Explaining Deep Neural Networks

Oana-Maria Camburu
Linacre College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Hilary 2020

To my parents, Geta and Marin, for their enormous love and support.

Parintţlor mei, Geta si Marin, pentru dragostea si sprijinul lor nemărginit.

Acknowledgements

I am most grateful to my advisors, Phil Blunsom and Thomas Lukasiewicz, for their ongoing guidance and support. They always provided useful advice and perspective, forming me as a researcher. I am also very grateful to Nando de Freitas for his guidance at the beginning of my DPhil.

I am also very thankful to all my collaborators, and in particular to Ana-Maria Creţu, Jakob Foerster, Eleonora Giunchiglia, Vid Kocijan, Pasquale Minervini, Tim Rocktäschel, and Brendan Shillingford (in alphabetical order) for the many inspiring research conversations.

I am grateful to my DPhil examiners, Shimon Whiteson and Kyunghyun Cho, for their careful examination and helpful comments on this thesis.

I also want to thank the Graduate Studies administrators, Julie Sheppard and Sarah Retz-Jones, for their ongoing helpful advice.

I am also very grateful to Linacre College for becoming my home away from home, for the wonderful friends I made in college, and for the support and care I received from all the college staff, in particular, from Jane Hoverd and Karren Morris.

This work has been supported by an Engineering and Physical Sciences Research Council Scholarship and a J.P. Morgan PhD Fellowship.

Lastly, my gratitude goes to my parents, Geta and Marin, who always believed in me, encouraged me to persevere, and supported me unconditionally, to my mentoring family from the École Polytechnique, Edouard and Ioana, who have been like a second family for me, and to all my friends who have been there for me throughout this journey.

Abstract

Deep neural networks are becoming more and more popular due to their revolutionary success in diverse areas, such as computer vision, natural language processing, and speech recognition. However, the decision-making processes of these models are generally not interpretable to users. In various domains, such as healthcare, finance, or law, it is critical to know the reasons behind a decision made by an artificial intelligence system. Therefore, several directions for explaining neural models have recently been explored.

In this thesis, I investigate two major directions for explaining deep neural networks. The first direction consists of feature-based post-hoc explanatory methods, that is, methods that aim to explain an already trained and fixed model (post-hoc), and that provide explanations in terms of input features, such as tokens for text and superpixels for images (feature-based). The second direction consists of self-explanatory neural models that generate natural language explanations, that is, models that have a built-in module that generates explanations for the predictions of the model. The contributions in these directions are as follows.

First, I reveal certain difficulties of explaining even trivial models using only input features. I show that, despite the apparent implicit assumption that explanatory methods should look for one specific ground-truth feature-based explanation, there is often more than one such explanation for a prediction. I also show that two prevalent classes of explanatory methods target different types of ground-truth explanations without explicitly mentioning it. Moreover, I show that, sometimes, neither of these

explanations is enough to provide a complete view of a decision-making process on an instance.

Second, I introduce a framework for automatically verifying the faithfulness with which feature-based post-hoc explanatory methods describe the decision-making processes of the models that they aim to explain. This framework relies on the use of a particular type of model that is expected to provide insight into its decision-making process. I analyse potential limitations of this approach and introduce ways to alleviate them. The introduced verification framework is generic and can be instantiated on different tasks and domains to provide off-the-shelf sanity tests that can be used to test feature-based post-hoc explanatory methods. I instantiate this framework on a task of sentiment analysis and provide sanity tests[1] on which I present the performances of three popular explanatory methods.

Third, to explore the direction of self-explanatory neural models that generate natural language explanations for their predictions, I collected a large dataset of ∼570K human-written natural language explanations on top of the influential Stanford Natural Language Inference (SNLI) dataset. I call this explanation-augmented dataset e-SNLI.[2] I do a series of experiments that investigate both the capabilities of neural models to generate correct natural language explanations at test time, and the benefits of providing natural language explanations at training time.

Fourth, I show that current self-explanatory models that generate natural language explanations for their own predictions may generate inconsistent explanations, such as "There is a dog in the image." and "There is no dog in the [same] image.". Inconsistent explanations reveal either that the explanations are not faithfully describing the decision-making process of the model or that the model learned a flawed decision-making process. I

[1]The sanity tests are available at `https://github.com/OanaMariaCamburu/CanITrustTheExplainer`.

[2]The dataset is publicly available at `https://github.com/OanaMariaCamburu/e-SNLI`.

introduce a simple yet effective adversarial framework for sanity checking models against the generation of inconsistent natural language explanations. Moreover, as part of the framework, I address the problem of adversarial attacks with exact target sequences, a scenario that was not previously addressed in sequence-to-sequence attacks, and which can be useful for other tasks in natural language processing. I apply the framework on a state of the art neural model on e-SNLI and show that this model can generate a significant number of inconsistencies.

This work paves the way for obtaining more robust neural models accompanied by faithful explanations for their predictions.

Publications

This thesis is based on the following publications:

(Camburu et al., 2018): Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, Phil Blunsom. *e-SNLI: Natural Language Inference with Natural Language Explanations.* In Advances in Neural Information Processing Systems 31, 2018.

(Camburu et al., 2019): Oana-Maria Camburu[3] Eleonora Giunchiglia[3], Jakob Foerster, Thomas Lukasiewicz, Phil Blunsom. *Can I Trust the Explainer? Verifying Post-hoc Explanatory Methods.* In the Neural Information Processing Systems (NeruIPS) Workshop Safety and Robustness in Decision Making, 2019.

(Camburu et al., 2020): Oana-Maria Camburu, Brendan Shillingford, Pasquale Minervini, Thomas Lukasiewicz, Phil Blunsom. *Make Up Your Mind! Adversarial Generation of Inconsistent Natural Language Explanations* In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), 2020.

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

(Mao et al., 2016): Junhua Mao, Jonathan Huang, Alexander Toshev, Oana-Maria Camburu, Alan L. Yuille, Kevin Murphy. *Generation and Comprehension of Unambiguous Object Descriptions.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[3]Equal first author.

(Kocijan et al., 2019a): Vid Kocijan, Oana-Maria Camburu, Ana-Maria Creţu, Yordan Yordanov, Phil Blunsom, Thomas Lukasiewicz. *WikiCREM: A Large Unsupervised Corpus for Coreference Resolution.* In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019.

(Kocijan et al., 2019b): Vid Kocijan, Ana-Maria Creţu, Oana-Maria Camburu, Yordan Yordanov, Thomas Lukasiewicz. *A Surprisingly Robust Trick for Winograd Schema Challenge.* In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), 2019.

(Yordanov et al., 2020): Yordan Yordanov, Oana-Maria Camburu, Vid Kocijan, Thomas Lukasiewicz. *Does the Objective Matter? Comparing Training Objectives for Pronoun Resolution.* In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.

(Do et al., 2020): Virginie Do, Oana-Maria Camburu, Zeynep Akata, Thomas Lukasiewicz. *e-SNLI-VE-2.0: Corrected Visual-Textual Entailment with Natural Language Explanations.* In the IEEE Computer Vision and Pattern Recognition (CVPR) Workshop on Fair, Data Efficient and Trusted Computer Vision, 2020.

Finally, during my DPhil, I also had the pleasure to co-supervise three MSc theses:

a. *Group-Sparse Sentence Representations.* Thesis written by Ana-Maria Creţu for the degree of Master of Science, Department of Computer Science, École Polytechnique Fédérale de Lausanne, 2018.

b. *Natural Language Inference Dataset Generation using Natural Language Explanations.* Thesis written by Vlad Barvinko for the degree of Master

of Science, Department of Computer Science, University of Oxford, 2019.

c. *Human-Readable Explanations of Neural Network Predictions.* Thesis written by Virginie Do for the degree of Master of Science, Oxford Internet Institute, University of Oxford, 2019.

Contents

List of Figures

Chapter 1

Introduction

Neural networks[1] are a class of computing systems designed to learn to perform tasks directly from raw data, such as images or text, without hard-coding task-specific knowledge. These systems have been gaining in popularity over recent years due to their versatility and revolutionary success in various domains. For example, in computer vision, deep neural networks improved the state of the art in object classification by a large margin (Krizhevsky et al., 2012). Similarly, neural networks have revolutionised acoustic modelling (Hinton et al., 2012) and machine translation (Sutskever et al., 2014), with significant improvements in performance over traditional machine learning methods. Learning without hard-coding task-specific knowledge is a breakthrough in artificial intelligence. For example, AlphaGo (Silver et al., 2016) learned to play the complex game of Go solely by watching a large number of games played by humans and then repeatedly playing against itself. In 2016, AlphaGo beat the 18-time world champion Lee Sedol 4-1. Furthermore, it has also been shown that neural networks are able to acquire artistic capabilities and to create art. For example, neural networks can recognise artistic styles and transfer them to other images (Gatys et al., 2015), and they can also generate music (Huang et al., 2018; Engel et al., 2019; Hawthorne et al., 2019).

[1]This thesis only addresses artificial neural networks, hence there is no risk of confusion with biological neural networks.

1.1 The Importance of Explanations for Deep Neural Networks

It has been shown that a key factor in the success of neural networks is their capability to be *deep*, i.e., the fact that successful neural networks can be composed of a very large number of non-linear functions (Telgarsky, 2016; Eldan and Shamir, 2015; He et al., 2016). Intuitively, multiple layers of non-linear functions allow the network to learn features at various levels of abstraction between the raw data and the prediction. However, this comes at the cost of explainability, since providing a human intelligible interpretation to an intricate composition of a large number of non-linear functions is a difficult open question. Thus, in safety-critical applications, such as health diagnosis, credit allowance, or criminal justice, one may still prefer to employ less accurate but human-interpretable models, such as linear regression and decision trees.

The doubts around the decision-making processes of neural networks are justified, as it has been shown that seemingly very accurate such systems can easily rely on spurious correlations in datasets (also called statistical bias, or artefacts) to provide correct answers (Cooper et al., 2005; Glockner et al., 2018a; Gururangan et al., 2018; Chen et al., 2016a; McCoy et al., 2019). A notorious example is that of predicting dire outcomes for patients with pneumonia, on which neural networks outperformed traditional methods by a wide margin (Cooper et al., 2005). However, it turned out that the training set contained the pattern that patients with a history of asthma were at lower risk of dying from pneumonia (Caruana et al., 2015). This pattern appeared because asthmatic patients were given quicker and higher attention (hence, they had lower mortality rate), because they were, in fact, at higher risk. Obviously, it would be very dangerous, in practice, to use a model that relies on such correlations.

Another source of mistrust in black-box systems comes from the potential subjective bias that these systems might develop, such as racism, sexism, or other kinds of discrimination and subjectivity (Buolamwini and Gebru, 2018). For example, Dressel and Farid (2018) cast doubt on the fairness of the widely used commercial risk assessment software COMPAS for recidivism prediction. Such biases may be learned

either from under-representation or irrelevant statistical correlations in the datasets that we train and test our models on. For example, Bolukbasi et al. (2016) showed that word embeddings exhibit female/male gender stereotypes, while Webster et al. (2018) showed that existing corpora and systems favour masculine entities.

Moreover, a large number of adversarial attacks have shown the fragility of the apparently highly accurate neural networks. For example, image processing neural networks can be fooled into changing *any* of their predictions into *any* other possible prediction only by making imperceptible alterations to the pixels of an image (Szegedy et al., 2014; Moosavi-Dezfooli et al., 2017). Adversarial attacks in neural networks have also had significant success rates in other domains, such as natural language processing (Jia and Liang, 2017a; Alzantot et al., 2018) and speech recognition (Carlini and Wagner, 2018). The fragility of deep neural networks revealed by adversarial attacks casts doubt on the underlying learned decision-making processes of these methods.

Therefore, for neural network systems to garner widespread public trust, as well as to ensure that these systems are indeed fair, we must have human-intelligible explanations for the decisions of these models (Dzindolet et al., 2003).

Firstly, explanations are necessary for providing justifications to end-users for whom the decisions are being taken, such as patients and clients. Not only is it ethical to provide such justifications, but, in some countries, it is even required by law, for example with the introduction of the "Right to explanation" in GDPR 2016.[2]

Secondly, explanations are useful for the employers of these systems such as doctors and judges, to better understand a system's strengths and limitations and to appropriately trust and employ the system's predictions.

Thirdly, explanations provide an excellent source of knowledge discovery. It is well known that neural networks are particularly good in finding patterns in data. Therefore, being able to explain the algorithms learned by neural networks may result in revealing valuable knowledge that would otherwise be difficult for humans to mine

[2] `https://www.privacy-regulation.eu/en/r71.htm`

from extremely large datasets. For example, Chandra and Omlin (2007) use decision trees to extract knowledge about conservation biology from trained neural networks.

Finally, being able to explain neural networks can help the developers/researchers of these methods to diagnose and further improve the systems. For example, Ribeiro et al. (2016) showed that, after seeing a number of explanations, even non-experts in machine learning were able to detect which words should be eliminated from a dataset in order to improve an untrustworthy classifier.

Given the above-mentioned usages of explanations, there is currently an increasing demand for human-intelligible explanations for the predictions of artificial intelligent systems, which has been the motivation to dedicate my thesis to this topic.

1.2 Research Questions and Outline

To advance the field of explaining deep neural networks, an appealing research direction is that of developing deep neural networks that provide their own natural language explanations for their decisions (Park et al., 2018; Ling et al., 2017). This type of model is similar to how humans both learn from explanations and explain their decisions. Indeed, humans do not learn solely from labelled examples supplied by a teacher, but seek a conceptual understanding of a task through both demonstrations and explanations (Lombrozo, 2012, 2006). Moreover, natural language is readily comprehensible to an end-user who needs to assert the reliability of a model. It is also easiest for humans to provide free-form language, eliminating the additional effort of learning to produce formal language, thus making it simpler to collect such datasets. Lastly, natural language justifications might be mined from the existing large-scale free-form text, thus putting to advantage models that are capable of learning from natural language explanations. Therefore, two of the three main research questions I investigate in this thesis are: *Do neural networks improve their behaviour and performance if they are additionally given natural language explanations for the ground-truth label at training time?* and *Can neural networks generate natural language explanations for their predictions at test time?*

Given the scarcity of datasets of natural language explanations, I first collected a large dataset of ∼570K instances of human-written free-form natural language explanations on top of SNLI, an existing influential dataset of natural language inference (Bowman et al., 2015). I call this explanations-augmented dataset e-SNLI. For example, a natural language explanation for the fact that the sentence "A woman is walking her dog in the park." entails the sentence "A person is in the park." is that "A woman is a person, and walking in the park implies being in the park.". I introduce neural models that learn from these explanations at training time and output such explanations at test time. I also investigate whether learning with the additional signal from natural language explanations can provide benefits in solving other downstream tasks. These contributions will be presented in Chapter 5.

While natural language explanations generated by a neural model can provide reassurance to users when they display correct argumentation for solving the task, they might not faithfully reflect the decision-making processes of the model. Therefore, in this thesis, I also investigate the following research question: *Can we verify if explanations are faithfully describing the decision-making processes of the deep neural networks that they aim to explain?* To address this question for the class of neural models that generate natural language explanations, two different approaches can be considered. The first approach is to investigate whether the generated natural language explanations are consistent per model. For example, if a model generates explanations that are mutually contradictory, such as "There is a dog in the image." and "There is no dog in the [same] image.", then there is a flaw in the model. The flaw can be either in the decision-making process or in the faithfulness with which the generated explanations reflect the decision-making process of the model (or in both). I introduce a framework that checks whether models that generate natural language explanations can generate inconsistent explanations. As part of this framework, I address the problem of adversarial attacks for full target sequences, a scenario that had not been previously addressed in sequence-to-sequence attacks and which can be useful for other kinds of tasks. I instantiate the framework on a state-of-the-art neural model

on e-SNLI, and show that this model is capable of generating a significant number of inconsistencies. These contributions will be presented in Chapter 6.

The second approach for explaining deep neural models consists of developing post-hoc explanatory methods, i.e., external methods that aim to explain already trained neural models. Post-hoc explanatory methods are widespread and currently form the majority of techniques for providing explanations for deep neural networks. While neural networks that generate their own natural language explanations may be corrupted by the same types of biases that cause the need for explaining such models, external explanatory methods may be less prone to these biases. Hence, an interesting approach in verifying the faithfulness of the natural language explanations generated by a model would be to investigate the correlation between the explanations given by an external explanatory method and the natural language explanations generated by the model itself. However, external explanatory methods may also not faithfully explain the decision-making processes of the models that they aim to explain. Therefore, in this thesis, I introduce a verification framework to verify the faithfulness of external explanatory methods. These contributions will be presented in Chapter 4.

In the process of developing the verification framework mentioned above, I identified certain peculiarities of explaining a model using only input features, i.e., units of the input such as tokens for text and superpixels[3] for images. For example, I show that sometimes there exists more than one ground-truth explanation for the same prediction of a model, contrary to what current works in the literature seem to imply. I also show that two prevalent classes of post-hoc explanatory methods target distinct ground-truth explanations and I reveal some of their strengths and limitations. These insights can have an important impact on how users choose explanatory methods to best suit their needs, as well as on verifying the faithfulness explanatory methods. These contributions will be presented in Chapter 3.

To put into perspective the scope of these contributions, before diving into the contributions of this thesis, I provide background knowledge on explainability for deep neural networks in Chapter 2.

[3]A superpixel is a group of pixels with common characteristics, such as pixel intensity or proximity.

At the end of each contribution chapter, I present the conclusions and the open questions that the findings in the chapter lead to. Finally, in Chapter 7, I provide general conclusions and perspectives on explainability for deep neural networks.

Chapter 2

Background on Explanatory Methods for Deep Neural Networks

This thesis assumes and does not describe basic knowledge of machine learning and, in particular, of neural networks. Readers interested in an in-depth introduction into deep neural networks can see, for example, the book "Deep Learning" by Goodfellow et al. (2016). In particular, I recommend Chapter 10 ("Sequence Modeling: Recurrent and Recursive Nets"), since recurrent neural networks are encountered extensively throughout this thesis.

The main goal of this chapter is to provide the necessary background on the current state of explanatory methods, in order to ease the reading of the rest of the thesis as well as to put into perspective the contributions hereby brought to the field.

2.1 Notation

This section describes the notation used throughout this thesis.

Variables. Scalars are usually denoted with non-bold letters, while vectors are in bold. Thus, I use bold lower-case Roman letters, such as $\mathbf{x}$, to denote 1-dimensional vectors, which are assumed to be column vectors. A component of a vector is denoted by subscripting the non-bold version of the variable letter with the index of the component. For example, the i-th component of the vector $\mathbf{x}$ is denoted by x_i.

Hyperparameters are usually denoted by lower-case Greek letters. For instance, μ and α typically refer to the learning rate and the coefficient of a loss term, respectively.

Data. To denote datasets, I use calligraphic, upper-case Roman letters, for example, $\mathcal{D}$. To enumerate instances (also called samples or datapoints) in a dataset, I use parenthesised superscripts. For example, the dataset $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1,N}$ contains N instances $\mathbf{x}^{(n)}$ with their associated scalar or categorical targets $y^{(n)}$. Thus, to refer to the i-th component of the n-th instance in the dataset $\mathcal{D}$, I use $\mathbf{x}_i^{(n)}$. Of particular importance in this thesis are datapoints that consist of variable length sequences, such as natural language text. For an input sequence $\mathbf{x}$, x_t denotes the t-th timestep in the sequence. For example, in a sentence $\mathbf{x}$, x_t can denote the t-th token or character. For discrete inputs, while the actual input into the neural network at each timestep t is an embedding vector (Mikolov et al., 2013) of the token x_t, I use the scalar notation x_t to refer to the raw input at timestep t.

Functions. In this thesis, the name of a function does not contain indications of the dimensionalities of its inputs and outputs. These dimensionalities are either stated or inferred from the context. Similarly to vectors, the components of the outputs of functions are referred to by using subscripts placed *before* the argument (if present). For example, the i-th component of the vector output of function f at point $\mathbf{x}$ is referred to as $f_i(\mathbf{x})$.

A commonly used type of function is the loss function used for training neural networks, which I denote by the uppercase letter L, which is typically a sum of loss terms.

Finally, neural networks themselves are functions. Generic models will be commonly referred as m. For readability, I use abbreviations or acronyms as names for the (parts of the) neural networks. For example, the encoder and decoder of a sequence-to-sequence neural network can be referred as $\text{enc}(\cdot)$ and $\text{dec}(\cdot)$, respectively, while a multilayer perceptron can be referred as $\text{MLP}(\cdot)$.

Big O notation. $O(\cdot)$ denotes the typical big O notation.

Vocabulary. In this thesis, it is important to make the distinction between three types of algorithms. First, a *target model* refers to a model that we want to explain. Second, an *explanatory method* (also called *explainer*) refers to a method that aims to explain a target model. Third, a *verification framework* refers to a framework for verifying if an explanatory method faithfully explain a target model.

2.2 Types of Explanatory Methods

Recently, an increasing number of diverse works are aiming to shed light on deep neural networks (Ribeiro et al., 2016; Lundberg and Lee, 2017; Lapuschkin et al., 2015; Arras et al., 2017; Shrikumar et al., 2017; Ribeiro et al., 2018; Plumb et al., 2018; Chen et al., 2018a; Kim et al., 2018b; Park et al., 2018; Lei et al., 2016; Yoon et al., 2019).[1] These methods differ substantially in various ways. Different groupings of explainers are presented below. However, I will not present all the possible groups of explanatory methods. This is because there is a large number of groups, and, since the aim of this chapter is to provide background for putting into perspective the rest of the thesis, I do not want to overwhelm the reader with information that would not be further necessary for the scope of this thesis. For the readers interested in more detailed reviews on categorisations of explanatory methods, I refer to the works of Gilpin et al. (2018) and Arrieta et al. (2020).

2.2.1 Post-Hoc versus Self-Explanatory

One of the most prominent distinctions among current explanatory methods is to divide them into two types.

(i) ***Post-hoc explanatory methods*** are stand-alone methods that aim to explain already trained and fixed target models (Ribeiro et al., 2016; Lundberg and Lee, 2017; Chen et al., 2018a; Ribeiro et al., 2018; Simonyan et al., 2014; Lapuschkin

[1]Some of these methods are not restricted to explaining only neural networks.

et al., 2015; Arras et al., 2017; Shrikumar et al., 2017; Sundararajan et al., 2017). For example, LIME (Ribeiro et al., 2016) is a post-hoc explanatory method that explains a prediction of a target model by learning an interpretable model, such as a linear regression, on a neighbourhood around the prediction of the model (I will explain the concept of neighbourhood of an instance in Section 2.3).

(ii) ***Self-explanatory models*** are target models which incorporate an explanation generation module into their architecture such that they provide explanations for their own predictions (Lei et al., 2016; Yoon et al., 2019; Chang et al., 2019; Kim et al., 2018b; Park et al., 2018; Hendricks et al., 2016; Ling et al., 2017; Rajani et al., 2019). At a high level, self-explanatory models have two interconnected modules: (i) a *predictor* module, i.e., the part of the model that is dedicated to making a prediction for the task at hand, and (ii) an *explanation generator* module, i.e., the part of the model that is dedicated to providing the explanation for the prediction made by the predictor. For example, Lei et al. (2016) introduced a self-explanatory neural network where the explanation generator selects a subset of the input features, which are then exclusively passed to the predictor that provides the final answer based solely on the selected features. Their model is also regularised such that the selection is short. Thus, the selected features are intended to form the explanation for the prediction.

    Self-explanatory models do not necessarily need to have supervision on the explanations. For example, the models introduced by Lei et al. (2016), Yoon et al. (2019), and Chang et al. (2019) do not have supervision on the explanations but only at the final prediction. On the other hand, the models introduced by Park et al. (2018), Kim et al. (2018b), Hendricks et al. (2016), and Gilpin et al. (2018) require explanation-level supervision.

In general, for self-explanatory models, the predictor and explanation generator are trained jointly, hence the presence of the explanation generator is influencing the training of the predictor. This is not the case for post-hoc explanatory methods, which

do not influence at all the predictions made by the already trained and fixed target models. Hence, for the cases where the augmentation of a neural network with an additional explanation generator results in a significantly lower task performance than that of the neural network trained only to perform the task, one may prefer to use the latter model followed by a post-hoc explanatory method. On the other hand, it can be the case that enhancing a neural network with an explanation generator and jointly training them results in a better performance on the task at hand. This can potentially be due to the additional guidance in the architecture of the model, or to the extra supervision on the explanations if available. For example, on the task of sentiment analysis, Lei et al. (2016) obtained that adding an intermediate explanation generator module without any supervision on the explanations does not hurt performance. On the task of commonsense question answering, Rajani et al. (2019) obtained a better performance by a self-explainable model with supervision on the explanations than by a neural network trained only to perform the task. Thus, the two types of explanatory methods have their advantages and disadvantages.

In Chapter 3, I will present a fundamental difference between two major types of post-hoc explanatory methods. In Chapter 4, I will introduce a verification framework for post-hoc explanatory methods which is based on self-explanatory target models. In Chapters 5 and 6, I will focus solely on self-explanatory models.

