it is hardly possible to find the argument a with $\sigma(a) = 0$ in practice. So, none of the relations are removed during post-processing.

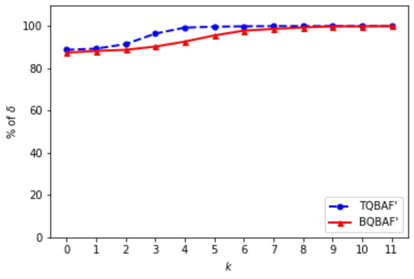
6.4.2. Sufficiency

Next, given a QBAF', we were interested in the number of supporting arguments needed in order to sufficiently explain the prediction. Here, *sufficiently* means that given the base score of δ and all the attacking arguments, the strengths given by these supporting arguments are enough to make the strength of δ greater than 0. This is analogous to the meaning of sufficiency for input rationales, defined in Section 3.2.1 Definition 7, with an adaptation to the context of argumentation frameworks. In other words, for each test example, we wanted to find the smallest k such that $S \subseteq \mathcal{R}^{+\prime}(\delta)$, $|S| = k$ and

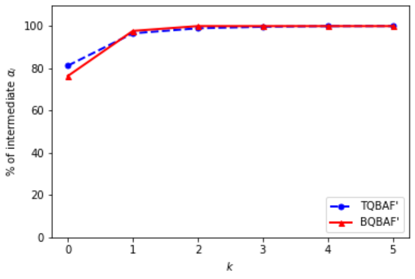
$$\tau'(\delta) + \sum_{b \in S} \frac{\sigma(b)}{\nu(b)} - \sum_{b \in \mathcal{R}^{-\prime}(\delta)} \frac{\sigma(b)}{\nu(b)} > 0 \tag{6.7}$$

Furthermore, we extended our question to other arguments in QBAF' which had at least one attacker or supporter. (We call them *intermediate arguments.*) We wondered how many supporting arguments were needed to make the strength of the argument greater than 0, taking into account the base score and all the strengths from the attackers. Knowing the answers to these questions helps us decide how many arguments we should show to the users for explaining the final prediction or the intermediate arguments, especially when we have limited space.

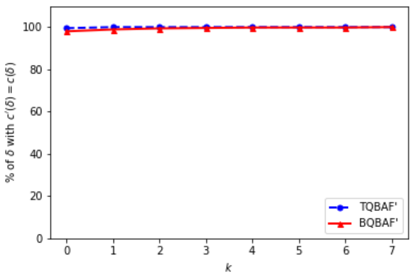
Figures 6.11-6.13 show the results of this experiment. The x-axis of each plot is the number of supporting arguments used (k), whereas the y-axis shows the percentage of arguments (default or intermediate) of which the strength can be greater than 0 by using only k supporting arguments. Considering plots on the left-hand side of the three figures, we can see that the numbers of supporting arguments needed for δ were different for each dataset. The SMS Spam Collection dataset seemed to need the least. However, this was the case only for examples predicted as negative, i.e., the supported class was not flipped after post-processing, corresponding to Figure 6.11(c). The reason was that the base score of δ was relatively high. It could outnumber the strengths from the attackers even without the strengths from the supporters. Nevertheless, this was not true when the supported class flips from negative to positive as it required 4-6 sup-



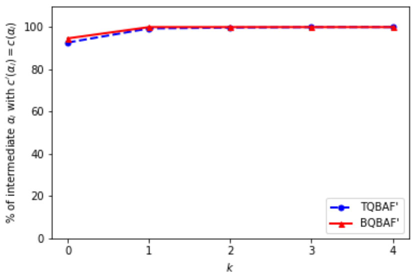
(a) Default δ (All)



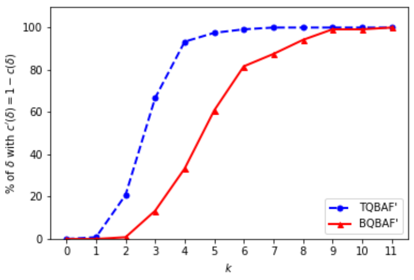
(b) Intermediate α_i (All)



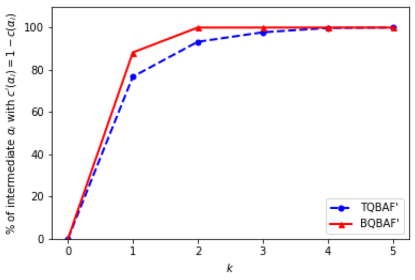
(c) Default δ (Class not flipped)



(d) Intermediate α_i (Class not flipped)



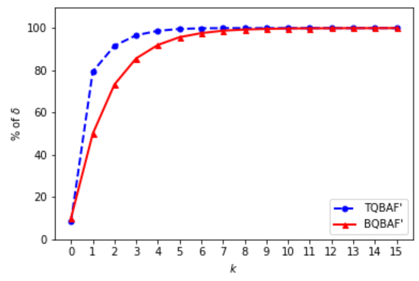
(e) Default δ (Class flipped)



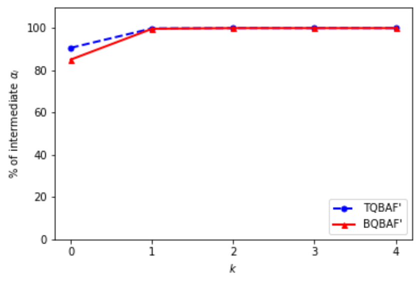
(f) Intermediate α_i (Class flipped)

Figure 6.11.: Plots showing the percentage of arguments (the default arguments δ or intermediate arguments α_i) of which the strength can be greater than 0 using only k supporting arguments. These arguments are extracted from test examples of the SMS Spam Collection dataset. *Class flipped* means the supported class changes after post-processing.

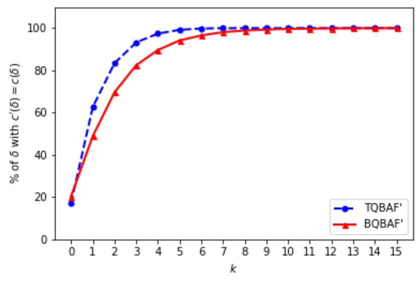
porting arguments to make 80% of the test set have sufficient explanations. Meanwhile, the Amazon Clothes and the Deceptive Hotel Reviews datasets required approximately 1-3 and 3-4 supporting arguments, respectively, for



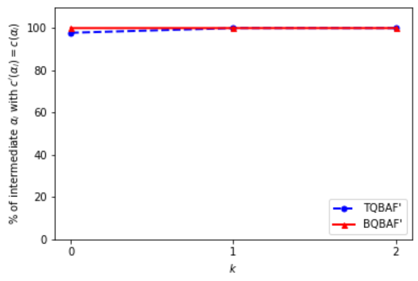
(a) Default δ (All)



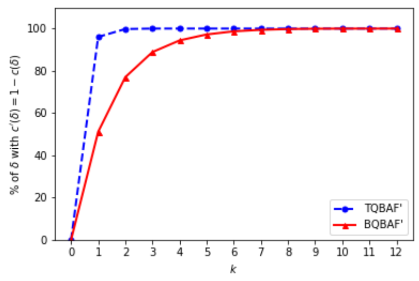
(b) Intermediate α_i (All)



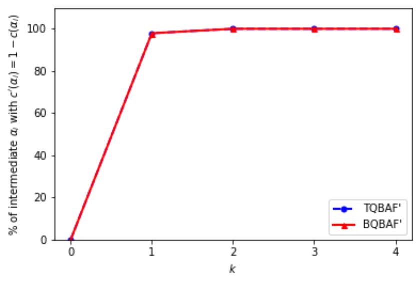
(c) Default δ (Class not flipped)



(d) Intermediate α_i (Class not flipped)



(e) Default δ (Class flipped)

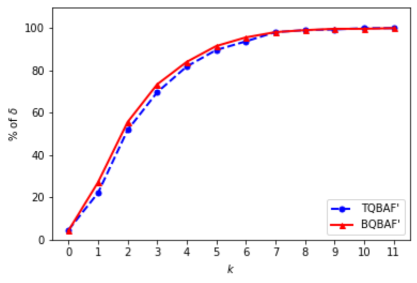


(f) Intermediate α_i (Class flipped)

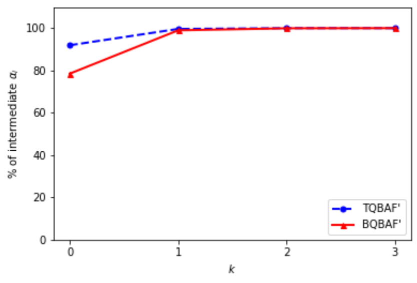
Figure 6.12.: Plots showing the percentage of arguments (the default arguments δ or intermediate arguments α_i) of which the strength can be greater than 0 using only k supporting arguments. These arguments are extracted from test examples of the Amazon Clothes dataset. *Class flipped* means the supported class changes after post-processing.

sufficient explanations of 80% of the test set (regardless of the predicted class).

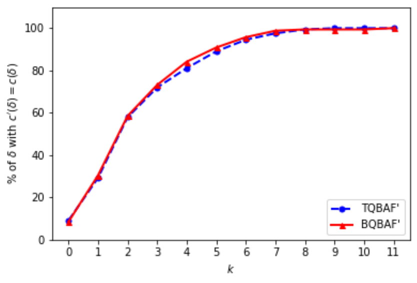
Besides, for the SMS Spam Collection and the Amazon Clothes datasets,



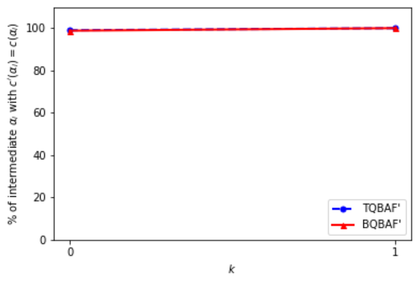
(a) Default δ (All)



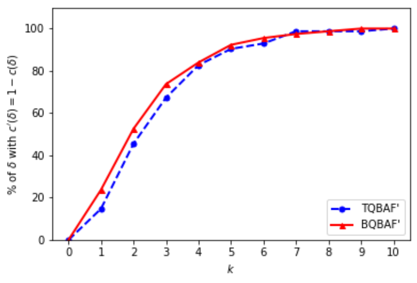
(b) Intermediate α_i (All)



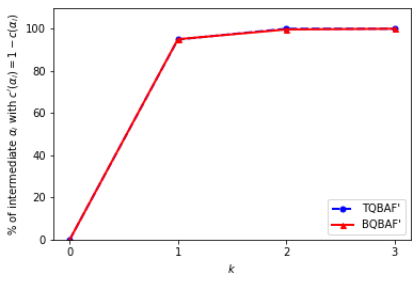
(c) Default δ (Class not flipped)



(d) Intermediate α_i (Class not flipped)



(e) Default δ (Class flipped)



(f) Intermediate α_i (Class flipped)

Figure 6.13.: Plots showing the percentage of arguments (the default arguments δ or intermediate arguments α_i) of which the strength can be greater than 0 using only k supporting arguments. These arguments are extracted from test examples of the Deceptive Hotel Reviews dataset. *Class flipped* means the supported class changes after post-processing.

BQBAF' required more supporting arguments than TQBAF'. This was likely because BQBAF' connected the arguments representing the most specific patterns to the default. For these two datasets, they outnumbered the most general patterns TQBAF' connected to the default. So, the default argument of BQBAF' had more supporters where the strengths were distributed. Therefore, more supporters were required to make the sufficient explanation.

Considering the plots on the right side of Figures 6.11-6.13, only one supporting argument was usually sufficient to explain the supported class of an intermediate argument α_i. Even without any supporters, only the base score was sufficient in most cases if the supported class is not flipped after post-processing. Hence, if a user wants to see supporting information for an intermediate argument, when the space is limited, showing only 1-2 supporters are totally acceptable.

6.5. Experiment 2: Plausibility

In this section, we aimed to evaluate the plausibility of AXPLR, compared to FLX, to confirm our hypothesis that it is essential to consider relations between features (i.e., patterns) when we generate local explanations. So, we compared the feature scores given by the explanation methods to scores reflecting how humans consider the features. For instance, if a machine explanation says that p_1 is a main reason for predicting the positive class and humans also think that p_1 is truly a sign of the positive class, we can say that the machine explanation has high plausibility (i.e., aligning well with human judgement). Therefore, in this experiment, we used FLX and AXPLR to generate explanations for a number of test examples. Then we collected humans' opinions on the explanations and computed the correlations between machine explanation scores and human judgement so as to quantify plausibility of the explanations.

6.5.1. Datasets and Materials

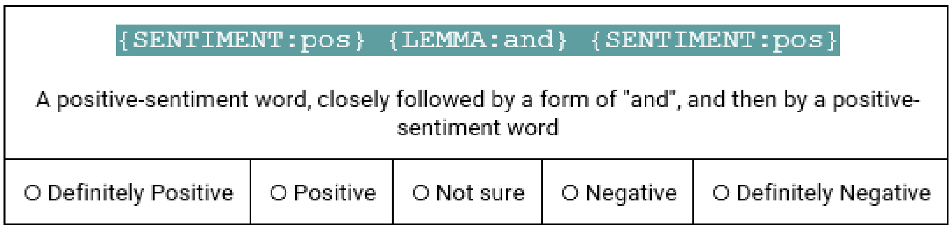
Datasets. We used the SMS Spam Collection (spam classification) and the Amazon Clothes (sentiment analysis) datasets since humans generally perform well on these two tasks and can identify evidence for each of the classes. In contrast, we did not conduct this experiment on the Deceptive Hotel Reviews dataset as lay humans are not adept at identifying deceptive reviews[6], so we cannot trust human judgement on machine explanations in this task. We would work on the deceptive review detection task in the next experiment instead.

Models and inputs. In this experiment, we still used the LR models explained in Section 6.3. For each of the two chosen datasets, we generated machine explanations for 500 test examples and then collected human opinions on these explanations. As in Section 4.1.2, the test examples we (randomly) selected must have the predicted probability of the output class greater than 0.9 to ensure that the bad quality of the explanation was not due to low model accuracy or text ambiguity.

Machine explanations. Concerning explanation methods, we compared AXPLR to FLX, which is the standard way to explain LR predictions. As discussed in Section 6.2.5, both FLX and AXPLR use $(p_j, \pi(p_j, x), s_j)$ triplets as explanations where s_j is the score of the pattern p_j or the match $\pi(p_j, x)$ in the input x. Hence, after we generated explanations for the input x, the pattern p_j and the matched phrase $\pi(p_j, x)$ would obtain s_j from both FLX and AXPLR. For FLX, s_j for the positive class equals $w_j f_j$. For AXPLR, we tried using both τ' and σ as s_j so that we could observe the usefulness of our logistic regression semantics. In fact, τ' and σ needs to be interpreted with the supported class. To simplify this issue, we adjusted s_j for AXPLR to be self-contained by multiplying $\tau'(\alpha_j)$ and $\sigma(\alpha_j)'$ of AXPLR with 1 if $c'(\alpha_j) = 1$, or with -1 if $c'(\alpha_j) = -1$. This made the higher s_j always imply the stronger evidence for the positive class (similar to FLX).

Questions. To measure plausibility, we then collected human scores h_j for p_j and $\pi(p_j, x)$ in order to compare them with s_j from FLX and AXPLR. To collect h_j for p_j, we generated two types of questions. The first type, so called *the pattern question*, showed p_j to human participants and asked them whether p_j was likely the evidence for the positive or the negative class. Since the pattern p_j only may be difficult to understand, we provided the translation to help the participants, as shown in Figure 6.14 (a). Even with the translation, humans may not be able to understand

[6]The human accuracy on deceptive review detection was only around 55% in (Lai et al., 2020).

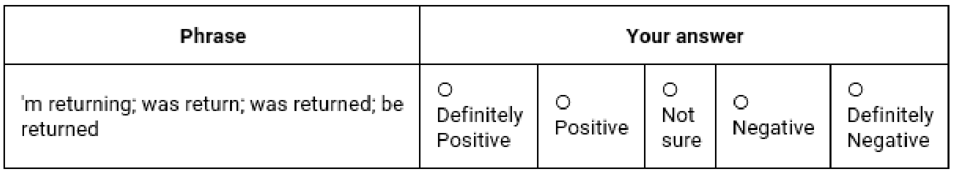


{SENTIMENT:pos} {LEMMA:and} {SENTIMENT:pos}

A positive-sentiment word, closely followed by a form of "and", and then by a positive-sentiment word

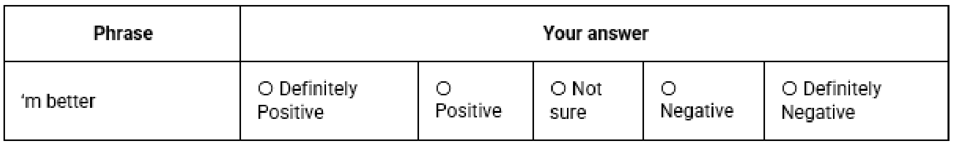
○ Definitely Positive	○ Positive	○ Not sure	○ Negative	○ Definitely Negative

(a) An example question for a pattern.



Phrase	Your answer				
'm returning; was return; was returned; be returned	○ Definitely Positive	○ Positive	○ Not sure	○ Negative	○ Definitely Negative

(b) An example question for a group of phrases sampled from the pattern.



Phrase	Your answer				
'm better	○ Definitely Positive	○ Positive	○ Not sure	○ Negative	○ Definitely Negative

(c) An example question for a matched phrase.

Figure 6.14.: Examples of questions (from the Amazon Clothes dataset) posted on Amazon Mechanical Turk to elicit human scores.

the pattern and its role in the classification task clearly. So, the second type of questions, so called *the samples question*, showed samples of phrases (from the training set) matched by the pattern p_j instead and asked the participants the same question, that is, whether the phrases were likely the evidence for the positive or the negative class. In each question, we showed five unique samples per pattern[7], as displayed in Figure 6.14 (b). To collect h_j for $\pi(p_j, x)$, we used the third type of questions, so called *the matched phrase question*. It was the simplest type of questions because it showed only a single matched phrase $\pi(p_j, x)$ and asked the participants the same question, as shown in Figure 6.14 (c).

For all question types, the participants were provided five options, similar to Chapter 4. For the sentiment analysis task, the options included definitely positive, positive, not sure, negative, and definitely negative. For the spam

[7]If we have less than five unique matched phrases in the training set, we just show all of them.

classification task, these options were instead definitely spam, spam, not sure, non-spam, and definitely non-spam. For both tasks, the corresponding human scores h_j of these options were 2, 1, 0, -1, and -2, respectively, so the higher h_j meant the stronger evidence for the positive class according to human judgement. Each question would be answered by five participants, and the scores were averaged before comparing with machine explanation scores.

