Overview Of SHapley Additive exPlanations (SHAP) In Natural Language Processing

Bachelor's Thesis

by

Johannes Widera

submitted to

Professorship for Computer Networks
Prof. Dr. Martin Mauve
Heinrich-Heine-University Düsseldorf

August 2023

Supervisor:
Marc Feger, M. Sc.

Abstract

In recent years, the application of Machine Learning (ML) in Natural Language Processing (NLP) has witnessed exponential growth. One challenge is to interpret the increasingly complex ML models to ensure trustworthy model deployment. One promising technique for interpreting ML models is the use of Shapley Additive exPlanation (SHAP). This technique assigns an importance score to each feature, indicating how much it contributed to the output of a certain model. The benefit of this technique is that it treats every model as a black box and ignores the underlying complexity, making it highly versatile. This thesis is motivated by the lack of in-depth investigation when it comes to understanding how SHAP can be applied effectively to NLP models. By explaining the evolution of SHAP and applying SHAP to various text-based models, we provide a comprehensive overview of the possibilities of applying it to this critical domain. Our findings underscore the importance of interpretability in ML models because it helps to debug models and provides explanations of why a model output was made, increasing trust in these models, and improving the reliability. However, we also found that SHAP still has a problem with fast computation of the result, especially when the input text and complexity of the model increase. Additionally, there is little reliance on whether the SHAP output is properly interpreted. SHAP provides valuable information to interpret ML models, but there is still a need for improvement in applying SHAP to the domain of NLP.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Motivation

In recent years, the application of Machine Learning (ML) in Natural Language Processing (NLP) has witnessed exponential growth. One challenge is explaining the models and understanding how they make their decision. As the dimensionality of the data increases, as seen in NLP, even simple models such as decision trees become less interpretable. However, especially in NLP, the trend is moving away from classical models towards the use of models that operate in the area of deep learning [1, 2]. Although impressive classification results can be achieved with these models, the question remains of which decisions and properties of the data the respective model uses for decision making (black boxes).

The rise of deep learning models, particularly language models like BERT, raises new concerns about the potential amplification of human biases present in the training data. An exemplary scenario for this is given: the model unconsciously associates certain occupations or emotions with certain genders [3]. To detect and avoid such bias effects, it is crucial to develop a thorough understanding of how such models work and possible spurious correlations before using them in real-world situations. Only by having a detailed knowledge of how they work we can ensure that models are used responsibly and accurately and that potentially negative effects can be avoided.

The Shapley Additive exPlanations (SHAP) technique has gained prominence as an effective method for explaining machine learning models, with extensive evaluation in various do-

mains [4, 5, 6]. However, despite the growing usage of SHAP in various domains to explain machine learning models, its potential application in NLP has not been comprehensively explored. This lack of in-depth investigation leaves a critical gap in our understanding of how SHAP can effectively interpret NLP models. Our research is motivated by the need to address this limitation and delve into the use of SHAP, for NLP, with the aim of providing valuable insights to bridge this knowledge gap. By shedding light on SHAP's usefulness in interpreting NLP models, we hope to equip researchers with the necessary knowledge and tools to effectively leverage this interpretability technique. Through our efforts, we aim to improve both the interpretability and performance of NLP models that contribute to advances in AI within this critical domain.

1.2 Thesis Outline

The thesis mainly contains four sections. In Section 2, we will introduce essential concepts to provide you with the necessary knowledge to follow along. We will begin with a brief introduction to Interpretable Machine Learning (IML), exploring its importance and taxonomy. Additionally, we will introduce NLP and look at specific challenges, along with a look at two common NLP tasks that will serve as a foundation for later explanations. Furthermore, we will introduce three ML models: Decision Tree, Logistic Regression, and Transformer. After this brief introduction, we look at the main topic of this thesis, SHAP. To understand SHAP, it is important to know SHAP's history in game theory and the problems it solves. Because SHAP differs slightly from domain to domain, we will present you at the end of Section 3 the specific concepts of SHAP in NLP. In Section 4 we will introduce two datasets on which we will train our models, which serve the purpose of presenting the applicability of SHAP in NLP later in Section 5. Lastly, in Section 6 we will conduct a critical examination of SHAP, discussing the advantages and disadvantages, with the aim of providing guidance to future researchers.

Chapter 2

Fundamentals

2.1 Interpretable Machine Learning

As an integral component of the broader data science process, **IML** emphasizes the importance of using interpretations to extract insights from ML models. It is a process that involves deducing meaningful knowledge from an ML model about the relationships inherent in the data or those learned by the model itself [7].

This knowledge is considered valuable if it provides illumination on a selected problem for a specific audience. Insights derived from this process guide communication, actions, and discovery, and can be represented in different representation formats, including visualizations, natural language, and mathematical equations, based on the specific context and intended audience [7]. For example, a doctor diagnosing a patient through an X-ray scan would require qualitatively distinct information compared to a computer scientist assessing whether a text classifier exhibits racial discrimination[7].