2.2.2 Black-Box versus White-Box

Another distinction among explanatory methods can be done in terms of the knowledge about the target model that the explainer requires. We have the following two categories.

(i) ***Black-box/model-agnostic explainers*** are explainers that assume access only to querying the target model on any input. LIME (Ribeiro et al., 2016), Anchors (Ribeiro et al., 2018), KernalSHAP (Lundberg and Lee, 2017), L2X (Chen et al., 2018a), and LS-Tree (Chen and Jordan, 2020) are a few examples of model-agnostic explanatory methods.

(ii) ***White-box/model-dependant explainers*** are explainers that assume access to the architecture of the target model. For example, LRP (Lapuschkin et al., 2015; Arras et al., 2017), DeepLIFT (Shrikumar et al., 2017), saliency maps (Simonyan et al., 2014), integrated gradients (Sundararajan et al., 2017), Grad-CAM (Selvaraju et al., 2019), and DeepSHAP and MaxSHAP (Lundberg and Lee, 2017) are a few examples of model-dependant explanatory methods.

This division mainly applies to post-hoc explanatory methods, since, by default, self-explanatory models exhibit a strong connection between the explanation generator and the predictor.

Black-box explainers have a larger spectrum of applicability than white-box explainers, because the former can be applied in cases where one does not have access to the internal structure of the target model. Black-box explainers are also usually quicker to use in an off-the-shelf manner, since one does not have to adapt a model-dependant technique to a particular architecture of a model or to a new type of layer. However, these advantages come at the expense of potentially less accurate explanations, since black-box explainers can infer correlations between inputs and predictions that do not necessarily reflect the true inner workings of the target model.

The verification framework that I will introduce in Chapter 4 can be applied to both black-box and white-box explainers.

2.2.3 Instance-Wise versus Global

Another way of dividing explanatory methods is according to the scope of the explanation. We have the following two types of explainers.

(i) ***Instance-wise explainers*** provide an explanation for the prediction of the target model on any *individual instance* (Ribeiro et al., 2016; Lundberg and Lee, 2017; Chen et al., 2018a; Plumb et al., 2018; Lapuschkin et al., 2015; Arras et al., 2017; Simonyan et al., 2014; Shrikumar et al., 2017). For example, LIME (Ribeiro et al., 2016) learns an instance-wise explanation for any instance by training a linear regression on a neighbourhood of the instance.

(ii) ***Global explainers*** explain the high-level inner workings of the entire target model (Ibrahim et al., 2019; Frosst and Hinton, 2017; Yang et al., 2018). For example, Frosst and Hinton (2017) provide global explanations by distilling a neural network into a soft decision tree.

Instance-wise explainers are particularly useful, for example, for use cases where end users require an explanation for the decisions taken for their particular case. Global explainers are particularly useful, for example, for quick model diagnostics of possible biases or for knowledge discovery. Since global explainers aim to explain the behaviour of the entire target model, usually via distilling the target model into an interpretable one, they implicitly provide instance-wise explanations as well. However, it is difficult, if not impossible, for an interpretable model to accurately capture all the irregularities learned by a highly non-linear model. Hence, instance-wise explanations derived from global explainers might not always be accurate. The majority of the current works in the literature focus on designing instance-wise explainers.

Conversely, instance-wise explanations can be a starting point for obtaining global explanations. For example, Ribeiro et al. (2016), and Ribeiro et al. (2018) introduced sub-modular pick techniques to derive a global explanation for the inner working of the model from instance-wise explanations, while Ibrahim et al. (2019) used clustering techniques to provide global explanations for sub-populations starting from instance-wise explanations.

This division is mainly relevant for post-hoc explanatory methods. By default, self-explanatory models are instance-wise explainers, since the built-in explanation generation module would be applied to each instance. However, the above mentioned techniques for going from instance-wise to global explanations can equally be applied to self-explanatory models.

This thesis addresses solely instance-wise explainers.

2.2.4 Forms of Explanations

Explanatory methods can also be divided into different groups depending on the form of the explanations that they provide. I will briefly describe below a few commonly used forms of explanations. I will discuss in more detail the feature-based and natural language explanations, since they are the focus of this thesis.

Feature-based explanations. Feature-based explanations are currently the most widespread form of explanations and consist of assessing the importance/contribution that each feature of an instance has in the model prediction on that instance. Common features include tokens for text and super-pixels for images.

There are two major types of feature-based explanations: importance weights and subsets of features. The importance weights explanations provide for each input feature in an instance a real number that represents the contribution of the feature to the prediction of the model on the instance. Subsets explanations provide for each instance the subset of most important features for the prediction of the model on the instance. For example, for a model that predicts that the sentence "The movie was very good." has a positive sentiment of 4 stars (out of 5), a subset explanation could be formed of the features {"very", "good"}, while an importance weights explanation could, for example, attribute to the feature "good" an importance weight of 3 and to the feature "very" an importance weight of 1 (the sum of importance weights matches with the total prediction of 4; this property is called feature-additivity and will be discussed in detail in the next section). I will describe each of these two types of feature-based explanations in Section 2.3.

There is a large number of explanatory methods providing feature-based explanations. For example, LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017), L2X (Chen et al., 2018a), LS-Tree (Chen and Jordan, 2020), Anchors (Ribeiro et al., 2018), LRP (Lapuschkin et al., 2015; Arras et al., 2017), DeepLIFT (Shrikumar et al., 2017), Integrated Gradients (Sundararajan et al., 2017), and Grad-CAM (Selvaraju et al., 2019) are just a few of the many feature-based post-hoc explanatory methods,

while self-explanatory methods with feature-based explanations include RCNN (Lei et al., 2016), INVASE (Yoon et al., 2019), CAR (Chang et al., 2019), as well as the influential class of attention models (Bahdanau et al., 2015), among others.

I investigate feature-based explanations in Chapters 3 and 4.

Natural language explanations. Natural language explanations consist of natural language sentences that provide human-like arguments supporting a prediction. For example, a natural language explanation for the fact that the sentence "A woman is walking her dog in the park." entails the sentence "A person is in the park." is that "A woman is a person, and walking in the park implies being in the park.".

To train explanatory methods to provide natural language explanations, new datasets of human-written natural language explanations have recently been collected to allow supervision on the explanations at training time, as well as to enable evaluation of the correctness of the generated explanations at test time (Park et al., 2018; Hendricks et al., 2016; Kim et al., 2018b; Ling et al., 2017; Jansen et al., 2018; Rajani et al., 2019). For example, Park et al. (2018) introduced ACT-X and VQA-X, two datasets that contain (besides visual explanations) natural language explanations for the tasks of visual activity recognition and visual question-answering, respectively. Further, Park et al. (2018) also introduced the self-explanatory model PJ-X (Pointing and Justification Explanation), that is trained to jointly provide a prediction, a feature-based explanation, and a natural language explanation, with both explanations supporting the prediction. Similarly, Kim et al. (2018b) introduced the BDD-X (Berkeley DeepDrive eXplanation) dataset consisting of natural language explanations supporting the decisions of a self-driving car. They train a self-explanatory model consisting of a vehicle controller (the predictor) and an explanation generator such that the decisions of a self-driving car are justified to the users with explanations such as "The car moves back into the left lane because the school bus in front of it is stopping.". Another relevant work is that of Jansen et al. (2018), who provide a dataset of natural language explanation graphs for elementary science questions. However, their corpus is very small, only 1680 pairs of questions and explanations.

Similarly, Ling et al. (2017) introduced a dataset of textual explanations for solving mathematical problems. Yet, the focus of this dataset is narrow and it is arguably difficult to transfer to more general natural understanding tasks.

While the majority of the natural language explanations methods are self-explanatory models, there are works, such as that of Hendricks et al. (2018), that aim to improve the quality of the natural language explanations in a post-hoc manner.

In Chapter 5, I introduce a new large dataset of ∼570K natural language explanations for the influential task of solving natural language inference (Bowman et al., 2015). In addition, I develop models that incorporate natural language explanations into their training process as well as generate such explanations at test time. Further, in Chapter 6, I draw attention to the fact that such models can generate inconsistent natural language explanations, which exposes flaws in the model. I introduce an adversarial framework for detecting pairs of inconsistent explanations and show that the best model trained in Chapter 5 can generate a significant amount of inconsistent explanations.

Concept-based explanations. Concept-based explainers aim to quantify the importance of a user-defined high-level concept, such as curls or strips, to the prediction of the model (Kim et al., 2018a; Akula et al., 2020).

Example-based explanations. Example-based explainers provide instances from which one can derive insights into the model predictions. For example, for a given instance, Koh and Liang (2017) aim to trace which instances from the training set influenced the most the model prediction on the current instance. Kanehira and Harada (2019) provide examples that show the model capabilities of distinguishing between the current instance and other instances which only have little yet essential differences with respect to the current instance.

Surrogate explanations. Surrogate explainers aim to provide an interpretable surrogate model of the target model. For example, Alaa and van der Schaar (2019)

used Meijer G-functions to parametrize their surrogate explainer.

Combinations of forms of explanations. A single explainer can also provide more than one type of explanations. For example, the PJ-X model introduced by Park et al. (2018) is a self-explanatory model that provides both feature-based and natural language explanations at the same time, and the joint training for both types of explanations was shown to improve each of the two types of explanations. Similarly, Kanehira and Harada (2019) introduce a self-explainable model that generates both natural language and example-based explanations.

2.3 Feature-Based Explanations

As mentioned above, there are two major types of feature-based explanations: importance weights and subsets. I will describe each of these below. Let m be a model and $\mathbf{x}$ an instance with a potentially variable number n of features $\mathbf{x} = (x_1, x_2, ..., x_n)$ (e.g., x_i is the i-th token in the sentence $\mathbf{x}$). A feature-based explanation can be seen as a mapping of $\mathbf{x}$ to an output space. For importance weights explanations, each feature x_i of $\mathbf{x}$ is mapped to a real number. For subsets explanations, $\mathbf{x}$ is mapped to a collection of subsets of $\mathbf{x}$, each of which is a potential alternative explanation.

2.3.1 Importance Weights

Importance weights explanations attribute to each feature x_i in an instance $\mathbf{x}$ a weight $w_i(m, \mathbf{x})$ which represents the importance (also called contribution) of feature x_i for the prediction $m(\mathbf{x})$ (Ribeiro et al., 2016; Lundberg and Lee, 2017; Sundararajan et al., 2017; Lapuschkin et al., 2015; Arras et al., 2017; Shrikumar et al., 2017; Simonyan et al., 2014; Chen and Jordan, 2020). The weights are signed real numbers and the sign indicates whether the feature pulled the model towards the prediction (the same sign as the sign of the prediction) or against the prediction (the opposite of the prediction sign).

2.3.1.1 Feature-Additive Weights

A property that has been encountered in a large number of the importance weights explainers is that of *feature-additivity*, which requires the sum of the importance weights of all the features present in an instance to be equal to the prediction of the model on that instance minus the bias of the model. For classification tasks, the prediction of a model is considered to be the probability of the predicted class, usually the highest probability among the predicted probabilities of each possible class. The bias of a model is the model prediction on an input that brings *no information*, usually referred to as the *reference* or *baseline* input. For example, the all-black image is a common baseline in computer vision, while the zero-vector embedding is a common baseline in natural language processing (Chen et al., 2018a; Sundararajan et al., 2017). Similarly, the information brought by any feature in an instance $\mathbf{x}$ can be eliminated either by *occlusion* (replacing the feature with a baseline feature, such as a black super-pixel or a zero-vector embedding) or by *omission* (removing the feature completely) (Kádár et al., 2017) if that is technically possible, for example, for models that admit variable length input sequences, such as natural language processing models. Both occlusion and omission may result in out-of-distribution inputs that may lead to unreliable importance weights, as mentioned by Sundararajan et al. (2017).

Formally, for a model m and an instance $\mathbf{x}$, under the feature-additivity constraint, the features $\{x_i\}_i$ are attributed weights $\{w_i(m, \mathbf{x})\}_i$ such that

$$\sum_{i=1}^{|\mathbf{x}|} w_i(m, \mathbf{x}) = m(\mathbf{x}) - m(\mathbf{b}), \tag{2.1}$$

where $\mathbf{b}$ is the baseline input. For regression models, $m(\mathbf{x})$ is the real value predicted by the model on instance $\mathbf{x}$, while for classification tasks, $m(\mathbf{x})$ is the probability of the predicted class.

A large number of explanatory methods rely on the feature-additivity property (Ribeiro et al., 2016; Lundberg and Lee, 2017; Lapuschkin et al., 2015; Arras et al., 2017; Shrikumar et al., 2017; Simonyan et al., 2014; Sundararajan et al., 2017). For example, LIME (Ribeiro et al., 2016) learns a linear regression model locally on a

neighbourhood of each instance $\mathbf{x}$, where the neighbourhood is defined by all instances formed by setting all possible combinations of features in the instance $\mathbf{x}$ to a baseline value.

Shapley values. Lundberg and Lee (2017) aim to unify the feature-additive explanatory methods by showing that the only set of feature-additive importance weights that verify three properties—*local accuracy*, *missingness*, and *consistency*—are given by the Shapley values from cooperative game theory (Shapley, 1951).

1. *Local accuracy* (also referred to as completeness in Sundararajan et al. (2017)) requires that the equality in Equation 2.1 holds. Note that not all feature-additive methods guarantee that the provided importance weights satisfy this equation, even if it is their goal. For example, when LIME learns a linear regression on the neighbourhood of an instance $\mathbf{x}$, this linear regression model might not be able to reproduce the same prediction on the instance $\mathbf{x}$ that it aims to explain.

2. *Missingness* requires features that are not present in an instance $\mathbf{x}$ to be given zero importance weight for the prediction $m(\mathbf{x})$. For example, for the prediction of model m on sentence $\mathbf{x}$, only the tokens present in the sentence $\mathbf{x}$ should be part of the explanation and hence can be given a non-zero importance weight, while any other token in the vocabulary that is not present in the sentence $\mathbf{x}$ has 0 importance weight for the prediction $m(\mathbf{x})$.

3. *Consistency* requires that if for two models m and m' the marginal contribution of a feature x_i of an instance $\mathbf{x}$ is higher for model m than for model m' for each subset of features of $\mathbf{x}$, then the importance weight of x_i for the instance $\mathbf{x}$ should be higher for the model m than for the model m'. Formally, if for all subsets of features $\mathbf{x_s} \subseteq \mathbf{x} \setminus \{x_i\}$ we have that

$$m(\mathbf{x_s} \cup \{x_i\}) - m(\mathbf{x_s}) \geq m'(\mathbf{x_s} \cup \{x_i\}) - m'(\mathbf{x_s}), \tag{2.2}$$

   then $w_i(m, \mathbf{x}) \geq w_i(m', \mathbf{x})$.

Lundberg and Lee (2017) show that the Shapley values are the only solution which satisfy all three above properties, and can be computed in closed form as follows:

$$w_i(m, \mathbf{x}) = \sum_{\mathbf{x_s} \subseteq \mathbf{x} \setminus \{x_i\}} \frac{|\mathbf{x_s}|!(|\mathbf{x}| - |\mathbf{x_s}| - 1)!}{|\mathbf{x}|!} [m(\mathbf{x_s} \cup \{x_i\}) - m(\mathbf{x_s})], \tag{2.3}$$

where the sum enumerates over all subsets $\mathbf{x_s}$ of features in $\mathbf{x}$ that do not contain feature x_i, and $|\cdot|$ denotes the number of features of its argument. For tasks where the order of features matters, the features are kept in the same order as they appeared in $\mathbf{x}$.

Since computing the Shapley values in closed form would require exponential time in the number of features, Lundberg and Lee (2017) provide both model-agnostic (KernalSHAP) and model-dependant (DeepSHAP and MaxSHAP) methods for approximating the Shapley values.

2.3.1.2 Non-Feature-Additive Weights

The majority of the importance weights explainers are based on the feature-additivity property. However, importance weights explainers that do not rely on feature-additivity are also being developed. For example, LS-Tree (Chen and Jordan, 2020) leverages parse trees for linguistic data to assign weights to each feature such that the weights can be used to detect and quantify interactions between the tokens in a sentence. Similarly to the Shapley methods, LS-Tree also takes inspiration from cooperative game theory. However, instead of using the Shapley values, LS-Tree is inspired by the Banzhaf values (Banzhaf-III, 1964).

2.3.2 Minimal Sufficient Subsets

Another popular way of explaining the prediction of a model m on an instance $\mathbf{x}$ is to provide a minimal subset of features $\text{mss}(m, \mathbf{x}) \subseteq \mathbf{x}$ such that these features alone suffice for the same prediction to be reached by the model if the information from all other features is missing (Carter et al., 2019). Formally,

$$m(\text{mss}(m, \mathbf{x})) = m(\mathbf{x}) \text{ and} \tag{2.4}$$

$$\forall \mathbf{x_s} \subset \text{mss}(m, \mathbf{x}), \text{ we have that } m(\mathbf{x_s}) \neq m(\mathbf{x}). \tag{2.5}$$

To compute the prediction of a model m on a subset of features $\mathbf{x_s}$, one performs either an occlusion or a deletion of the features in $\mathbf{x} \setminus \mathbf{x_s}$, as explained above.

The minimality condition is necessary to ensure that one does not provide the trivial sufficient subset formed by all features.

It may not always be the case that a target model relies only on a subset of features, as opposed to needing all the features in an instance. If the whole input is necessary to explain the prediction, then such type of explanation would be uninformative. Nonetheless, it can be the case that a model relies only on a subset of the input features for each instance. For example, in computer vision, arguably not all pixels are essential for a model to classify an object or for providing an answer to a question. Similarly, in sentiment analysis, certain sub-phrases can be enough to identify the sentiment.

There can be multiple minimal sufficient subsets for one instance. Then, each minimal sufficient subset is one independent potential explanation for the model prediction on the instance. In this case, to provide a complete view of the model, one would ideally provide the collection of all minimal sufficient subsets for a model m and an instance $\mathbf{x}$, i.e., $\mathcal{S}(m, \mathbf{x}) = \{\text{mss}_k(m, \mathbf{x})\}_k$.In Chapter 3, I will expose problems with providing certain minimal sufficient subsets as explanations.

Explanatory methods such as L2X (Chen et al., 2018a), SIS (Carter et al., 2019), Anchors (Ribeiro et al., 2018), and INVASE (Yoon et al., 2019) aim to provide this type of explanation. For example, L2X learns a minimal sufficient subset by maximising the mutual information between the prediction of the model on only a subset of features, i.e., $m(\mathbf{x_s})$, and the prediction on the full instance $m(\mathbf{x})$. However, L2X assumes that the cardinality of a minimal sufficient subset is known in advanced and that it is the same for all instances, which is a major limitation in practice which may result in both the minimality and the sufficiency conditions being violated. In contrast, by using an actor-critic methodology (Peters and Schaal, 2008), INVASE allows the cardinality of the minimal sufficient subsets to differ for each instance. Both L2X and INVASE

provide only one minimal sufficient subset. On the other hand, SIS provides a set of minimal sufficient subsets that do not overlap. Thus, SIS might identify only a subset $\mathcal{S}'$ of the full collection $\mathcal{S}(m, \mathbf{x})$.

In Chapter 3, I provide more insight into the strengths and limitations of the Shapley-based explanations and the minimal sufficient subsets explanations.

2.4 Properties of Explanations

The purpose of having explanations for deep neural networks is that humans understand the decision-making processes of these models. Therefore, there are two main properties that explanations should satisfy: explanations should be easy for humans to interpret and should be faithfully describing the decision-making processes of the target models that they aim to explain.

Easiness to interpret. An explanation for a model prediction would not be useful if humans cannot easily interpret it. At the extreme, since neural networks are a composition of functions, one could explain any prediction by providing the full chain of functions applied to the instance, but this explanation would not be of any use to a human.

Each form of explanation has its own interpretation. For example, a minimal sufficient subset explanation is to be interpreted as a subset of features that suffice to lead the model to the same prediction when the information from all the rest of the features is eliminated and no other subset of this minimal sufficient subset would suffice for the model to reach the same conclusion.

However, not all explanatory methods are easy to interpret by humans. For example, explanations in the form of Shapley values give for each feature a weighted average of the marginal contributions of the feature inside each possible subset of features of the original instance. Arguably, such quantity can be difficult to interpret by humans, who may simply resort to looking at the relative importance of features given by the magnitudes of the Shapley values and at the direction in which each

feature pulls the model given by the signs of the Shapley values. I will discuss more on this topic in Chapter 3.

Faithfulness. Faithfulness of an explanation refers to the accuracy with which the explanation describes the decision-making process of the target model. Faithfulness of an explanation should not be confused with the property of an explanation to provide ground-truth argumentation for solving the task at hand, which is independent of a model decision-making process. I will call this latter property correctness (more details on this in Chapter 5). For example, a natural language explanation is faithful if the target model internally made use only of the argumentation provided by the explanation (encoded in a model-intelligible way) to reach its prediction.

Explanations directly influence the perception and trust of users in target models. Hence unfaithful explanations may be dangerous because can either encourage users to trust unreliable and potential hazardous models or discourage users from trusting perfectly reliable models.

As mentioned before, verifying faithfulness of explanations is one of the two main goals of this thesis, and it will be specifically addressed in Chapters 3, 4, and 6.

Chapter 3

Difficulties and Subjectivity of Explaining with Feature-Based Explanations

One of the main goals in this thesis is to verify whether explanations are faithfully describing the decision-making processes of the models that they aim to explain. In the first place, my focus is to introduce a framework to verify the faithfulness of the explanations provided by feature-based post-hoc explanatory methods. To do so, the first challenge is to define what means a faithful feature-based explanation. In Section 3.1 of this chapter, I show that for certain models and instances, there exist more than one ground-truth feature-based explanation. Hence, the faithfulness of an explanation (or of an explainer) depends on the type of ground-truth explanation preferred in practice.

Moreover, I show that two influential classes of explainers, namely, Shapley explainers and minimal sufficient subsets explainers, target different types of ground-truth explanations. I also demonstrate that in certain cases, neither of them is enough to provide a complete view of a decision-making process. Knowing that there can be more than one ground-truth feature-based explanation, as well as knowing to which ground-truth explanation each explainer adheres to, have not, to my knowledge, been previously emphasised in the literature. However, these are critical pieces of information for both users — to pick the class of explainers that best suits their needs and interpret the explanations correctly — and researchers — to tell to users

the intended behaviour, strengths, and limitations of the explainers, as well as to provide fair comparisons when automatically evaluating explanatory methods. On the contrary, influential works seem to imply that there is only one ground-truth feature-based explanation that all explainers aim to find.

The second challenging aspect in verifying the faithfulness of explanations is precisely the fact that, in general, we do not know the decision-making process of a neural model. To address this challenge, in Section 3.2 of this chapter, I investigate to what extent a particular type of feature-based self-explanatory model that is expected to provide faithful explanations actually does so. This type of model will further be employed as a testbed for verifying feature-based explanatory methods in Chapter 4.

The findings in this chapter, which are partially based on (Camburu et al., 2019), are not only important as a foundation for the verification framework that I will introduce in Chapter 4, but also by themselves because they point to fundamental aspects of explainability.

Illustrating examples. Before diving into the core of this chapter, I introduce the reader to the type of illustrating example that I will use extensively hereafter. More precisely, I will use hypothetical models for the task of textual sentiment analysis to exemplify the points made in this chapter. Such models take as input a review, i.e., a piece of text that contains an opinion on an object, and outputs a score that reflects the sentiment of the review towards the object or towards one aspect of the object. This task is a salient task in natural language processing with important real-world applications (McAuley et al., 2012; Maas et al., 2011). Similarly to Lei et al. (2016), I will treat the task as a regression, with the score being a real number linearly reflecting the intensity of the sentiment. In the examples, -1 will be the most negative score, 1 the most positive score, hence 0 reflects the neutral score. We assume a difference of at least 0.1 to be significant.[1] An important type of sentiment analysis task is that of identifying the sentiment towards one aspect in a multi-aspect review,

[1]In practice, this can happen, for example, if the scores are continuous transformations of star ratings that accompany the review. If one can choose to give half-star ratings of up to 5 stars, then a difference of 0.1 in a sentiment score corresponds to different star ratings.

m: IF "very good" IN INPUT: RETURN 0.9;
ELSE IF "nice" IN INPUT: RETURN 0.7;
ELSE IF "good" IN INPUT: RETURN 0.6;
ELSE RETURN 0.

$\mathbf{x}^{(1)}$: "The movie was good, it was actually nice."
$m(\mathbf{x}^{(1)}) = 0.7$

Shapley values	Minimal sufficient subsets
"nice": 0.4	{"nice"}
"good": 0.3	
rest of tokens: 0	

$\mathbf{x}^{(2)}$: "The movie was nice, in fact, it was very good."
$m(\mathbf{x}^{(2)}) = 0.9$

Shapley values	Minimal sufficient subsets
"good": 0.417	{"good", "very"}
"nice": 0.367	
"very": 0.116	
rest of tokens: 0	

Figure 3.1: Examples of cases with at least two ground-truth feature-based explanations given by Shapley explainers and minimal sufficient subsets explainers, respectively. The Shapley values are computed from Equation 2.3. The assumption is that the scores are linearly reflecting the intensity of the sentiment and that a difference of 0.1 is significant.

i.e., a review that contains opinions on multiple aspects of the object. For example, the BeerAdvocate dataset (McAuley et al., 2012) contains human-written reviews that cover four aspects of a beer, namely, appearance, palate, taste, and smell. In this chapter, I will use several examples of hypothetical models for both overall and multi-aspect sentiment analysis as illustrating examples.