6.5.2. Participants and Procedure

We recruited human participants via Amazon Mechanical Turk (MTurk), and we also used MTurk to host our questions. With two datasets and three question types, we created and posted six tasks on MTurk. As with the case in Chapter 5, each task contained a short description, and an MTurk worker could select to do as many tasks/HITs as s/he wanted. Each HIT consisted of four sections including the instructions, three example questions with explanations, a set of actual questions, and a text box for an optional feedback. For the tasks of pattern questions, we also included descriptions on how to read the patterns (such as the attribute types and the structure) in the instructions. Concerning the number of questions per HIT, for the first and the second types of questions (patterns and samples), a single HIT contained 10 questions to be answered. For the third type of questions (matched phrases) which was easier to answer, a single HIT contained 20 questions. The worker was required to answer all the assigned questions before clicking submit. After that, s/he could accept to perform the next HIT if they wanted. As in Chapter 5, we set three required qualifications for MTurk workers who wanted to perform our tasks: (1) currently living in the US, the UK, Australia, or New Zealand, (2) having at least 50 approved HITs and (3) having more than 97% HITS approval rate so far.

Concerning the payment for answering questions, we paid the MTurk workers \$0.30 per 10 pattern questions, \$0.20 per 10 group-of-phrases questions, and \$0.20 per 20 matched phrase questions. At the end, there were 256 participants helping complete 1,726 HITs in total. The minimum and the maximum numbers of HITs per participant are 1 and 90, respectively, while the mean and the standard deviation are 4.92 and 12.06, respectively.

6.5.3. Data Collection and Analysis

As each question was answered by five participants, we averaged their scores before comparing it with s_j. Additionally, we calculated the inter-rater agreement measure (Fleiss' kappa) (Fleiss, 1971) of the human answers. After that, for each p_j in an input x, we had three (averaged) h_j for the pattern, the samples, and the matched phrase. Using as an evaluation measure for plausibility, we calculated Pearson's correlation of the three h_j and the machine explanation scores s_j for different explanation methods and settings. A higher correlation score implies a higher plausibility of the explanation method.

Turning to consider machine explanations, because any p_j with a relatively high FLX score s_j can be chosen as a part of FLX explanation, we compared s_j of FLX with the three averaged human scores h_j for every p_j. By contrast, shallow AXPLR uses only arguments in the top level of the underlying QBAF, i.e., arguments attacking or supporting δ, as explanations. Meanwhile, deep AXPLR can use any arguments in the QBAF. So, for AXPLR, we calculated the Pearson's correlation for both cases, i.e., using s_j for top-level arguments only and using s_j for all arguments (except δ). Furthermore, since the s_j of AXPLR depends on whether the QBAF is TQBAF or BQBAF, we computed the correlation for both scenarios. Note that s_j of AXPLR could be either τ' or σ. However, we did not have the setting where τ' is used as s_j for arguments from all levels since this setting is actually equivalent to FLX.

6.5.4. Results

Tables 6.8 and 6.9 report the Pearson's correlations between the machine explanation scores and the human scores collected from Amazon Mechanical Turk for both datasets. The last row of each table shows inter-rater agreement measures (Fleiss' kappa). We observe that the agreement measures for the SMS Spam Collection dataset were very close to zero (especially for the questions for patterns and samples), while the agreement rates for the Amazon Clothes dataset (sentiment analysis task) were significantly higher. This was likely because evidence from the sentiment analysis task (including patterns, samples, matched phrases) usually conveys clear meanings even without contexts, whereas evidence from the spam detection task often re-

	Human scores		
Explanation scores	Pattern	Samples	Matched Phrase
FLX	0.227	0.175	-0.022
TQBAF'			
$s_j = \tau'(\alpha_j)$ (top level)	**0.687**	**0.533**	-0.019
$s_j = \sigma(\alpha_j)'$ (top level)	0.520	0.462	**0.125**
$s_j = \sigma(\alpha_j)'$ (all levels)	0.176	0.100	0.005
BQBAF'			
$s_j = \tau'(\alpha_j)$ (top level)	0.197	-0.005	-0.047
$s_j = \sigma(\alpha_j)'$ (top level)	0.240	0.175	0.046
$s_j = \sigma(\alpha_j)'$ (all levels)	0.271	0.308	0.053
Fleiss κ	0.001	0.068	0.118

Table 6.8.: Pearson's correlation between explanation scores and human scores for the SMS Spam Collection dataset.

	Human scores		
Explanation scores	Pattern	Samples	Matched Phrase
FLX	0.503	0.525	0.529
TQBAF'			
$s_j = \tau'(\alpha_j)$ (top level)	0.423	0.491	0.487
$s_j = \sigma(\alpha_j)'$ (top level)	**0.632**	**0.693**	**0.688**
$s_j = \sigma(\alpha_j)'$ (all levels)	0.490	0.503	0.501
BQBAF'			
$s_j = \tau'(\alpha_j)$ (top level)	0.442	0.466	0.486
$s_j = \sigma(\alpha_j)'$ (top level)	0.599	0.621	0.634
$s_j = \sigma(\alpha_j)'$ (all levels)	0.610	0.627	0.627
Fleiss κ	0.210	0.297	0.358

Table 6.9.: Pearson's correlation between explanation scores and human scores for the Amazon Clothes dataset.

quires contexts for humans to make decisions. For example, *upset*, *worthless*, and *disappointed* were surely for negative reviews. In contrast, *mobile*, *win*, and *call* could appear both in spam and non-spam texts. This caused higher disagreements in human answers though the model used these words certainly as evidence for the spam class. As a result, the human scores for the spam task were less reliable than the scores for the sentiment analysis task. Consequently, the overall correlations in Table 6.8 were also less than the scores in Table 6.9.

Hence, we focused on discussing the results in Table 6.9 with more reliable human scores. For each row, the correlation between the explanations and the human scores for patterns was lower than for samples and matched phrase. Hence, we should show not only the patterns but also some matched samples of the patterns to generate better plausible explanations. In addition, for TQBAF' and BQBAF', the strengths of top-level arguments $\sigma(\alpha_j)'$ were better than the base scores $\tau'(\alpha_j)$ in terms of the alignment with human judgement. The correlations were also significantly higher than FLX. This confirmed the advantage of the prominent feature of AXPLR, i.e., considering interactions between patterns when generating local explanations. However, by extending from arguments in the top level to all levels in the QBAF', only the correlations in BQBAF' remained high, while the correlations in TQBAF' dropped. Therefore, deep AXPLR, utilizing arguments of all levels in the graph, would go along better with BQBAF' than TQBAF'. It also implied that the base scores of the most specific patterns, which equaled their strengths in TQBAF', required some adjustments to align well with human judgement.

6.6. Experiment 3: Tutorial and Real-time Assistance

Among the three datasets, the deceptive review detection task is the most difficult tasks for humans. In this experiment, we follow Lai et al. (2020) to evaluate how effective AXPLR can be used to teach and support humans to perform deceptive review detection.

6.6.1. Participants and Procedure

We recruited participants via Amazon Mechanical Turk with the same set of required qualifications as in Experiment 2. After MTurk workers accepted to perform the task, we redirected them to our a survey created using Qualtrics[8]. The survey aimed to assess the capability of humans to detect deceptive hotel reviews before and after they learn from explanations. It consisted of five parts.

1. Attention-check questions (4 questions) – The participant needed to answer all the questions in this part correctly to proceed.

[8]`https://imperial.eu.qualtrics.com/`

2. Pre-test (10 questions) – For each question, the participant was asked whether a given hotel review was truthful or deceptive.

3. Tutorial (10 questions) – The format was the same as part 2, but then, we revealed to the participant the correct answer and the AI-generated prediction and explanation for them to learn from.

4. Post-test (20 questions) – For the first ten questions, the questions and the format were the same as part 2. We additionally showed what the participant had answered during the pre-test as a reference. The next ten questions were the same as the first ten except that we also provided AI explanations (without the predictions) for these questions, as *real-time assistance* (Lai et al., 2020). The format of the explanations was the same as what s/he had seen during the tutorial phase. The corresponding previous answer (from the first ten questions) was also provided when the participant answered each of the last ten questions.

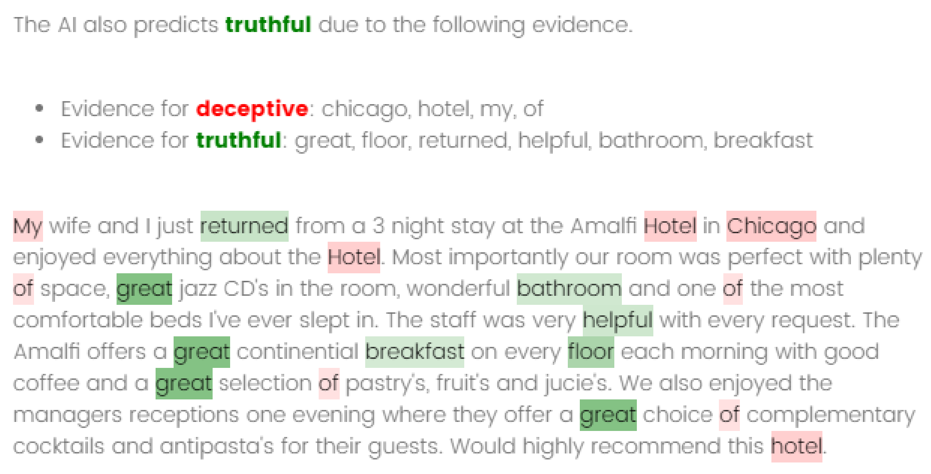
5. Additional questions (5 questions) – The participant was asked general questions before finishing the survey. These include, for example, how they detected deceptive and truthful reviews and any (free-text) feedback they might want to tell us.

At the end of the survey, each participant was given a Reference ID as a proof that s/he had completed the task (i.e., the HIT) for claiming the reward from the MTurk system. The improved performance of humans after being trained and assisted by the explanations showed how useful the explanations were. To motivate the participants to pay attention to the tasks, we divided the payment into two parts.

- A guaranteed reward ($2.00) was given after the participant completed the whole survey.

- A bonus reward – The participant was given an additional bonus reward of $0.10 for each question answered correctly (both in the pre-test and in the post-test). Therefore, the maximum bonus reward each participant could get was $0.10 x 30 = $3.00.

In total, we had 100 participants (to be explained in Section 6.6.3). At the end of the experiment, the average of the bonus payments was $1.756 per person, whereas the standard deviation was $0.333.


The AI also predicts **truthful** due to the following evidence.

- Evidence for **deceptive**: chicago, hotel, my, of
- Evidence for **truthful**: great, floor, returned, helpful, bathroom, breakfast

My wife and I just returned from a 3 night stay at the Amalfi Hotel in Chicago and enjoyed everything about the Hotel. Most importantly our room was perfect with plenty of space, great jazz CD's in the room, wonderful bathroom and one of the most comfortable beds I've ever slept in. The staff was very helpful with every request. The Amalfi offers a great continential breakfast on every floor each morning with good coffee and a great selection of pastry's, fruit's and jucie's. We also enjoyed the managers receptions one evening where they offer a great choice of complementary cocktails and antipasta's for their guests. Would highly recommend this hotel.

Figure 6.15.: Example of SVM explanation during the tutorial phase

6.6.2. Materials: Explanations

We compared four explanation methods in this experiment including SVM, FLX, shallow AXPLR, and deep AXPLR. We selected linear SVM since Lai et al. (2020) had shown that tutorials from simple models such as linear SVM worked better than tutorials from deep models such as BERT (Devlin et al., 2019). To train the SVM, we used TF-IDF vectorizer and employed exhaustive search to find the best hyperparameter $C \in \{1, 10, 100, 1000\}$. As a result, the model achieved the accuracy and the macro F1 of 0.891. We generated the explanations for the SVM model by showing the most important 10 words according to the absolute value of SVM coefficients. We also highlighted these words in text with the color and the intensity reflecting the sign and the magnitude of the coefficient, respectively. An example of SVM explanations during the tutorial phase is shown in Figure 6.15.

FLX, shallow AXPLR, and deep AXPLR were extracted from the same pattern-based LR model, of which the performance was shown in Table 6.4. Note that the LR model underperformed the SVM model, with the accuracy of 0.853 and 0.891, respectively. We decided to use BQBAF' for both shallow and deep AXPLR due to two reasons. First, the top-level arguments of BQBAF' provided more contexts than those of TQBAF', and the experiments in chapter 4 demonstrated that n-gram explanations were better than word-level explanations thanks to more contexts provided. Second, deep

AXPLR went along better with BQBAF' than TQBAF' as discussed in Section 6.5. Both FLX and shallow AXPLR showed top 10 patterns/arguments and share the same presentation, as shown in Figure 6.9. Deep AXPLR also started from the top 10 arguments but allowed the users to expand them to see attacking and supporting arguments, as shown in Figure 6.10. Moreover, we provided the input text with highlights similar to SVM explanations to help the users locate where the patterns appear in the input. The intensity of the highlight represented the sum of the explanation scores of all patterns that the word matched. For AXPLR, we summed the scores from only the top-level patterns as the scores from other levels had been aggregated into the top level.

6.6.3. Materials: Question Selection

For test questions, we randomly selected 50 questions from the test set of the Deceptive Hotel Reviews dataset. Then we partitioned them into five question sets. One participant was assigned one set of test questions and one explanation method (for tutorial and real-time assistance). Each pair of explanation method and question set was assigned to five people. Overall, we had 4 explanation methods × 5 question sets × 5 annotations = 100 surveys in total. So, we recruited exactly 100 participants on MTurk without allowing a participant to do the survey twice.

For the ten tutorial questions for each explanation method, we selected them from the development set of the Deceptive Hotel Reviews dataset using submodular pick (Ribeiro et al., 2016) to ensure that the selected examples covered important features of the task. Although submodular pick is a greedy algorithm, it provides a constant-factor approximation guarantee of $1 - e^{-1}$ to the optimum (Krause and Golovin, 2014). This made the tutorial questions different for each explanation method except that shallow AXPLR and deep AXPLR share the same set of tutorial questions.

6.6.4. Data Collection and Analysis

We are interested in the number of questions humans answered correctly in the pre-test, the post-test without real-time assistance, and the post-test with real-time assistance for different explanation methods. Particularly, the most interesting part is the jump from the pre-test scores to each of the two post-test scores. As each explanation method had 25 surveys (5 question sets × 5 annotations), we averaged the scores before comparing the results. Moreover, we compared the human scores to AI scores (i.e., accuracy scores of the SVM and the pattern-based LR models) to see how large the performance gap was after training and real-time assistance.

6.6.5. Results

The average scores of human participants are displayed in Table 6.10. Using the pre-test scores as a baseline, we observe that the tutorial phase only did not help the participants perform better as the post-test scores without real-time assistance were not significantly greater than the baseline. However, the real-time assistance after the tutorial indeed helped. By the approximate randomization test with 1,000 iterations and a significance level of 0.05 (Noreen, 1989; Graham et al., 2014), the post-test scores with real-time assistance from the explanations were significantly higher than the pre-test scores and the post-test scores with no assistance of the same explanation methods. Nevertheless, we see no significant difference across explanation groups, so we can conclude only that FLX, shallow AXPLR, and deep AXPLR are competitive with SVM for providing explanations to teach and support humans to detect deceptive reviews.

The last column in Table 6.10 shows the average performance of the underlying AI model on the same set of questions. SVM achieved 9 out of 10 in three question sets and 10 out of 10 in the other two, whereas the LR model (underlying FLX and AXPLR) got 7, 7, 8, 9, and 10 for the five question sets (regardless of the order). The total numbers of people that scored better than or equal to the AI during the pre-test, post-test with no assistance, and post-test with assistance are 6, 8, and 26 out of 100 people, respectively. This again shows the effectiveness of real-time assistance after the participants learned from the tutorial.

Finally, we asked the participants in the final part of the survey how they detected deceptive and truthful reviews. We manually selected interesting answers from the participants who got 8 correct answers or more during the post-test with AI assistance. The answers are shown in Tables 6.11 and 6.12. As expected, participants learning from SVM explanations rarely mentioned patterns but individual words. Some used the majority of high-

Explanation	Pre-test score	Post-test score		Model
		No assistance	+ assistance	
SVM	5.68±1.60	5.12±1.24	6.56±1.87	9.40±0.50
FLX	5.64±1.38	5.68±1.44	6.56±1.73	8.20±1.19
Shallow AX.	5.40±1.53	5.60±1.35	6.56±1.89	8.20±1.19
Deep AX.	5.24±1.36	5.44±1.61	6.76±1.79	8.20±1.19

Table 6.10.: Scores of the human participants (Average ± SD) in the tutorial and real-time assistance experiment using the Deceptive Hotel Reviews dataset. AX. stands for AXPLR. The last column shows the average score of the model that provides real-time assistance. The maximum score is 10.

lighting colors as a heuristic (which was surprisingly effective, probably due to the good performance of SVM). Since FLX was extracted from the LR model with GrASP patterns, we noticed some patterns and generalizations noted by participants who learned from FLX such as "when my was closely followed by 1 and hotel was followed by different words" and "the language used, and symbols and punctuation". Similarly, we also saw patterns noted by participants who learned from both types of AXPLR such as "It uses pronouns closely together" and "The patterns of specific words close together stood out, like luxury hotel.", as well as implicit patterns such as "There's also much less usage of city and hotel names". On the other hand, they could also cover word-level cues, as we can see from the comments like "I would also assume it was deceptive if the reviewer said "I" a lot.". However, there was no participant in the deep AXPLR group mentioning the usefulness of sub-patterns (which could be expanded or collapsed). Also, the average scores of both types of AXPLR were not significantly different. It could imply that shallow AXPLR is already sufficient for tutorial and real-time assistance, without the need to go deep. Last but not least, we found two interesting comments from the deep AXPLR group. One contrasted deceptive and truthful reviews – "If the review said "location" as apposed to naming the city, I was more likely to assume it was true, or if it mentioned the elevators or doormen. If it said "we" instead of "I" I was usually more inclined say it was truthful.". The other theorized the reason behind prominent patterns – "human phrasing that doesn't have hallmarks of being algorithmically generated or designed with the obvious intent to be picked up by a search engine (repeatedly mentioning the word Chicago was

Explanation	Score	Answer
SVM	10	If it seems too biased and sounds exaggerated.
	9	Certain key words are used repeatedly and unnaturally.
	8	If it has more red than green.
FLX	9	Extreme and/or superlative language.
	8	when my was closely followed by 1 and hotel was followed by different words
	8	because of the words used, and naming the location, etc
Shallow AXPLR	10	A city was not capitalized or the overuse and closeness of “my” and “I”.
	9	There were methods to look at the text or the type, as well as sentiment and identify some un natural responses. The patterns of specific words close together stood out, like luxury hotel.
	8	It uses pronouns closely together, uses proper names for hotels and cities oddly, and so on.
Deep AXPLR	9	The review mentioned the city by name a few times and was accompanied by odd sounding and separated facts.
	8	If the review kept mentioning the name of the city or referring to things as being luxurious or smelly, then I would generally assume that the review was deceptive. I would also assume it was deceptive if the reviewer said “I” a lot.
	8	It uses certain turns of phrase that are highly improbable or likely to come from a genuine human. Syntax issues can also be indicative of a deceptive review.

Table 6.11.: Some answers from the participants on how they knew that a review was **deceptive**. These answers were manually picked from the participants who got 8 correct answers or more during the post-test with AI assistance. We also show the explanation method they were assigned and the final scores they got.

one example of this used).”.

Explanation	Score	Answer
SVM	10	It sounds truthful and may sometimes talk about both the good and bad of the experience.
	9	Words are not frequently repeated and they are used in a natural manner.
	8	If it has more green than red
FLX	9	Down to earth. Pros and cons are expressed in a balanced, not hyperbolic way.
	8	when the text wasnt too long and sounded realistic
	8	the language used, and symbols and punctuation
Shallow AXPLR	10	The use of brackets or parentheses.
	9	The way the sentence was structured was far different then the other ones. The deceptive ones tried to appear truthful but the other ones just came off as natural.
	8	It describes the layout and number of things in a more detailed fashion. There's less of a focus on repetitious usage of pronouns. There's also much less usage of city and hotel names.
Deep AXPLR	9	The review spoke on a personal level and did mention city names many times.
	8	If the review said "location" as apposed to naming the city, I was more likely to assume it was true, or if it mentioned the elevators or doormen. If it said "we" instead of "I" I was usually more inclined say it was truthful. I also just payed attention to the overall vibe of the review.
	8	Review features ordinary, human phrasing that doesn't have hallmarks of being algorithmically generated or designed with the obvious intent to be picked up by a search engine (repeatedly mentioning the word Chicago was one example of this used).

Table 6.12.: Some answers from the participants on how they knew that a review was **truthful**. These answers were manually picked from the participants who got 8 correct answers or more during the post-test with AI assistance. We also show the explanation method they were assigned and the final scores they got.

6.6.6. Discussion

We may conclude from the results of Experiment 3 that AXPLR is competitive with SVM and FLX in terms of assisting humans in detecting deceptive reviews. Also, according to the qualitative analysis, AXPLR helps humans capture non-obvious patterns which are helpful to perform the task to some degree. Still, there is a gap between human performance and model performance as we can notice in the last two columns of Table 6.10. To narrow down this gap further, there are some interesting directions that could be explored. First, how could we make the tutorial part more effective? We hypothesize that submodular pick might not be the best method to select tutorial questions. In fact, Lai et al. (2020) have tried the *spaced repetition* strategy where humans are presented with important features repeatedly (with some space in-between). However, it cannot be concluded from their experiment that spaced repetition is significantly better than submodular pick when it comes to selecting tutorial examples. It would be interesting to study whether there is a better method to select and arrange tutorial questions for supporting human learning.

Additionally, in our experiment, AXPLR transformed QBAF' into a local input-based explanation, identifying important parts in the input together with the associated patterns. However, there are other forms of explanations which could be extracted from QBAF' and might be more suitable for this task. One is counterfactual explanation, showing which arguments should be added or removed from the current QBAF' in order to change the model prediction. This may help humans better learn relative importance of the patterns. It is likely possible to extract counterfactual explanation from our QBAF', in line with a recent work by Albini et al. (2021) extracting counterfactual explanations from argumentation frameworks for PageRank (Page et al., 1999). Besides, if needed, we could generate synthetic example(s) and/or QBAF'(s) to teach humans cases which are interesting but do not exist in the training data. For example, an input x_1 has a group of patterns strongly supporting the positive class, while an input x_2 has another group of patterns strongly supporting the negative class. What would happen if the two groups of patterns appeared in the same input? The answer to this question could aid humans in prioritizing knowledge learned from individual real examples. Combining groups of patterns is easier to do with AXPLR, but not FLX since FLX does not group related patterns together. Thus, overall, although AXPLR did not outperform existing methods significantly in this experiment, the experiment is a first step towards several possible extensions of AXPLR that may be worth exploring to better support human learning of new tasks.

6.7. General Considerations on AXPLR

In this section, we discuss complexity of AXPLR as well as its deployability in other settings.

6.7.1. Method Complexity

According to Section 6.2, AXPLR consists of four steps including QBAF extraction, computing dialectical strengths, post-processing, and generating explanations. Practically, if we want to explain many predictions of the same pattern-based LR classifier, we can pre-compute all possible arguments and relations of the classifier beforehand in order to speed up the local explanation generation phase. Let us analyze the complexity of both phases. For the pre-computation phase, assume we are given a trained LR classifier with d pattern features learned from GrASP. There are three steps we need to do. First, based on the parameters of the trained model, we initialize all possible arguments α_i with necessary information, i.e., the corresponding pattern p_i, the base score $\tau(\alpha_i)$, and the original supported class $c(\alpha_i)$. Since we have d pattern features, the runtime of this step is in $O(d)$. Second, we check specificity relations of all the arguments which will later become either attack or support relations in the QBAF. The number of argument pairs we need to check is in $O(d^2)$. The time complexity of the specificity check process depends on the method used. In our implementation, we employ the most basic method, finding a set of training examples matched by each corresponding pattern and checking if one set is a subset of the other.[9] For each pair of arguments, it takes a runtime in $O(|\mathcal{D}|)$ where $|D|$ is the number of training examples. Hence, the runtime of this second

[9]This method relies on the training dataset, so it may not be perfect especially when the training set is small. Checking specificity of two patterns formally is challenging as it needs to take into account the semantics of the attributes and the number of gaps allowed. We leave this problem as a possible future work.

step is in $O(d^2|\mathcal{D}|)$. Finally, we remove non-immediate specificity relations. To explain, for $p_i \succ p_k \succ p_j$, the second step also returns that $p_i \succ p_j$, but this relation cannot appear in the QBAF since there is p_k in between (according to the definitions of $\mathcal{R}^-$ and $\mathcal{R}^+$ (of both TQBAF and BQBAF) in Definition 10). This step takes $O(dE)$ runtime where E is the number of all specificity relations from the second step including the non-immediate ones. All in all, for the pre-computation phase, the runtime is in $O(d^2|\mathcal{D}| + dE)$. In general, the number of training examples $|\mathcal{D}|$ is significantly larger than the number of specificity relations E, making the second step dominate the whole pre-computation phase. Hence, we may conclude that the runtime is in $O(d^2|\mathcal{D}|)$. This number is quite large but still acceptable since the pre-computation phase is a one-time computation. However, we believe that with good data structures, this phase could be further optimized (though this optimization is not a focus of this thesis).

Concerning the local explanation generation phase, as AXPLR is a post-hoc explanation method, we can assume that the model has already computed the prediction of the target input x. Therefore, we know the feature vector $\mathbf{f} \in \{0,1\}^d$ of x. Let us analyze the four steps of AXPLR. First, to construct the QBAF, we need to find the set of arguments using $O(|\mathcal{A}|)$ runtime, find all the relations between pattern arguments using $O(|\mathcal{A}|^2)$ runtime, and connect some pattern arguments with the default argument using $O(|\mathcal{A}|)$ runtime. Thus, the overall runtime for QBAF construction is $O(|\mathcal{A}|^2)$. Next, to compute the dialectical strengths of the arguments, we perform topological sort using $O(|\mathcal{A}| + |\mathcal{R}|)$ runtime to find the order for strength computation and then compute the strength for all the arguments using $O(|\mathcal{A}|)$ runtime. In total, this step uses $O(|\mathcal{A}| + |\mathcal{R}|)$ runtime. After that, the post-processing step checks every argument to refine the supported class using $O(|\mathcal{A}|)$ runtime and checks every relation to refine the relation type (attack or support) using $O(|\mathcal{R}|)$ runtime. Hence, the post-processing step also takes $O(|\mathcal{A}| + |\mathcal{R}|)$ runtime. Lastly, for the explanation generation step, shallow AXPLR needs arguments in the top layer, so it uses $O(|\mathcal{A}|)$ runtime. Meanwhile, deep AXPLR needs to perform either breadth first search or depth first search to compute the hierarchical explanation, so it takes $O(|\mathcal{A}|+|\mathcal{R}|)$ runtime. In conclusion, most of the steps in this phase use $O(|\mathcal{A}| + |\mathcal{R}|)$ runtime except QBAF construction which uses $O(|\mathcal{A}|^2)$. When the graph is sparse, the runtime of QBAF construction can be reduced to

$O(|\mathcal{A}| + E_*)$ where E_* is the number of all possible relations of any QBAF for this LR model. In other words, E_* is E after the non-immediate specificity relations are removed, so $E_* \leq E$. As a result, for a sparse graph, $O(|\mathcal{A}| + E_*)$ also becomes the time complexity of this phase overall since the other steps use $O(|\mathcal{A}| + |\mathcal{R}|)$ and $|\mathcal{A}| - 1 \leq |\mathcal{R}| < |\mathcal{A}| - 1 + E_*$.

Given the size of actual $\mathcal{A}$ and $\mathcal{R}$ in Tables 6.5-6.7, it can be seen that the runtime for AXPLR generation would be fast for these three datasets. Increasing the number of features d affects mostly the pre-computation phase (computed once), but not much the local explanation generation phase (which relies more on $|\mathcal{A}|$ that is task-dependent). Concerning the size of the QBAF, the maximum number of $|\mathcal{A}|$ is $d+1$, occurring when all the patterns appear in the input text. Since the QBAF is a DAG, the maximum number of $|\mathcal{R}|$ is $0.5 \times |\mathcal{A}| \times (|\mathcal{A}| - 1)$, although it is usually a lot less in practice as GrASP tries to learn distinct patterns. Thus, overall, we believe that the complexity of deploying AXPLR is acceptable.

Concerning the user experience, AXPLR could be tailored to the cognitive requirements of users. In particular, even if the QBAF is large, we can indicate the numbers of arguments to be included in the final AXPLR in order to ensure that the presented information is not too overwhelming to the users.

6.7.2. AXPLR in Other Contexts

According to Experiment 2, AXPLR renders highly plausible explanations compared to FLX, the traditional explanation method of LR. One possible reason for AXPLR not shining in Experiment 3 is that plausibility is not necessary for the tutorial and real-time assistance task. The task only requires humans to learn and apply useful patterns though they may not know the reasons why such patterns are for the genuine class or the deceptive class. On the other hand, AXPLR would be more suitable for the task where plausibility is needed. For example, if we use the classifier as a decision support tool, we want the explanation to provide insights about the input text that align well with human reasoning. Even though the prediction is correct, if the explanation does not make sense, it is possible that the humans distrust the model and make a wrong final decision, leading to undesirable consequences.

Another context where AXPLR could be useful is explanation-based human debugging of the model. The individual model weight w_i for the pattern feature p_i may not make sense to humans when p_i is in fact related to other pattern features (as we can see in Experiment 2, where $\tau(\alpha_i)$ does not quite correlate with human reasoning). This may cause misunderstanding in the humans and pave the way to their feedback in a way that is harmful to the overall model performance. The QBAFs of AXPLR would provide a more accurate view of how the pattern features have been used by the model. So, we believe that it is more likely leading to a successful model debugging than FLX. Moreover, with the argumentative structure of AXPLR, it would be interesting to see whether and how AXPLR could let humans argue with the model, contributing to a richer way of human-AI collaboration for reversing an undesirable output or improving the model.

6.8. Summary

To generate local explanations for pattern-based logistic regression models, we proposed AXPLR, an explanation method enabled by quantitative bipolar argumentation frameworks we defined (TQBAFs and BQBAFs), capturing interactions among the patterns. We proved that the extracted and post-processed frameworks underpinning AXPLR are faithful to the LR model and satisfy many desirable properties. After that, we proposed two presentations of AXPLR, shallow and deep, specifying whether we present only the top-level arguments or all the arguments in the explanations. Concerning the experiments, we conducted both empirical and human studies. The former one discussed the statistics of the underlying argumentation frameworks for all input texts in the test sets and analyzed sufficiency of the explanations in terms of the number of supporting arguments needed. The latter one assessed whether AXPLR is more plausible and helpful for human learning than traditional explanation methods for pattern-based LR models. The results show that taking into account relations between arguments as AXPLR does indeed helps the explanations align better with human judgement, particularly in the sentiment analysis task. Though AXPLR performed competitively with traditional explanation methods in tutoring and supporting humans to detect deceptive hotel reviews, there were many participants learning from AXPLR that could recall well-generalized patterns and important but implicit patterns deemed useful for the task.

7. Conclusions

Human-AI collaboration is the key to many successful uses of AI, while explanations can enable effective collaboration between AIs and humans. In addition, NLP, particularly text classification, has tons of useful applications, many of which can benefit from explainability and human involvement. These motivated us to study *Explainable NLP for Human-AI Collaboration* in this thesis. We summarize our contributions to the field of explainable NLP towards human-AI collaboration in Section 7.1. Next, in Section 7.2, we discuss potential future work, where we highlight interesting and related open problems in the field and explain how our work has set the ground or provided foundations to address them.

7.1. Summary of Contributions

Our contributions align well with three research directions of explainable NLP – generating explanations, evaluating explanations, and applying explanations to downstream tasks (i.e., model debugging in our case) – while keeping the ultimate goal of fostering human-AI collaboration in mind.

7.1.1. Generating Explanations

Although there are existing explanation methods for any classifiers (e.g., LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017)) and for neural networks in general (e.g., integrated gradients (Sundararajan et al., 2017), LRP (Bach et al., 2015), and DeepLIFT (Shrikumar et al., 2017)), there has not been any explanation method that is applicable specifically to 1D CNN text classifiers used in many NLP applications. Therefore, we proposed two novel local explanation methods to fill this gap. One is Grad-CAM-Text which we adapted from Grad-CAM (Selvaraju et al., 2017) (originally devised for 2D CNNs in the computer vision domain) to work with 1D CNNs for text classification. The other method, DTs, uses a decision tree to mimic the classification part of the CNN and presents input n-grams corresponding to decision nodes in the tree as explanations. We also conducted experiments to evaluate these two new methods and found that they are good at justifying model predictions to humans.

Apart from that, we turned to focus on explanation methods for interpretable models. This topic has been rarely studied in the community because the models' inherent explanations are usually obvious and faithful. However, there is no guarantee that they will satisfy other desirable properties for explanations. So, we wanted to contribute to this area, focusing mainly on plausibility of the explanations (Wiegreffe and Pinter, 2019). In particular, we proposed a new local explanation method, namely AXPLR, for binary logistic regression using GrASP patterns (Shnarch et al., 2017) as features. We showed that inherent explanations for logistic regression sometimes do not align well with human judgement as they consider every pattern independently from each other although the patterns in fact are dialectical related. This motivated us to apply computational argumentation to model the relations between patterns and generate better explanations (i.e., AXPLR). We did so by *(i)* extracting a quantitative bipolar argumentation framework (QBAF) from the model and the input text; *(ii)* computing strengths of the arguments to take dialectical relations between arguments into account; *(iii)* post-processing the results to make the QBAF satisfy more desirable properties; and *(iv)* generating deep and shallow AXPLR explanations from the post-processed QBAF, making it suitable for human consumption. In the empirical evaluation, we analyzed statistics of the QBAFs for test examples of three binary text classification datasets. We also analyzed their sufficiency to help decide how many arguments we would need to show in AXPLR in general. Lastly, we conducted two human experiments. The first one showed that using the computed strengths of arguments to generate explanations helps increase the plausibility of the explanations. The second human experiment showed that AXPLR performed competitively with traditional explanation methods in tutoring and supporting humans to perform a task they are not much capable of (i.e., detect deceptive hotel reviews).

We invented two local explanation methods, adapted from Grad-CAM and Decision Trees, specifically for CNN text classifiers. Also, we proposed another local explanation method for pattern-based binary logistic regression, generating more plausible explanations using computational argumentation.

7.1.2. Evaluating Explanations

With many explanation methods proposed in the literature, evaluation and comparison of these methods are essential so that we can choose the right method for a task at hand. There have been existing works comparing post-hoc explanation methods in NLP focusing on intrinsic properties of the explanations (Poerner et al., 2018b; Nguyen, 2018; DeYoung et al., 2020). However, the community still lacks comparisons of various post-hoc explanation methods with respect to specific explanation usage (such as human decision support and model inspection) (Ribeiro et al., 2016; Lai and Tan, 2019). Therefore, we addressed this issue by proposing three human-grounded tasks to evaluate input-based explanations for text classification. Each of the tasks targets different purposes of explanation usage for human-AI collaboration. Task 1 aims to use explanations to reveal model behavior. So, we trained two classifiers, one is significantly better than the other and provided to humans local explanations for both classifiers on the same test examples with the same predictions. The explanation method is desirable if humans identify the better model correctly after seeing its explanation. Task 2 aims to use explanations to justify model predictions. So, we asked humans to select the most likely class for an unseen input text based only on top-m evidence texts chosen by explanation methods of interest. Answers from humans that match model predictions implied that the explanations were good to justify the predictions. Task 3 aims to use explanations to help humans make decisions when the model predictions are uncertain. Without revealing the full inputs, the explanation method is desirable if humans can answer the actual class correctly by looking at evidence and counter-evidence chosen by the explanations. Using 1D CNNs as target models, we evaluated nine forms of explanation methods (including Grad-CAM-Text and DTs we newly proposed) using the three proposed tasks. The results showed that none of the methods performed well on Task 1, LIME was the most prominent method for Task 2, and LRP at n-gram level worked fairly well on Task 3.

We proposed three novel human-grounded tasks to evaluate input-based explanations for text classification with respect to different purposes of explanation usage to support human-AI collaboration.

7.1.3. Explanation-Based Human Debugging

Because explanations provide insights about the underlying model, existing works have used them to enable model debugging via human feedback. These existing works, nonetheless, mostly rely on local explanations and iterative improvement, preventing the humans from perceiving the big picture of the model while providing feedback (Ribeiro et al., 2016; Teso and Kersting, 2019; Wu et al., 2019b). Also, none of them has explored the use of global explanations to debug deep NLP models. To narrow this gap, we proposed FIND, a human-in-the-loop debugging framework for simple deep text classifiers. FIND divides the classifiers into two parts – the feature extraction part (transforming an input text to a feature vector) and the classification part (applying a linear layer to predict the output from the feature vector). Given a trained model, first, FIND analyzes each neuron in the feature vector using LRP to understand the patterns it mainly captures. Then it uses a word cloud to present these patterns to the users and asks them whether the texts shown are relevant to the task or not. For a feature of which the word cloud is irrelevant to the task according to human feedback, it will be disabled using a masking matrix. Finally, the model is re-trained with the feature extraction part kept frozen so that the re-trained model uses only features deemed relevant to the task.

We conducted three human experiments on CNN text classifiers with three different bug scenarios (including small training data, training data with biases, and dataset shift) to demonstrate the usefulness of FIND. The results showed that some of the learned features in CNNs are more useful to the task than the others, and the word clouds generated by using LRP enabled humans to distinguish between the useful and the irrelevant (or even harmful) ones. These resulted in the improvement in the model predictive performance and the decrease in unintended biases in the model after we disabled irrelevant features and retrained the classification part of the model.