3.1 Multiple Types of Ground-Truth Feature-Based Explanations

In this section, I show that there are cases of models and instances for which there exist more than one ground-truth feature-based explanation. I also show that two prevalent classes of explainers each promote different ground-truth explanations without explicitly mentioning it. This has not, to my knowledge, been emphasised in the literature so far. On the contrary, current influential works seem to imply that there is only one ground-truth feature-based explanation for a prediction of a model. Nonetheless, it is critical for both users and researchers alike to be aware of these differences.

Hereafter, I will label as a Shapley explainer any explainer that explicitly aims to provide Shapley values as explanations. As we saw in Chapter 2, Lundberg

and Lee (2017) argued that all feature-additive explainers should aim for Shapley values as explanations. However, given the findings in this chapter, I want to allow feature-additive explainers to adhere to different views.

To best illustrate the existence of more than one ground-truth feature-based explanation for the prediction of a model on an instance, in Figure 3.1, I present an example of a hypothetical sentiment analysis regression model m. This model makes its predictions as follows: the mere presence of the substring "very good" in the input text leads to a very positive score of 0.9. In the absence of "very good", if "nice" is present in the input text, then the model provides a positive sentiment of 0.7. If neither "very good" nor "nice" are present but "good" is present in the input text, then the model provides a score of 0.6, and finally, if none of these positive-indicator tokens are present, then the model defaults to the neutral score of 0. This is, therefore, a trivial model for which we know its decision-making process. Applying this model to the instance $\mathbf{x}^{(1)}$: "The movie was good, it was actually nice.", the model predicts $m(\mathbf{x}^{(1)}) = 0.7$ because "nice" is present in the input instance. Hence, one can argue that "nice" is the only important feature for this prediction of 0.7, while "good" would have been the only important feature for a different prediction (of 0.6). However, one may argue that "good" also has to be flagged as important for this prediction for the following reason: if "nice" is to be eliminated, then the model would rely on "good" to provide a score as high as 0.6 instead of the much lower default of 0. Therefore, we see that there are two equally valid perspectives on what a ground-truth feature-based explanation should be, even for this trivial model. Therefore, it is subjective which perspective is preferred in practice.

The difficulty faced when trying to explain model m with feature-based explanations is even more pronounced on the instance $\mathbf{x}^{(2)}$: "The movie was nice, in fact, it was very good.". The model predicts $m(\mathbf{x}^{(2)}) = 0.9$ because "very good" is a substring of the input instance. Hence, the features "very" and "good" form one ground-truth explanation for this prediction. However, for this instance, if "good" is eliminated, the model relies on "nice" (and not on "very") to provide a score as high as 0.7, while if both "good" and "nice"are eliminated, then the score drops all the way to 0. Hence,

from this perspective, "nice" can be seen as more important than "very", and the explanation that provides "good", "nice", and "very" in this order of importance is also a ground-truth feature-based explanation.

Shapley values vs. minimal sufficient subsets. The two types of ground-truth explanations described above are separately advocated by the two influential classes of Shapley and minimal sufficient subsets explainers. On one hand, Shapley explainers (Lundberg and Lee, 2017) tell us that "nice" is the most important feature for this model on this instance $\mathbf{x}^{(1)}$, with a weight of 0.4, but also that "good" has a significant contribution of 0.3 (see Equation 2.3). On the other hand, minimal sufficient subsets explainers find that only the feature "nice" is important (see Equations 2.4 and 2.5). Similarly, for the prediction of model m on instance $\mathbf{x}^{(2)}$, the Shapley explanation tells us that "good" and "nice" are the two most important features with very close importance weights, and that "very" is about three times less important than "nice". On the other hand, the minimal sufficient subsets perspective tells us that the two most important features are instead "good" and "very", while "nice" would not be mentioned as important at all.

The difference between the two types of ground-truth explanations stems from the fact that the explanation aligned with the Shapley values aims to provide *average importance weights*[2] for the features on a *neighbourhood* of the instance, while the explanation aligned with the minimal sufficient subsets aims to provide the *pointwise* features used by the model on the instance *in isolation*. The Shapley values come from cooperative game theory (Shapley, 1951), where they were introduced to promote *fairness* in distributing a total gain of a coalition among its players. This is why the Shapley values take into account the *capabilities* of players to perform in any sub-coalition, via the consistency condition in Equation 2.2. On the other hand, the minimal sufficient subsets perspective rewards only the players that are absolutely necessary inside the full coalition. In the example in Figure 3.1, for the model m,

[2]More precisely, a weighted average of the marginal contributions of the feature inside each possible subset of features of the original instance.

"nice" is not necessary in the presence of "very" and "good" appearing next to each other, even if "nice" alone would *win* a significant amount of 0.7.

I will thereafter refer to these two types of ground-truth explanations as *neighbourhood explanation* and *pointwise explanation*, respectively. These names are meant to reflect the high-level fundamental difference between the two ground-truth explanations, such as considering "nice" as more important than "very" versus considering "nice" as not important at all for explaining the prediction of m on $\mathbf{x}^{(2)}$, as opposed to low-level differences, such as the exact values of the Shapley weights or the fact that minimal sufficient subsets perspective provide subsets instead of weights, for which I will explicitly refer to as Shapley explanation (or Shapley explainer) and minimal sufficient subsets explanations (or minimal sufficient subsets explainer) to avoid confusion.

Not always distinct. Even though the two types of ground-truth explanations are distinct for certain models and instances, such as the ones mentioned above, in many cases they can coincide. For example, for the same model m but applied to instances that contain only one of the three key sub-phrases "very good", "nice", and "good", such as "The movie was good.", both the neighbourhood explanation and the pointwise explanation point towards the same important features.

Not emphasised in the literature. To my knowledge, the fundamental difference between the two types of ground-truth feature-based explanations illustrated above was only briefly alluded by Sundararajan et al. (2017) in their example of explaining the function $\min(x_1, x_2)$ on the instance $x_1 = 1, x_2 = 3$. Their method, called integrated gradients, attributes the whole importance weight (of $1 = \min(1, 3)$) to the *critical* feature x_1 (and hence 0 importance to x_2), while Shapley-Shubik (Shapley and Shubik, 1971) attributes $0.5 = \frac{1}{2}\min(1, 3)$ weight to each feature. They also mention that preferring one explanation over the other is a subjective matter.

On the other hand, the influential work of Lundberg and Lee (2017) implies, only with a graph of results (their Figure 4), that in their user study on Amazon Mechanical

Turk, *all* participants provided as explanation for the function $\max(x_1, x_2, x_3)$[3] on the input $x_1 = 5, x_2 = 4, x_3 = 0$ that x_1 has an importance weight of 3, x_2 of 2, and x_3 of 0. The authors do not mention the number or characteristics of the participants in their user study, nor do they provide the precise guidelines that they gave to the participants. More extensive and well-documented user studies are necessary to conclude the usefulness of one ground-truth explanation over the other.

Another pattern present in current works that may hinder the acknowledgement of both the existence of more than one ground-truth explanation and the fundamental difference between the Shapley and minimal sufficient subsets explainers is the fact the explainers from the different classes are directly compared on their ability to identify ground-truth features used by models for their predictions. For example, both Chen et al. (2018a) and Yoon et al. (2019) compare L2X, a minimal sufficient subsets explainer, with Shapley methods (Lundberg and Lee, 2017) on identifying the ground-truth features used by a model trained on synthetic datasets where the ground-truth features are known, called 2-dimensional XOR, Orange Skin, and Switch Feature. While these particular synthetic datasets happen not to violate either of the two ground-truth explanations described above, this information is not mentioned at the time of comparison. Such comparisons risk inducing the idea that there is always only one ground-truth feature-based explanation that all feature-based explainers should aim to find.

3.1.1 Strengths and Limitations

This section reveals strengths and limitations of the two types of feature-based explanations presented above.

Redundant features. By looking only at the Shapley explanation for $m(\mathbf{x}^1)$ in Figure 3.1, one cannot know whether (1) the model requires *both* features "nice" and "good" to make its prediction of 0.7 (which is not the case for m), or (2) one of these

[3]Devised as a story of three people making money based on the maximum score that any of them achieved.

1. ASPECT_INDICATORS = {"taste", "smell", "appearance"} (AND ANY VARIATION, SUCH AS "Tastes")
2. SENTIMENT_INDICATORS = {"amazing" → 1, "good" → 0.6, "refreshing" → 0.6, "bad" → −0.6, "peculiar"→ −0.3, "horrible" → −1}
3. A SENTIMENT INDICATOR IS ASSOCIATED TO ITS CLOSEST ASPECT (OCCLUDED TOKENS ARE COUNTED).
4. AN OCCLUDED TOKEN IS CONSIDERED TO BE NEUTRAL.
5. IF MORE SENTIMENT INDICATORS ARE ASSOCIATED TO AN ASPECT, THEN
   (i) IF ALL ARE OF THE SAME SIGN: THE SCORE FOR THAT ASPECT IS THE SCORE OF THE STRONGEST SENTIMENT.
   (ii) IF THERE ARE BOTH POSITIVE AND NEGATIVE SENTIMENTS ASSOCIATED TO THE ASPECT: THE SCORE FOR THAT ASPECT IS THE THRESHOLDED SUM OF SCORES (MAX(MIN(SUM_SCORES, 1), −1)).

m^{O}: s = SUM OF SCORES OF ASPECTS
RETURN MAX(MIN(s, 1), −1)

$\mathbf{x}^{\mathrm{O}}$: "The beer has an amazing appearance, a good smell, a bad taste."
$m^{\mathrm{O}}(\mathbf{x}^{\mathrm{O}}) = 1$

Shapley explanation	MSS explanation
1. "amazing": 0.52	{"amazing", "appearance"}
2. "good": 0.40	
3. "bad": −0.23	
4. "smell": 0.15	
5. "appearance": 0.12	
6. "taste": 0.03	

m^{S}: RETURN SCORE OF SMELL

$\mathbf{x}^{\mathrm{S1}}$: "Tastes horrible, peculiar smell."
$m^{\mathrm{S}}(\mathbf{x}^{\mathrm{S1}}) = -0.3$

Shapley explanation	MSS explanation
1. "smell": −0.29	{"peculiar", "smell"}
2. "Tastes": 0.26	
3. "horrible": −0.14	
4. "peculiar": −0.13	

$\mathbf{x}^{\mathrm{S2}}$: "Tastes amazing, peculiar smell."
$m^{\mathrm{S}}(\mathbf{x}^{\mathrm{S2}}) = -0.3$

Shapley explanation	MSS explanation
1. "peculiar": −0.27	{"peculiar", "smell"}
2. "smell": −0.10	
3. "amazing": 0.05	
4. "Tastes": 0.02	

m^{T}: RETURN SCORE OF TASTE

$\mathbf{x}^{\mathrm{T1}}$: "Tastes good, refreshing."
$m^{\mathrm{T}}(\mathbf{x}^{\mathrm{T1}}) = 0.6$

Shapley explanation	MSSs explanations
1. "Tastes": 0.4	{"Tastes", "good"},
2. "good": 0.1, "refreshing": 0.1	{"Tastes", "refreshing"}

$\mathbf{x}^{\mathrm{T2}}$: "Tastes amazing. The smell is also amazing."
$m^{\mathrm{T}}(\mathbf{x}^{\mathrm{T2}}) = 0.6$

Shapley explanation	MSSs explanations
1. "Tastes": 0.58	{"Tastes", "amazing^{1}"},
2. "amazing^{1}": 0.42	{"Tastes", "amazing^{2}"}
3. "amazing^{2}": 0.08	
4. "smell": −0.08	

Figure 3.2: Examples illustrating the strengths and limitations of the two types of feature-based explanations, as presented in Section 3.1.1. The five rules are common to all three models. The Shapley values were computed via Equation 2.3, and written in decreasing order of their importance (absolute value); the non-mentioned features received 0 weight. MSS stands for minimal sufficient subset. In the last example, the superscript of "amazing" differentiates between its two occurrences.

features is redundant in the presence of the other (which is the case for m). In contrast, minimal sufficient subset explanations do not contain redundant features (as they would violate Equation 2.5), and hence, the minimal sufficient subset explanation for $m(\mathbf{x}^1)$ is able to distinguish between the two scenarios.

Feature cancellations: genuine vs. artefacts. In certain cases, there exist features that cancel each other out. Consider the model m^{O} in Figure 3.2, which predicts the overall sentiment on a beer from a multi-aspect review by adding up the scores that it associates to each aspect in the review. On the instance $\mathbf{x}^{\mathrm{O}}$, m^{O} predicts 1 by taking into account all three aspects. However, the minimal sufficient subset explanation is {"amazing", "appearance"}—it does not contain the features "bad", "taste", "good", and "smell", due to Equation 2.5. Arguably, users may want to see the features from such a *genuine cancellation* in the explanation. Note that these features are flagged as important by the Shapley explanation, which, nonetheless, does

not clearly indicate the perfect cancellation between "good smell" and "bad taste". Moreover, the Shapley explanation gives the impression that "smell" and "appearance" are much more important (0.15 and 0.12) than "taste" (0.03), when, by design, m^{O} equally takes into account all aspects. Hence, neither the Shapley nor the minimal sufficient subset explanation is well reflecting the decision-making process for $m^{\mathrm{O}}(\mathbf{x}^{\mathrm{O}})$.

Artefacts may occur when eliminating features from an instance, distorting the importance of certain features. Model m^{S} in Figure 3.2 illustrates such an example. When m^{S} is applied to $\mathbf{x}^{\mathrm{S1}}$, it predicts -0.3, and the minimal sufficient subset explanation is {"peculiar", "smell"}, which, arguably, best reflects the decision-making process for $m^{\mathrm{S}}(\mathbf{x}^{\mathrm{S1}})$. However, in the Shapley explanation ,"Tastes" appears to be twice more important than "peculiar", and "horrible" appears as important as "peculiar", even though "peculiar" is the actual sentiment indicator for smell. Furthermore, note how the Shapley importance weights dramatically change when only the sentiment on taste is changed in instance $\mathbf{x}^{\mathrm{S2}}$, even though m^{S} does not rely on the sentiment on taste to predict the sentiment on smell.

Multiple minimal sufficient subsets: genuine vs. artefacts. In certain cases, there can exist multiple minimal sufficient subsets explanations for one prediction. For example, for the model m^{T} and instance $\mathbf{x}^{\mathrm{T1}}$ in Figure 3.2, either of the features "good" and "refreshing" leads to the score of 0.6. Ideally, minimal sufficient subsets explainers provide all the genuine minimal sufficient subsets, e.g., both {"Tastes", "good"} and {"Tastes", "refreshing"}. However, many minimal sufficient subsets explainers are designed to retrieve only one minimal sufficient subset (Chen et al., 2018a; Yoon et al., 2019). An exception is the SIS explainer (Carter et al., 2019), which retrieves a set of disjoint minimal sufficient subsets, which might also not be exhaustive (e.g., SIS would not retrieve the second minimal sufficient subset explanation for $m^{\mathrm{T}}(\mathbf{x}^{\mathrm{T1}})$, because "Tastes" is already taken by the first minimal sufficient subset). On the other hand, the Shapley explanation gives the same importance to both "good" and "refreshing". However, with this explanation alone, one would not be able to know whether both

"good" and "refreshing" are necessary to be present or if each individually suffices for the prediction of 0.6.

Artefacts occurring when eliminating features can also lead certain subsets of features to appear as minimal sufficient subsets. For example, for m^{T} and $\mathbf{x}^{\mathrm{T2}}$ in Figure 3.2, either of the two occurrences of "amazing" forms an minimal sufficient subset, but the second one is not reflecting the decision-making process of the model. The Shapley explanation makes the distinction in this case.

In this section, we saw that for certain cases, there can be more than one ground-truth feature-based explanation for a prediction of a model. I identified two such types of explanations and called them neighbourhood and pointwise explanations. We also saw that the Shapley explainers aim to provide neighbourhood explanations, while the minimal sufficient subsets explainers aim to provide pointwise explanations. Both types of explanations and explainers give valuable insights into the decision-making process of a target model, and have their strengths and limitations. Hence, one type of explanation may be preferred over the other in different real-world use-cases, and the choice of ground-truth is therefore best left in the hands of the users. Therefore, users need to be informed of the strengths, limitations, and differences among various types of explanations and explainers, in order to pick and use them accordingly.

Finally, using both neighbourhood and pointwise explanations together may bring a complete picture of the decision-making process of a model. The investigation on combining the two types of explanations can therefore be an interesting future work. Additionally, future work may include a comprehensive user study to decide the extent to which users could benefit from each of these types of explanations, as well as from their combination.

Most importantly, the work described in this section encourages researchers to pay particular attention to the assumptions behind the explanations that their explanatory methods aim to provide, and to state them directly in their works. For example, a good practice could be that each work introducing an explainer or a verification

framework for explainers has a section *Specifications* where the authors present the intended behaviour, strengths, and limitations of their method(s).

3.2 Trusting Selector-Predictor Models

A potential way of verifying the faithfulness of post-hoc explainers is to use feature-based self-explanatory models as testbed, since these models are supposed to provide the features that are relevant for their predictions. In this section, I investigate a particular type of feature-based self-explanatory model, which I call a *selector-predictor* model. A selector-predictor model is composed of two sequential modules: a *selector* followed by a *predictor*. The selector makes a *hard* selection of a subset of the input features, and *only* the selected features are passed along to the predictor, which outputs the final answer. There is no restriction on the types of neural networks used for modelling the selector and the predictor. Crucially, there is also no need for supervision on the selected features, as long as the selector and the predictor are trained jointly with supervision only on the final answer. Therefore, such a feature-based self-explanatory model is very appealing. Models following this high-level architecture have been introduced, for example, by Lei et al. (2016), Yoon et al. (2019), and Chang et al. (2019).

I investigate the type of selector-predictor self-explanatory model, since it seemed to conveniently provide, for any instance, the set of relevant features (the selected ones) and irrelevant features (presumably the non-selected ones). Therefore, it appeared that one could test if explainers correctly identify the irrelevant features as less important than the relevant ones. However, we will see below that this type of self-explanatory model may not always faithfully explain itself.

Formally, let $\mathcal{D}$ be a dataset for which any instance $\mathbf{x}$ has a potentially variable number n of features $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. For example, $\mathbf{x}$ can be a sentence, and the feature x_i the i-th token in the sentence. The selector, which we call $sel(\cdot)$, takes the input $\mathbf{x}$ and returns a subset of features $\mathbf{S}^m_\mathbf{x} \subseteq \mathbf{x}$. The predictor, which we call $pred(\cdot)$, takes as input *only* the features $\mathbf{S}^m_\mathbf{x} := sel(\mathbf{x})$ and makes its prediction based

exclusively on these features. It thus holds that

$$m(\mathbf{x}) = pred(sel(\mathbf{x})) = pred(\mathbf{S}^m_\mathbf{x}).$$

We also call $\mathbf{N}^m_\mathbf{x}$ the non-selected features, i.e., $\mathbf{N}^m_\mathbf{x} = \mathbf{x} \setminus \mathbf{S}^m_\mathbf{x}$.

For example, for a model tackling a sentiment analysis task, the selector may learn to select only the token "amazing" if this token is in the input text, and the predictor may learn that when it receives the input $\mathbf{S}^m_\mathbf{x} = \{\text{"amazing"}\}$, it has to predict the most positive sentiment score.

Selector-predictor models aim for minimal sufficient subsets explanations. To encourage the selector to do a meaningful job and not simply return all features (unless necessary), one can, for example, use a regularizer that penalises the model proportionally to the number of selected features, as was done in the work of Lei et al. (2016). Thus, such a selector-predictor model aims for a minimal sufficient subset explanation, as defined by Equations 2.4 and 2.5. For example, a selector-predictor model that learns to mimic the decision-making process of the model m from Figure 3.1, would likely not select the feature "nice" if "very good" is a substring of the input. This is because selecting "nice" in addition to "very" and "good" would not change the prediction, but it would increase the penalisation on the number of selected features.

Minimality is not guaranteed. While a regularizer can incentives the minimality on the number of selected features, it is not guaranteed that a selector-predictor model would never select features that are not strictly necessary for the prediction, as this depends on how well the model learns to do so during training time. Moreover, it is an open question whether checking if the set of selected features is minimal can be done with less than $O(2^{|\mathbf{S}^m_\mathbf{x}|})$ queries to the model, since any subset of the selected features may be a minimal sufficient subset.

From the perspective of a self-explanatory model, not complying with the minimality condition may sometimes be crucial. For example, if the model learned to rely on

a spurious correlation, such as learning that the word "bottle" alone implies a positive sentiment, and if the selected features include ground-truth sentiment-indicator words, then the explanation formed by the selected features would hide from the users the spurious correlation learned by the model. On the other hand, as mentioned in Section 3.1, minimality implies considering features that cancel each other out as irrelevant, and this may not always be desired. Hence, the fact that a selector-predictor model may not always fulfil the minimality condition might sometimes be useful in practice.

From a verification perspective, if certain selected features are irrelevant for the prediction, then it is a mistake to penalise an explainer for not presenting those features as more important than other equally irrelevant features.

```
IF "very good" IN INPUT: SELECT {"very"} & RETURN 1;
ELSE IF "not good" IN INPUT: SELECT {"not"} & RETURN -0.8;
ELSE IF "good" IN INPUT: SELECT {"good"} & RETURN 0.8;
ELSE SELECT ∅ & RETURN 0.
```

Figure 3.3: Example of a selector-predictor model that does not always select sufficient subsets of features. For instances containing "very good" or "not good", "good" is a crucial feature for the prediction, yet the model does not select it. However, the model would not present this undesired behaviour for any instance that does not contain "very good" or "not good".

Sufficiency is not guaranteed. Another concerning issue of selector-predictor models is the fact that selected features are not guaranteed to form a sufficient subset, i.e., the model can rely on features it does not select. More precisely, m may learn an internal emergent communication protocol (Foerster et al., 2016) between its selector and predictor such that non-selected features are crucially influencing the prediction via a hidden encoding that the selector and predictor agree upon during training. An example of a model that does not select sufficient subsets is given in Figure 3.3. For instances containing the substrings "very good" or "not good", such as $\mathbf{x}^{(1)}$: "The movie is very good." and $\mathbf{x}^{(2)}$: "The movie is not good.", the feature "good" is influencing the prediction, but the model does not select it. However, the same model is not exhibiting the undesired behaviour for any instance that does not contain "very

good" or "not good" as substrings. In practice, spurious correlations in datasets may lead to even more misleading selections of features done by a selector-predictor model. Unlike checking the minimality condition, fortunately, it is far less time-consuming to check whether a selector-predictor model m selects a sufficient subset on an instance $\mathbf{x}$. By definition, we have that

$$\mathbf{S}^m_{\mathbf{x}} \text{ is a sufficient subset } \iff m(\mathbf{S}^m_{\mathbf{x}}) = m(\mathbf{x}). \tag{3.1}$$

Hence, for using a selector-predictor model either as a self-explanatory model or as a testbed for verifying explainers, it is crucial to check the sufficiency condition beforehand.

Inconsistency in selecting features that cancel each other out. We saw that a selector-predictor is not guaranteed to select a minimal set of sufficient features. Hence, on each instance, such a model may select certain groups of features that cancel each other out, while it may not select others. This inconsistency can have advantages and disadvantages. On one hand, such a model has a chance to select the features that form genuine cancellations and not select the features that form artefact-caused cancellations. Neither minimal sufficient subsets explainers nor Shapley explainers are capable of doing so. On the other hand, the opposite may also happen, that is, a selector-predictor model may select features that form artefact-caused cancellations and not select features that form genuine cancellations. Hence, users need to be aware of this risk. It is an open question whether one could check in less than an exponential number of queries to the model whether there are non-selected features that cancel each other out.

3.3 Conclusions and Open Questions

In this chapter, I unveiled certain difficulties of explaining models with feature-based explanations. First, I showed that there can be more than one ground-truth feature-based explanation for the prediction of a model on an instance. I also showed that two prevalent classes of post-hoc feature-based explanatory methods aim to provide

different ground-truth explanations, and I uncovered some of their strengths and limitations.

Second, I investigated a type of feature-based self-explanatory model, called selector-predictor, and showed its strengths and limitations in providing its own explanations.

The findings in this chapter encourage researchers to directly state the specifications of their explainers and verification frameworks. They also pave the way to further investigations, such as: (1) which type of feature-based ground-truth explanations or which combination of them best suit users in different circumstances, and (2) whether selector-predictor models can be regularised during training to enforce the sufficiency condition on the selected features.

Chapter 4

Verifying Feature-Based Post-Hoc Explanatory Methods

In this chapter, which is based on (Camburu et al., 2019), I introduce a framework for verifying the faithfulness with which feature-based post-hoc explanatory methods describe the decision-making processes of the models that they aim to explain.

4.1 Motivation

As we saw in Chapter 2, a large number of feature-based post-hoc explanatory methods have recently been developed with the goal of shedding light on the decision-making processes learned by neural models (Ribeiro et al., 2016; Lundberg and Lee, 2017; Lapuschkin et al., 2015; Shrikumar et al., 2017; Sundararajan et al., 2017; Simonyan et al., 2014; Ribeiro et al., 2018; Chen et al., 2018a; Chen and Jordan, 2020). While current explainers manage to point out catastrophic biases of certain models, such as relying on the background of an image to discriminate between *Wolf* and *Husky* (Ribeiro et al., 2016), it is an open question how to verify if these methods are faithfully describing the decision-making process of a model that has a less obvious bias. This is a difficult task precisely because, generally, the decision-making process of deep neural networks is not known. Consequently, the automatic verification of explanatory methods usually relies either on oversimplistic scenarios (an interpretable yet trivial target model such as a linear regression model, or a non-trivial target model but trained on synthetic datasets), or on the assumption that accurate target models

must behave reasonably, i.e., that they do not rely on spurious correlations in the datasets. For example, in their verification test based on morphosyntactic agreement, Poerner et al. (2018) assume that a model which predicts whether a verb should be singular or plural given the tokens before the verb must be doing so by focusing on a noun that the model must have identified as the subject. Such assumptions may be poor since recent works show that neural networks can learn to heavily rely on surprising spurious correlations even when they provide correct answers (Gururangan et al., 2018; McCoy et al., 2019; Glockner et al., 2018a). Therefore, it is not reliable to verify the faithfulness of an explainer only based on whether the explanations appear coherent for the task at hand.

In order to overcome these issues, I propose to use the type of selector-predictor model introduced in Chapter 3, since it has the potential to provide a set of relevant and irrelevant features for each of its predictions. Also, this type of model is a non-trivial neural model that can be trained on real-world datasets. However, as we saw in the previous chapter, selector-predictor models can have certain limitations that one needs to take into account when using them as testbeds for verifying explainers. Therefore, I introduce checks to account for some of these limitations, as well as further improvements that can be done to further alleviate the remaining limitations.

I instantiate the SelPredVerif framework on the selector-predictor model introduced by Lei et al. (2016) and trained on the BeerAdvocate dataset (McAuley et al., 2012) for the task of multi-aspect sentiment analysis. I thereby produce three sanity tests, one for each aspect in BeerAdvocate. Further, I test three popular explainers, namely, LIME (Ribeiro et al., 2016), KernalSHAP (Lundberg and Lee, 2017), and L2X (Chen et al., 2018a), and provide their performances on each of these three sanity tests to raise awareness of the potential unfaithful explanations that these explainers can produce.

The three sanity tests can be used off-the-shelf[1] for testing existing and future feature-based post-hoc explanatory methods. Moreover, SelPredVerif is generic and can be instantiated on other tasks, domains, and neural architectures for the

[1]The tests are available at `https://github.com/OanaMariaCamburu/CanITrustTheExplainer`.

selector and the predictor networks, thus allowing the generation of a large number of sanity tests.

4.2 Related Work

Despite the increasing number of proposed explanatory methods, it is still an open question how to thoroughly validate their faithfulness to the target models that they aim to explain. There are four types of verification commonly performed:

1. **Interpretable yet simple target models.** Typically, explainers are tested on linear regression and decision trees (e.g., in Ribeiro et al. (2016)) or support vector representations (e.g., in Plumb et al. (2018)). While this evaluation accurately assesses the faithfulness of the explainer to the target model, these very simple models may not be representative of the large and intricate neural networks used in practice.

2. **Synthetic setups.** Another popular setup is to create synthetic tasks where the set of important features is controlled (Chen et al., 2018a; Yoon et al., 2019; Lundberg and Lee, 2017). For example, Chen et al. (2018a) performed evaluations on four synthetic tasks. While there is no limit on the complexity of the target models trained in these setups, their synthetic nature may still prompt the target models to learn simpler functions than the ones needed for real-world applications. This, in turn, may ease the job for the explainers.

3. **Assuming a reasonable behaviour of the target model.** In this setup, one identifies certain intuitive heuristics that a high-performing target model is assumed to follow. For example, Chen et al. (2018a) assume that a neural network that performs well on sentiment analysis must rely on tokens that convey the predicted sentiment. Similarly, Poerner et al. (2018) assumes that a neural network that performs well on deciding whether a verb is singular or plural must rely on a noun of the same number that the model must have identified as the subject of the verb. Thus, under this evaluation procedure,

an explainer is assessed based on whether it points out the features that are in ground-truth correlation with the prediction of the model. However, neural networks may rely on surprising spurious correlations even when they obtain a high test accuracy (Gururangan et al., 2018; Ribeiro et al., 2016; McCoy et al., 2019; Glockner et al., 2018a). Hence, this type of verification is not reliable for assessing the faithfulness of the explainers to the target model.

4. **Improved human understanding of a model.** A non-automatic way to evaluate explainers is to check how their explanations help humans predict the behaviour of target models. In this evaluation, humans are presented with a series of a predictions of a model and explanations from different explainers, and are asked to infer the predictions that the model would make on a separate set of examples. One concludes that an explainer e_1 is better than an explainer e_2 if humans are better at predicting the output of the model after seeing explanations from e_1 than after seeing explanations from e_2 (Ribeiro et al., 2018). While this type of evaluation is arguably the most effective since it simulates the real-world usage of explanatory methods, it is nonetheless expensive and requires considerable human effort if it is to be applied to complex real-world neural network models. Hence, it is desirable to have complementary automatic sanity tests that can be used before human evaluations.

In contrast to the above, SelPredVerif generates automatic sanity tests in which the target model is a non-trivial neural network trained on real-world datasets and for which we know part of its decision-making process.

SelPredVerif is similar in goal to the framework introduced by Adebayo et al. (2018). However, their framework tests for the basic requirement that an explainer should provide different explanations for a target model trained on the real data than for the same target model trained on randomised data, or for target models that were not trained at all. SelPredVerif is more challenging and requires a stronger fidelity of the explainer to the target model. Similarly to the concurrent work of Yang and Kim (2019), SelPredVerif checks for false positives, i.e., whether the explainers

are ranking less relevant features higher than more relevant ones. However, their framework uses semi-synthetic datasets, since they paste objects randomly over scene images, while SelPredVerif can be applied to any real-world dataset.

4.3 Verification Framework

As in the previous chapter, let $\mathcal{D}$ be a dataset for which any instance $\mathbf{x}$ has a potentially variable number of features $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. Let $m(\mathbf{x}) = \text{pred}(\text{sel}(\mathbf{x}))$ be a selector-predictor model, and denote by $\mathbf{S}_\mathbf{x}^m := \text{sel}(\mathbf{x})$ the selected features and by $\mathbf{N}_\mathbf{x}^m := \mathbf{x} \setminus \mathbf{S}_\mathbf{x}^m$ the non-selected features. The goal is to find a set of instances $\mathcal{G}^m \subseteq \mathcal{D}$ such that, for each instance $\mathbf{x} \in \mathcal{G}^m$, we know a subset of features that are relevant to the prediction of m on $\mathbf{x}$ and a subset of features that are irrelevant to this prediction. With a dataset $\mathcal{G}^m$ with such guarantees on the relevance of the features in each instance, one can verify whether an explainer is wrongly attributing more importance to irrelevant features rather than to relevant ones.

In the previous chapter, I showed that selector-predictor models aim to provide a minimal sufficient subset as an explanation, via the selected features. Hence, such models aim for the pointwise type of ground-truth explanations. Consequently, this verification framework is tailored for this kind of ground-truth explanations. However, we also saw that neither the minimality nor the sufficiency of the selected features is guaranteed. Therefore, it is possible that the sanity tests produced by SelPredVerif do not penalise neighbourhood explanations if redundant features are selected (such as "nice" in the instance $\mathbf{x}^{(2)}$ in Figure 3.1).

Below I show how one can use such a model as a testbed for explainers, together with the strengths and limitations that come along.

Finding a set of relevant features. It is an open question how to check whether the subset of selected features is minimal without an exponential number of queries to the model (on each subset of the selected features). Not being able to guarantee minimality may bring the following issues:

(I1) certain selected features may be irrelevant to the prediction,

(I2) certain groups of features that cancel each other out may be selected while others may not be selected.

Issue (I1) can lead to wrongly penalising an explainer for not placing an irrelevant feature higher than another equally irrelevant feature. To avoid this, I identify the subset of selected features whose individual absence significantly changes the prediction. More precisely, I identify the subset $\mathbf{SIR}^m_{\mathbf{x}} \subseteq \mathbf{S}^m_{\mathbf{x}}$ as

$$\mathbf{SIR}^m_{\mathbf{x}} = \{x_i \in \mathbf{S}^m_{\mathbf{x}} : \ \text{abs}(\text{pred}(\mathbf{S}^m_{\mathbf{x}} \setminus \{x_i\}) - m(\mathbf{x})) \geq \tau\}, \tag{4.1}$$

where τ is the significance threshold, and $\text{abs}(\cdot)$ is the absolute value function. The name $\mathbf{SIR}^m_{\mathbf{x}}$ stands for Selected Independently Relevant features, emphasising the fact that the relevance of these features is determined by whether their absence alone influences the prediction. It is important to note that simply because a selected feature alone did not make a significant change in prediction does not mean that this feature is not relevant, as it may be essential in combination with other features. Hence, $\mathbf{SIR}^m_{\mathbf{x}}$ is not to be regarded as the only relevant set of features. Yet, considering only this subset of relevant features was enough to obtain conclusive results, as we will see in Section 4.4.

On the other hand, in Chapter 3, we saw that the elimination of a feature can influence the prediction as part of an artefact-caused cancellation. Hence, this can be the case for features in $\mathbf{SIR}^m_{\mathbf{x}}$. Moreover, given the issue (I2), one may end up with cases for which the explainers are (arguably wrongly) penalised for ranking features from genuine cancellations as more important than features from artefact-caused cancellations, which happens in case the latter are not selected. This is an important limitation of the proposed verification framework. Hopefully, future work will provide a solution to this corner case. In Section 4.6, I provide a way to decrease the probability of this corner case happening.

For any instance $\mathbf{x}$, the check in Equation 4.1 takes linear time in the number of selected features $|\mathbf{S}^m_{\mathbf{x}}|$ (which is usually significantly less than the total number of

features $|\mathbf{x}|$). However, this check needs to be done only once in order to obtain an off-the-shelf sanity test that can be applied to any number of explainers.

Sufficiency of the selected features. As we saw in Chapter 3, the sufficiency of the selected features can be checked relatively easily by probing whether $m(\mathbf{S}^m_{\mathbf{x}}) = m(\mathbf{x})$. To avoid any noise that can appear in practice from checking this equality, for this verification framework, I check the stricter condition of

$$\mathbf{S}^m_{\mathbf{S}^m_{\mathbf{x}}} = \mathbf{S}^m_{\mathbf{x}}, \tag{4.2}$$

which implies $m(\mathbf{S}^m_{\mathbf{x}}) = m(\mathbf{x})$.

Proof: If $\mathbf{S}^m_{\mathbf{S}^m_{\mathbf{x}}} = \mathbf{S}^m_{\mathbf{x}}$ then, by applying the pred() module we have that $\text{pred}(\mathbf{S}^m_{\mathbf{S}^m_{\mathbf{x}}}) = \text{pred}(\mathbf{S}^m_{\mathbf{x}})$.

Since by construction $m(\mathbf{x}) = \text{pred}(\mathbf{S}^m_{\mathbf{x}})$, the above equality translates to $m(\mathbf{S}^m_{\mathbf{x}}) = m(\mathbf{x})$. □

More instances from $\mathcal{D}$ may be preserved if one instead checks directly for $m(\mathbf{S}^m_{\mathbf{x}}) = m(\mathbf{x})$ with an appropriate tolerance threshold for the equality. It is an open question whether (and why) a selector-predictor model would change its selection of features yet provide the same final prediction.

Final dataset $\mathcal{G}^m$. To obtain the desired dataset $\mathcal{G}^m$, we first prune $\mathcal{D}$ to a dataset where for each instance, the selected features $\mathbf{S}^m_{\mathbf{x}}$ form a sufficient subset via Equation 4.2. From this pruned dataset, we further remove the instances for which $\mathbf{SIR}^m_{\mathbf{x}} = \emptyset$. Hence, we obtain the dataset $\mathcal{G}^m$ such that for each instance $\mathbf{x} \in \mathcal{G}^m$ we have (1) the selected features are a sufficient subset, and (2) there is a subset of these features $\mathbf{SIR}^m_{\mathbf{x}} \subseteq \mathbf{S}^m_{\mathbf{x}}$ that are independently relevant features.

SelPredVerif tests if an explainer ranks irrelevant features higher than relevant ones, under two important assumptions: (i) that any feature that is not part of a sufficient subset is considered irrelevant (with the warning that groups of feature that cancel each other out may not be part of the sufficient subset), and (ii) that any

selected feature that is independently relevant is considered relevant (with the warning that features from artefact-caused cancellations may be considered relevant).

Metrics. Let $f_1^{e,m}(\mathbf{x}), f_2^{e,m}(\mathbf{x}), \ldots, f_{|\mathbf{x}|}^{e,m}(\mathbf{x})$ be the features in $\mathbf{x}$ sorted in decreasing order of importance as returned by the explainer e. For example, feature $f_1^{e,m}(\mathbf{x})$ is the feature considered by the explainer e as the most relevant for the prediction of m on $\mathbf{x}$. To quantify the extent to which an explainer ranks irrelevant features higher than relevant ones, one can use the following error metrics:[2]

(M1) **Percentage of instances for which the most relevant feature provided by the explainer is among the irrelevant features:**

$$\mathrm{PCT}_{\mathrm{first}} = \frac{1}{|\mathcal{G}^m|} \sum_{\mathbf{x} \in \mathcal{G}^m} \mathbb{1}_{\{f_1^{e,m}(\mathbf{x}) \in \mathbf{N}_{\mathbf{x}}^m\}}\,, \tag{4.3}$$

where $\mathbb{1}$ is the indicator function.

(M2) **Percentage of instances for which at least one irrelevant feature is ranked higher than an independently relevant feature:**

$$\mathrm{PCT}_{\mathrm{misrnk}} = \frac{1}{|\mathcal{G}^m|} \sum_{\mathbf{x} \in \mathcal{G}^m} \mathbb{1}_{\{\exists i<j | f_i^{e,m}(\mathbf{x}) \in \mathbf{N}_{\mathbf{x}}^m \text{ and } f_j^{e,m}(\mathbf{x}) \in \mathbf{SIR}_{\mathbf{x}}^m\}}\,. \tag{4.4}$$

(M3) **Average number of irrelevant features ranked higher than any independently relevant feature:**

$$\mathrm{AVG}_{\mathrm{misrnk}} = \frac{1}{|\mathcal{G}^m|} \sum_{\mathbf{x} \in \mathcal{G}^m} \sum_{i<r'} \mathbb{1}_{\{f_i^{e,m}(\mathbf{x}) \in \mathbf{N}_{\mathbf{x}}^m\}}\,, \tag{4.5}$$

where $r' = \mathrm{argmax}_j\{f_j^{e,m}(\mathbf{x}) \in \mathbf{SIR}_{\mathbf{x}}^m\}$ is the lowest rank of an independently relevant feature.

Metric $\mathrm{PCT}_{\mathrm{first}}$ shows the percentage of instances for which the explainer tells us that the most relevant feature is one that was irrelevant for the prediction. Metric $\mathrm{PCT}_{\mathrm{misrnk}}$ shows the percentage of instances for which there is at least one error in

[2]The limitations of these errors will be discussed in Section 4.5.

Table 4.1: Error rates of the explainers on the sanity tests generated by SelPredVerif. For an explainer to pass this sanity tests, its errors should be all 0. Best results (lowest errors) are in bold, although the tests are not comprehensive, so one cannot fully compare the explainers based on their performance on these tests. $\text{AVG}_{\text{misrnk}}$ reports an average with the standard deviation in parentheses.

	APPEARANCE			AROMA			PALATE		
Model	$\text{PCT}_{\text{first}}$	$\text{PCT}_{\text{misrnk}}$	$\text{AVG}_{\text{misrnk}}$	$\text{PCT}_{\text{first}}$	$\text{PCT}_{\text{misrnk}}$	$\text{AVG}_{\text{misrnk}}$	$\text{PCT}_{\text{first}}$	$\text{PCT}_{\text{misrnk}}$	$\text{AVG}_{\text{misrnk}}$
LIME	**4.24**	24.39	7.02 (24.12)	14.79	32.08	12.74 (33.54)	2.92	13.93	**3.48** (17.38)
KernalSHAP	4.74	**16.81**	**1.16** (7.75)	**4.24**	**13.53**	**0.83** (7.10)	**2.65**	**9.20**	9.25 (9.70)
L2X	6.58	28.85	3.54 (12.66)	12.95	31.61	4.41 (16.25)	12.77	29.83	3.70 (13.05)

the ranking. Finally, metric $\text{AVG}_{\text{misrnk}}$ gives the average number of irrelevant features per instance that are ranked higher than any independently relevant feature.

SelPredVerif can be summarised in the following steps, which one can use to create new sanity tests based on other tasks, datasets, or architectures for the selector and predictor modules.

1. Choose a task and a dataset $\mathcal{D}$.
2. Choose a selector-predictor model m.
3. Train m on $\mathcal{D}$.
4. Using the trained and fixed model m, prune the dataset $\mathcal{D}$ to $\mathcal{G}^m \subseteq \mathcal{D}$ by first eliminating the instances $\mathbf{x}$ that do not satisfy Equation 4.2, and then by further eliminating the instances for which $\mathbf{SIR}^m_{\mathbf{x}} = \emptyset$ according to Equation 4.1.
5. Using the metrics defined above, test an explainer on m applied to instances in $\mathcal{G}^m$.

4.4 Experiments

To prove the effectiveness of SelPredVerif, I instantiate it on the RCNN model introduced by Lei et al. (2016) and apply it to the task of sentiment analysis in natural language.

The RCNN. The RCNN is a selector-predictor type of model, for which:

1. the selector (called generator in their paper) takes as input a piece of text $\mathbf{x}$ and, for each feature (token in our case) $x_i \in \mathbf{x}$, outputs $p_i \in [0, 1]$. Then, p_i is used as the parameter of a Bernoulli distribution modelling the probability that the token x_i is selected, and

2. the predictor (called encoder in their paper) takes as input $\mathbf{S}_{\mathbf{x}}^m = \{x_i \sim \text{Bernoulli}(p_i)\}$ (at test time $\mathbf{S}_{\mathbf{x}}^m = \{x_i | p_i \geq 0.5\}$) and returns the prediction as a real number in $[0, 1]$.

Both the selector and the predictor are recurrent convolutional neural networks (Lei et al., 2015) (hence, the name of RCNN). The selector has the additional property of being bidirectional, so that the decision to select a token is based on the entire context around the token. There is no direct supervision on the subset selection, and the selector and predictor are trained jointly, with supervision only on the final prediction. To train the RCNN, two regularizers are deployed: one to encourage the selection of fewer tokens, and a second one to encourage the selection of a subphrase, rather than disconnected tokens. The latter is employed because in the dataset on which this model is applied, the relevant tokens usually form a subphrase. At training time, to circumvent the non-differentiability introduced by the intermediate sampling, the gradients for the selector are estimated using the REINFORCE procedure (Williams, 1992).

Dataset. In this experiments, I use the BeerAdvocate corpus,[3] on which the RCNN was initially evaluated (Lei et al., 2016). BeerAdvocate consists of a total of ∼100K human-generated multi-aspect beer reviews, where the three considered aspects are: appearance, aroma, and palate. Even though it appeared to be only one dataset with each review containing information about all three aspects, in the provided link to the corpus there are separate datasets for each aspect, which appeared to be slightly different (for example, the training set for the appearance aspect has 10K

[3] `http://people.csail.mit.edu/taolei/beer/`

Table 4.2: Statistics of the datasets $\mathcal{G}^a$ for each aspect. $|\mathcal{G}^a|$ is the number of instances after pruning dataset $\mathcal{D}$, $|\mathbf{x}|$ is the average length of the reviews in $\mathcal{G}^a$. $|\mathbf{S}^a_{\mathbf{x}}|$, $|\mathbf{SIR}^a_{\mathbf{x}}|$, and $|\mathbf{N}^a_{\mathbf{x}}|$ are the number of selected tokens, independently relevant selected tokens, and non-selected tokens, respectively. In parentheses are the standard deviations. The column $\%(\mathbf{S}^a_{\mathbf{S}^a_{\mathbf{x}}} \neq \mathbf{S}^a_{\mathbf{x}})$ provides the percentage of instances eliminated due to the possibility that the model might not have selected a sufficient subset of tokens. Finally, $\%(\mathbf{SIR}^a_{\mathbf{x}} = \emptyset)$ shows the percentage of instances (out of the remaining ones after the first pruning) further eliminated due to the absence of selected tokens that alone change the prediction by at least 0.1.

Aspect (a)	$\lvert\mathcal{G}^a\rvert$	$\lvert\mathbf{x}\rvert$	$\lvert\mathbf{S}^a_{\mathbf{x}}\rvert$	$\lvert\mathbf{SIR}^a_{\mathbf{x}}\rvert$	$\lvert\mathbf{N}^a_{\mathbf{x}}\rvert$	$\%(\mathbf{S}^a_{\mathbf{S}^a_{\mathbf{x}}} \neq \mathbf{S}^a_{\mathbf{x}})$	$\%(\mathbf{SIR}^a_{\mathbf{x}} = \emptyset)$
APPEARANCE	20508	145 (79)	16.9 (8.4)	1.33 (0.70)	121 (56)	15.9	73.2
AROMA	7621	139 (74)	11.15 (6.48)	1.16 (0.49)	123 (57)	72.0	58.0
PALATE	16494	153 (76)	9.14 (5.38)	1.21 (0.55)	137 (59)	39.2	66.5

more instances than the training set for the other two aspects). The reviews are accompanied by fractional ratings between 0 and 5 for each aspect independently, which I rescale between 0 and 1, similarly to Lei et al. (2016). Also following the procedure in Lei et al. (2016), I train three separate RCNN models, one for each aspect independently, with the code from the original paper. I did not do any hyperparameter search but used the default settings in their code.[4]

For each aspect a and the trained model RCNN_a, I obtain a pruned version $\mathcal{G}^a$ according to step 4.[5] To obtain the independently relevant tokens, I choose a threshold of $\tau = 0.1$ in Equation 4.1. Since the scores are in $[0, 1]$ and the ground-truth ratings correspond to $\{0, 0.1, 0.2, \ldots, 1\}$, a change in prediction of 0.1 is significant for this dataset.

Table 4.2 shows the statistics of the $\mathcal{G}^a$ datasets. We see that we detect only one or two independently relevant tokens per instance, showing that the threshold of 0.1 is likely very strict. However, it is better to be more conservative in order to ensure that the sanity tests do not wrongly penalise explainers. We also see that the percentages of instances eliminated in order to ensure the sufficiency condition (Equation 4.2)

[4] `https://github.com/taolei87/rcnn`

[5] Note that each $\mathcal{G}^a$ depends on the trained RCNN_a model. For the same RCNN architecture trained by starting from another initialisation, or with different hyperparameters, one may obtain a different $\mathcal{G}^a$.

can be quite large, up to 72% for the aroma aspect. Therefore, as future work, it would be interesting to check whether the model indeed provides significantly different predictions for such a large proportion of instances, or whether it just makes slightly different selections that still lead to the same final prediction. Nonetheless, for the scope of this work, we are left with a large number of instances for each sanity check.

Verifying explainers. I test three popular explainers: LIME (Ribeiro et al., 2016), KernalSHAP (Lundberg and Lee, 2017), and L2X (Chen et al., 2018a). I use the code of each explainer as provided in the original repositories,[6] with their default settings for text explanations, with the exceptions that: (1) for L2X, I set the dimension of the word embeddings to 200 (the same as in the RCNN) and I increase the maximum number of training epochs from 5 to 30, and (2) for LIME and KernalSHAP, I increase the number of samples per instance to 10K, since the length of the input text was relatively large (∼240 tokens per instance). Due to time constraints, I run each explainer only once, yet the number of instances on which the explainers are tested is rather large (∼44.5K in total over the three aspects).