We made global explanations more actionable by using them to enable humans to switch off irrelevant features in simple deep text classifiers (i.e., CNNs) for fixing model bugs.

7.1.4. Broader Implications

Overall, this thesis encourages people who use or work on explainable AI to consider the next step after obtaining explanations. In particular, explanations for text classification can be utilized in many downstream applications, supporting human-AI collaboration. We demonstrated some of them in this thesis, e.g, revealing model behaviors to humans, assisting humans to investigate uncertain predictions, debugging the classification model via human feedback, and training humans to perform a challenging task. Nevertheless, our results show that some explanation methods may be more effective for some downstream tasks while performing poorly on other tasks. Hence, it is important to choose the method to use appropriately. Furthermore, despite many available explanation methods, there is room for new explanation methods that focus on specific desirable properties or applications. Even inherent explanations from interpretable models may not be the best option for every situation, as we can see from our experiments with pattern-based logistic regression models for example. All in all, research on the intersection of explainable AI and human-AI collaboration still has remaining challenges and open problems with regards to generating, evaluating, and utilizing explanations awaiting to be studied.

7.2. Future Work

We have already mentioned some items of future work in the previous chapters. In this section, we summarize major themes of the future work that are shared across our three main contributions.

7.2.1. Generalization to Other Models

Previously, we experimented on CNN text classifiers when we evaluated explanation methods (with our proposed human tasks) and demonstrated our debugging framework, FIND. Also, we touched upon debugging bidirectional LSTMs in Appendix C. In contrast, when it comes to AXPLR, we focused on explaining pattern-based logistic regression. One interesting question is "How can we make the proposed methods and the presented results in each chapter generalize to other models?"

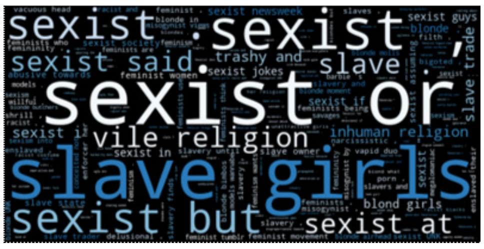
Human Evaluations of Explanation Methods. As the three evaluation tasks we proposed do not have any requirements concerning the underlying model(s), we believe that the three tasks are already generalizable beyond CNNs. In other words, the assumptions backing up the three tasks are model-agnostic (i.e., good explanations can reveal model behavior, justify the predictions, and help humans investigate uncertain predictions). However, whether the findings of our experiments are also generalizable beyond CNNs straightforwardly is not yet clear. So, in the future, it would be interesting to conduct the three evaluation tasks using other deep learning models as testbeds and compare the findings to what we have already known from CNNs. Future experiments may also include more recent explanation methods, e.g., contextual decomposition (Murdoch et al., 2018) and input marginalization (Kim et al., 2020), to monitor progress in the field.

Human-in-the-Loop Debugging. We have shown that FIND works effectively with CNNs. There are many factors for this success. First, CNN filters are not too complex to interpret, and a single word cloud is sufficient to capture the behavior of each filter. Second, it is clear where we should divide the model into M_f and M_c. Third, the CNN models we experimented on were not too large, so the human effort required to investigate and disable features was manageable. In the future, it would be very interesting to generalize FIND to models that are more complicated than CNNs, and the key challenge is how to carry these success factors of CNNs onwards.

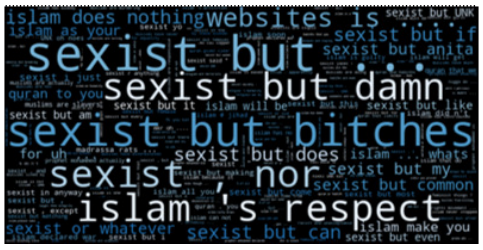
There are a few questions that are worth discussing in this regards. First, what is an effective way to understand each feature in other models? A single word cloud may not always be effective, especially when the feature value could be either positive or negative. In that situation, we need to consider not only patterns that the feature favors but also patterns that the feature reacts in the opposite way (i.e., leading to strong negative values). We exemplified this case with two word clouds representing each BiLSTM feature in Appendix C.1, and we plan to experiment with advanced visualizations such as LSTMVis (Strobelt et al., 2018) in the future. Second, can we make the model features more interpretable? For example, using ReLU as activation functions in LSTM cells (instead of tanh) renders the features non-negative. So, they can be summarized using one word cloud which is more practical for debugging. Third, which layer should we set as the target features $\mathbf{f}$ to be investigated and disabled? It may not be feasible to apply FIND to BERT-base (Devlin et al., 2019) at the standard feature representation level with 768 features since it would require too much human effort to investigate the features and provide feedback. A more proper level of debugging could be at the attention heads rather than individual features (Voita et al., 2019).

In general, the principle of FIND is understanding the features and then disabling the irrelevant ones. The process makes *visualizations* and *interpretability* more actionable. Over the past few years, we have seen rapid growth of scientific research in both topics (visualizations and interpretability) aiming to understand many emerging advanced models including the popular transformer-based models (Jo and Myaeng, 2020; Hoover et al., 2020). We believe that they are good materials for further enhancing FIND to support debugging of more complex models.

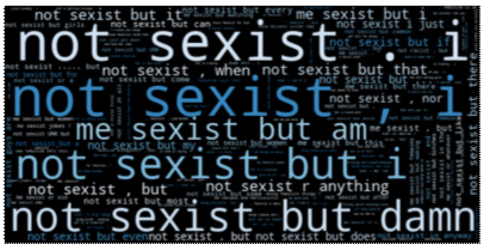
Argumentative Explanations for Text Classification. AXPLR is useful when features used by the model are not independent. Obviously, this is the case when we use GrASP patterns as features for logistic regression. However, we argue that the dependency between features could happen with deep learning models too. For example, Figure 7.1 shows word clouds of 4 out of 30 features of a CNN trained on the Waseem dataset (from one of our debugging experiments). We can see apparently that these features are not independent though they are not written in explicit forms, unlike the interpretable GrASP patterns. Still, we may approximate patterns from the word clouds as follows: feature 7 = [[`TEXT:sexist`]], feature 18 = [[`TEXT:sexist`], [`TEXT:but`]], feature 23 = [[`TEXT:not`], [`TEXT:sexist`], [`TEXT:but`]], feature 24 = [[`TEXT:i`], [`HYPERNYM:be`], [`TEXT:not`], [`TEXT:sexist`]]. Because the



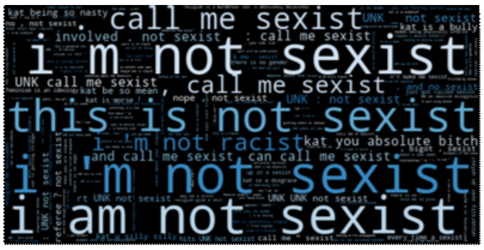
(a) Feature 7 (-0.210; 0.140)



(b) Feature 18 (0.080; 0.346)



(c) Feature 23 (-0.345; 0.065)



(d) Feature 24 (-0.463; -0.203)

Figure 7.1.: Word clouds of 4 out of 30 features of a CNN trained on the Waseem dataset in Experiment 2 of Chapter 5 (see Section 5.4). Each (x; y) pair is the weights of the feature in the last linear layer of the CNN model where x and y are the weights for the Not abusive and the Abusive classes, respectively.

last linear layer of the CNN is inherently interpretable, we believe that it is not impossible to extract QBAF from CNNs to generate AXPLR. However, open questions are "How to model the specificity relation between two CNN features given that the patterns are not explicit?" and "How to extract base scores from the model when it is not binary logistic regression?"[1]. Answering these questions are potential future work needed to generalize AXPLR beyond pattern-based logistic regression.

7.2.2. Knowledge Injection

Our FIND framework allows humans to disable learned features which are irrelevant from the model. Nevertheless, what we have not achieved using FIND is injecting new knowledge into the model via human feedback. For instance, in the dataset shift context, if a user had known the target dataset

[1]The last linear layer of the CNN in Figure 7.1 acts like a multiclass logistic regression since it uses the softmax function (rather than the sigmoid function) at the end.

we wanted to apply the model to, s/he would have wanted to suggest a new textual pattern to the model. Under the current FIND framework, the challenge of knowledge injection is twofold. First, how can we represent a piece of knowledge given by a human in the improved model? This is not obvious because the human represents the pattern in a discrete representation whereas the model represents patterns using CNN convolutional filters (with a continuous representation). Second, how can we set or train the weight for the newly added feature? The reason that the suggested pattern does not appear in the original model could be that the training data does not have a strong signal of this pattern. Even though we re-train the model containing the injected pattern on the training data again, there is no guarantee that the learned weight of the injected pattern will be accurate. In the context of interpretable models such as pattern-based logistic regression, which we used with AXPLR, the first challenge is resolved because humans can represent the knowledge to be injected using GrASP patterns. Still, the second challenge remains difficult since the weights of injected features cannot be properly learned using existing training data. Overall, we believe that injecting new knowledge is more difficult than removing learned knowledge as we did in FIND. Future work might need to incorporate techniques from other relevant research fields to enable knowledge injection using human feedback such as neuro-symbolic learning (Hilario, 1997) and knowledge integration into learning systems (von Rueden et al., 2021).

7.2.3. Reliable Comparison of Human Experiments

Human experiments are naturally difficult to replicate as they are inevitably affected by choices of user interfaces, phrasing, population, incentives, etc. (Lakkaraju et al., 2020). Further, research in machine learning rarely adopts practices from the human-computer interaction community (Abdul et al., 2018), limiting the possibility to compare across studies. For explanation-based human debugging, for example, most existing work including ours only considers model performance before and after debugging or compare the results among several configurations of a single proposed framework. This leads to little knowledge in the community about which explanation types or feedback mechanisms are more effective across several settings. Similar difficulties apply to evaluation of explanations as well. Thus, one promising research direction would be proposing a standard setup or a benchmark for evaluating and comparing explainable NLP techniques for human-AI tasks reliably across different settings.

Bibliography

Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y Lim, and Mohan Kankanhalli. 2018. Trends and trajectories for explainable, accountable and intelligible systems: An hci research agenda. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–18.

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2017. QUINT: Interpretable question answering over knowledge bases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–66, Copenhagen, Denmark. Association for Computational Linguistics.

Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160.

Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. 2018. innvestigate neural networks! *arXiv preprint arXiv:1808.04260*.

Emanuele Albini, Pietro Baroni, Antonio Rago, and Francesca Toni. 2021. Interpreting and explaining pagerank through argumentation semantics. *Intelligenza Artificiale*, 15(1):17–34.

Emanuele Albini, Piyawat Lertvittayakumjorn, Antonio Rago, and Francesca Toni. 2020. Dax: Deep argumentative explanation for neural networks. *arXiv preprint arXiv:2012.05766*.

Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262.

Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Leila Amgoud and Jonathan Ben-Naim. 2018. Evaluation of arguments in weighted bipolar graphs. *International Journal of Approximate Reasoning*, 99:39–55.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany. Association for Computational Linguistics.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. 2019. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*.

Katie Atkinson, Pietro Baroni, Massimiliano Giacomin, Anthony Hunter, Henry Prakken, Chris Reed, Guillermo Simari, Matthias Thimm, and Serena Villata. 2017. Towards artificial argumentation. *AI magazine*, 38(3):25–36.

Marco Aurisicchio, Pietro Baroni, Dario Pellegrini, and Francesca Toni. 2015. Comparing and integrating argumentation-based with matrix-based decision support in arg&dec. In *International Workshop on Theory and Applications of Formal Argumentation*, pages 1–20. Springer.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, Brussels, Belgium. Association for Computational Linguistics.

Pietro Baroni, Antonio Rago, and Francesca Toni. 2019. From fine-grained properties to broad principles for gradual argumentation: A principled spectrum. *International Journal of Approximate Reasoning*, 105:252–286.

Osbert Bastani, Carolyn Kim, and Hamsa Bastani. 2017. Interpretability via model extraction. *arXiv preprint arXiv:1706.09773*.

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

Trevor Bench-Capon, Henry Prakken, and Giovanni Sartor. 2009. Argumentation in legal reasoning. In *Argumentation in artificial intelligence*, pages 363–382. Springer.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Umang Bhatt, Adrian Weller, and José M. F. Moura. 2020a. Evaluating and aggregating feature-based model explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages

3016–3022. International Joint Conferences on Artificial Intelligence Organization. Main track.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. 2020b. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657.

Monica Bianchini and Franco Scarselli. 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8):1553–1565.

Jacob Bien and Robert Tibshirani. 2011. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424.

Or Biran and Kathleen R McKeown. 2017. Human-centric justification of machine learning predictions. In *IJCAI*, volume 2017, pages 1461–1467.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. 2016. A comparative study of ranking-based semantics for abstract argumentation. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Faeze Brahman, Vered Shwartz, Rachel Rudinger, and Yejin Choi. 2021. Learning to rationalize for nonmonotonic reasoning with distant supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12592–12601.

K. Branting, B. Weiss, B. Brown, C. Pfeifer, A. Chakraborty, L. Ferro, M. Pfaff, and A. Yeh. 2019. Semi-supervised methods for explainable legal prediction. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*, ICAIL '19, page 22–31, New York, NY, USA. Association for Computing Machinery.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. Classification and regression trees. *Wadsworth International Group*, 8:452–456.

Adrian Bussone, Simone Stumpf, and Dympna O'Sullivan. 2015. The role of explanations on trust and reliance in clinical decision support systems. In *2015 international conference on healthcare informatics*, pages 160–169. IEEE.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: natural language inference with natural language explanations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9560–9572.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD* *international conference on knowledge discovery and data mining*, pages 1721–1730.

Claudette Cayrol and Marie-Christine Lagasquie-Schiex. 2005. On the acceptability of arguments in bipolar argumentation frameworks. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 378–389. Springer.

Martin Chapman, Panagiotis Balatsoukas, Nadin Kökciyan, Kai Essers, Isabel Sassoon, Mark Ashworth, Vasa Curcin, Sanjay Modgil, Simon Parsons, and Elizabeth I Sklar. 2019. Computational argumentation-based clinical decision support. In *18th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2019*, pages 2345–2347. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. 2020. Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7530–7541, Online. Association for Computational Linguistics.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2018. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Minseok Cho, Gyeongbok Lee, and Seung-won Hwang. 2019. Explanatory and actionable debugging for machine learning: A tableqa demonstration. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1333–1336.

Oana Cocarascu, Antonio Rago, and Francesca Toni. 2019. Extracting dialogical explanations for review aggregations with argumentative dialogical agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1261–1269. Association for Computing Machinery.

Oana Cocarascu, Andria Stylianou, Kristijonas Čyras, and Francesca Toni. 2020. Data-empowered argumentation for dialectically explainable predictions. In *ECAI 2020*, pages 2449–2456. IOS Press.

Andrea Cohen, Sebastian Gottifredi, Alejandro J García, and Guillermo R Simari. 2014. A survey of different approaches to support in argumentation systems. *The Knowledge Engineering Review*, 29(5):513–550.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th*

Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Henriette Cramer, Vanessa Evers, Satyan Ramlal, Maarten Van Someren, Lloyd Rutledge, Natalia Stash, Lora Aroyo, and Bob Wielinga. 2008. The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-adapted interaction*, 18(5):455.

Kristijonas Čyras, Antonio Rago, Emanuele Albini, Pietro Baroni, and Francesca Toni. 2021. Argumentative xai: A survey. *arXiv preprint arXiv:2105.11266*.

Kristijonas Cyras, Ken Satoh, and Francesca Toni. 2016. Abstract argumentation for case-based reasoning. In *Fifteenth international conference on the principles of knowledge representation and reasoning*.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, Online. Association for Computational Linguistics.

Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. 2020. A survey of the state of explainable AI for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, Suzhou, China. Association for Computational Linguistics.

Shubhomoy Das, Travis Moore, Weng-Keen Wong, Simone Stumpf, Ian Oberst, Kevin McIntosh, and Margaret Burnett. 2013. End-user feature labeling: Supervised and semi-supervised approaches based on locally-weighted logistic regression. *Artificial Intelligence*, 204:56–74.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 53–59, Uppsala, Sweden. Association for Computational Linguistics.

Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 120–128, New York, NY, USA. Association for Computing Machinery.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. 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. Association for Computational Linguistics.

Adam Dejl, Peter He, Pranav Mangal, Hasan Mohsin, Bogdan Surdu, Eduard Voinea, Emanuele Albini, Piyawat Lertvittayakumjorn, Antonio Rago, and Francesca Toni. 2021. Argflow: A toolkit for deep argumentative explanations for neural networks. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, page 1761–1763, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Li Deng and Yang Liu. 2018. *Deep learning in natural language processing*. Springer.

Joseph F DeRose, Jiayao Wang, and Matthew Berger. 2020. Attention flows: Analyzing and comparing attention mechanisms in language models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1160–1170.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Shipi Dhanorkar, Yunyao Li, Lucian Popa, Kun Qian, Christine T Wolf, and Anbang Xu. 2020. Explainability for natural language processing. *AACL-IJCNLP 2020*.

Yannis Dimopoulos, Paul Bourret, and Sovan Lek. 1995. Use of some sensitivity criteria for choosing networks with good generalization ability. *Neural Processing Letters*, 2(6):1–4.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.

Phan Minh Dung. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. 2004. Least angle regression. *The Annals of statistics*, 32(2):407–499.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2009. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1.

Xiuyi Fan and Francesca Toni. 2015. On computing explanations in argumentation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Jinyue Feng, Chantal Shaib, and Frank Rudzicz. 2020. Explainable clinical decision support from text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1478–1489, Online. Association for Computational Linguistics.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. 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. Association for Computational Linguistics.

Lorenzo Ferrone and Fabio Massimo Zanzotto. 2020. Symbolic, distributed, and distributional representations for natural language processing in the era of deep learning: A survey. *Frontiers in Robotics and AI*, 6:153.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Tim Fountaine, Brian McCarthy, and Tamim Saleh. 2019. Building the ai-powered organization. *Harvard Business Review*, 97(4):62–73.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90, Vancouver, BC, Canada. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. 2014. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367–382.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. 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.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. 2014. Randomized significance tests in machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 266–274, Baltimore, Maryland, USA. Association for Computational Linguistics.

Filip Graliński, Anna Wróblewska, Tomasz Stanisławek, Kamil Grabowski, and Tomasz Górecki. 2019. GEval: Tool for debugging NLP datasets and models. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 254–262, Florence, Italy. Association for Computational Linguistics.

Han Guo, Nazneen Fatema Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2020. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

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

Xing Han and Joydeep Ghosh. 2020. Model-agnostic explanations using minimal forcing subsets. *arXiv preprint arXiv:2011.00639*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, Online. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.

Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic Rishabh Krishnan, and Dawn Song. 2020. Pretrained transformers improve out-of-distribution robustness. In *Association for Computational Linguistics*.

Melanie Hilario. 1997. An overview of strategies for neurosymbolic integration. *Connectionist-Symbolic Integration: From Unified to Hybrid Approaches*, pages 13–36.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. 2018. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*.

Pollawat Hongwimol, Peeranuth Kehasukcharoen, Pasit Laohawarutchai, Piyawat Lertvittayakumjorn, Aik Beng Ng, Zhangsheng Lai, Timothy Liu, and Peerapon Vateekul. 2021. ESRA: Explainable scientific research assistant. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 114–121, Online. Association for Computational Linguistics.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2020. exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 187–196, Online. Association for Computational Linguistics.

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

Alon Jacovi and Yoav Goldberg. 2021. Aligning Faithful Interpretations with their Social Attribution. *Transactions of the Association for Computational Linguistics*, 9:294–310.

Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. 2021. 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.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 56–65, Brussels, Belgium. Association for Computational Linguistics.

Ayush Jaiswal, Daniel Moyer, Greg Ver Steeg, Wael AbdAlmageed, and Premkumar Natarajan. 2020. Invariant representations through adversarial forgetting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4272–4279.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Jae-young Jo and Sung-Hyon Myaeng. 2020. Roles and utilization of attention heads in transformer-based neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3404–3417, Online. Association for Computational Linguistics.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado. Association for Computational Linguistics.

Patrick Juola. 2007. Future trends in authorship attribution. In *IFIP International Conference on Digital Forensics*, pages 119–132. Springer.

Dan Jurafsky and James H. Martin. 2020. Speech & language processing, 3rd edition.

Kasidis Kanwatchara, Thanapapas Horsuwan, Piyawat Lertvittayakumjorn, Boonserm Kijsirikul, and Peerapon Vateekul. 2021. Rational LAMOL: A rationale-based lifelong learning framework. 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 2942–2953, Online. Association for Computational Linguistics.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078.

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. 2019. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3382–3390. PMLR.

Siwon Kim, Jihun Yi, Eunji Kim, and Sungroh Yoon. 2020. Interpretation of NLP models through input marginalization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3154–3167, Online. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Andreas Krause and Daniel Golovin. 2014. Submodular function maximization. *Tractability*, 3:71–104.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*, pages 126–137.

Todd Kulesza, Simone Stumpf, Margaret Burnett, Weng-Keen Wong, Yann Riche, Travis Moore, Ian Oberst, Amber Shinsel, and Kevin McIntosh. 2010. Explanatory debugging: Supporting end-user debugging of machine-learned programs. In *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 41–48. IEEE.

Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. 2013. Too much, too little, or just right? ways explanations impact end users' mental models. In *2013 IEEE Symposium on visual languages and human centric computing*, pages 3–10. IEEE.

Todd Kulesza, Weng-Keen Wong, Simone Stumpf, Stephen Perona, Rachel White, Margaret M Burnett, Ian Oberst, and Andrew J Ko. 2009. Fixing the program my computer learned: Barriers for end users, challenges for the machine. In *Proceedings of the 14th international conference on Intelligent user interfaces*, pages 187–196.

Vivian Lai, Han Liu, and Chenhao Tan. 2020. "why is' chicago'deceptive?" towards building model-driven tutorials for humans. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Vivian Lai and Chenhao Tan. 2019. On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 29–38.

Himabindu Lakkaraju, Julius Adebayo, and Sameer Singh. 2020. Explaining machine learning predictions: State-of-the-art, challenges, and opportunities. NeurIPS 2020 Tutorial.

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. 2017. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*.

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. 2019. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

John Lawrence and Chris Reed. 2019. Argument mining: A survey. *Computational Linguistics*, 45(4):765–818.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Breiman Leo, Jerome H Friedman, Richard A Olshen, and Charles J Stone. 1984. Classification and regression trees. *Wadsworth International Group.*

Piyawat Lertvittayakumjorn, Daniele Bonadiman, and Saab Mansour. 2021a. Knowledge-driven slot constraints for goal-oriented dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3407–3419, Online. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn, Leshem Choshen, Eyal Shnarch, and Francesca Toni. 2021b. Grasp: A library for extracting and exploring human-interpretable textual patterns. *arXiv preprint arXiv:2104.03958.*

Piyawat Lertvittayakumjorn, Ivan Petej, Yang Gao, Yamuna Krishnamurthy, Anna Van Der Gaag, Robert Jago, and Kostas Stathis. 2021c. Supporting complaints investigation for nursing and midwifery regulatory agencies. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 81–91, Online. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn, Lucia Specia, and Francesca Toni. 2020. FIND: Human-in-the-Loop Debugging Deep Text Classifiers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 332–348, Online. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn and Francesca Toni. 2019. Human-grounded evaluations of explanation methods for text classification. 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 5195–5205, Hong Kong, China. Association for Computational Linguistics.

Piyawat Lertvittayakumjorn and Francesca Toni. 2021. Explanation-based human debugging of nlp models: A survey. *Transactions of the Association for Computational Linguistics (Accepted, forthcoming).*

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016a. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

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

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020a. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S Yu, and Lifang He. 2020b. A survey on text classification: From shallow to deep learning. *arXiv preprint arXiv:2008.00364*.

Tal Linzen. 2019. What can linguistics and deep learning contribute to each other? response to pater. *Language*, 95(1):e99–e108.

Frederick Liu and Besim Avci. 2019. Incorporating priors with feature attribution on text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6274–6283, Florence, Italy. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Hui Liu, Qingyu Yin, and William Yang Wang. 2019a. Towards explainable NLP: A generative explanation framework for text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5570–5581, Florence, Italy. Association for Computational Linguistics.

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

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774.

Oisin Mac Aodha, Shihan Su, Yuxin Chen, Pietro Perona, and Yisong Yue. 2018. Teaching categories to human learners with visual explanations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3820–3828.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Joseph E Mercado, Michael A Rupp, Jessie YC Chen, Michael J Barnes, Daniel Barber, and Katelyn Procci. 2016. Intelligent agent transparency in human–agent teaming for multi-uxv management. *Human factors*, 58(3):401–415.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In

Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38.

Sanjay Modgil and Martin Caminada. 2009. Proof theories and algorithms for abstract argumentation frameworks. In *Argumentation in artificial intelligence*, pages 105–129. Springer.

Sina Mohseni, Jeremy E Block, and Eric D Ragan. 2018. A human-grounded evaluation benchmark for local explanations of machine learning. *arXiv preprint arXiv:1801.05075*.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. 2019. Layer-wise relevance propagation: an overview. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209. Springer.

W James Murdoch, Peter J Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *International Conference on Learning Representations*.

Seil Na, Yo Joong Choe, Dong-Hyun Lee, and Gunhee Kim. 2019. Discovery of natural language concepts in individual units of cnns. In *International Conference on Learning Representations*.

Menaka Narayanan, Emily Chen, Jeffrey He, Been Kim, Sam Gershman, and Finale Doshi-Velez. 2018. How do humans understand explanations from machine learning systems? an evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1802.00682*.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. 2016. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *Advances in neural information processing systems*, 29:3387–3395.

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

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses.* Wiley New York.

Myle Ott, Claire Cardie, and Jeffrey T. Hancock. 2013. Negative deceptive opinion spam. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 497–501, Atlanta, Georgia. Association for Computational Linguistics.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA. Association for Computational Linguistics.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Devi Parikh and C Zitnick. 2011. Human-debugging of machines. *NIPS WCSSWC*, 2(7):3.

Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2018a. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8779–8788.

Ji Ho Park, Jamin Shin, and Pascale Fung. 2018b. Reducing gender bias in abusive language detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2799–2804, Brussels, Belgium. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, and Franco Turini. 2019. Meaningful explanations of black box ai decision systems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9780–9784.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Nina Poerner, Benjamin Roth, and Hinrich Schütze. 2018a. Interpretable textual neuron representations for NLP. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 325–327, Brussels, Belgium. Association for Computational Linguistics.

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018b. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Melbourne, Australia. Association for Computational Linguistics.

Nico Potyka. 2021. Interpreting neural networks as quantitative argumentation frameworks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6463–6470.

Pearl Pu and Li Chen. 2006. Trust building with explanation interfaces. In *Proceedings of the 11th international conference on Intelligent user interfaces*, pages 93–100.

B. Pustejovsky, J. Pustejovsky, B. Boguraev, and A.P.C.S.J. Pustejovsky. 1996. *Lexical Semantics: The Problem of Polysemy*. Clarendon paperbacks. Clarendon Press.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. 2009. *Dataset Shift in Machine Learning*. The MIT Press.

Antonio Rago, Francesca Toni, Marco Aurisicchio, and Pietro Baroni. 2016. Discontinuity-free decision support with quantitative argumentation debates. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

S Ransbotham, S Khodabandeh, D Kiron, F Candelon, M Chu, and B LaFountain. 2020. Expanding ai's impact with organizational learning. *MIT Sloan Management Review and Boston Consulting Group*.

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

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018a. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018b. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. 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. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Alexis Ross, Ana Marasović, and Matthew Peters. 2021. Explaining NLP models via minimal contrastive editing (MiCE). In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3840–3852, Online. Association for Computational Linguistics.

Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215.

Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Michal Walczak, Julius Pfrommer, Annika Pick, et al. 2021. Informed machine learning-a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. 2018. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services*, 1(1):39–48.

James Schaffer, John O'Donovan, James Michaelis, Adrienne Raglin, and Tobias Höllerer. 2019. I can do better than your ai: expertise and explanations. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages 240–251.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

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

Naziha Sendi, Nadia Abchiche-Mimouni, and Farida Zehraoui. 2019. A new transparent ensemble method based on deep learning. *Procedia Computer Science*, 159:271–280.

Mattia Setzu, Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. 2021. Glocalx-from local to global explanations of black box ai models. *Artificial Intelligence*, 294:103457.

Eyal Shnarch, Ran Levy, Vikas Raykar, and Noam Slonim. 2017. GRASP: Rich patterns for argumentation mining. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1345–1350, Copenhagen, Denmark. Association for Computational Linguistics.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153, International Convention Centre, Sydney, Australia. PMLR.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.

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

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

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

Alison Smith-Renner, Ron Fan, Melissa Birchfield, Tongshuang Wu, Jordan Boyd-Graber, Daniel S Weld, and Leah Findlater. 2020. No explainability without accountability: An empirical study of explanations and feedback in interactive ml. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Kacper Sokol and Peter Flach. 2020. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 56–67.

Neha Soni, Enakshi Khular Sharma, Narotam Singh, and Amita Kapoor. 2020. Artificial intelligence in business: From research and innovation to market deployment. *Procedia Computer Science*, 167:2200–2210.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2018. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676.

Simone Stumpf, Vidya Rajaram, Lida Li, Weng-Keen Wong, Margaret Burnett, Thomas Dietterich, Erin Sullivan, and Jonathan Herlocker. 2009. Interacting meaningfully with machine learning systems: Three experiments. *International journal of human-computer studies*, 67(8):639–662.

Purin Sukpanichnant, Antonio Rago, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. Lrp-based argumentative explanations for neural networks. In *Proceedings of the 2nd Italian Workshop on Explainable Artificial Intelligence*, pages 71–85.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Madhumita Sushil, Simon Šuster, and Walter Daelemans. 2018. Rule induction for global explanation of trained models. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 82–97, Brussels, Belgium. Association for Computational Linguistics.

Madhumita Sushil, Simon Suster, and Walter Daelemans. 2021. Contextual explanation rules for neural clinical classifiers. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 202–212, Online. Association for Computational Linguistics.

Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. 2009. Moviexplain: a recommender system with explanations. *RecSys*, 9:317–320.

Sarah Tan, Rich Caruana, Giles Hooker, and Yin Lou. 2018. Distill-and-compare: Auditing black-box models using transparent model distillation. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 303–310.

Zheng Tang and Mihai Surdeanu. 2021. Interpretability rules: Jointly bootstrapping a neural relation extractorwith an explanation decoder. In *Proceedings of the First Workshop on Trustworthy Natural Language Processing*, pages 1–7, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118, Online. Association for Computational Linguistics.

Stefano Teso and Kristian Kersting. 2019. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 239–245.

Nithum Thain, Lucas Dixon, and Ellery Wulczyn. 2017. Wikipedia talk labels: Toxicity.

Alexandros Vassiliades, Nick Bassiliades, and Theodore Patkos. 2021. Argumentation and explainable artificial intelligence: a survey. *The Knowledge Engineering Review*, 36.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Douglas Walton. 2009. Argumentation theory: A very short introduction. In *Argumentation in artificial intelligence*, pages 1–22. Springer.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019.*

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Sholom M Weiss, Nitin Indurkhya, Tong Zhang, and Fred Damerau. 2010. *Text mining: predictive methods for analyzing unstructured information.* Springer Science & Business Media.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of Abusive Language: the Problem of Biased Datasets. 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 602–608, Minneapolis, Minnesota. Association for Computational Linguistics.

Sarah Wiegreffe and Ana Marasović. 2021. Teach me to explain: A review of datasets for explainable nlp. *arXiv preprint arXiv:2102.12060.*

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. 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 11–20, Hong Kong, China. Association for Computational Linguistics.

Chao Wu, Guolong Wang, Jiangcheng Zhu, Piyawat Lertvittayakumjorn, Simon Hu, Chilie Tan, Hong Mi, Yadan Xu, and Jun Xiao. 2019a. Exploratory analysis for big social data using deep network. *IEEE Access*, 7:21446–21453.

Tongshuang Wu, Daniel S Weld, and Jeffrey Heer. 2019b. Local decision pitfalls in interactive machine learning: An investigation into feature selection in sentiment analysis. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 26(4):1–27.

Wenting Xiong, Iftitahu Ni'mah, Juan MG Huesca, Werner van Ipenburg, Jan Veldsink, and Mykola Pechenizkiy. 2018. Looking deeper into deep learning model: Attribution-based explanations of textcnn. *arXiv preprint arXiv:1811.03970*.

Linyi Yang, Eoin Kenny, Tin Lok James Ng, Yi Yang, Barry Smyth, and Ruihai Dong. 2020. Generating plausible counterfactual explanations for deep transformers in financial text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6150–6160, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Huihan Yao, Ying Chen, Qinyuan Ye, Xisen Jin, and Xiang Ren. 2021. Refining neural networks with compositional explanations. *arXiv preprint arXiv:2103.10415*.

Roozbeh Yousefzadeh and Dianne P O'Leary. 2019. Debugging trained machine learning models using flip points. In *ICLR 2019 Debugging Machine Learning Models Workshop*.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 260–267, Rochester, New York. Association for Computational Linguistics.

Jeffrey C Zemla, Steven Sloman, Christos Bechlivanidis, and David A Lagnado. 2017. Evaluating everyday explanations. *Psychonomic bulletin & review*, 24(5):1488–1500.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. 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 1031–1040, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Yunfeng Zhang, Q Vera Liao, and Rachel KE Bellamy. 2020. Effect of confidence and explanation on accuracy and trust calibration in ai-assisted decision making. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 295–305.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, New Orleans, Louisiana. Association for Computational Linguistics.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929.

Hugo Zylberajch, Piyawat Lertvittayakumjorn, and Francesca Toni. 2021. HILDIF: Interactive debugging of NLI models using influence functions. In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 1–6, Online. Association for Computational Linguistics.

Appendix

A. Additional Background

A.1. Standard Metrics for Text Classification

Apart from *accuracy* or *classification rate*, which is the percentage of correct predictions, we can evaluate the model performance for each specific class $c \in \mathcal{C}$ using the class *precision*, *recall*, and *F-measure* (mostly F1), defined as follows (Jurafsky and Martin, 2020, chapter 4).

$$\text{Precision}_c = P_c = \frac{TP_c}{TP_c + FP_c} = \frac{|\{(x_i, y_i) \in \mathcal{D}' | \hat{y}_i = y_i = c\}|}{|\{(x_i, y_i) \in \mathcal{D}' | \hat{y}_i = c\}|} \tag{A.1}$$

$$\text{Recall}_c = R_c = \frac{TP_c}{TP_c + FN_c} = \frac{|\{(x_i, y_i) \in \mathcal{D}' | \hat{y}_i = y_i = c\}|}{|\{(x_i, y_i) \in \mathcal{D}' | y_i = c\}|} \tag{A.2}$$

$$\text{F}_{\beta,c} = \frac{(\beta^2 + 1) P_c R_c}{\beta^2 P_c + R_c} \tag{A.3}$$

$$\text{F1}_c = \frac{2 P_c R_c}{P_c + R_c} \text{ (i.e., Eq. A.3 with } \beta = 1) \tag{A.4}$$

where TP_c (true positives) and FP_c (false positives) are the number of examples predicted as class c by the classifier that are correct and incorrect predictions, respectively. Besides, FN_c (false negatives) is the number of examples with class c as their true label but which the model does not predict correctly.

There are two ways to aggregate the class-specific metrics to be the metrics for the overall performance. First, *micro-averaging* combines the TP,

FP, and FN of all classes before computing precision and recall.

$$\text{Micro Precision} = \frac{\sum_{c\in\mathcal{C}} TP_c}{\sum_{c\in\mathcal{C}} TP_c + \sum_{c\in\mathcal{C}} FP_c} \tag{A.5}$$

$$\text{Micro Recall} = \frac{\sum_{c\in\mathcal{C}} TP_c}{\sum_{c\in\mathcal{C}} TP_c + \sum_{c\in\mathcal{C}} FN_c} \tag{A.6}$$

$$\text{Micro F1} = \frac{2 \times \text{Micro Precision} \times \text{Micro Recall}}{\text{Micro Precision} + \text{Micro Recall}} \tag{A.7}$$

Second, *macro-averaging* averages out precision and recall scores of all the classes so all the classes are weighted equally regardless of their size.

$$\text{Macro Precision} = \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} P_c \tag{A.8}$$

$$\text{Macro Recall} = \frac{1}{|\mathcal{C}|} \sum_{c\in\mathcal{C}} R_c \tag{A.9}$$

$$\text{Macro F1} = \frac{2 \times \text{Macro Precision} \times \text{Macro Recall}}{\text{Macro Precision} + \text{Macro Recall}} \tag{A.10}$$

Normally, when we work with datasets that are class imbalanced, we want the model to work well for all the classes, not just for the majority class. Therefore, we often use macro F1 as the main evaluation metric in addition to the classification accuracy.

A.2. Deep Learning for NLP

Deep learning is a subfield of machine learning that uses a composition of multiple non-linear modules (layers) so as to learn effective representations of inputs for achieving a given target task (LeCun et al., 2015). Unlike traditional machine learning methods, deep learning does not require humans to perform feature extraction or feature engineering because it can directly learn from raw input data (i.e., input texts in our case). The composition of multiple non-linear layers will transform an input text into multiple level of representations before the final output is produced at the last layer. In the last decade, deep learning has become a mainstream approach of many research areas including NLP (Deng and Liu, 2018) since there are sufficient amount of data and computing resources (such as GPUs) to train such complicated models, many of which can achieve state-of-the-art performance in plenty of NLP tasks (Zhang et al., 2015; Wang et al., 2019). One of the important concepts of deep learning for text classification is word embeddings.