As mentioned in Chapter 3, LIME and KernalSHAP adhere to the neighbourhood type of ground-truth explanation, hence, this verification is not directly targeting these explainers. However, we see in Table 4.1 that, in practice, LIME and KernalSHAP outperformed L2X on the majority of the metrics, even though L2X is a minimal sufficient subsets explainer. There are a couple of hypotheses that could explain why this is the case. First, it might be the case that the RCNN models learn to select most of the redundant features, or that there are no redundant features in the dataset. Hence, it is possible that the sanity tests are equally valid for both the pointwise and the neighbourhood type of ground-truth explanations. In Section 4.6, I provide a further check that can be done to examine this. Second, a major limitation of L2X is the requirement to know in advance the number of sufficient features per instance. Indeed, L2X learns a distribution over the set of features by maximising the mutual

[6] `https://github.com/marcotcr/lime`, `https://github.com/slundberg/shap`, `https://github.com/Jianbo-Lab/L2X/tree/master/imdb-token`.

information between the prediction of the target model on subsets of K features and the prediction of the target model on the full instance, where K is assumed to be known and the same for all instances. In practice, one usually does not know how many features per instance a model relies on. To test L2X in real-world circumstances, I set K to the average number of tokens highlighted by human annotators on the subset manually annotated in McAuley et al. (2012). I obtained an average K of 23, 18, and 13 for the three aspects, respectively.

In Table 4.1, we see that, on metric $\text{PCT}_{\text{first}}$, all explainers are prone to stating that the most relevant token is an irrelevant token, as much as 14.79% of the time for LIME and 12.95% of the time for L2X in the aroma aspect. The results on metric $\text{PCT}_{\text{misrnk}}$ show that all explainers tend to rank at least one irrelevant token higher than an independently relevant one, i.e., there is at least one error in the predicted ranking. Finally, the results on metric $\text{AVG}_{\text{misrnk}}$ show that, on average, KernalSHAP only places one irrelevant token ahead of any independently relevant token for the first two aspects but as much as 9 tokens for the third aspect, while L2X places around 3-4 irrelevant tokens ahead of an independently relevant token for all three aspects.

Qualitative analysis. In Figure 4.1, I present an example from the palate sanity test, i.e., the sanity test created with the RCNN trained for the palate aspect. Examples from the aroma and appearance sanity tests are in Appendix A.

Notice that all explainers are prone to rank tokens that were irrelevant for the prediction of the model higher than tokens on which the model actually relied on for its prediction. For example, "gorgeous", the only token used by the model on the given instance, was given almost zero weight by LIME, and did not even make it into the top five tokens given by L2X. Instead, L2X gives "mouthfeel", "lacing", "and", and even "," as most important tokens. Note that such an explanation would likely give humans a less good impression and diminish their trust in the model, even though the model actually relied on the informative token "gorgeous".

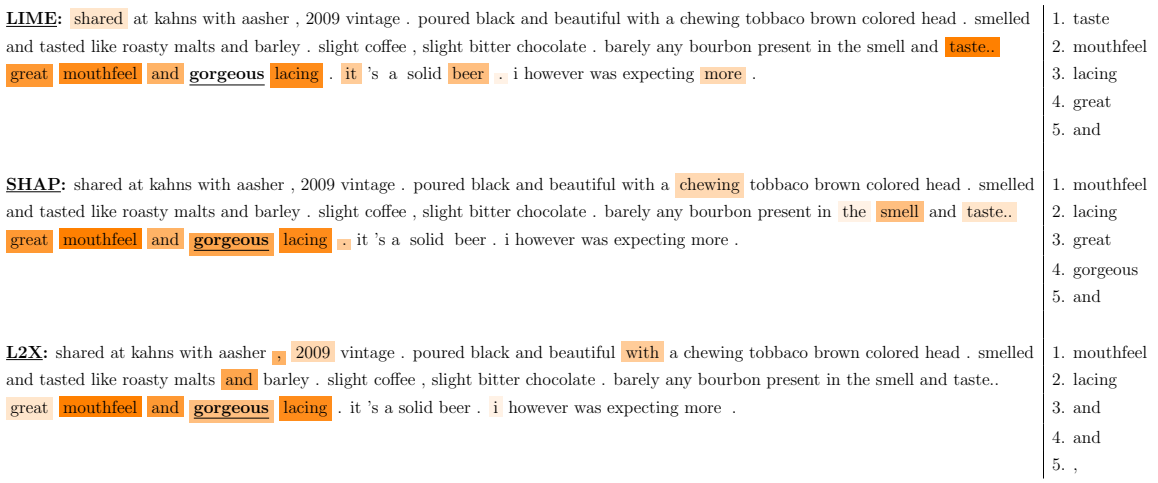

Figure 4.1: Explainers rankings on an instance from the palate sanity test. The top 5 features ranked by each explainer are on the right-hand side. Additionally, the heatmap corresponds to the ranking provided by each explainer, where the intensity of the colour decreases linearly with the rank of the token.[7] For visibility reasons, only the first $K = 10$ ranked tokens are shown. The tokens selected by the model, $\mathbf{S}^a_{\mathbf{x}}$, are in bold (in this case only one), and the detected independently relevant tokens, $\mathbf{SIR}^a_{\mathbf{x}}$, are additionally underlined (also one in this case). The non-bold tokens should not be ranked higher than any bold and underlined token. KernalSHAP is simply referred as SHAP.

4.5 Specifications

As mentioned in Chapter 3, it is essential for works introducing explainers or verification frameworks for explainers to state the intended behaviour, strengths, and limitations of their methods. Hence, in this section, I provide the specifications for SelPredVerif.

4.5.1 Intended Behaviour

Type of ground-truth explanation. SelPredVerif is tailored for the pointwise type of ground-truth explanations. Hence, it *might* penalise an explainer for ranking a feature that is redundant in the presence of other features higher than a strictly necessary feature. For example, SelPredVerif might penalise an explainer that ranks "nice" higher than "very" for the model m on the instance $\mathbf{x}^{(2)}$ in Figure 3.1. However, if the selector-predictor model also selects redundant features (in our example, if it selects "nice" in addition to "very" and "good"), then the framework does not

penalise an explainer regardless of the order in which it ranks these three features. In Section 4.6, I provide a way of checking whether for a selector-predictor model and an instance, there exist redundant features that are not selected. This check will allow the extension of the framework to verifying explainers in the case when neighbourhood explanations are desired. Meanwhile, the current framework and results should be used only when pointwise ground-truth explanations are desired. Consequently, what is called *error* in this chapter refers to an error under the pointwise perspective, which, as we saw, may not be an error under the neighbourhood perspective.

A sanity check, not a complete evaluation. The sanity tests provided by this framework cannot be used as a complete evaluation for concluding the faithfulness of explainers in full generality. This is because we do not know the full behaviour of the model and thus cannot provide a ranking of all features. Furthermore, similarly to the BAM framework (Yang and Kim, 2019), the SelPredVerif framework does not guarantee to identify *all* false positives. This is because certain features from $\mathbf{S}_{\mathbf{x}}^{m}$ might have been relevant for the prediction even if they do not pass the independently relevant check from Equation 4.1. Thus, SelPredVerif generates necessary but not sufficient sanity tests.

Ranking. The current metrics assume that the explanatory methods give a ranking of the features in terms of their relevance to the prediction of the target model. However, the metrics can be adapted to methods that provide only subsets of important features.

4.5.2 Strengths

Non-trivial neural model, real-world dataset, and partially known decision-making process. As mentioned before, an important challenge in verifying explanatory methods is how to automatically sanity check these methods on neural models similar to the ones deployed in the real world and without speculating the decision-making processes of these networks. To my knowledge, the verification framework introduced in this chapter is the first to take a step in this direction. While the

current version of the framework still contains certain unknowns with respect to the decision-making process of the testbed model, the directions of further improvements provided in Section 4.6 aim to alleviate most of these unknowns.

Artefact-caused minimal sufficient subsets. Another important strength of this verification framework is that it provides (potentially a superset of) the exact minimal sufficient subset on which the target model relied for its prediction on an instance. This allows us to penalise explainers for providing artefact-based minimal sufficient subsets, which, as we saw in Chapter 3, would not be faithful explanations and can consequently distort the perception and trust of users in the model. This would not be possible to be checked simply by probing whether a subset provided by an explainer is sufficient.

4.5.3 Limitations

The current version of the framework has two major limitations that I describe below. While these cases might not happen often in practice, it is important to acknowledge them. In Section 4.6, I provide checks that can alleviate these limitations.

Genuine cancellations. If the selector-predictor model does not select groups of features that form a genuine cancellation, then the framework would penalise an explainer for placing these features higher than an independently relevant selected feature. If additionally, this independently relevant selected feature is part of an artefact-caused cancellation, then this would arguably be a wrong penalisation. However, for this to happen, two conditions need to be met: the model would have to (1) select features from artefact-caused cancellations, and (2) not select features from genuine cancellations. Hence, it is reasonable to believe that this would not happen often in practice.

Genuine minimal sufficient subsets. If for an instance there are several genuine minimal sufficient subsets and if the selector-predictor model does not select all of

them, then our framework would penalise an explainer for ranking features from a non-selected yet equally genuine minimal sufficient subset higher than features from selected genuine minimal sufficient subsets. This would arguably be an incorrect penalisation for the explainer. It is unclear how often this scenario happens in practice.

4.6 Directions and Guidelines of Improvement

In this section, I provide three directions for extending and improving the SelPredVerif framework, with concrete guidelines.

Check for non-selected redundant features. To adapt SelPredVerif to the cases where the neighbourhood type of explanation is desired, one can further eliminate certain instances that contain non-selected redundant features based on the following observation: if the selector-predictor model applied to the instance formed only by the non-selected features gives a prediction that is significantly different from the prediction of the model on the baseline input, then there is a redundant feature among the originally non-selected features. Formally, if for a selector-predictor m and an instance $\mathbf{x}$, we have that $m(\mathbf{N}_{\mathbf{x}}^{m}) \neq m(\mathbf{b})$, where $\mathbf{b}$ is the baseline input for the task at hand, then there must be a redundant feature among $\mathbf{N}_{\mathbf{x}}^{m}$, and this feature would be among the newly selected features $\text{sel}(\mathbf{N}_{\mathbf{x}}^{m})$.

However, $m(\mathbf{N}_{\mathbf{x}}^{m}) = m(\mathbf{b})$ does not guarantee the absence of redundant features among $\mathbf{N}_{\mathbf{x}}^{m}$. This is because there might be redundant features that require the presence of certain originally selected features in order to *be activated*. For example, assume that "nice" is redundant in the presence of "very good" and that the model also requires the presence of the feature "tastes" in order to activate these sentiment-providing features. Then, on an instance such as "The beer tastes nice, very good." the model might select "tastes", "very", and "good", and the check above would not identify "nice" as a redundant feature because "tastes" is missing from $\mathbf{N}_{\mathbf{x}}^{m}$.

Future work may focus on how to guarantee the absence of redundant features instead of simply decreasing the probability of their existence.

Check for other genuine minimal sufficient subsets. Similarly to the check above, if $m(\mathbf{N}_{\mathbf{x}}^m) = m(\mathbf{x})$, then there is another genuine minimal sufficient subset among the non-selected features. However, this check does not account for genuine equivalent subsets that intersect with the originally selected features. As before, this check only reduces the probability of wrongly penalising explainers. Future work may find ways to provide a guarantee.

Check for non-selected features that cancel each other out. To decrease the probability of having groups of features that cancel each other out among the non-selected features, one can check if the elimination of each non-selected feature alone changes the prediction of the model. More precisely, if there exists a non-selected feature $x_i \in \mathbf{N}_{\mathbf{x}}^m$ such that $m(\mathbf{N}_{\mathbf{x}}^m \setminus \{x_i\}) \neq m(\mathbf{x})$, then there exists a group of features that cancel each other out that the model did not select. However, this check is again not enough to guarantee the absence of such a group of features among the non-selected features. This is because there can exist groups of features that cancel each other out but for which we need to eliminate more than one feature at a time to observe a change in prediction. For example, if the cancellation follows a rule such as ["amazing" or "awesome" cancel out with either "horrible" or "awful"], then for an instance that contains all the four features, eliminating either of them in isolation does not change the prediction.

Further improvements. Doing all of the above eliminations might lead to a small number of instances in the final dataset $\mathcal{G}^m$. In this case, one may salvage more instances in the following ways. First, one can check the sufficiency condition using the equality in prediction $m(\mathbf{S}_{\mathbf{x}}^m) \neq m(\mathbf{x})$ rather than the stronger condition in Equation 4.2. While, in theory, there is no guarantee that any instance at all would be salvaged in this way, in practice, I noticed that the difference between the two predictions was often very small even when the newly selected features were different than the originally selected ones. Second, instead of eliminating instances for which these conditions do not hold, one can adjust the sets of relevant and irrelevant features

according to the newly selected and non-selected features. Such investigation is left as future work.

4.7 Conclusions and Open Questions

In this chapter, I introduced a framework capable of generating automatic sanity tests to verify whether post-hoc explainers are ranking features that are irrelevant for the prediction of a model on an instance higher than features that are relevant for the prediction of the model on that instance. I instantiated the framework on a task of sentiment analysis and produced three sanity tests, on which I tested three explainers, namely, LIME, KernalSHAP, and L2X.

This framework unveiled certain problems of current explanatory methods, such as the fact that they may point to an irrelevant feature as the most important feature for a prediction. Identifying such problems is a first step towards improving these methods in the future.

is generic and can be instantiated on other tasks and domains, while the three sanity tests can be used off-the-shelf to test other existing and future explainers.

Future work includes performing the improvements described in Section 4.6, instantiating SelPredVerif on other tasks and modalities, as well as testing more explanatory methods.

Chapter 5

Neural Networks that Generate Natural Language Explanations

In this chapter, which is based on (Camburu et al., 2018), I explore self-explanatory neural models. More precisely, I investigate whether neural models improve their behaviour if they are additionally given natural language explanations for the ground-truth labels at training time, and whether these models can generate such explanations for their predictions at test time.

5.1 Motivation

As mentioned in Chapter 1, models trained simply to obtain a high accuracy on held-out sets can often learn to rely on shallow input statistics, resulting in brittle models. For example, Ribeiro et al. (2016) present a document classifier that distinguishes between *Christianity* and *Atheism* with a test accuracy of 94%. However, on close inspection, the model spuriously separates classes based on words contained in the headers, such as "Posting", "Host", and "Re". Spurious correlations in both training and test sets allow for such undesired models to obtain high accuracies. Much more complex hidden correlations may be present in any arbitrarily large and human-annotated dataset (Gururangan et al., 2018; Bhathena et al., 2020; Chen et al., 2016a; Levy and Dagan, 2016; Cai et al., 2017). Such correlations may be difficult to spot, and even when one identifies them, it is an open question how to mitigate them (Belinkov et al., 2019).

In this chapter, I investigate a direction that has the potential to both steer

neural models away from relying on spurious correlations and provide explanations for the predictions of these models. This direction is that of enhancing neural models with the capability to learn from natural language explanations during training time and to generate such explanations at test time. For humans, it has been shown that explanations play a key role in structuring conceptual representations for categorisation and generalisation (Williams and Lombrozo, 2010; Lombrozo, 2009). Humans also benefit tremendously from reading explanations before acting in an environment for the first time (Tsividis et al., 2017). Thus, explanations may also be used to set a model in a better initial position to further learn the correct functionality. Meanwhile, at test time, generating correct argumentation in addition to obtaining a high accuracy has the potential to endow a model with a higher level of transparency and trust.

Incorporating external knowledge into a neural model was shown to result in more robust models (Glockner et al., 2018b). Free-form natural language explanations are a form of external knowledge that has the following advantages over formal language. First, it is easy for humans to provide free-form language, eliminating the additional effort of learning to produce formal language, thus making it simpler to collect such datasets. Secondly, natural language explanations might potentially be mined from existing large-scale free-form text. Finally, natural language is readily comprehensible to an end-user who needs to assert the reliability of a model.

Despite the potential for natural language explanations to improve both learning and transparency, there is a scarcity of such datasets in the community, as discussed in Section 2.2.4. To address this deficiency, I collected a large corpus of ∼570K human-annotated explanations for the SNLI dataset (Bowman et al., 2015). I chose SNLI because it constitutes an influential corpus for natural language understanding that requires deep assimilation of fine-grained nuances of commonsense knowledge. I call this explanation-augmented dataset e-SNLI, which I release publicly[1] to advance research in the direction of training with and generation of free-form natural language explanations.

[1]The dataset can be found at `https://github.com/OanaMariaCamburu/e-SNLI`.

Secondly, I show that it is much more difficult for a neural model to produce correct natural language explanations based on spurious correlations than it is for it to produce correct labels based on such correlations.

Thirdly, I develop models that predict a label and generate an explanation for their prediction, and I investigate the correctness of the generated explanations.

Finally, I investigate whether training a neural model with natural language explanations can result in better universal sentence representations produced by this model and in better performance on out-of-domain datasets.

Remark. In this chapter, I use the concept of correct explanation to refer to the correct argumentation for the ground-truth label on an instance. This should not be confused with the concept of faithful explanation, which refers to the accuracy with which an explanation describes the decision-making process of a model, as described in Section 2.4. The capability of a neural model to generate correct explanations is an important aspect of the development of such models. For example, correct argumentation may sometimes be needed in practice, alongside the correct final answer. Hence, in this chapter, I inspect the correctness of the explanations generated by the introduced neural models. In the next chapter, I will take a step towards verifying the faithfulness of these explanations.

5.2 Background

The task of recognising textual entailment is a critical natural language understanding task. Given a pair of sentences, called the premise and hypothesis, the task consists of classifying their relation as either (a) *entailment*, if the premise entails the hypothesis, (b) *contradiction*, if the hypothesis contradicts the premise, or (c) *neutral*, if neither entailment nor contradiction hold. For example, the premise "*Two doctors perform surgery on patient.*" and the hypothesis "*Two doctors are performing surgery on a man.*" constitute a neutral pair (because one cannot infer from the premise if the patient is a man). The SNLI dataset (Bowman et al., 2015), containing ∼570K

instances of human-generated triples (premise, hypothesis, label), has driven the development of a large number of neural models (Rocktäschel et al., 2015; Nie and Bansal, 2017; Parikh et al., 2016; Chen et al., 2016b; Liu et al., 2016; Chen et al., 2018b; Conneau et al., 2017).

Moreover, the power of SNLI transcends the task of natural language inference. Conneau et al. (2017) showed that training a model to produce universal sentence representations on SNLI can be both more efficient and more accurate than certain training approaches on orders of magnitude larger but unsupervised datasets (Kiros et al., 2015; Hill et al., 2016). I take this approach one step further and, in Section 5.4.4, I investigate whether the additional layer of natural language explanations can bring further improvement.

Recently, an increasing amount of analysis has been carried out on the spurious correlations in the SNLI dataset and on how different models rely on these correlations (Bhathena et al., 2020; Gururangan et al., 2018; Glockner et al., 2018b). In particular, Gururangan et al. (2018) show that specific words in the hypothesis tend to be strong indicators of the label, e.g., "friends" and "old" appear very often in neutral hypotheses, "animal" and "outdoors" appear most of the time in entailment hypotheses, while "nobody" and "sleeping" appear mostly in contradiction hypothesis. They also show that a premise-agnostic model, i.e., a model that only takes as input the hypothesis and predicts the label, obtains 67% test accuracy. In Section 5.4.1, I show that it is much more difficult to rely on spurious correlations to generate explanations than to generate labels.

5.3 The e-SNLI Dataset

In this section, I present the methodology that I used to collect e-SNLI on Amazon Mechanical Turk. The main question that this dataset had to answer was: *Why is a pair of sentences in a relation of entailment, neutrality, or contradiction?* I encouraged the annotators to focus on the salient elements that induce the given relation, and not on the parts that are repeated identically in the premise and hypothesis. I also asked

PREMISE: An adult dressed in black holds a stick. HYPOTHESIS: An adult is walking away, empty-handed. LABEL: contradiction EXPLANATION: Holds a stick implies using hands so it is not empty-handed.
PREMISE: A child in a yellow plastic safety swing is laughing as a dark-haired woman in pink and coral pants stands behind her. HYPOTHESIS: A young mother is playing with her daughter in a swing. LABEL: neutral EXPLANATION: Child does not imply daughter and woman does not imply mother.
PREMISE: A man in an orange vest leans over a pickup truck. HYPOTHESIS: A man is touching a truck. LABEL: entailment EXPLANATION: Man leans over a pickup truck implies that he is touching it.

Figure 5.1: Examples from the e-SNLI dataset. Annotators were given the premise, the hypothesis, and the label. They highlighted the words that they considered essential for the label and provided the explanations.

them to explain all the parts of the hypothesis that do not appear in the premise. Additionally, I asked the annotators to provide self-contained explanations, as opposed to sentences that would make sense only after reading the premise and hypothesis. For example, an explanation of the form "Anyone can knit, not just women." would be preferred to the explanation "It cannot be inferred they are women."

The main challenge of collecting this dataset is that, in crowd-sourcing, it is difficult to control the quality of free-form annotations. As a solution, I preemptively blocked the submission of obviously incorrect explanations by doing in-browser checks. For example, the annotators were not allowed to submit unless each explanation contained at least three tokens and it was not a copy of the premise or of the hypothesis. A second way of guiding the annotators to provide correct explanations was by asking them to proceed in two steps. First, they were required to highlight the words from the premise or hypothesis that they considered essential for the given relation. Secondly, annotators had to formulate each explanation using the words that they highlighted. In-browser checks were performed to ensure that a minimal number of words were highlighted and that at least half of the highlighted words were used in the explanation, so that the explanation is on-topic. To account for the particularities of each relation,

the minimal number of words required to be highlighted depended on the relation. For entailment pairs, annotators were required to highlight at least one word from the premise. The annotators were also encouraged (but not required) to highlight words from the hypothesis. For contradiction pairs, they were required to highlight at least one word in both the premise and in the hypothesis. For neutral pairs, they were required to highlight at least one word in the hypothesis, and they were not allowed to highlight words from the premise. This specific constraint was introduced to prevent workers from confusing the premise with the hypothesis. Neutral pairs were often confusing, since annotators were easily prone to focus on the details from the premise that could not be found in the hypothesis instead of the other way around. This was not the case for the contradiction pairs, since a contradiction relation between entities is usually symmetrical, nor for the entailment pairs, for which the relation between entities is intuitive (e.g., a human would naturally say that "A dog is an animal." instead of "An animal is a dog."). Hence, requiring annotators to highlight words with restrictions specific for each relation was especially useful to place the annotators into the correct mindset, and additionally provided a way to filter incorrect explanations. Moreover, the highlighted words may also provide a valuable future resource, for example, for providing supervision for attention models or for evaluating them (Rocktäschel et al., 2015; Parikh et al., 2016). Finally, an in-browser check also verified that the annotators used other words than the ones highlighted, since a correct explanation would need to articulate a link between these words.

I collected one explanation for each instance in the training set and three explanations for each instance in the validation and test sets. Table 5.1 shows examples of collected explanations. There were 6325 workers with an average of 86 ± 403 explanations per worker.

Note that an explanation that could also be generated automatically, such as "Just because [entire premise] doesn't mean [entire hypothesis]" (for neutral pairs), "[entire premise] implies [entire hypothesis]" (for entailment pairs), or "It can either be [entire premise] or [entire hypothesis]" (for contradiction pairs), would be uninformative. Therefore, I assembled a list of possible such templates, given in Appendix B, which I

used for filtering the dataset of such uninformative explanations. More precisely, I filtered an explanation if its edit distance to one of the templates was less than 10 characters. I ran this template detection on the entire dataset and reannotated the detected explanations (11% in total).

Quality of collected explanations. In order to measure the quality of the collected explanations, I selected a random sample of 1000 examples and manually graded their correctness between 0 (incorrect) and 1 (correct). For entailment, an explanation received a potential partial score of $\frac{k}{n}$ if exactly k out of n required arguments were mentioned. For neutral and contradiction pairs, one correct argument is enough to conclude a correct explanation.

The annotation resulted in a total error rate of 9.6%, with 19.5% on entailment, 7.3% on neutral, and 9.4% on contradiction pairs. The higher error rate on the entailment pairs is firstly due to partial explanations, as annotators had an incentive to provide shorter inputs, so they often only mentioned one argument. A second reason is that many of the entailment pairs have the hypothesis as almost a subset of the premise, prompting the annotators to paraphrase the premise or hypothesis. Strictly speaking, this would not be an error given that in those cases, there was nothing salient to explain. However, it was considered an error in this annotation because one could still formulate an argumentation-like explanation rather than simply paraphrasing the input. Future work may investigate automatic ways to detect the incorrect explanations, so that one can reannotate them.

5.4 Experiments

In this section, I perform a series of experiments to investigate the capabilities of neural models to learn from the natural language explanations in e-SNLI as well as to generate such explanations at test time. First, I show that a model that relies on spurious correlations in SNLI to provide correct labels is not able to also provide correct explanations based on these correlations. Further experiments elucidate whether

models trained on e-SNLI are able to: (i) predict a label and generate an explanation for the predicted label, (ii) generate an explanation then predict the label given only the generated explanation, (iii) learn better universal sentence representations, and (iv) perform better on out-of-domain natural language inference datasets.

Throughout the experiments, the models largely follow the architecture of the model called BiLSTM-Max from Conneau et al. (2017).[2] More precisely, BiLSTM-Max separately encodes the premise and the hypothesis using two bidirectional long short-term memory recurrent units (BiLSTM) with internal sizes of 2048 (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997). Max-pooling over timesteps is used for obtaining the vector representations of each sentence (of dimension $2 \cdot 2048 = 4096$, due to bidirectionality). Let $\mathbf{u}$ be the vector representation of the premise, and $\mathbf{v}$ the vector representation of the hypothesis. The final vector of features is $\mathbf{f} = [\mathbf{u}, \mathbf{v}, |\mathbf{u}-\mathbf{v}|, \mathbf{u} \odot \mathbf{v}]$, which is passed to a multilayer perceptron (MLP) with three layers of dimension 512 each and without non-linearities[3], which predicts the label. This is a typical high-level architecture often employed on SNLI (Bowman et al., 2015).

To generate explanations, I use a one-layer LSTM module, whose internal size is a hyperparameter with values among {512, 1024, 2048, 4096}. I also experimented with gated recurrent units (GRUs) (Cho et al., 2014) for the decoder, however, the LSTMs units performed best in all experiments. Recurrent dropout (Srivastava et al., 2014) was also applied to the explanations decoder, with a rate fixed to 0.5.

To reduce the size of the output vocabulary for generating explanations, tokens that appear less than 15 times in the training set of explanations are replaced with `<UNK>`, resulting in an output vocabulary of ∼12K tokens.

The preprocessing and optimisation were kept the same as in Conneau et al. (2017). In particular, to input sentences into a model, the already trained GloVe word embeddings were used and fixed throughout the experiments (Pennington et al., 2014). The premises and hypotheses were cut at a maximum of 84 tokens, while and

[2] I build on top of their code, which is available at `https://github.com/facebookresearch/InferSent`.

[3] The default in the original BiLSTM-Max code.

the explanations at a maximum of 40 tokens (these limits were decided based on the statistics of dataset).

All models were trained with stochastic gradient descent (SGD) with the learning rate starting at 0.1 and decayed by a factor of 0.99 every epoch. I have experimented with the Adam optimizer (Kingma and Ba, 2014), however SGD performed best in all experiments. The batch size was fixed at 64.

For the majority of the experiments, I use five seeds for the random number generator and provide the average performance with the standard deviation in parentheses. If no standard deviation is reported, the results are from only one seed.

5.4.1 Premise Agnostic

Gururangan et al. (2018) show that a neural model that *only* has access to the hypotheses can predict the correct label 67% of the times, by relying on spurious correlations in SNLI. Therefore, it is interesting to evaluate to what extent a model can also rely on spurious correlations to generate correct explanations.

Model. The model Hyp2Expl is formed of one BiLSTM encoder with max-pooling, which provides a vector representation for the hypothesis, and one LSTM decoder that takes this representation as its initial state as well as additional input at every timestep, and generates an explanation.

For a fair comparison, I also separately train a model with the same architecture for the encoder of the hypothesis, but followed by a 3-layer MLP for predicting the label instead of generating an explanations. I call this model Hyp2Lbl. No hyperparameter search is performed for this model.

Model selection. The hyperparameters for Hyp2Expl are the internal size of the decoder (as mentioned above, with values of 512, 1024, 2048 and 4096). The model selection is performed via perplexity on the validation set, which strictly decreased when the decoder size was increased. However, for practical reasons, the decoder

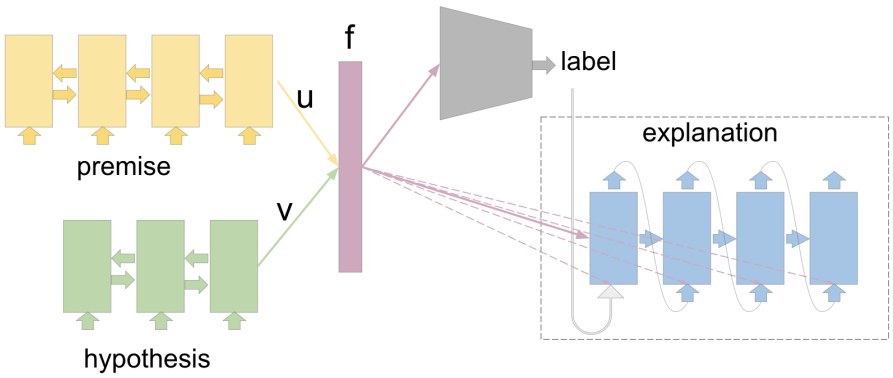

Figure 5.2: Architecture of the BiLSTM-Max-PredExpl model.

internal size was not increased beyond 4096, which is therefore the chosen value for this hyperparameter for this model.

Results. I manually inspected the explanations generated by Hyp2Expl for the first 100 test instances, and obtained a correctness score of only 6.83.[4] On the other hand, Hyp2Lbl obtained 66 correct labels for the same instances. This comforts the intuition that it is much more difficult (approx. 10 times for this architecture) to rely on spurious correlations to generate correct explanations than to rely on these correlations to predict correct labels.

5.4.2 Predict then Explain

In this experiment, I investigate the capability of a neural model to generate natural language explanations for its predictions.

Model. I enhance the BiLSTM-Max model with an explanation generator by simply connecting the feature vector $\mathbf{f}$ to a one-layer LSTM decoder, both as an initial state and concatenated to the word embedding input at every timestep. I call this model BiLSTM-Max-PredExpl. To condition the explanation on the label, the label is inputted at the first timestep of decoder (as the word "entailment", "contradiction", or "neutral"). At training time, the gold label is provided, while at

[4]Partial scoring is explained in Section 5.3.

test time, the label predicted by the model is used. This architecture is depicted in Figure 5.2.

Loss function. Negative log-likelihood is used for both the classification loss and the explanation loss. Note that the explanation loss is much larger in magnitude than the classification loss, due to the summation of negative log-likelihoods over a large number of the words in the explanations. To account for this difference during training, a weighting coefficient $\alpha \in [0, 1]$ is used, such that the overall loss is

$$L_{\text{total}} = \alpha L_{\text{label}} + (1 - \alpha) L_{\text{explanation}}. \tag{5.1}$$

Model selection. The hyperparameters in this experiment are the decoder internal size, α (for which I chose values from 0.1 to 0.9 with a step of 0.1) and the learning rate (again, with values among 512, 1024, 2048, and 4096). To investigate how well a model can generate explanations without sacrificing prediction accuracy, in this experiment, only the label accuracy is used as the criterion for model selection. As future work, one can inspect different trade-offs between label and explanation performance. The best validation accuracy, of 84.37%, was obtained for $\alpha = 0.6$ and the decoder internal size of 512.

Results. The test accuracy of BiLSTM-Max over five seeds is 84.01% (0.25). The BiLSTM-Max-PredExpl model obtains essentially the same test accuracy, of 83.96% (0.26), which shows that one can get additional explanations without sacrificing label accuracy.

For the generated explanations, the test perplexity is 10.58 (0.40) and the BLEU (bilingual evaluation understudy) score (Papineni et al., 2002) is 22.40 (0.70). Since e-SNLI has three explanations for each instance in the validation and test sets, an inter-annotator BLEU score can be obtained, for example, by computing the BLEU score of the third explanation with respect to the first two explanations. In this way, I obtained an inter-annotator BLEU score of 22.51. For consistency, I use the same two

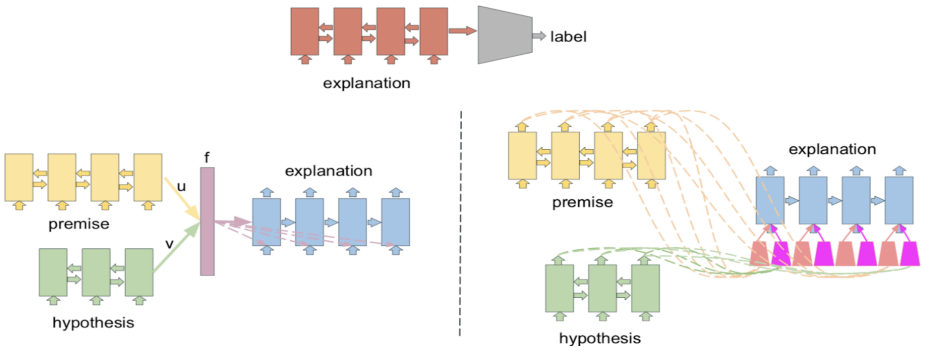

Figure 5.3: Architectures of the two versions of BiLSTM-Max-ExplPred. At the top is the ExplToLbl model that predicts a label from an explanation alone. On the left is the BiLSTM-Max-ExplPred-Seq2Seq model. On the right is the BiLSTM-Max-ExplPred-Att model.

explanations as only references when computing the BLEU score for the generated explanations.

Given the low inter-annotator score and the fact that the BLEU score of the generated explanations almost matches the inter-annotator BLEU score, we can see that this metric is not reliable for assessing the quality and correctness of the generated explanations. Therefore, I manually annotated the first 100 instances in the test set (following the same partial scoring as in Section 5.3). Since the explanation is conditioned on the predicted label, it is not expected that the model generates correct explanations when it predicts incorrect labels. Therefore, an appropriate correctness measure for the explanations is the percentage of correct explanations among the subset for which the predicted label is correct. For the first 100 instances in the test set, the percentage of correct explanations for BiLSTM-Max-PredExpl is only 34.68% (out of the 80 instances for which the model predicts correct labels). While this percentage is quite low, one should keep in mind that the selection criteria for BiLSTM-Max-PredExpl was only the label accuracy. In the next experiment, I show how selecting (and training) only for generating explanations results in a higher percentage of correct explanations when the predicted label is also correct..

5.4.3 Explain then Predict

In the previous experiment, BiLSTM-Max-PredExpl generated an explanation conditioned on its predicted label. However, this type of model might not provide explanations that are faithful to the decision-making process of the model. For instance, the explanation generator does not have access to the inner workings of the classifier (see Figure 5.2). To increase the probability of obtaining faithful explanations, one can reverse the order between the label prediction and the explanation generation. Thus, I propose a model that first generates an explanation given a pair of (premise, hypothesis), and then predicts a label given only the generated explanation. I call this type of model BiLSTM-Max-ExplPred, and I will provide two versions of it below. This is a sensible decomposition for the e-SNLI dataset, due to the following key observation: for e-SNLI, one can easily detect for which label an explanation has been provided. This might not be the case in general, as the same explanation can correctly argue for different labels, depending on the premise and hypothesis. For example, "A woman is a person." would be a correct explanation both for the entailment pair ("A woman is in the park.", "A person is in the park.") and for the contradiction pair ("A woman is in the park.", "There is no person in the park."). However, there are multiple ways of formulating an explanation. In our example, for the contradiction pair, one could also explain that "There cannot be no person in the park if a woman is in the park.", which read alone would allow one to infer that the pair was a contradiction. The latter kind of explanation is dominant in the e-SNLI dataset.

Model. For the part of the model BiLSTM-Max-ExplPred that predicts an explanation given a premise and a hypothesis, I propose two architectures. First, BiLSTM-Max-ExplPred-Seq2Seq is a simple sequence-to-sequence model. Essentially, it is the same as BiLSTM-Max-PredExpl but without the MLP classifier (and no label prepended to the explanation). Second, BiLSTM-Max-ExplPred-Att is an attention version of this BiLSTM-Max-ExplPred-Seq2Seq. Attention mech-

anisms in neural networks brought consistent improvements over the non-attention counter-parts in various areas, such as computer vision (Xu et al., 2015), speech (Chan et al., 2016), and natural language processing (Gong et al., 2018; Bahdanau et al., 2015). Similarly to BiLSTM-Max-ExplPred-Seq2Seq, the encoders are two 2048-BiLSTM units and the decoder is a one-layer LSTM. The difference is that BiLSTM-Max-ExplPred-Att uses two separate but identical attention heads to attend over the tokens from the premise and hypothesis while generating the explanation. Finally, I train a neural model, called ExplToLbl, that given only an explanation predicts a label. ExplToLbl consists of a 2048-BiLSTM with max-pulling encoder and a 3-layer MLP classifier, and obtains a test accuracy of 96.83%. This high test accuracy proves that it is reasonable to decompose this model into two separate modules. An illustration of the two versions of this model is depicted in Figure 5.3.

Model selection. The only hyperparameter in this experiment is the internal size for the decoder. The model selection criterion is the perplexity on the validation set of e-SNLI. The best perplexity for both BiLSTM-Max-ExplPred-Seq2Seq and BiLSTM-Max-ExplPred-Att was obtained for an internal size of 1024.

Results. With the described setup, the label accuracy drops from 83.96% (0.26) in BiLSTM-Max-PredExpl to 81.59% (0.45) in BiLSTM-Max-ExplPred-Seq2Seq and 81.71% (0.36) in BiLSTM-Max-ExplPred-Att. However, manual annotation of the generated explanations for the first 100 instances in the test set, resulted in significantly higher percentages of correct explanations: 49.8% for BiLSTM-Max-ExplPred-Seq2Seq and 64.27% for BiLSTM-Max-ExplPred-Att. Note that performing model selection only on the perplexity on the generated explanation, and using attention mechanisms significantly increases the quality of the explanations. This experiment shows that, despite a small decrease in label accuracy, one can increase the correctness of the explanations generated for the instances for which a model predicts the correct label.

The results of all models are summarised in Table 5.1.

Table 5.1: The performance of BiLSTM-Max-PredExpl, BiLSTM-Max-ExplPred, and of the BiLSTM-Max baseline. The averages are over five seeds and the standard deviations are in parentheses. Expl@100 is the percentage of correct explanations in the subset of instances for which the model predicted the correct label in the first 100 instances from the e-SNLI test set. The correctness of the explanations was obtained after manual annotation. In bold are the best results.

Model	Label Accuracy	Perplexity	BLEU	Expl@100
BiLSTM-Max	**84.01** (0.25)	-	-	-
BiLSTM-Max-PredExpl	83.96 (0.26)	10.58 (0.40)	22.40 (0.70)	34.68
BiLSTM-Max-ExplPred-Seq2Seq	81.59 (0.45)	8.95 (0.03)	24.14 (0.58)	49.8
BiLSTM-Max-ExplPred-Att	81.71 (0.36)	**6.1** (0.00)	**27.58** (0.47)	**64.27**

Qualitative analysis. In Table 5.2, one can see examples of explanations generated by (a) BiLSTM-Max-PredExpl, (b) BiLSTM-Max-ExplPred-Seq2Seq, and (c) BiLSTM-Max-ExplPred-Att. At the end of each explanation, in brackets is the score that I manually allocated, as explained in Section 5.3. Notice that the explanations are mainly on topic for all the three models, with minor exceptions, such as the mention of "camouflage" in (1c). Also notice that even when incorrect, the generated explanations are sometimes frustratingly close to being correct. For example, explanation (2b) is only one word (out of its 20 words) away from being correct.

It is also interesting to inspect the explanations provided when the predicted label is incorrect. For example, in (1a), we see that the network omitted the information of "facing the camera" from the premise and therefore classified the pair as neutral, which is backed up by an otherwise correct explanation in itself. We also see that model BiLSTM-Max-ExplPred-Seq2Seq correctly classifies this pair as entailment. However, it only generates one out of the three necessary arguments that lead to entailment, and it also picks arguably the easiest argument. On the other hand, the attention model label the instance (1c) as neutral and argues for the subtle fact that "standing" and "facing a camera" are not enough to conclude "posing for a picture", while humans labelled this instance as entailment. However, the explanations also mentions "camouflage", which is not necessarily the same as hoodies. Therefore, I labelled this explanations as incorrect (score of 0 in the brackets).

Table 5.2: Examples of predicted labels and generated explanations from (a) BiLSTM-Max-PredExpl, (b) BiLSTM-Max-ExplPred-Seq2Seq, and (c) BiLSTM-Max-ExplPred-Att. In brackets is the score of correctness, with partial scoring as described in Section 5.3.

(1) Premise: 3 young man in hoods standing in the middle of a quiet street facing the camera. Hypothesis: Three hood wearing people pose for a picture. Gold label: entailment		
(a) Predicted label: neutral Explanation: Just because the men are in the middle of a street doesn't mean they are posing for a picture. [0]	(b) Predicted label: entailment Explanation: three young men are people. [0.33]	(c) Predicted label: neutral Explanation: Just because three young man in camouflage standing in the middle of a quiet street facing the camera does not mean they pose for a picture. [0]
(2) Premise: Three firefighter come out of subway station. Hypothesis: Three firefighters putting out a fire inside of a subway station. Gold label: neutral		
(a) Predicted label: contradiction Explanation: The firefighters can not be putting out a fire station and putting out a fire at the same time. [0]	(b) Predicted label: neutral Explanation: The fact that three firemen are putting out of a subway station doesn't imply that they are putting out a fire. [0]	(c) Predicted label: neutral Explanation: The firefighters may not be putting out a fire inside of the subway station. [1]
(3) Premise: A blond-haired doctor and her African American assistant looking threw new medical manuals. Hypothesis: A man is eating pb and j. Gold label: contradiction		
(a) Predicted label: contradiction Explanation: A man is not a woman. [1]	(b) Predicted label: contradiction Explanation: One can not be looking and eating simultaneously. [0]	(c) Predicted label: contradiction Explanation: A person can not be looking at a medical and a book at the same time. [0]

While more research is needed to know whether the generated explanations provide reliable clues about the decision-making process of the model, it is reassuring to see that the explanations are plausible.

5.4.4 Universal Sentence Representations

Obtaining universal sentence representations is the task of training an encoder to provide semantically meaningful fixed-length representations of phrases/sentences (Kiros et al., 2015; Hill et al., 2016; Conneau et al., 2017). These representations are further used as features in other downstream tasks, particularly for tasks where labelled training data are scarce. In computer vision, pretrained ImageNet-based encoders provide standard image feature extractors for other downstream tasks (Deng et al., 2009). However, in natural language processing, there is still no consensus on general-purpose sentence encoders. It is an open question on which task and dataset should such an encoder be trained. Traditional approaches make use of very large

unsupervised datasets, taking weeks to train (Kiros et al., 2015). Recently, Conneau et al. (2017) showed that training only on natural language inference is both more accurate and more time-efficient than training on orders of magnitude larger but unsupervised datasets. Their results encourage the idea that more supervision can be more beneficial than larger but unsupervised datasets. Therefore, it is interesting to investigate whether an additional layer of supervision in the form of natural language explanations can further improve the learning of universal sentence representations.

Model. I use the BiLSTM-Max-PredExpl model trained in Section 5.4.2. To ensure that a potential improvement of BiLSTM-Max-PredExpl over BiLSTM-Max comes from the explanations and not simply from the addition of a language decoder, I introduce the BiLSTM-Max-AutoEnc model as an additional baseline. BiLSTM-Max-AutoEnc follows the same architecture as BiLSTM-Max-PredExpl, but instead of decoding explanations, it decodes the premise and hypothesis from their corresponding vector representations using a shared LSTM decoder. For BiLSTM-Max-AutoEnc, I use the same hypeparameters as for BiLSTM-Max-PredExpl.

Evaluation metrics. Typically, sentence representations are evaluated by using them as fixed features on top of which shallow classifiers are trained to perform a series of downstream tasks. Conneau et al. (2017) provide an excellent tool for evaluating sentence representations on 10 diverse tasks: movie reviews (MR), product reviews (CR), subjectivity/objectivity (SUBJ), opinion polarity (MPQA), question-type (TREC), sentiment analysis (SST), semantic textual similarity (STS), paraphrase detection (MRPC), entailment (SICK-E), and semantic relatedness (SICK-R). MRPC is evaluated with accuracy/F1-score. STS14 is evaluated with the Pearson/Spearman correlations. SICK-R is evaluated with the Pearson correlation. For all the rest of the tasks, accuracy is used for performance evaluation. A detailed description of each of these tasks and of the evaluation tool can be found in their paper.

Table 5.3: Performance of BiLSTM-Max-PredExpl and of the baselines, BiLSTM-Max and BiLSTM-Max-AutoEnc, on the 10 downstream tasks. Results are the averages of 5 runs with different seeds, with the standard deviations shown in parentheses. The best result for every task is indicated in bold. '*' indicates a significant difference at level 0.05 with respect to the BiLSTM-Max baseline.

Model	MR	CR	SUBJ	MPQA	SST2	TREC	MRPC	SICK-E	SICK-R	STS14
BiLSTM-Max	**78.18**	81.28	**92.46**	88.46	**82.12**	89.32	74.82 / 82.74	**85.96**	0.887	0.65 / 0.63
	(0.25)	(0.15)	(0.15)	(0.21)	(0.22)	(0.5)	(0.66 / 0.27)	(0.32)	(0.002)	(0 / 0)
BiLSTM-Max-AutoEnc	75.94*	79.26*	91.72*	88.16	80.9*	**90.52***	**76.2*** / 82.48	85.58	0.88*	0.5* / 0.5*
	(0.18)	(0.36)	(0.28)	(0.26)	(0.48)	(0.52)	(0.93 / 1.23)	(0.33)	(0)	(0.02 / 0.02)
BiLSTM-Max-PredExpl	77.76	**81.3**	92.14*	**88.78***	81.84	90	75.56 / **83.24***	85.92	**0.89***	**0.68** / **0.65***
	(0.44)	(0.16)	(0.21)	(0.22)	(0.4)	(0.51)	(0.62 / 0.24)	(0.52)	(0)	(0.01 / 0.01)

Results. Table 5.3 shows the results (as averages over 5 models trained with different seeds, with the standard deviations in parentheses) of BiLSTM-Max-PredExpl, BiLSTM-Max, and BiLSTM-Max-AutoEnc on the 10 downstream tasks mentioned above. To test whether the differences in performance of BiLSTM-Max-AutoEnc and BiLSTM-Max-PredExpl relative to the BiLSTM-Max baseline are significant, I performed a Welch's t-test.[5] The results that appeared significant under the significance level of 0.05 are marked with '*'.

We see that BiLSTM-Max-AutoEnc performs significantly worse than BiLSTM-Max on six tasks and significantly outperforms it on only two tasks. This indicates that adding a language generator can actually hurt performance. Instead, BiLSTM-Max-PredExpl significantly outperforms BiLSTM-Max on four tasks, while it is significantly outperformed only on one task. Therefore, one can conclude that training with explanations helps the model to learn overall better sentence representations.

5.4.5 Performance on Out-of-Domain Datasets

It is known that models trained on one distribution of instances do not perform well on instances from another distribution. For example, Bowman et al. (2015) obtained an accuracy of only 46.7% when training a model on SNLI and evaluating it on SICK-E (Marelli et al., 2014). Therefore, it is interesting to investigate whether explanations can help models to improve their performance on out-of-domain datasets, in both label prediction and explanation generation. I investigate this using the SICK-E (Marelli

[5]Using the implementation in `scipy.stats.ttest_ind` with `equal_var=False`.

Table 5.4: The performance without fine-tuning of BiLSTM-Max-PredExpl and the two baselines, BiLSTM-Max and BiLSTM-Max-AutoEnc, on SICK-E and MultiNLI. Results are average accuracies over 5 seeds, with standard deviations in parentheses. expl@100 shows the correctness of generated explanations via manual annotation.

Model	SICK-E acc/expl@100	MultiNLI acc/expl@100
BiLSTM-Max	53.27 (1.65) / -	57 (0.41) / -
BiLSTM-Max-AutoEnc	52.9 (1.77) / -	55.38 (0.9) / -
BiLSTM-Max-PredExpl	**53.54** (1.43) / 30.64	**57.16** (0.51) / 1.92

et al., 2014) and MultiNLI (Williams et al., 2018) datasets for natural language inference.

Model. I use the BiLSTM-Max-PredExpl model trained in Section 5.4.2. As baselines, I again use the already trained BiLSTM-Max and BiLSTM-Max-AutoEnc models.

Results. Table 5.4 shows the performance of BiLSTM-Max-PredExpl, BiLSTM-Max, and BiLSTM-Max-AutoEnc when evaluated without fine-tuning on SICK-E and MultiNLI. We see that the improvements obtained with BiLSTM-Max-PredExpl are very small. Therefore, we cannot conclude on the usefulness of explanations in improving the performance of this model on out-of-domain datasets.

To also evaluate the capability of BiLSTM-Max-PredExpl to generate correct natural language explanations, I manually annotated the explanations generated by this model for the first 100 instances in test sets of SICK-E and MultiNLI. The percentage of correct explanations in the subset where the label was predicted correctly is 30.64% for SICK-E and only 1.92% for MultiNLI. During annotation, I noticed that the explanations in SICK-E, even when wrong, were generally on-topic and valid statements, while the ones in MultiNLI were generally nonsense or off-topic. This is not surprising, since SICK-E is less complex and more similar to SNLI than MultiNLI is.

5.5 Conclusions and Open Questions

In this chapter, I introduced e-SNLI, a large dataset of ∼570K human-written natural language explanations for the influential task of natural language inference. I showed that it is much more difficult for a neural model that generates correct labels based on spurious correlations in SNLI to also generate correct explanations based on these correlations. This brings empirical evidence that models that generate correct explanations are more reliable than models that only predict correct labels.

I implemented various models that generate natural language explanations for their label predictions and I quantified the capability of these models to generate correct explanations when they predict the correct label. I also investigated the usefulness of providing these explanations at training time for obtaining better universal sentence representations and for improving the performance of a model on out-of-domain datasets.