Word Embeddings

Traditional NLP methods (before the era of deep learning) usually represent words using *discrete representation* where each word corresponds to one dimension in a vector of which the size equals the number of words in vocabulary. This results in very sparse word vectors which are unable to capture the degree of similarity and difference among words. Furthermore, the discrete representation cannot deal with new words or changes of word meanings effectively.

The deep learning approach does not adopt the discrete representation but *distributed representation* to overcome these issues. The distributed representation represents each word with a dense vector of ~50–1,000 dimensions. These vectors are known as *word embeddings* and learned from large text corpora so that similar words appear closely in the vector space (Ferrone and Zanzotto, 2020). The learning process relies on the *distributional semantics hypothesis*, proposing that words used in similar contexts tend to possess similar meanings (Harris, 1954).

Different word embedding methods realize the distributional semantics hypothesis in different ways. Mikolov et al. (2013) proposed **word2vec** with two neural architectures to learn word embeddings. One is the *continuous bag-of-words* model (CBOW) aiming to predict a target word from given context words of a fixed window size. The other is the *skip-gram* model aiming to predict context words given a target word. Training examples of both architectures can be prepared from any corpus of running texts. Word embeddings are then obtained from the learned weight matrices of these architectures. Bojanowski et al. (2017) proposed **fastText**, further improving word2vec to handle out-of-vocabulary words. fastText represents a word with its character n-grams and learns embeddings of all the character n-grams in the corpus using word2vec (skip-gram). The embedding of a given word then equals the sum of the embeddings of its character n-grams.

Another well-known word embedding method is **GloVe** (Global Vectors for Word Representation) (Pennington et al., 2014). GloVe creates, from a large corpus, a word-word co-occurrence matrix $\mathbf{X} \in \mathbb{R}^{V \times V}$ where V is the number of words in vocabulary and $\mathbf{X}_{ij}$ is the number of times word j occurs in the context of word i (within a fixed window size). Next, it creates and optimizes two matrices $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{V \times d}$ so that $\mathbf{u}_i^T \mathbf{v}_j$ (plus some learned biases) can be used to estimate $\log \mathbf{X}_{ij}$. Then the embedding of word i is $\mathbf{u}_i + \mathbf{v}_i \in \mathbb{R}^d$.

Word embeddings trained on general corpora (e.g., Wikipedia[1] and Common Crawl[2]) are applicable to any downstream applications. Although the training process may take a long time, it is a one-off effort and the results can be shared and reused widely. Word2vec[3], fastText[4], and GloVe[5] all provide pre-trained embeddings for the public to download and use.

A.3. Recurrent Neural Networks

Apart from CNNs discussed in Section 2.1.2, **Recurrent Neural Networks (RNNs)** are deep learning models that have shown good performance in many NLP tasks due to their capability to process sequential data (Mikolov et al., 2010). As shown in Figure A.1, an RNN starts from an initial hidden state $\mathbf{h}_0 \in \mathbb{R}^m$. For each time step t, it reads an input vector $\mathbf{x}_t \in \mathbb{R}^d$ (i.e., the embedding of the input word at time t) and processes it together with the hidden state at the previous time step $\mathbf{h}_{t-1} \in \mathbb{R}^m$ to obtain the updated hidden state at time t, i.e., $\mathbf{h}_t \in \mathbb{R}^m$. Mathematically,

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + b) \tag{A.11}$$

where $\mathbf{W} \in \mathbb{R}^{m \times d}$ and $\mathbf{U} \in \mathbb{R}^{m \times m}$ are weight matrices, $\mathbf{b}$ is a bias vector, and g is a non-linear activation function (e.g., sigmoid function).

For the tasks where we want an output out of every time step, we can apply a linear layer on top of $\mathbf{h}_t$ to get it ($\mathbf{out}_t$).

$$\mathbf{out}_t = \text{softmax}(\mathbf{W}_o\mathbf{h}_t + b_o) \tag{A.12}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are the weight matrix and the bias vector for the output

[1]`https://en.wikipedia.org/`
[2]`https://commoncrawl.org/`
[3]`https://code.google.com/archive/p/word2vec/`
[4]`https://github.com/facebookresearch/fastText`
[5]`https://nlp.stanford.edu/projects/glove/`

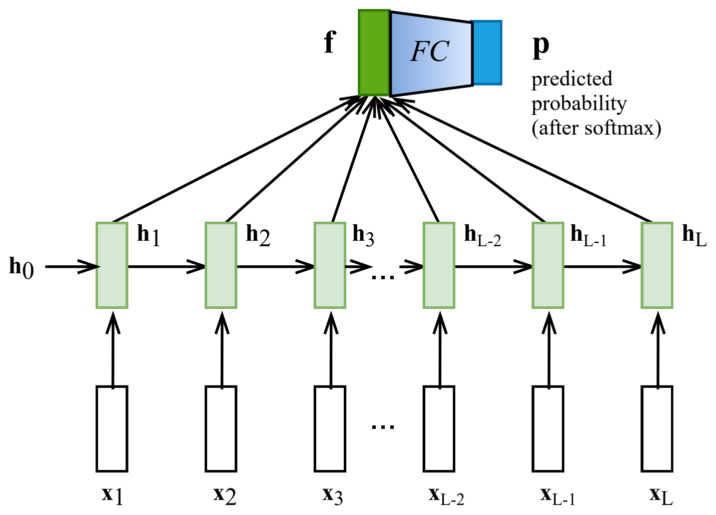


Figure A.1.: An RNN for text classification.

layer, respectively. In contrast, for the task where we want only one output per sequence (such as text classification), we need to derive one feature vector $\mathbf{f}$ for the input from all the hidden states $\mathbf{h}_1, ..., \mathbf{h}_t$. Possible ways to do so include *(i)* using only the last hidden state ($\mathbf{h}_t$) as $\mathbf{f}$; *(ii)* averaging all the hidden states to be $\mathbf{f}$; and *(iii)* pooling the maximum value for each dimension of the hidden states to be $\mathbf{f}$. Then we can perform classification with $\mathbf{f}$ in the same way as we do using Equations 2.1 or 2.5.

To train an RNN, we perform backpropagation through time (BPTT), i.e., unfolding the recurrent network into a chain of (non-)linear transformations with shared parameters and backpropagating the gradient through the chain. However, when the input text is too long, RNNs usually suffer from the vanishing gradient problem, i.e., the gradient from faraway becomes smaller and smaller when propagated back through time and the parameter updates for long-term dependencies become insignificant due to the too small gradient (Bengio et al., 1994). A popular solution to tackle this problem is using *gated RNNs* which learn when to forget or preserve the old state. With their additive gradient structure and gating mechanism, gated RNNs can mitigate the gradient vanishing problem. Next, we review a class of gated RNNs which is the long short-term memory (LSTM) model used in the thesis.

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber,

1997) is a type of RNN where there are, at the end of each time step t, the *hidden state* $\mathbf{h}_t \in \mathbb{R}^m$ and the *cell state* $\mathbf{c}_t \in \mathbb{R}^m$. To obtain those states, after reading the input vector $\mathbf{x}_t$, the new cell content $\widetilde{\mathbf{c}}_t \in \mathbb{R}^m$ is computed, and it will then be used, together with the previous cell state $\mathbf{c}_{t-1}$, to compute $\mathbf{c}_t$. In particular, $\mathbf{c}_t$ is a combination of $\widetilde{\mathbf{c}}_t$ and $\mathbf{c}_{t-1}$, controlled by the input gate $\mathbf{i}_t \in \mathbb{R}^m$ and the forget gate $\mathbf{f}_t \in \mathbb{R}^m$, respectively. Formally,

$$\widetilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right) \tag{A.13}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}}_t \tag{A.14}$$

The cell state $\mathbf{c}_t$ carries long-term information, some of which is passed to the hidden state $\mathbf{h}_t$, and this is controlled by the output gate $\mathbf{o}_t \in \mathbb{R}^m$:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right) \tag{A.15}$$

The values of all the three gates depend on the input vector $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. Specifically,

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)
\end{aligned} \tag{A.16}$$

To sum up, $\mathbf{W}_*$, $\mathbf{U}_*$, and $\mathbf{b}_*$ (for $* \in \{i, f, o, c\}$) are the parameters of the LSTM model that need to be learned using the training data. Each element in the gate vectors stays within the range (0, 1) as a result of the sigmoid function σ. While the vanilla RNNs always update the hidden states with the new input vectors, the gates in LSTMs help the model decide when to keep or forget the old information and when to update it with the new information, also mitigating the gradient vanishing problem.

One weakness of RNNs in general and LSTMs in particular is that the hidden state at time t relies only on the left context (i.e., the input words at the time steps $t, t-1, ...$) although, in practice, the meaning of the word at time t can also be influenced by the right context. To effectively utilize the right context, we could use the **bidirectional** architecture (Schuster and Paliwal, 1997) consisting of two RNNs, as displayed in Figure A.2. One of them (i.e., the forward RNN) reads the input text from left to right,

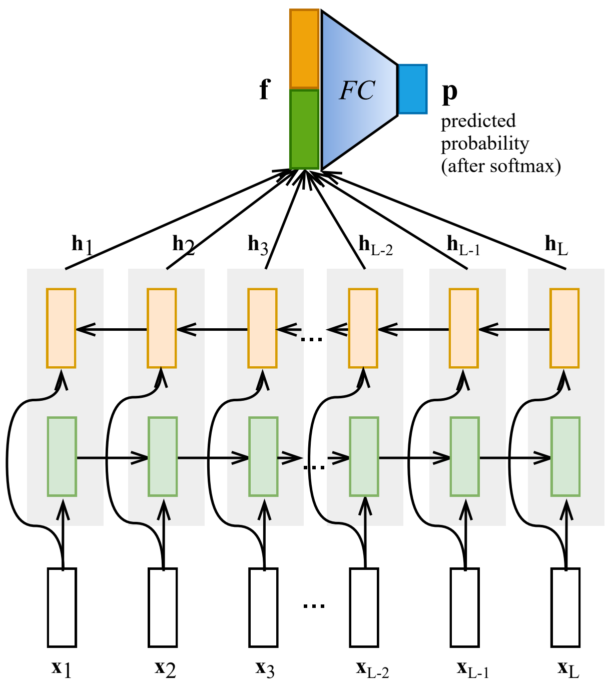


Figure A.2.: A bidirectional RNN for text classification.

whereas the other (i.e., the backward RNN) reads from right to left. The final hidden state $\mathbf{h}_t$ is the concatenation of the hidden states from the two RNNs:

$$\begin{aligned}
\overrightarrow{\mathbf{h}}_t &= \text{RNN}_{FW}(\mathbf{h}_{t-1}, \mathbf{x}_t) \\
\overleftarrow{\mathbf{h}}_t &= \text{RNN}_{BW}(\mathbf{h}_{t+1}, \mathbf{x}_t) \\
\mathbf{h}_t &= [\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t]
\end{aligned} \tag{A.17}$$

where RNN_{FW} and RNN_{BW} are the forward and backward RNNs respectively, and they do not share weights. Note that these RNNs could be vanilla RNNs, LSTMs, or other gated RNNs such as GRUs (Cho et al., 2014).

A.4. LIME

To explain a given example, Local Interpretable Model-agnostic Explanations (LIME) (Ribeiro et al., 2016) generates neighbor instances of the example (by perturbing the input text) and learns an interpretable model over interpretable features (such as words) to mimic the behavior of the original classifier with respect to this set of neighbor examples (including the target example to be explained). Therefore, the learned interpretable model is locally faithful to the original classifier and can be used to explain the prediction. To be precise, LIME for text classification works in three steps.

Given an input text x and the classifier M, first, LIME perturbs x for k times to obtain $z_1, \dots z_k$ where z_i is x with some words randomly selected to be removed or replaced with a special unknown token. Hence, some z_i could be more similar to the original x than others. We can define a distance function $\pi_x : \text{Text} \to [0,1]$ to measure the distance between z_i to x. The higher $\pi_x(z_i)$ is, the more similar x and z_i are. One possible distance function is $\pi_x(z_i) = 1 - \frac{n_r}{n_x}$ where n_r is the number of tokens in x being replaced/removed to generate z_i and n_x is the number of all tokens in x. Regardless of the distance $\pi_x(z_i)$, we call $\mathcal{Z} = \{z_i | i \in \{1, \dots, k\}\}$ a set of neighbor examples of x.

The second step is to apply the model M to $\mathcal{Z}$ to know how the model works on the neighbor examples of x. After this step, LIME obtains $m(z_i)$, i.e., the prediction of M on z_i before softmax. In the third step, LIME trains a linear regression classifier g by optimizing

$$\mathcal{L} = \sum_{z_i \in \mathcal{Z}} \left(\pi_x(z_i)(m(z_i) - g(z_i))^2 \right) + \Omega(g) \tag{A.18}$$

where $\Omega(g)$ is the regularization term of g, enforcing that g should not use too many features. Here, LIME uses K-LASSO (Efron et al., 2004) as Ω. Using this objective function and least squares optimization, g will behave similarly to m especially for neighbor examples staying closer to x (i.e., locally faithful to M). Since g is inherently interpretable, we can use the weights of g as the local explanation, representing how M sees the importance of words in the input x. Because LIME perturbs the input x to understand how M responses to changes in the input, LIME can be categorized as a *perturbation-based method* for computing input-based explanations.

A.5. DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) is also an input-based explanation method relying on relevance propagation (Shrikumar et al., 2017).

However, what makes DeepLIFT different from LRP is that DeepLIFT sets up a reference input $\bar{x}$ where the output computed by the model M is $\bar{y}$. Then, to compute the relevance scores for input features x_i in the target example x where the output is y, DeepLIFT back propagates $\Delta y = y - \bar{y}$ to find the contribution of each $\Delta x_i = x_i - \bar{x_i}$. In other words, the relevance score of the input x_i is the contribution of Δx_i towards Δy, i.e., $R_{\Delta x_i \leftarrow \Delta y}$. At the end of the day, $\sum_i R_{\Delta x_i \leftarrow \Delta y} = \Delta y$.

Again, consider a neuron k whose value is computed using n neurons in the previous layer as in Equation 2.12. While applying the model to the reference input, we obtain

$$\bar{a_k} = g(\sum_{j=1}^{n} \bar{a_j} w_{jk} + b_k). \tag{A.19}$$

Because the activation function g does not change the relevance score during back propagation, we can disregard it for now and obtain

$$\begin{aligned} \Delta a_k = a_k - \bar{a_k} &= (\sum_{j=1}^{n} a_j w_{jk} + b_k) - (\sum_{j=1}^{n} \bar{a_j} w_{jk} + b_k) \\ &= \sum_{j=1}^{n} \Delta a_j w_{jk}. \end{aligned} \tag{A.20}$$

Therefore, in terms of the relevance scores, we get

$$\begin{aligned} R_{\Delta a_j \leftarrow \Delta a_k} &= \frac{\Delta a_j w_{jk}}{\sum_{j'=1}^{n} \Delta a_{j'} w_{j'k}} \Delta a_k \\ &= \frac{a_j w_{jk} - \bar{a_j} w_{jk}}{\sum_{j'=1}^{n} \left(a_{j'} w_{j'k} - \bar{a_{j'}} w_{j'k} \right)} \Delta a_k \\ &= \frac{a_j w_{jk} - \bar{a_j} w_{jk}}{\sum_{j'=1}^{n} a_{j'} w_{j'k} - \sum_{j'=1}^{n} \bar{a_{j'}} w_{j'k}} \Delta a_k. \end{aligned} \tag{A.21}$$

Again, we can obtain the relevance score of Δa_j in total by summing up $R_{\Delta a_j \leftarrow \Delta a_k}$ for all k in the next layer. If we back propagate until j is at the input layer, we will obtain the relevance score of the input feature j as desired.

B. Human-Grounded Evaluations of Explanation Methods

B.1. Examples of the Explanations

Example 1: Amazon Dataset, (Actual: Positive, Predicted: Negative)

"Source hip hop hits Volume 3: THe songs listed aren't even on the CD! I bought it for Bling Bling and it wasn't on the CD. the other songs are good, but not what I was looking for. Amazon needs to get the info right on this listing."

Top-5 evidence texts

- Random (W): . / get / hip / was / I
- Random (N): the CD ! I / the CD / needs to get the / info right on this / for .
- LIME: not / bought / 3 / info / Bling
- LRP (W): it / bought / . / listed / :
- LRP (N): ! I bought it / : THe songs listed / was looking for . / right on this listing / not what I
- DeepLIFT (W): it / bought / . / listed / :
- DeepLIFT (N): ! I bought it / : THe songs listed / was looking for . / right on this listing / not what I
- Grad-CAM-Text: n't even on the / not what I was / hits Volume 3 : / CD ! I / . Amazon needs to
- DTs: n't even on the / CD ! I

Example 2: ArXiv Dataset, (Actual: Physics (PH), Predicted: Computer Science (CS))

"Multiple-valued Logic (MVL) circuits are one of the most attractive applications of the Monostable-to-Multistable transition Logic (MML), and they are on the basis of advanced circuits for communications. The operation of such quantizer has two steps : sampling and holding. Once the quantizer samples the signal, it must maintain the sampled value even if the input changes. However, holding property is not inherent to MML circuit topologies. This paper analyses the case of an MML ternary inverter used as a quantizer, and determines the relations that circuit representative parameters must verify to avoid this malfunction."

Top-5 evidence texts

- Random (W): not / This / one / basis / MML
- Random (N):) , and / , holding property is / are one of / sampled value even / circuit topologies
- LIME: paper / Logic / circuits / communications / applications
- LRP (W): paper / - / communications / topologies / the
- LRP (N): topologies . This paper / to - Multistable transition / valued Logic (MVL / circuits for communications . / the quantizer samples the
- DeepLIFT (W): paper / - / communications / Logic / the
- DeepLIFT (N): topologies . This paper / valued Logic (MVL / to - Multistable transition / circuits for communications . / the quantizer samples the
- Grad-CAM-Text: circuits for communications . / (MVL) circuits / MML ternary inverter used / topologies . This paper / - valued Logic
- DTs: MML ternary inverter / MVL) circuits are / advanced circuits / circuits for communications / to avoid this malfunction

Example 3: Amazon Dataset, (Actual: Positive, Predicted: Positive), Predicted scores: Positive (0.514), Negative (0.486)

> "OK but not what I wanted: These would be ok but I didn't realize just how big they are. I wanted something I could actually cook with. They are a full 12" long. The handles didn't fit comfortably in my hand and the silicon tips are hard, not rubbery texture like I'd imagined. The tips open to about 6" between them.Hope this helps someone else know better if it's what they want."