Thus, the work described in this chapter paves the way for future development of robust neural models that can learn from natural language explanations at training time as well as generate correct natural language explanations at test time. Moreover, the e-SNLI dataset can also be exploited for other goals. For example, similar to the evaluation performed for visual question answering in Das et al. (2016), the highlighted tokens in e-SNLI may provide a source of supervision and evaluation for attention models (Rocktäschel et al., 2015; Parikh et al., 2016).

It also remains an open question how to automatically evaluate the quality of the generated explanations, since we have seen that automatic measures, such as the BLEU score, are not suitable.

It is also left as open question how to verify if the generated explanations are faithfully describing the decision-making process of the model.

Chapter 6

Verifying Neural Networks that Generate Natural Language Explanations

In this chapter, which is based on (Camburu et al., 2020), I introduce a simple yet effective adversarial framework to verify if neural models can generate inconsistent natural language explanations.

6.1 Motivation

In order to explain the predictions produced by black-box neural models, a growing number of works propose extending these models with natural language explanation generation modules, thus obtaining models that explain themselves in human language (Hendricks et al., 2016; Park et al., 2018; Kim et al., 2018b; Jansen et al., 2018; Ling et al., 2017; Rajani et al., 2019).

In this chapter, I first draw attention to the fact that such models, while appealing, are nonetheless prone to generating inconsistent explanations. Two explanations are considered to be inconsistent if they provide contradictory arguments about the instances and predictions that they aim to explain. For example, consider a visual question answering (VQA) task (Park et al., 2018) and two instances where the image is the same but the questions are different, say "Is there an animal in the image?" and "Can you see a Husky in the image?". If for the first instance a model predicts "Yes." and generates the explanation "Because there is a dog in the image.",

while for the second instance the *same* model predicts "No." and generates the explanation "Because there is no dog in the image.", then the model is producing a pair of inconsistent explanations.

Inconsistent explanations reveal at least one of the following undesired behaviours: (i) at least one of the explanations is not faithfully describing the decision-making process of the model, or (ii) the model relied on a faulty decision-making process for at least one of the instances. For a pair of inconsistent explanations, further investigation is needed to conclude which of these two behaviours is the actual one (and might vary for each instance). Indeed, a pair of inconsistent explanations does not necessarily imply at least one unfaithful explanation. In the previous example, if the image contains a dog, it is possible that the model identifies the dog when it processes the image together with the first question, and that the model does not identify the dog when it processes the image together with the second question, hence both explanations would faithfully reflect the decision-making process of the model even if they are inconsistent. Similarly, a pair of inconsistent explanations does not necessarily imply that the model relies on a faulty decision-making process, because the explanations may not faithfully describe the decision-making process of the model.

Before investigating the problem of identifying which of the two undesired behaviours is true for a given pair of inconsistent explanations, it is sensible to first examine whether a model is capable of generating inconsistent explanations. To this goal, I introduce the INCONSISTVERIF framework. For a given model m and an instance $\mathbf{x}$, INCONSISTVERIF aims to generate at least one input $\hat{\mathbf{x}}$ that causes m to generate an explanation that is inconsistent with the explanation generated by m for $\mathbf{x}$. Thus, INCONSISTVERIF falls under the category of *adversarial methods*, i.e., methods searching for inputs that cause a model to produce undesired answers (Biggio et al., 2013; Szegedy et al., 2014).

As part of INCONSISTVERIF, I address the problem of adversarial attacks with exact target sequences, a scenario that has not been previously addressed in sequence-to-sequence attacks, and which can be useful for other areas, such as dialog systems. Finally, I apply the framework to BILSTM-MAX-EXPLPRED-ATT presented in the

previous chapter, and show that this model can generate a significant number of inconsistent explanations.

6.2 Related Work

Verifying models that generate natural language explanations. Works on verifying models that generate natural language explanations are very scarce. Hendricks et al. (2018) identify the risk of generating natural language explanations that mention attributes from a strong class prior without any evidence being present in the input. In this chapter, I bring awareness to another risk, which is of generating inconsistent explanations.

Generating adversarial examples. Generating adversarial examples is an active research area in natural language processing (Zhang et al., 2020; Wang et al., 2019). For instance, Jia and Liang (2017b) analyse the robustness of extractive question answering models on examples obtained by adding adversarially generated distracting text. However, most works build on the requirement that the adversarial input should be a small perturbation of an original input (Belinkov and Bisk, 2018; Hosseini et al., 2017; Cheng et al., 2018), or should be preserving the semantics of the original input (Iyyer et al., 2018). Our setup does not have this requirement, and any pair of task-realistic inputs that causes the model to produce inconsistent explanations suffices. Most importantly, to my knowledge, no previous adversarial attack for sequence-to-sequence models generates *exact target sequences*, i.e., given a sequence, find an input that causes the model to generate the exact given sequence. Closest to this goal, Zhao et al. (2018) propose an adversarial framework for removing or adding tokens in the target sequence for the task of machine translation. Similarly, Cheng et al. (2018) require the presence of pre-defined tokens anywhere in the target sequence. They only test with up to three required tokens, and their success rate dramatically drops from 99% for one token to 37% for three tokens for the task of

automatic summarisation. Hence, their method would likely not generalise to exact target sequences.

Finally, Minervini and Riedel (2018) attempted to find inputs where a model trained on SNLI (Bowman et al., 2015) violates a set of logical constraints. This scenario may, in theory, lead to finding inputs that cause the generation of inconsistent explanations. However, their method needs to enumerate and evaluate a potentially very large set of perturbations of the inputs, e.g., removing sub-trees or replacing tokens with their synonyms, thus being computational expensive. Moreover, their scenario does not address the question of automatically producing undesired (inconsistent) sequences.

6.3 Framework

Consider a model m that, for each instance $\mathbf{x}$, generates a natural language explanation for its prediction on the instance. We refer to the explanation generated by m for $\mathbf{x}$ as $\mathbf{e}_m(\mathbf{x})$. I propose a framework that, for an instance $\mathbf{x}$, aims to generate new instances for which the model produces explanations that are inconsistent with $\mathbf{e}_m(\mathbf{x})$.

InconsistVerif consists of the following high-level steps. Given an instance $\mathbf{x}$, (A) create a list of explanations that are inconsistent with $\mathbf{e}_m(\mathbf{x})$, and (B) given an inconsistent explanation from the list created in A, find an input that causes m to generate this precise inconsistent explanation.

Setup. The setup we are facing has three desired properties that make it different from commonly researched adversarial settings in natural language processing:

- At step (B), the model has to generate a exact target sequence, since the goal is to generate the *exact* explanation that was identified at step (A) as inconsistent with the explanation $\mathbf{e}_m(\mathbf{x})$.

- Adversarial inputs do not have to be a paraphrase or a small perturbation of the original input, since the objective is to generate inconsistent explanations rather than incorrect predictions — these can happen as a byproduct.

- Adversarial inputs have to be realistic to the task at hand.

To my knowledge, this work is the first to tackle this setup, especially due to the challenging requirement of generating a exact target sequence, as described in Section 6.2.

Context-dependent inconsistencies. In certain tasks, instances consist of a context (such as an image or a paragraph), and some assessment to be made about the context (such as a question or a hypothesis). Since explanations may refer (sometimes implicitly) to the context, the assessment of whether two explanations are inconsistent may also depend on it. For example, in VQA, the inconsistency of the two explanations "Because there is a dog in the image." and "Because there is no dog in the image." depends on the image. However, if the image is the same, the two explanations are inconsistent regardless of which questions were asked on that image.

For the above reasons, given an instance $\mathbf{x}$, I differentiate between parts of the instance that will remain fixed in InconsistVerif (referred to as *context parts* and denoted as $\mathbf{x}_c$) and parts of the instance that InconsistVerif will vary in order to obtain inconsistencies (referred to as *variable parts* and denoted as $\mathbf{x}_v$). Hence, $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_v)$. In the VQA example above, $\mathbf{x}_c$ would be the image, and $\mathbf{x}_v$ would be the question.

Stand-alone inconsistencies. There are cases for which explanations are inconsistent regardless of the input. For example, explanations formed purely of background knowledge such as "A woman is a person." and "A woman is not a person."[1] are always inconsistent (and sometimes outrageous), regardless of the instances that lead to them. For these cases, InconsistVerif can treat the whole input as variable, i.e., $\mathbf{x}_c = \emptyset$ and $\hat{\mathbf{x}}_v = \mathbf{x}$.

Steps. InconsistVerif consists of the following steps:

[1]Which was generated by the model in our experiments.

1. Reverse the explanation generator module of model m by training a RevExpl model to map from the generated explanation and the context part of the input to the variable part of the input, i.e., $\text{RevExpl}(\mathbf{x}_c, \mathbf{e}_m(\mathbf{x})) = \mathbf{x}_v$.

2. For each explanation $\mathbf{e} = \mathbf{e}_m(\mathbf{x})$:

   (a) Create a list of statements that are inconsistent with $\mathbf{e}$, referred as $\mathcal{I}_\mathbf{e}$.

   (b) Query RevExpl on each $\hat{\mathbf{e}} \in \mathcal{I}_\mathbf{e}$ and the context $\mathbf{x}_c$. Get the new variable part $\hat{\mathbf{x}}_v = \text{RevExpl}(\mathbf{x}_c, \hat{\mathbf{e}})$ of a *reverse input* $\hat{\mathbf{x}} = (\mathbf{x}_c, \hat{\mathbf{x}}_v)$, which *may* cause m to produce inconsistent explanations.

   (c) Query m on each reverse input to get a *reverse explanation* $\mathbf{e}_m(\hat{\mathbf{x}})$.

   (d) Check if each reverse explanation $\mathbf{e}_m(\hat{\mathbf{x}})$ is indeed inconsistent with $\mathbf{e}$ by checking if $\mathbf{e}_m(\hat{\mathbf{x}}) \in \mathcal{I}_\mathbf{e}$.

To execute step (2a), for any task, one may define a set of logical rules to transform an explanation into an inconsistent counterpart, such as negation or replacement of task-essential tokens with task-specific antonyms. For example, in explanations for self-driving cars (Kim et al., 2018b), one can replace "green light" with "red light", or "the road is empty" with "the road is crowded" (which are task-specific antonyms), to get inconsistent — and hazardous — explanations such as "The car accelerates because there is a red light.".

Another strategy to obtain inconsistent explanations consists of swapping explanations from mutually exclusive labels. For example, assume a recommender system predicts that movie X is a bad recommendation for user Y "because X is a horror movie.", implying that user Y does not like horror movies. If the same system also predicts that movie Z is a good recommendation to the same user Y "because Z is a horror movie.", then we have an inconsistency, as the latter would imply that user Y likes horror movies.

While this step requires a degree of specific adjustment to the task at hand, it is arguably a small price to pay to ensure that the deployed system is coherent. An interesting direction for future work would be automatise this step, for example, by

training a neural network to generate task-specific inconsistencies after crowd-sourcing a dataset of inconsistent explanations for the task at hand.

To execute step (2d), InconsistVerif currently checks for an exact string match between a reverse explanation and any of the inconsistent explanations created at step (2a). Alternatively, one can train a model to identify if a pair of explanations forms an inconsistency.

Finally, while the framework itself does not directly enforce the adversarials to be realistic for the task at hand, the fact that they are generated by a model that is trained on realistic data induces the majority of the adversarials to be realistic, as we will see in the experiments below.

6.4 Experiments

I test InconsistVerif on the model BiLSTM-Max-ExplPred-Att introduced in the previous chapter. I set $\mathbf{x}_c$ as the premise (as this represents the given context for the task of solving natural language inference) and $\mathbf{x}_v$ as the hypothesis. However, due to the nature of SNLI for which decisions are based mostly on commonsense knowledge, the explanations are most of the time independent of the premise, such as "A dog is an animal.". Hence, it would be possible to also reverse the premise and not just the hypothesis, which is left as future work.

For the RevExpl model, I use the same neural architecture and hyperparameters used for BiLSTM-Max-ExplPred-Att. Thus, RevExpl takes as input a premise-explanation pair and generates a hypothesis. The trained RevExpl model is able to reconstruct *exactly the same* (i.e, string matching) hypothesis with 32.78% test accuracy.

Creating $\mathcal{I}_\mathbf{e}$. To execute step (2a), I employ negation and swapping explanations. For negation, I simply remove the tokens "not" and "n't" if they are present. If these tokens appear more than once in an explanation, I create multiple inconsistencies by

Table 6.1: Examples of inconsistent explanations detected by InconsistVerif on the model BiLSTM-Max-ExplPred-Att.

Premise: A guy in a red jacket is snowboarding in midair.	
Original Hypothesis: A guy is outside in the snow. Predicted Label: entailment Original explanation: **Snowboarding is done outside.**	Reverse Hypothesis: The guy is outside. Predicted label: contradiction Reverse explanation: **Snowboarding is not done outside.**
Premise: A man talks to two guards as he holds a drink.	
Original Hypothesis: The prisoner is talking to two guards in the prison cafeteria. Predicted Label: neutral Original explanation: **The man is not necessarily a prisoner.**	Reverse Hypothesis: A prisoner talks to two guards. Predicted Label: entailment Reverse explanation: **A man is a prisoner.**
Premise: Two women and a man are sitting down eating and drinking various items.	
Original Hypothesis: Three women are shopping at the mall. Predicted label: contradiction Original explanation: **There are either two women and a man or three women.**	Reverse Hypothesis: Three women are sitting down eating. Predicted label: neutral Reverse explanation: **Two women and a man are three women.**
Premise: Biker riding through the forest.	
Original Hypothesis: Man riding motorcycle on highway. Predicted Label: contradiction Original explanation: **Biker and man are different.**	Reverse Hypothesis: A man rides his bike through the forest. Predicted label: entailment Reverse explanation: **A biker is a man.**
Premise: A hockey player in helmet.	
Original Hypothesis: They are playing hockey Predicted Label: entailment Original explanation: **A hockey player in helmet is playing hockey.**	Reverse Hypothesis: A man is playing hockey. Predicted Label: neutral Reverse explanation: **A hockey player in helmet doesn't imply playing hockey.**
Premise: A blond woman speaks with a group of young dark-haired female students carrying pieces of paper.	
Original Hypothesis: A blond speaks with a group of young dark-haired woman students carrying pieces of paper. Predicted label: entailment Original explanation: **A woman is a female.**	Reverse Hypothesis:The students are all female. Predicted label: neutral Reverse explanation: **The woman is not necessarily female.**
Premise: The sun breaks through the trees as a child rides a swing.	
Original Hypothesis: A child rides a swing in the daytime. Predicted label: entailment Original explanation: **The sun is in the daytime.**	Reverse Hypothesis: The sun is in the daytime. Predicted label: neutral Reverse explanation: **The sun is not necessarily in the daytime.**
Premise: A family walking with a soldier.	
Original Hypothesis: A group of people strolling. Predicted label: entailment Original explanation: **A family is a group of people.**	Reverse Hypothesis: A group of people walking down a street. Predicted label: contradiction Reverse explanation: **A family is not a group of people.**

removing only one occurrence at a time. I do not attempt to add negation tokens, as this may result in grammatically incorrect sentences.

For swapping explanations on e-SNLI, we note that the explanations in this dataset largely follow a set of label-specific templates. This is a natural consequence of the task and the SNLI dataset and not a requirement in the collection of the e-SNLI. For example, annotators often used "One cannot X and Y simultaneously." to explain a contradiction, "Just because X, doesn't mean Y." for neutral, or "X implies Y." for entailment. Since any two labels are mutually exclusive, transforming an explanation from one template to a template of another label should automatically create an

inconsistency. For example, for the explanation of the contradiction "One cannot eat and sleep simultaneously.", one matches X to "eat" and Y to "sleep", and creates the inconsistent explanation "Eat implies sleep." using the entailment template "X implies Y.". Thus, for each label, I created a list of the most used templates that I manually identified for the explanations e-SNLI, which can be found in Appendix C. A running example of creating inconsistent explanations by swapping is given in Appendix C. If there is no negation and no template match, the instance is discarded. In our experiments, only 2.6% of the e-SNLI test set were discarded for this reason. This procedure may result in grammatically or semantically incorrect inconsistent explanations. However, as we will see below, RevExpl performed well in generating correct and relevant reverse hypotheses even when its input explanations were not correct. This is not surprising, since RevExpl has been trained to output ground-truth hypotheses.

The rest of the steps follow as described in (2b) - (2d). More precisely, for each inconsistent explanation in $\mathcal{I}_{\mathbf{e}}$, RevExpl returns a reverse hypothesis, which is fed back to BiLSTM-Max-ExplPred-Att to get a reverse explanation. To check whether a reverse explanation is inconsistent with the initial explanation, I checked for an exact string match with the explanations from the list of inconsistent explanations from step (2a).

Results and discussion. InconsistVerif identifies a total of 1044 pairs of inconsistent explanations starting from the e-SNLI test set, which contains of 9824 instances. Nonetheless, there are, on average, 1.93 ± 1.77 distinct reverse hypotheses giving rise to the same pair of inconsistent explanation. Since the hypotheses are distinct, each of these instances is a separate valid adversarial input. However, if one is strictly interested in the number of distinct pairs of inconsistent explanations, then, after eliminating duplications, 540 pairs of distinct inconsistencies remain.

Since the generation of natural language is always best evaluated by humans, I manually annotated 100 random distinct pairs. I found that 82% of the reverse hypotheses form realistic instances together with the premise. I also found that the

majority of the unrealistic instances are due to a repetition of a token in the hypothesis. For example, "A kid is riding a helmet with a helmet on training." is a generated reverse hypothesis which is very close to a perfectly valid hypothesis.

Given the estimation of 82% of the detected inconsistencies to be caused by realistic reverse hypotheses, ∼443 distinct pairs of inconsistent explanations are detected by InconsistVerif. While this means that InconsistVerif only has a success rate of ∼4.51% on BiLSTM-Max-ExplPred-Att, it is nonetheless alarming that this simple and under-optimised adversarial framework detects a significant number of inconsistencies on a model trained on ∼570K examples. In Table 6.1, we see examples of detected inconsistencies. Note how the reverse hypotheses are realistic given the associated premises.

Manual scanning. It is interesting to investigate to what extent one can find inconsistencies via brute-force manual scanning. I performed three such experiments, with no success. On the contrary, I noticed a good level of robustness against inconsistencies when scanning through the generic adversarial hypotheses introduced by Carmona et al. (2018).

In the first experiment, I manually analysed the first 50 instances in the test set without finding any inconsistency. However, these examples were involving different concepts, thus decreasing the likelihood of finding inconsistencies. To account for this, in the second experiment, I constructed three groups around the concepts of *woman*, *prisoner*, and *snowboarding*, respectively, by simply selecting the explanations in the test set containing these words. I selected these concepts, because InconsistVerif detected inconsistencies about them (see Table 6.1).

For *woman*, there were 1150 instances in the test set on which BiLSTM-Max-ExplPred-Att generated explanations containing this word. I looked at a random sample of 20 explanations, among which I did not find any inconsistency. For *snowboarding*, there were 16 instances in the test set for which BiLSTM-Max-ExplPred-Att generated explanations containing this word, and no inconsistency among them.

For *prisoner*, there was only one instance in the test set for which BiLSTM-Max-ExplPred-Att generated an explanation containing this word, so there was no way to find out that the model is inconsistent with respect to this concept simply by scanning the test set.

I only looked at the test set for a fair comparison with InconsistVerif that was only applied on this set.

Even if the manual scanning were successful, it should not be regarded as a proper baseline, since it does not bring the same benefits as InconsistVerif. Indeed, manual scanning requires considerable human effort to look over a large set of explanations in order to find if any two are inconsistent. Even a group of only 50 explanations required a non-negligible amount of time. Moreover, restricting ourselves to the instances in the original dataset would clearly be less effective than being able to generate new instances. The InconsistVerif framework addresses these issues and directly provides pairs of inconsistent explanations. Nonetheless, this experiment is useful for illustrating that BiLSTM-Max-ExplPred-Att does not provide inconsistent explanations in a frequent manner.

In the third experiment of manual scanning, I experimented with a few manually created hypotheses from the dataset introduced by Carmona et al. (2018). These hypothesis had been shown to induce confusion at the label level. However, BiLSTM-Max-ExplPred-Att showed a good level of robustness against inconsistencies on the inspected hypotheses. For example, for the neutral pair (premise: "A bird is above water.", hypothesis: "A swan is above water."), BiLSTM-Max-ExplPred-Att generates the explanation "Not all birds are a swan.", while when interchanging "bird" with "swan", i.e., for the pair (premise: "A swan is above water.", hypothesis: "A bird is above water."), BiLSTM-Max-ExplPred-Att generates the explanation "A swan is a bird.", showing a good understanding of the relationship between the entities "swan" and "bird". Similarly, interchanging "child" with "toddler" in (premise: "A small child watches the outside world through a window.", hypothesis: "A small toddler watches the outside world through a window.") does not confuse the model, which generates "Not every child is a toddler." and "A toddler is a small child.",

respectively. Further investigation on whether the model can be tricked on concepts where it seems to exhibit robustness, such as *toddler* or *swan*, is left for future work.

6.5 Conclusions and Open Questions

In this chapter, I drew attention to the fact that models generating natural language explanations are prone to producing inconsistent explanations. This concern is general and can have a large practical impact. For example, users would likely not accept a self-driving car if its explanation module is prone to state both that "The car accelerates because there is no one crossing the intersection." and that "The car accelerates because there are people crossing the intersection.".

I introduced a generic framework for identifying such inconsistencies and showed that BiLSTM-Max-ExplPred-Att, the model that produced the highest parentage of correct explanations on e-SNLI in Chapter 5, can generate a significant number of inconsistencies.

Valuable directions for future work include: (1) developing more advanced frameworks for detecting inconsistencies, (2) investigating whether inconsistencies are due to unfaithful explanations or to a faulty decision-making process of the model, and (3) developing more robust models that do not generate inconsistencies.

Chapter 7

General Conclusions and Perspectives

In this thesis, I investigated two major directions for explaining deep neural networks: feature-based post-hoc explanatory methods and self-explanatory neural models that generate natural language explanations for their predictions. For both directions, I investigated the question of verifying the faithfulness with which the explanations describe the decision-making processes of the models that they aim to explain. In addition, for the class of self-explanatory models that generate natural language explanations, I investigated whether these models exhibit an improved behaviour due to the additional supervision in the form of natural language explanations at training time. The results are as follows.

First, I showed that, despite the apparent implicit assumption that there is only one ground-truth feature-based explanation for the prediction of a model on an instance, there are often more such ground-truths. Moreover, I showed that two important classes of post-hoc explanatory methods aim to provide different types of ground-truth feature-based explanations, without explicitly stating so, and I unveil certain strengths and limitations for each of these types. Furthermore, for certain cases, none of these types of explanations is enough to provide a non-ambiguous view of the decision-making process of a model. I also showed that the selector-predictor type of neural model, which is expected to provide the relevant features for each prediction, has certain limitations in doing so, and I provided solutions that can alleviate some of

these limitations.

Future work on rigorous specifications of explanatory methods would be necessary for proper usage of these methods in practice. More investigation into how explanations can provide a complete view of the decision-making process of a model would also be valuable. Moreover, robustifying selector-predictor models to provide faithful insight into their decision-making processes would also be an important direction of research.

Second, I introduced a framework for verifying the faithfulness of the explanations provided by feature-based post-hoc explanatory methods. This framework relies on the use of selector-predictor models as target models, after the limitations of this type of model are alleviated. This framework is automatic and verifies explainers on a realistic scenario, i.e., on non-trivial neural models that are trained on real-world datasets. The framework is also generic and has, therefore, the advantage to generate a potentially very large number of off-the-shelf sanity tests by being instantiated on other tasks and other architectures of the selector and the predictor modules. The results of testing three popular explanatory methods on this framework showed its potential to reveal unfaithful explanations provided by these methods. I also presented ways for further improving this framework.

As future work, the introduced verification framework can further be improved, for example, using the guidelines that I introduced in Section 4.6. Also, other types of feature-based self-explanatory methods might also be exploited as a testbed for verifying explainers. Moreover, recall that the introduced framework is a sanity check and not an evaluation framework for providing the performance of explainers in full generality. Complete evaluation frameworks would also be highly desired.

Third, I investigated the class of self-explanatory neural models that generate natural language explanations for their predictions. To address the scarcity of datasets of natural language explanations, I introduced a new and large dataset of ∼570K human-written natural language explanations, which I called e-SNLI. Further, I showed that it is more difficult for neural models to rely on spurious correlations to provide correct explanations than to provide correct labels. This empirically supports the intuition that models that can generate correct argumentation in addition to correct predictions

are more reliable than models that just provide correct predictions. Furthermore, I showed that a series of models that generate natural language explanations at test time provide relatively low performance in generating correct explanations. However, I also showed that improvements are possible with better architectures. Finally, I showed that the presence of additional explanations at training time guide models into learning better universal sentence representations and into having better capabilities to solve out-of-domain instances, even though the improvements were minor for the tested models.