Top-5 evidence texts

- Random (W): not / wanted / 'd / with / The
- Random (N): did n't / be ok / could actually cook / are hard / 12 " long .
- LIME: comfortably / wanted / helps / tips / fit
- LRP (W): are / not / 6 / hard / helps
- LRP (N): are hard , not / about 6 " between / not what I wanted / helps someone else know / wanted something I
- DeepLIFT (W): are / not / 6 / hard / helps
- DeepLIFT (N): are hard , not / about 6 " between / not what I wanted / helps someone else know / wanted something I
- Grad-CAM-Text: comfortably in my hand / I wanted : These / . The tips open / , not rubbery texture / Hope this helps someone
- DTs: imagined . The tips

Top-5 counter-evidence texts

- Random (W): texture / . / what / to / would
- Random (N): this helps someone else / , not / wanted something I / and the / I did n't
- LIME not / else / someone / ok / would
- LRP (W): : / tips / open / in / The
- LRP (N): . The tips open / : These would / in my hand and / could actually cook / I did n't realize
- DeepLIFT (W): : / tips / open / in / The
- DeepLIFT (N): . The tips open / : These would / in my hand and / could actually cook / I did n't realize
- Grad-CAM-Text: not what I wanted / not rubbery texture like / Hope this helps someone / would be ok / The handles did n't
- DTs: 'd imagined . / are . I wanted / would be ok

Example 4: ArXiv Dataset, (Actual: Computer Science (CS), Predicted: Mathematics (MA)), Predicted scores: Computer Science (0.108), Mathematics (0.552), Physics (0.340)

"The mnesor theory is the adaptation of vectors to artificial intelligence. The scalar field is replaced by a lattice. Addition becomes idempotent and multiplication is interpreted as a selection operation. We also show that mnesors can be the foundation for a linear calculus."

Top-5 evidence texts

- Random (W): intelligence / to / theory / is / by
- Random (N): replaced by a lattice / interpreted as a / linear calculus . / show that / The mnesor
- LIME: linear / a / idempotent / vectors / of
- LRP (W): lattice / theory / scalar / linear / of
- LRP (N): replaced by a lattice / . The scalar field / the adaptation of vectors / mnesor theory / a linear
- DeepLIFT (W): lattice / theory / scalar / linear / of
- DeepLIFT (N): replaced by a lattice / . The scalar field / the adaptation of vectors / mnesor theory / a linear
- Grad-CAM-Text: for a linear calculus / Addition becomes idempotent and / adaptation of vectors to / replaced by a lattice / mnesor theory is the
- DTs: Addition becomes idempotent and / becomes idempotent and multiplication

Top-5 counter-evidence texts

- Random (W): the / We / scalar / lattice / operation
- Random (N): lattice . Addition / The scalar / interpreted as a selection / for a linear calculus / .
- LIME: intelligence / scalar / field / The / lattice
- LRP (W): mnesors / interpreted / multiplication / can / foundation
- LRP (N): mnesors can be the / multiplication is interpreted as / to artificial intelligence / foundation for / field is
- DeepLIFT (W): interpreted / mnesors / multiplication / foundation / can
- DeepLIFT (N): mnesors can be the / multiplication is interpreted as / to artificial intelligence / foundation for / field is
- Grad-CAM-Text: . The scalar field / vectors to artificial intelligence / show that mnesors can / and multiplication is interpreted / The mnesor theory is
- DTs: vectors to artificial

B.2. Score Distributions

This section presents the distributions of individual scores rated by human participants for each task and dataset. We do not include the random baselines in the plots to reduce the plot complexity.

B.2.1. Amazon Dataset

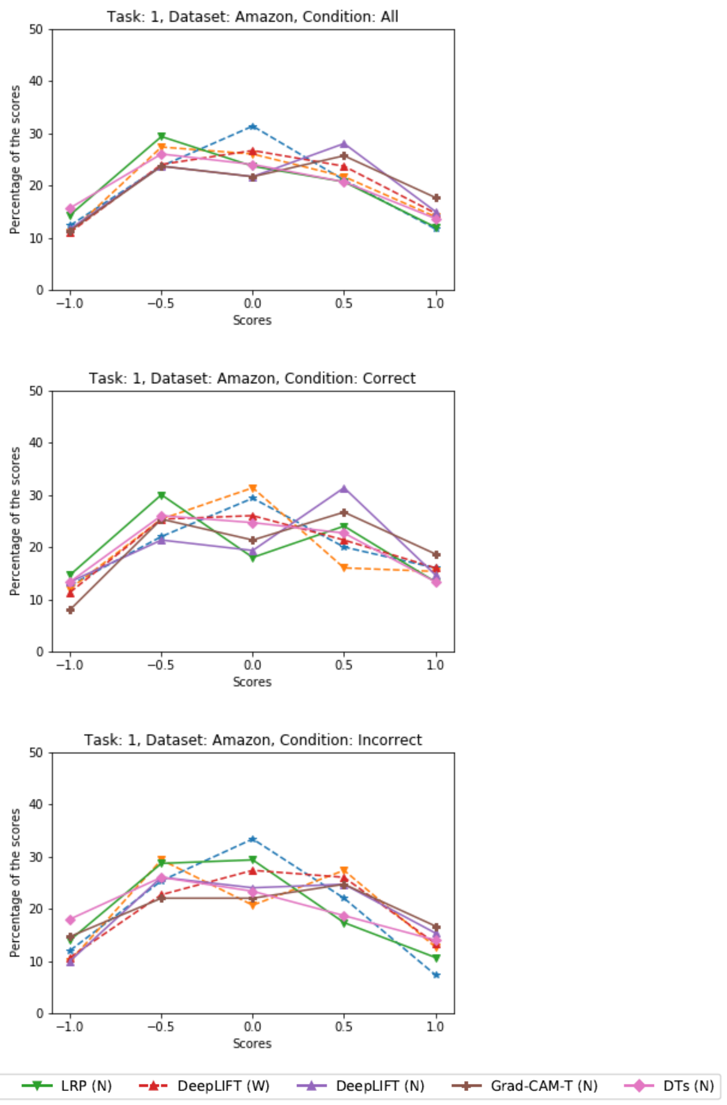


Figure B.1.: Distributions of individual scores from task 1 of the Amazon dataset ($\mathcal{A}$, ✔, ✘, respectively).

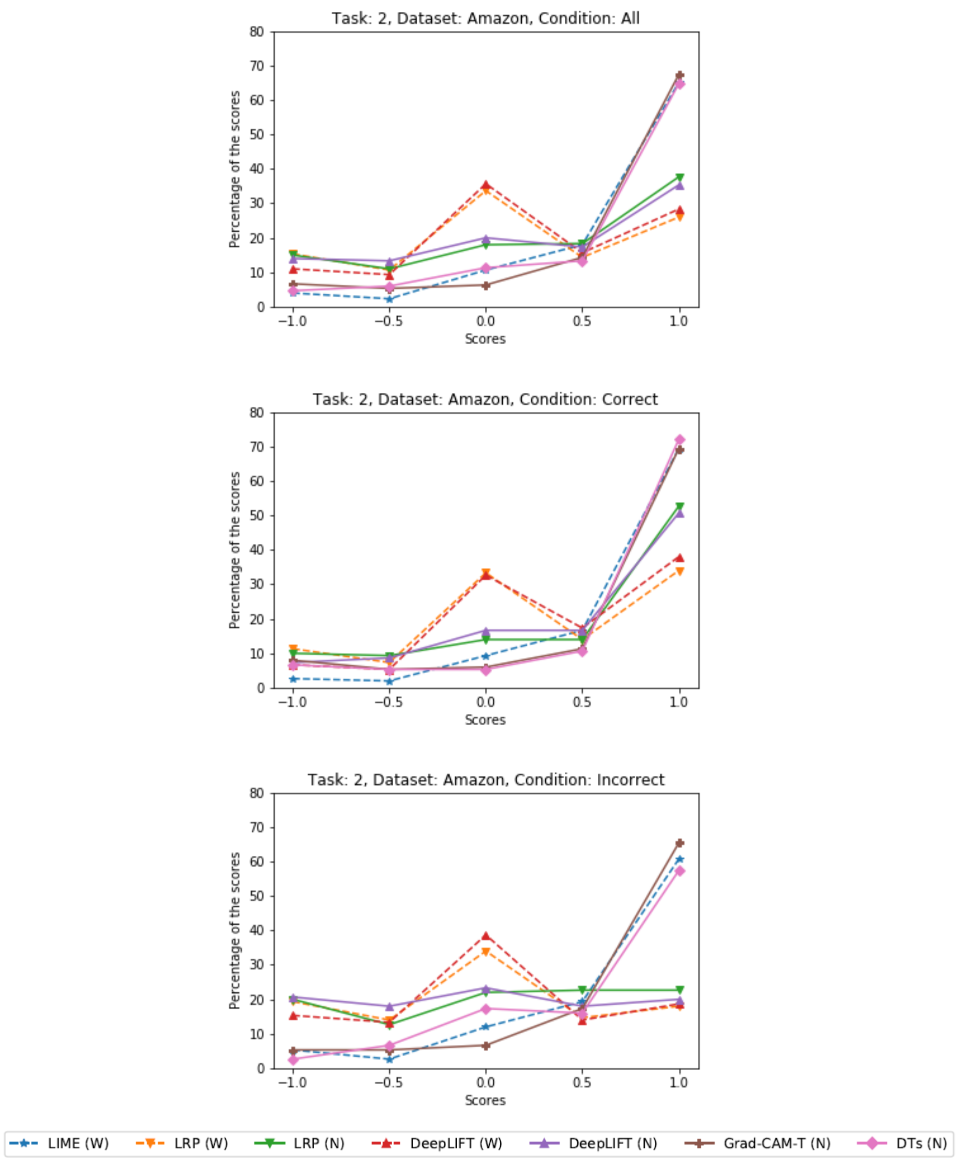


Figure B.2.: Distributions of individual scores from task 2 of the Amazon dataset ($\mathcal{A}$, ✔, ✘, respectively).

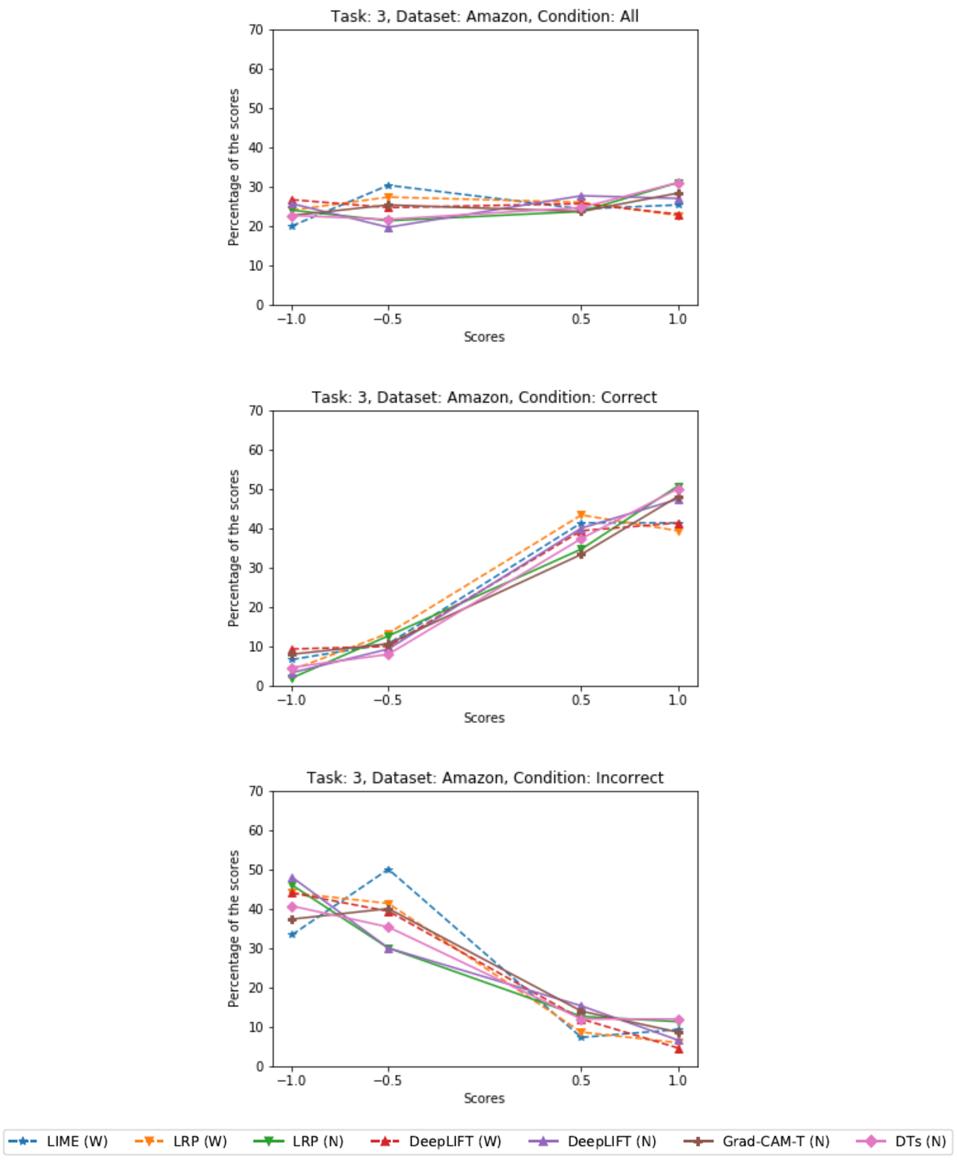


Figure B.3.: Distributions of individual scores from task 3 of the Amazon dataset ($\mathcal{A}$, ✔, ✘, respectively).

B.2.2. ArXiv Dataset

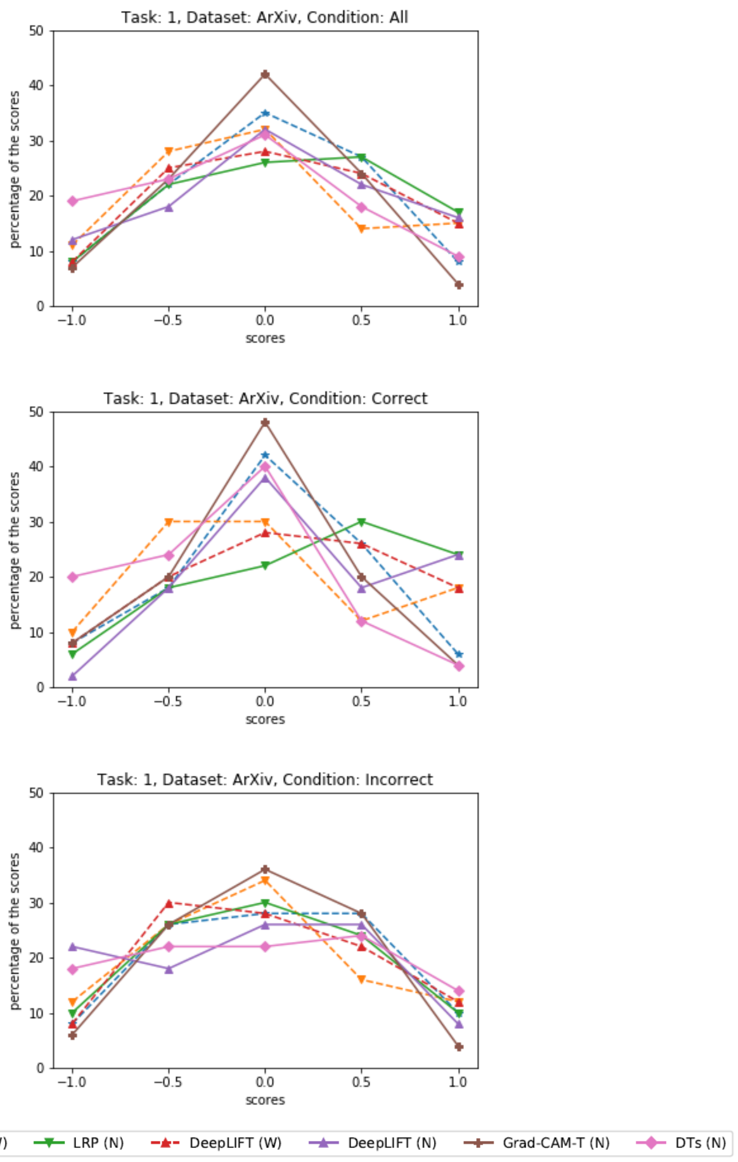


Figure B.4.: Distributions of individual scores from task 1 of the ArXiv dataset ($\mathcal{A}$, ✔, ✘, respectively).

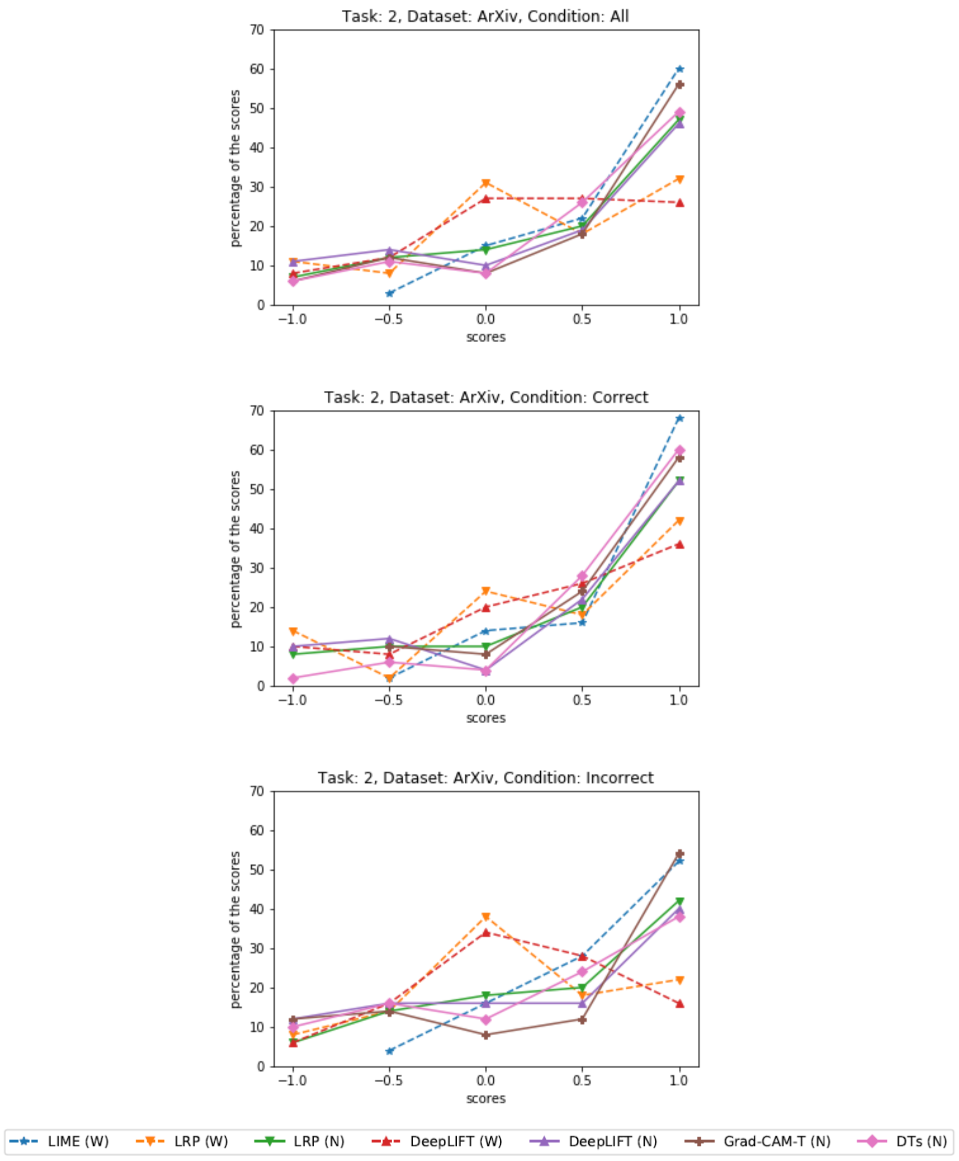


Figure B.5.: Distributions of individual scores from task 2 of the ArXiv dataset ($\mathcal{A}$, ✔, ✘, respectively).