Future work would further exploit the e-SNLI dataset to advance research in the direction of training with and generation of natural language explanations. The models introduced in this thesis, while providing encouraging results, are not yet deployable in the real world, and more work is needed to advance this promising direction. New datasets of natural language explanations would also be highly beneficial.

Fourth, as a step towards verifying the faithfulness of the generated natural language explanations, I introduced an adversarial framework to check whether neural models generate inconsistent natural language explanations. Inconsistent explanations expose either an unfaithful explanation or a faulty decision-making process of the model, either of which is undesirable. I applied the framework to the model that produced the highest percentage of correct explanations on e-SNLI and I showed that this model is capable of generating a significant number of inconsistent explanations. Moreover, as part of the introduced framework, I addressed the problem of adversarial attacks with full target sequences, a scenario that has not been previously investigated in sequence-to-sequence attacks, and which can be useful for other natural language tasks.

Verifying the faithfulness of natural language explanations generated by self-explanatory models remains an open question. The adversarial framework introduced in this thesis checks whether models can generate inconsistent natural language explanations. However, further work is needed to find whether an inconsistency is due to unfaithful explanations or to a flaw in the decision-making process of the model. Moreover, future work on building more robust models that do not generate

inconsistent explanations would also be very valuable. My hope is that, in the future, we will have robust and accurate neural models that faithfully explain themselves in human language.

Appendix A

Examples from the Aroma and Appearance Sanity Tests

Figures A.1 and A.2 each provide an example from our sanity tests from the appearance and aroma aspects, respectively.

In Figures A.1, we see that LIME and L2X detected "laces" as the most important token, yet this token is an irrelevant token for the prediction. KernalSHAP however managed to detect "fizzy" as most important token, which appears as a better behaviour since "fizzy" is an independently relevant token. However, "nice" is also an independently relevant token, yet KernalSHAP flags it as less important than "laces". SelPredVerif penalises the explainers for these errors.

Similarly, in the example in Figure A.2, we see that LIME ranks the irrelevant tokens "and" and "flavour" higher than the independently relevant token "rich". SelPredVerif penalises LIME for this. However, the fact that KernalSHAP and L2X placed "aroma" (with both its occurrences for SHAP) higher than "rich" is not penalised by SelPredVerif because both occurrences of "aroma" were selected tokens.

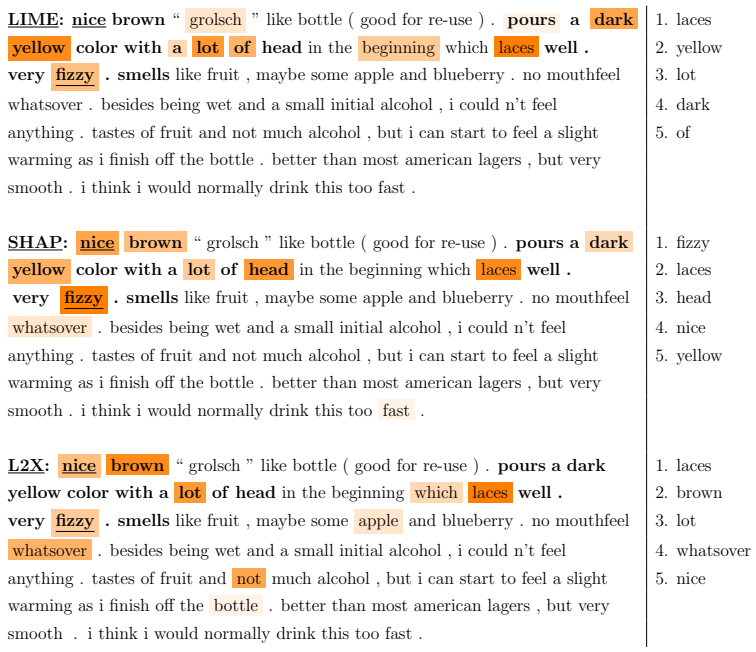

Figure A.1: Explainers rankings (with top 5 features on the right-hand side) on an instance from the appearance aspect in our test. The selected tokens are in bold, and the independently relevant tokens are additionally underlined. The non-bold tokens should not be ranked higher than any bold and underlined token. KernalSHAP is simply referred as SHAP.

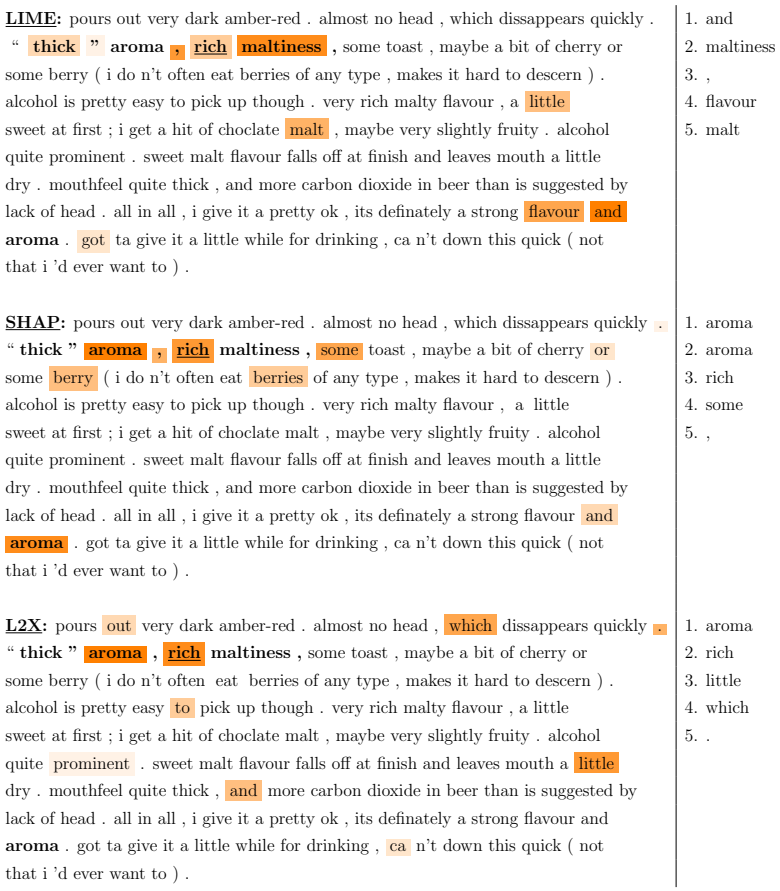

Figure A.2: Explainers rankings (with top 5 features on the right-hand side) on an instance from the aroma aspect in our test. The selected tokens are in bold, and the independently relevant tokens are additionally underlined. The non-bold tokens should not be ranked higher than any bold and underlined token. KernalSHAP is simply referred as SHAP.

Appendix B

Templates to Filter Uninformative Explanations

This Appendix presents the list of templates used to filter uninformative explanations in Section 5.3. These templates were identified by manually scanning through the dataset.

General templates

<PREMISE>

<HYPOTHESIS>

<HYPOTHESIS> <PREMISE>

<PREMISE> <HYPOTHESIS>

Sentence 1 states <PREMISE>. Sentence 2 is stating <HYPOTHESIS>

Sentence 2 states <HYPOTHESIS>. Sentence 1 is stating <PREMISE>

There is <PREMISE>

There is <HYPOTHESIS>

Entailment templates

<PREMISE> implies <HYPOTHESIS>

If <PREMISE> then <HYPOTHESIS>

<PREMISE> would imply <HYPOTHESIS>

<HYPOTHESIS> is a rephrasing of <PREMISE>

<PREMISE> is a rephrasing of <HYPOTHESIS>

In both sentences <HYPOTHESIS>

<PREMISE> would be <HYPOTHESIS>

<PREMISE> can also be said as <HYPOTHESIS>

<HYPOTHESIS> can also be said as <PREMISE>

<HYPOTHESIS> is a less specific rephrasing of <PREMISE>

This clarifies that <HYPOTHESIS>

If <PREMISE> it means <HYPOTHESIS>

<HYPOTHESIS> in both sentences

<HYPOTHESIS> in both

<HYPOTHESIS> is same as <PREMISE>

<PREMISE> is same as <HYPOTHESIS>

<PREMISE> is a synonym of <HYPOTHESIS>

<HYPOTHESIS> is a synonym of <PREMISE>.

Neutral templates

Just because <PREMISE> doesn't mean <HYPOTHESIS>

Cannot infer the <HYPOTHESIS>

One cannot assume <HYPOTHESIS>

One cannot infer that <HYPOTHESIS>

Cannot assume <HYPOTHESIS>

<PREMISE> does not mean <HYPOTHESIS>

We don't know that <HYPOTHESIS>

The fact that <PREMISE> doesn't mean <HYPOTHESIS>

The fact that <PREMISE> does not imply <HYPOTHESIS>

The fact that <PREMISE> does not always mean <HYPOTHESIS>

The fact that <PREMISE> doesn't always imply<HYPOTHESIS>.

Contradiction templates

In sentence 1 <PREMISE> while in sentence 2 <HYPOTHESIS>

It can either be <PREMISE> or <HYPOTHESIS>

It cannot be <HYPOTHESIS> if <PREMISE>

Either <PREMISE> or <HYPOTHESIS>

Either <HYPOTHESIS> or <PREMISE>

<PREMISE> and other <HYPOTHESIS>

<HYPOTHESIS> and other <PREMISE>

<HYPOTHESIS> after <PREMISE>

<PREMISE> is not the same as <HYPOTHESIS>

<HYPOTHESIS> is not the same as <PREMISE>

<PREMISE> is contradictory to <HYPOTHESIS>

<HYPOTHESIS> is contradictory to <PREMISE>

<PREMISE> contradicts <HYPOTHESIS>

<HYPOTHESIS> contradicts <PREMISE>

<PREMISE> cannot also be <HYPOTHESIS>

<HYPOTHESIS> cannot also be <PREMISE>

Either <PREMISE> or <HYPOTHESIS>

Either <PREMISE> or <HYPOTHESIS> not both at the same time

<PREMISE> or <HYPOTHESIS> not both at the same time.

Appendix C

Templates Identified in e-SNLI

This Appendix presents the list of templates that I manually found to match most of the e-SNLI explanations. During the collection of the dataset, no template was imposed, they were a natural consequence of the task and SNLI dataset.

"*subphrase1/subphrase2/...*" means that a separate template is to be considered for each of the subphrases. X and Y are the key elements that we want to identify and use in the other templates in order to create inconsistencies. "[...]" is a placeholder for any string, and its value is not relevant. Subphrases placed between parentheses (for example, "(the)" or "(if)") are optional, and two distinct templates are formed one with and one without that subphrase.

Entailment Templates

- X is/are a type of Y
- X implies Y
- X is/are the same as Y
- X is a rephrasing of Y
- X is a another form of Y
- X is synonymous with Y
- X and Y are synonyms/synonymous

- X and Y is/are the same thing
- (if) X , then Y
- X so Y
- X must be Y
- X has/have to be Y
- X is/are Y

Neutral Templates

- not all X are Y
- not every X is Y
- just because X does not/n't mean/imply Y
- X is/are not necessarily Y
- X does not/n't have to be Y
- X does not/n't imply/mean Y

Contradiction Templates

- ([...]) cannot/can not/ca n't (be) X and Y at the same time/simultaneously
- ([...]) cannot/can not/ca n't (be) X and at the same time Y
- X is/are not (the) same as Y
- ([...]) is/are either X or Y
- X is/are not Y
- X is/are the opposite of Y

- ([...]) cannot/can not/ca n't (be) X if (is/are) Y
- X is/are different than Y
- X and Y are different ([...])

Running Example

Below, I present a running example for creating inconsistencies by swapping between templates of explanations.

Consider the explanation **e**: "Dog is a type of animal." which may arise from a model explaining the instance **x**: (premise: "A dog is in the park.", hypothesis: "An animal is in the park."). One identifies that **e** matches the template "X is/are a type of Y" with X = "dog" and Y = "animal". The list $\mathcal{I}_{\mathbf{e}}$ is then generated by replacing X and Y in each of the neutral and contradictory templates listed above with the exception of those that contain "[...]" in order to avoid guessing the placeholder.

Neutral inconsistencies

- not all dog are animal
- not every dog is animal
- just because dog does not/n't mean/imply animal
- dog is/are not necessarily animal
- dog does not/n't have to be animal
- dog does not/n't imply/mean animal

Contradiction inconsistencies

- cannot/can not/ca n't (be) dog and animal at the same time/simultaneously
- cannot/can not/ca n't (be) dog and at the same time animal
- dog is/are not (the) same as animal

- is/are either dog or animal
- dog is/are not animal
- dog is/are the opposite of animal
- cannot/can not/ca n't (be) dog if (is/are) animal
- dog is/are different than animal
- dog and animal are different

Bibliography

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. (2018). Sanity checks for saliency maps. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 9505–9515. Curran Associates, Inc.

Akula, A., Wang, S., and Zhu, S. (2020). CoCox: Generating conceptual and counterfactual explanations via fault-lines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2594–2601.

Alaa, A. M. and van der Schaar, M. (2019). Demystifying black-box models with symbolic metamodels. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 11304–11314. Curran Associates, Inc.

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B.-J., Srivastava, M., and Chang, K.-W. (2018). Generating natural language adversarial examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2890–2896.

Arras, L., Montavon, G., Müller, K.-R., and Samek, W. (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.

Arrieta, A. B., Diaz-Rodriguez, N., Ser, J. D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., and Herrera, F.

(2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82 – 115.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Banzhaf-III, J. F. (1964). Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev., 19:317*.

Belinkov, Y. and Bisk, Y. (2018). Synthetic and natural noise both break neural machine translation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Belinkov, Y., Poliak, A., Shieber, S. M., Durme, B. V., and Rush, A. M. (2019). Don't take the premise for granted: Mitigating artifacts in natural language inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 877–891.

Bhathena, H., Willis, A., and Dass, N. (2020). Evaluating compositionality of sentence representation models. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 185–193.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) (3)*, volume 8190 of *Lecture Notes in Computer Science*, pages 387–402. Springer.

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., and Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4349–4357. Curran Associates, Inc.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 632–642.

Buolamwini, J. and Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 77–91.

Cai, Z., Tu, L., and Gimpel, K. (2017). Pay attention to the ending: Strong neural baselines for the ROC story cloze task. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 616–622.

Camburu, O.-M., Giunchiglia, E., Foerster, J., Lukasiewicz, T., and Blunsom, P. (2019). Can I trust the explainer? Verifying post-hoc explanatory methods. In *Neural Information Processing Systems Workshop on Safety and Robustness in Decision Making*.

Camburu, O.-M., Rocktäschel, T., Lukasiewicz, T., and Blunsom, P. (2018). e-snli: Natural language inference with natural language explanations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 9539–9549. Curran Associates, Inc.

Camburu, O.-M., Shillingford, B., Minervini, P., Lukasiewicz, T., and Blunsom, P. (2020). Make up your mind! adversarial generation of inconsistent natural language explanations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4157–4165.

Carlini, N. and Wagner, D. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7.

Carmona, V. I. S., Mitchell, J., and Riedel, S. (2018). Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of*

the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT), pages 1975–1985.

Carter, B., Mueller, J., Jain, S., and Gifford, D. K. (2019). What made you do this? Understanding black-box decisions with sufficient input subsets. *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. (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.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

Chandra, R. and Omlin, C. (2007). Knowledge discovery using artificial neural networks for a conservation biology domain. *Proceedings of International Conference on Data Mining*, pages 221–227.

Chang, S., Zhang, Y., Yu, M., and Jaakkola, T. (2019). A game theoretic approach to class-wise selective rationalization. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32*, pages 10055–10065. Curran Associates, Inc.

Chen, D., Bolton, J., and Manning, C. D. (2016a). A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2358–2367.

Chen, J. and Jordan, M. I. (2020). LS-Tree: Model interpretation when the data are linguistic. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Chen, J., Song, L., Wainwright, M., and Jordan, M. (2018a). Learning to explain: An information-theoretic perspective on model interpretation. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 80:883–892.

Chen, Q., Zhu, X., Ling, Z., Wei, S., and Jiang, H. (2016b). Enhancing and combining sequential and tree LSTM for natural language inference. *CoRR*, abs/1609.06038.

Chen, Q., Zhu, X., Ling, Z.-H., Inkpen, D., and Wei, S. (2018b). Neural natural language inference models enhanced with external knowledge. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2406–2417.

Cheng, M., Yi, J., Zhang, H., Chen, P., and Hsieh, C. (2018). Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Cooper, G. F., Abraham, V., Aliferis, C. F., Aronis, J. M., Buchanan, B. G., Caruana, R., Fine, M. J., Janosky, J. E., Livingston, G., Mitchell, T., Monti, S., and Spirtes, P. (2005). Predicting dire outcomes of patients with community acquired pneumonia. *Journal of Biomedical Informatics*, 38(5):347 – 366.

Das, A., Agrawal, H., Zitnick, L., Parikh, D., and Batra, D. (2016). Human attention in visual question answering: Do humans and deep networks look at the same

regions? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 932–937.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li, F. F. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Do, V., Camburu, O.-M., Akata, Z., and Lukasiewicz, T. (2020). e-SNLI-VE-2.0: Corrected visual-textual entailment with natural language explanations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop on Fair, Data Efficient and Trusted Computer Vision*.

Dressel, J. and Farid, H. (2018). The accuracy, fairness, and limits of predicting recidivism. *Science Advances*, 4(1).

Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., and Beck, H. P. (2003). The role of trust in automation reliance. *International Journal of Human-Computer Studies*, 58(6):697 – 718.

Eldan, R. and Shamir, O. (2015). The power of depth for feedforward neural networks. *CoRR*, abs/1512.03965.

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., and Roberts, A. (2019). GANSynth: Adversarial neural audio synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc.

Frosst, N. and Hinton, G. E. (2017). Distilling a neural network into a soft decision tree. *CoRR*, abs/1711.09784.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *CoRR*, abs/1508.06576.

Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., and Kagal, L. (2018). Explaining explanations: An overview of interpretability of machine learning. *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*.

Glockner, M., Shwartz, V., and Goldberg, Y. (2018a). Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 650–655.

Glockner, M., Shwartz, V., and Goldberg, Y. (2018b). Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 650–655.

Gong, Y., Luo, H., and Zhang, J. (2018). Natural language inference over interaction space. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. (2018). Annotation artifacts in natural language inference data. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 107–112.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., and Darrell, T. (2016). Generating visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 9908 of *LNCS*, pages 3–19.

Hendricks, L. A., Hu, R., Darrell, T., and Akata, Z. (2018). Grounding visual explanations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 11206, pages 269–286.

Hill, F., Cho, K., and Korhonen, A. (2016). Learning distributed representations of sentences from unlabelled data. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1367–1377.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., and Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hosseini, H., Xiao, B., and Poovendran, R. (2017). Deceiving Google's cloud video intelligence API built for summarizing videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1305–1309. IEEE Computer Society.

Huang, C. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Hawthorne, C., Dai, A. M., Hoffman, M. D., and Eck, D. (2018). An improved relative self-attention mechanism for transformer with application to music generation. *CoRR*, abs/1809.04281.

Ibrahim, M., Louie, M., Modarres, C., and Paisley, J. W. (2019). Global explanations of neural networks: Mapping the landscape of predictions. In *Proceedings of the AAAI Conference on AI, Ethics, and Society*, pages 279–87.

Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1875–1885.

Jansen, P., Wainwright, E., Marmorstein, S., and Morrison, C. (2018). WorldTree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*.

Jia, R. and Liang, P. (2017a). Adversarial examples for evaluating reading comprehension systems. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2021–2031.

Jia, R. and Liang, P. (2017b). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2021–2031.

Kádár, Á., Chrupała, G., and Alishahi, A. (2017). Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Kanehira, A. and Harada, T. (2019). Learning to explain with complemental examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., and Sayres, R. (2018a). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kim, J., Rohrbach, A., Darrell, T., Canny, J. F., and Akata, Z. (2018b). Textual explanations for self-driving vehicles. *Proceedings of the European Conference on Computer Vision (ECCV)*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 3294–3302. Curran Associates, Inc.

Kocijan, V., Cretu, A.-M., Camburu, O.-M., Yordanov, Y., Blunsom, P., and Lukasiewicz, T. (2019a). WikiCREM: A large unsupervised corpus for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Hong Kong.

Kocijan, V., Cretu, A.-M., Camburu, O.-M., Yordanov, Y., and Lukasiewicz, T. (2019b). A surprisingly robust trick for the Winograd schema challenge. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4837–4842.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Lapuschkin, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10:e0130140.

Lei, T., Barzilay, R., and Jaakkola, T. (2015). Molding CNNs for text: non-linear, non-consecutive convolutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1565–1575.

Lei, T., Barzilay, R., and Jaakkola, T. (2016). Rationalizing neural predictions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 107–117.

Levy, O. and Dagan, I. (2016). Annotating relation inference in context via question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 249–255.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. (2017). Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 158–167.

Liu, P., Qiu, X., Zhou, Y., Chen, J., and Huang, X. (2016). Modelling interaction of sentence pair with coupled-LSTMs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1703–1712.

Lombrozo, T. (2006). The structure and function of explanations. *Trends in Cognitive Sciences*, 10:464–470.

Lombrozo, T. (2009). Explanation and categorization: How "why?" informs "what?". *Cognition 110(2):248–253*.

Lombrozo, T. (2012). Explanation and abductive inference. *Oxford Handbook of Thinking and Reasoning*, page 260–276.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 142–150.

Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A. L., and Murphy, K. (2016). Generation and comprehension of unambiguous object descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., and Zamparelli, R. (2014). A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)*, pages 216–223.

McAuley, J. J., Leskovec, J., and Jurafsky, D. (2012). Learning attitudes and attributes from multi-aspect reviews. *CoRR*, abs/1210.3926.

McCoy, T., Pavlick, E., and Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3428–3448.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Minervini, P. and Riedel, S. (2018). Adversarially regularising neural NLI models to integrate logical background knowledge. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL)*, pages 65–74.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. (2017). Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nie, Y. and Bansal, M. (2017). Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*, pages 311–318.

Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2249–2255.

Park, D. H., Hendricks, L. A., Akata, Z., Rohrbach, A., Schiele, B., Darrell, T., and Rohrbach, M. (2018). Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, J. and Schaal, S. (2008). Natural actor-critic. *Neurocomputing*, 71:1180–1190.

Plumb, G., Molitor, D., and Talwalkar, A. S. (2018). Model agnostic supervised local explanations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 2515–2524. Curran Associates, Inc.

Poerner, N., Schütze, H., and Roth, B. (2018). Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 340–350.

Rajani, N. F., McCann, B., Xiong, C., and Socher, R. (2019). Explain yourself! Leveraging language models for commonsense reasoning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4932–4942.

Ribeiro, M. T., Singh, S., and Guestrin, C. (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.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kociský, T., and Blunsom, P. (2015). Reasoning about entailment with neural attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing 45:2673-2681*.

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. (2019). Grad-CAM: Why did you say that? Visual explanations from deep networks via gradient-based localization. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Shapley, L. S. (1951). Notes on the n-person game – II: The value of an n-person game. *RAND Corporation*.

Shapley, L. S. and Shubik, M. (1971). The assignment game : The core. *International Journal of Game Theory*, 1(1):111–130, 1971.

Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3145–3153.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529:484–489.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In Bengio, Y. and LeCun, Y., editors, *2nd International Conference on Learning Representations (ICLR) Workshop Track Proceedings*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

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

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Telgarsky, M. (2016). Benefits of depth in neural networks. In Feldman, V., Rakhlin, A., and Shamir, O., editors, *Proceedings of Machine Learning Research*, volume 49, pages 1517–1539. PMLR.

Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., and Gershman, S. J. (2017). Human learning in Atari. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Wang, W., Tang, B., Wang, R., Wang, L., and Ye, A. (2019). Towards a robust deep neural network in texts: A survey. *CoRR*, abs/1902.07285.

Webster, K., Recasens, M., Axelrod, V., and Baldridge, J. (2018). Mind the GAP: A balanced corpus of gendered ambiguous pronouns. In *Transactions of the Association for Computational Linguistics (TACL)*, pages 6:605–617.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HTL)*, pages 1112–1122.

Williams, J. J. and Lombrozo, T. (2010). The role of explanation in discovery and generalization: Evidence from category learning. *Cognitive Science 34(5):776-806*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 37:2048–2057.

Yang, C., Rangarajan, A., and Ranka, S. (2018). Global model interpretation via recursive partitioning. In *Proceedings of the IEEE International Conference on High Performance Computing and Communications*, pages 1563–1570.

Yang, M. and Kim, B. (2019). BIM: Towards quantitative evaluation of interpretability methods with ground truth. *CoRR*, abs/1907.09701.

Yoon, J., Jordon, J., and van der Schaar, M. (2019). INVASE: Instance-wise variable selection using neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yordanov, Y., Camburu, O.-M., Kocijan, V., and Lukasiewicz, T. (2020). Does the objective matter? comparing training objectives for pronoun resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhang, W. E., Sheng, Q. Z., Alhazmi, A., and Li, C. (2020). Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3).

Zhao, Z., Dua, D., and Singh, S. (2018). Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR)*.