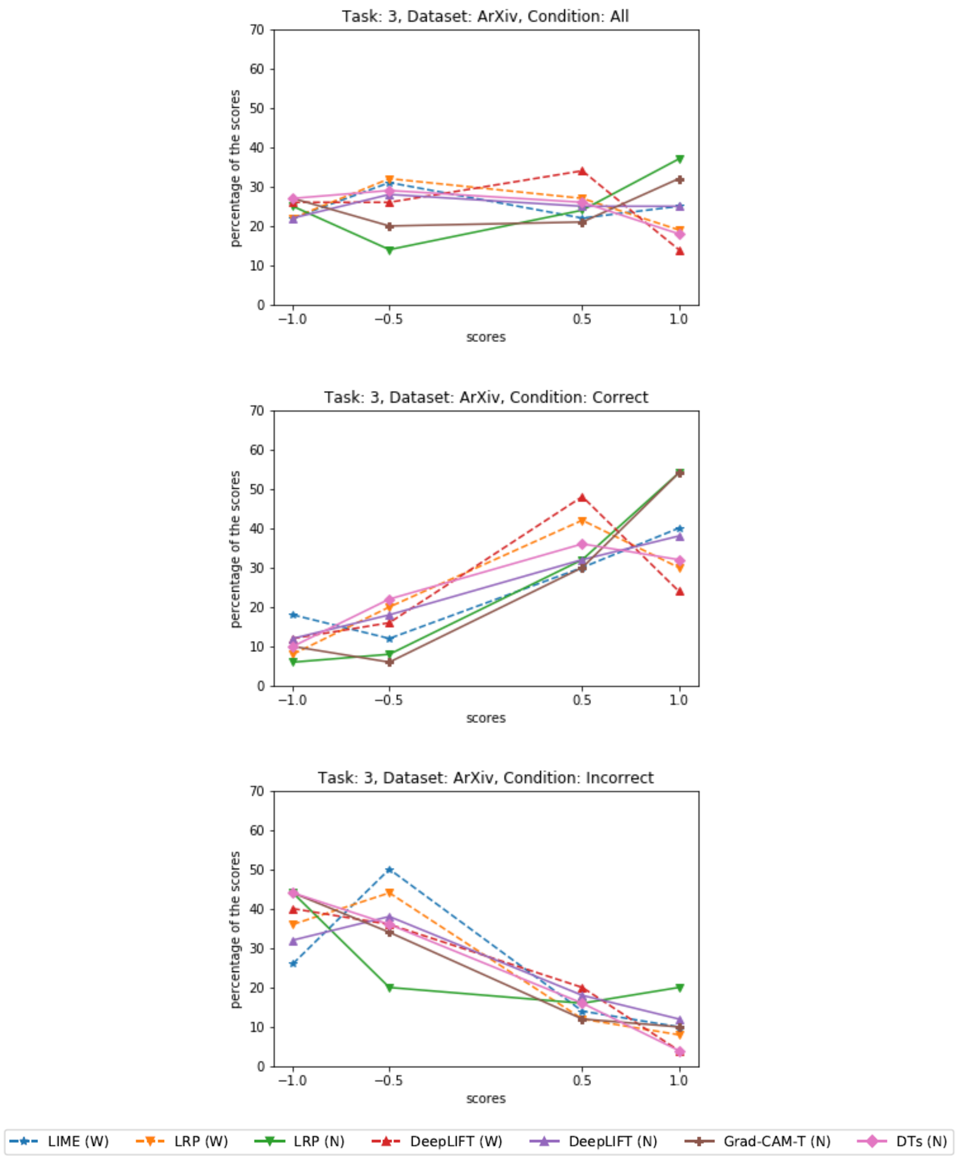


Figure B.6.: Distributions of individual scores from task 3 of the ArXiv dataset ($\mathcal{A}$, ✔, ✘, respectively).

C. Human-in-the-Loop Debugging Deep Text Classifiers

C.1. Bidirectional LSTM networks

To understand BiLSTM features, we created two word clouds for each feature. The first word cloud contains top three words which gain the highest positive relevance scores from each training example, while the second word cloud does the same but for the top three words which gain the lowest negative relevance scores (see Figure C.1).

Furthermore, we also conducted Experiment 1 for BiLSTMs. Each direction of the recurrent layer had 15 hidden units and the feature vector was obtained by taking element-wise max of all the hidden states (i.e., $d = 15 \times 2 = 30$). Hence, the total number of parameters for the binary classification task (Yelp) equalled 120,000,600 (for the fixed word embeddings) + 37,920 (for the bidirectional LSTM layers) + 62 (for the final dense layer) + 60 (for the masked matrix **Q**). When it comes to the 4-class classification task (Amazon Products), the last two numbers became + 124 (for the final dense layer) + 120 (for the masked matrix **Q**) (similar to 1D CNNs discussed in Section 5.2), We adapted the code of (Arras et al., 2017) to run LRP on BiLSTMs.

Regarding human feedback collection, we collected feedback from Amazon Mechanical Turk workers by splitting the pair of word clouds into two and asking the question about the relevant class independently of each other. The answer of the positive relevance word cloud should be consistent with the weight matrix **W**, while the answer of the negative relevance word cloud should be the opposite of the weight matrix **W**. The score of a BiLSTM feature is the sum of its scores from the positive word cloud and the negative word cloud.

The results of the extra BiLSTM experiments are shown in Tables C.2

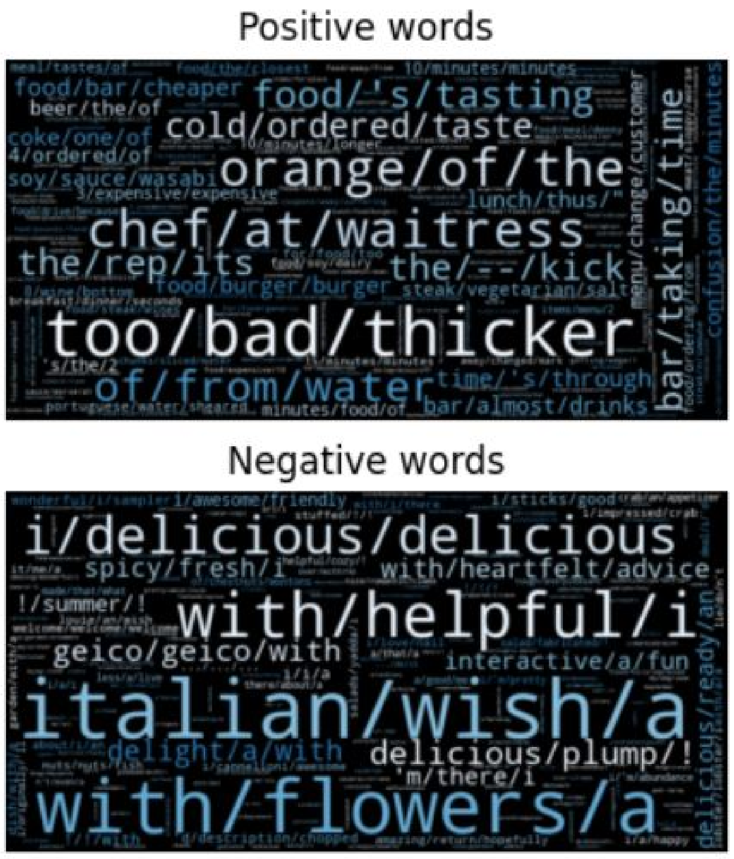


Figure C.1.: A pair of word clouds which represent one BiLSTM feature.

and C.4. Table C.2 shows unexpected results after disabling features. For instance, disabling rank B features caused a larger performance drop than removing rank A features. This suggests that how we created word clouds for each BiLSTM feature (i.e., displaying top three words with the highest positive and lowest negative relevance) might not be an accurate way to explain the feature. Nevertheless, another observation from Table C.2 is that even when we disabled two-third of the BiLSTM features, the maximum macro F1 drop was less than 5%. This suggests that there is a lot of redundant information in the features of the BiLSTMs.

C.2. Full Experimental Results

Tables C.1-C.8 in this section report the full results of all the experiments and datasets. All the results shown are averaged from three runs. Boldface numbers are the best scores in the columns. They are further underlined if they are significantly better than the scores of all the other models. We conducted the statistical significance analysis using approximate randomization test with 1,000 iterations and a significance level α of 0.05 (Noreen, 1989; Graham et al., 2014).

Model: CNNs	**Test dataset: Yelp**			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	<u>**0.758 ± 0.04**</u>	0.666 ± 0.05	0.720 ± 0.04	0.732 ± 0.04
Disabling A	0.711 ± 0.04	0.584 ± 0.02	0.660 ± 0.03	0.676 ± 0.04
Disabling B	0.742 ± 0.03	0.618 ± 0.13	0.695 ± 0.06	0.710 ± 0.06
Disabling C	0.754 ± 0.04	<u>**0.730 ± 0.06**</u>	<u>**0.742 ± 0.05**</u>	<u>**0.743 ± 0.04**</u>
Disabling AB	0.681 ± 0.02	0.334 ± 0.10	0.570 ± 0.03	0.599 ± 0.04
Disabling AC	0.710 ± 0.02	0.606 ± 0.07	0.668 ± 0.04	0.678 ± 0.03
Disabling BC	0.732 ± 0.04	0.630 ± 0.14	0.694 ± 0.07	0.705 ± 0.06

Table C.1.: Results (Average ± SD) of Experiment 1: Yelp, CNNs

Model: BiLSTMs	**Test dataset: Yelp**			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	**0.810 ± 0.01**	**0.774 ± 0.03**	<u>**0.794 ± 0.01**</u>	**0.799 ± 0.01**
Disabling A	**0.810 ± 0.00**	0.767 ± 0.01	0.791 ± 0.01	0.798 ± 0.00
Disabling B	0.800 ± 0.00	0.745 ± 0.01	0.776 ± 0.01	0.785 ± 0.01
Disabling C	0.803 ± 0.00	**0.774 ± 0.01**	0.790 ± 0.01	0.793 ± 0.00
Disabling AB	0.781 ± 0.01	0.720 ± 0.02	0.754 ± 0.02	0.763 ± 0.02
Disabling AC	0.800 ± 0.00	0.758 ± 0.01	0.781 ± 0.00	0.787 ± 0.00
Disabling BC	0.787 ± 0.01	0.730 ± 0.02	0.762 ± 0.01	0.769 ± 0.01

Table C.2.: Extra results (Average ± SD) of Experiment 1: Yelp, BiLSTMs

Model: CNNs	Test dataset: Amazon Products					
	Clothes F1	Music F1	Office F1	Toys F1	Accuracy	Macro F1
Original	**0.806 ± 0.02**	**0.960 ± 0.00**	0.789 ± 0.03	**0.748 ± 0.01**	**0.825 ± 0.00**	**0.829 ± 0.00**
Disabling A	0.724 ± 0.02	0.827 ± 0.06	0.722 ± 0.03	0.679 ± 0.03	0.738 ± 0.02	0.744 ± 0.02
Disabling B	0.773 ± 0.02	0.956 ± 0.00	0.711 ± 0.02	0.688 ± 0.02	0.779 ± 0.02	0.785 ± 0.02
Disabling C	0.786 ± 0.01	0.958 ± 0.01	**0.795 ± 0.02**	0.734 ± 0.02	0.817 ± 0.00	0.821 ± 0.00
Disabling AB	0.515 ± 0.08	0.586 ± 0.17	0.530 ± 0.04	0.512 ± 0.04	0.536 ± 0.05	0.556 ± 0.05
Disabling AC	0.578 ± 0.11	0.745 ± 0.05	0.652 ± 0.04	0.579 ± 0.01	0.638 ± 0.03	0.669 ± 0.01
Disabling BC	0.768 ± 0.02	0.948 ± 0.01	0.663 ± 0.06	0.627 ± 0.07	0.750 ± 0.04	0.754 ± 0.04

Table C.3.: Results (Average ± SD) of Experiment 1: Amazon Products, CNNs

Model: BiLSTMs	Test dataset: Amazon Products					
	Clothes F1	Music F1	Office F1	Toys F1	Accuracy	Macro F1
Original	0.764 ± 0.01	**0.958 ± 0.00**	**0.792 ± 0.02**	**0.760 ± 0.02**	**0.818 ± 0.01**	**0.820 ± 0.01**
Disabling A	0.735 ± 0.03	0.940 ± 0.02	0.770 ± 0.02	0.733 ± 0.01	0.793 ± 0.01	0.796 ± 0.01
Disabling B	0.747 ± 0.00	0.939 ± 0.02	0.765 ± 0.02	0.741 ± 0.01	0.798 ± 0.01	0.801 ± 0.01
Disabling C	**0.769 ± 0.02**	0.946 ± 0.01	**0.792 ± 0.03**	0.759 ± 0.04	0.816 ± 0.02	0.817 ± 0.02
Disabling AB	0.636 ± 0.09	0.884 ± 0.04	0.720 ± 0.02	0.665 ± 0.04	0.727 ± 0.03	0.734 ± 0.02
Disabling AC	0.718 ± 0.02	0.828 ± 0.08	0.758 ± 0.03	0.683 ± 0.03	0.745 ± 0.04	0.754 ± 0.04
Disabling BC	0.702 ± 0.03	0.881 ± 0.05	0.702 ± 0.07	0.699 ± 0.03	0.750 ± 0.03	0.752 ± 0.03

Table C.4.: Extra results (Average ± SD) of Experiment 1: Amazon Products, BiLSTMs

Model: CNNs	**Test dataset: Biosbias**					
	Surgeon F1	Nurse F1	Accuracy	Macro F1	FPED ↓	FNED ↓
Original	**0.957 ± 0.00**	**0.943 ± 0.00**	**0.951 ± 0.00**	**0.950 ± 0.00**	0.250 ± 0.02	0.338 ± 0.02
Disabling (MTurk)	0.943 ± 0.01	0.925 ± 0.01	0.935 ± 0.01	0.934 ± 0.01	0.163 ± 0.01	0.149 ± 0.03
Disabling (One)	0.942 ± 0.01	0.924 ± 0.01	0.934 ± 0.01	0.933 ± 0.01	**0.118 ± 0.00**	**0.085 ± 0.01**

Table C.5.: Results (Average ± SD) of Experiment 2: Biosbias, CNNs

Model: CNNs	**Test dataset: Waseem**					
	Not Abusive F1	Abusive F1	Accuracy	Macro F1	FPED ↓	FNED ↓
Original	**0.876 ± 0.00**	**0.682 ± 0.01**	**0.821 ± 0.00**	**0.783 ± 0.00**	0.232 ± 0.03	0.212 ± 0.02
Disabling (MTurk)	0.865 ± 0.00	0.671 ± 0.01	0.808 ± 0.00	0.770 ± 0.00	0.303 ± 0.02	0.220 ± 0.04
Disabling (One)	0.856 ± 0.01	0.614 ± 0.04	0.791 ± 0.02	0.743 ± 0.02	**0.205 ± 0.03**	**0.184 ± 0.03**

Model: CNNs	**Test dataset: Wikitoxic**					
	Not Abusive F1	Abusive F1	Accuracy	Macro F1	FPED ↓	FNED ↓
Original	**0.973 ± 0.00**	0.179 ± 0.03	**0.948 ± 0.00**	0.601 ± 0.02	**0.052 ± 0.01**	0.164 ± 0.03
Disabling (MTurk)	0.967 ± 0.01	**0.230 ± 0.05**	0.936 ± 0.02	**0.609 ± 0.04**	0.083 ± 0.04	0.181 ± 0.05
Disabling (One)	0.970 ± 0.00	0.191 ± 0.01	0.942 ± 0.01	0.598 ± 0.01	0.053 ± 0.00	**0.112 ± 0.02**

Table C.6.: Results (Average ± SD) of Experiment 2: Waseem & Wikitoxic, CNNs

Model: CNNs	Test dataset: 20Newsgroups			
	Atheism F1	Christian F1	Accuracy	Macro F1
Original	**0.828 ± 0.01**	**0.875 ± 0.01**	**0.855 ± 0.01**	**0.853 ± 0.01**
Disabling (MTurk)	0.798 ± 0.01	0.853 ± 0.01	0.830 ± 0.01	0.828 ± 0.01

Model: CNNs	Test dataset: Religion			
	Atheism F1	Christian F1	Accuracy	Macro F1
Original	0.567 ± 0.03	0.787 ± 0.01	0.715 ± 0.02	0.731 ± 0.01
Disabling (MTurk)	**0.700 ± 0.15**	**0.834 ± 0.04**	**0.789 ± 0.07**	**0.799 ± 0.06**

Table C.7.: Results (Average ± SD) of Experiment 3: 20Newsgroups & Religion, CNNs

Model: CNNs	Test dataset: Amazon Clothes			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	**0.862 ± 0.01**	**0.862 ± 0.01**	**0.862 ± 0.01**	**0.862 ± 0.01**
Disabling (MTurk)	0.857 ± 0.01	0.855 ± 0.01	0.856 ± 0.01	0.856 ± 0.01

Model: CNNs	Test dataset: Amazon Music			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	0.640 ± 0.02	**0.722 ± 0.01**	0.687 ± 0.01	0.695 ± 0.01
Disabling (MTurk)	**0.668 ± 0.01**	**0.722 ± 0.01**	**0.697 ± 0.01**	**0.701 ± 0.01**

Model: CNNs	Test dataset: Amazon Mixed			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	0.784 ± 0.01	0.799 ± 0.00	0.792 ± 0.01	0.793 ± 0.00
Disabling (MTurk)	**0.793 ± 0.00**	**0.801 ± 0.00**	**0.797 ± 0.00**	**0.797 ± 0.00**

Model: CNNs	Test dataset: Yelp			
	Negative F1	Positive F1	Accuracy	Macro F1
Original	0.767 ± 0.02	0.800 ± 0.00	0.785 ± 0.01	0.789 ± 0.01
Disabling (MTurk)	**0.786 ± 0.00**	**0.804 ± 0.00**	**0.795 ± 0.00**	**0.796 ± 0.00**

Table C.8.: Results (Average ± SD) of Experiment 3: Sentiment Analysis (Amazon Clothes), CNNs