2.1.1 Importance of Interpretability

Machine learning systems are being used in complex applications, leading to a growing interest in optimizing them not only for task performance but also for other important factors such as non-discrimination [8], safety [9], and the provision of the right to explanation [10]. To guarantee the secure utilization of ML systems, it is essential not only to meet performance

requirements, but also to adhere to these supplementary criteria. However, these present a challenge, as they are harder to quantify compared to performance measures such as accuracy [11].

ML systems may not always require interpretability, as in the case of recommendation engines, postal code sorting, and weather prediction models. All of these systems produce their outputs automatically without human intervention. There is no need for an explanation, either because the potential consequences of unacceptable results are not significant or because the problem has been extensively studied instilling trust in the system's decisions [11]. But when is it important to use interpretability and what are the benefits? Interpretability is essential in scenarios where there is an "incompleteness in the problem formalization" [11]. This means that when a problem could not be fully specified or there are gaps in its formulation, such as argument detection, interpretability is the way to go. The demand for interpretability and explanations is motivated, among others, by the following factors [11]:

- **Scientific Understanding**: Scientists' goal is to gain knowledge. A model that performs classification is not sufficient; as scientists, we desire to comprehend why it provides a specific output for a given input, and this helps us to strengthen our scientific understanding.

- **Ethics**: Humans might want to provide fairness in their models, but the human's notion of fairness might be too abstract to be fully formalized. Interpretability is a useful tool for finding biases in the model.

- **Safety**: If a machine learning model is used in safety-critical industries (e.g. military) where human intervention is possible, it is necessary that the model provides information to decision-makers about why a certain output was produced.

This illustrates that model performance is not always the only priority when it comes to model quality. In the presence of incompleteness, explanations are one way to make the gaps in problem formalization visible to us [11].

2.1.2 Taxonomy of Interpretability Methods

It is important to have a fundamental understanding of the taxonomy to be able to distinguish between the various interpretability techniques. Different criteria have been used to categorize interpretability approaches [12].

Intrinsic vs. Post hoc

Interpretability in machine learning can be achieved through two distinct approaches. Intrinsic interpretability involves achieving interpretability by limiting the complexity of the model itself. For instance, this can be achieved through the use of simple structures like short decision trees or sparse linear models. Post hoc interpretability relies on methods applied after the model's training to analyze and understand its decision-making process. In this case, an additional operation is performed after the prediction is made in order to effectively interpret the model [13].

Model-Specific vs. Model-Agnostic

Model-specific interpretation tools are designed to work only with specific model classes. For example, the interpretation of regression weights in a linear model falls under model-specific interpretation. It is worth noting that intrinsically interpretable models always fall into the category of model-specific interpretations, as their interpretability is inherently tied to their specific model structure. Model-agnostic interpretation tools perform on all kinds of model and are applied after the model has made a prediction (post hoc) [12]. To interpret any model, the model agnostic tools change the input to the model and monitor how the output changes.

Local vs. Global

Global methods provide a holistic understanding of the behavior of the model. These methods help you understand which features of the model contribute the most to the overall predictions of the model. One popular global method is the Partial Dependence Plot (PDP).

PDPs are graphical representations that show the marginal effect of one or two features on the predicted outcome of a machine learning model while keeping all other features constant [12]. However, local methods are used to explain individual predictions. A well-known local method is the Local Interpretable Model-Agnostic Explanations (LIME) [14]. LIME explains a prediction by fitting an easy-to-interpret surrogate model.

2.1.3 What is a Good Explanation?

The primary purpose of explanation in ML is to help humans understand the output of the model. To serve this purpose, it is essential to investigate the properties that an explanation should have to be understandable by humans. In a survey of publications on explanations [15], four discoveries were made in the field of explainable AI. Firstly, the survey emphasizes the significance of **contrastive** explanations in the context of AI, which focuses on clarifying why one event occurred instead of another. Secondly, it highlights the importance of **selectivity** in explanations, as individuals tend to choose a limited number of causes from numerous possibilities to provide explanations. Thirdly, the survey suggests that **causal explanations** are more effective than relying solely on probabilities and statistical relationships to improve understanding. Lastly, it underscores the **social nature** of explanations and, therefore, highlights the role of interaction and conversation between the explainer and the explainee. These findings emphasize the crucial contextual and interactive aspects of explanations, which contribute to the development of genuinely explainable AI systems in practice.

In order to evaluate which interpretability technique to use and how well they perform, it is worth taking a look at the properties of explanation methods [16].

Properties of Explanation Methods

Expressive Power is a term used to describe the variety of languages that can be used to express explanations. The language of the explanation could be propositional logic (IF-THEN rules), a histogram, natural language, or highlighted pixels in an image [12].

Translucency stand for the depth to which the model's internal workings are explored in an explanation technique. High translucency offers the advantage of utilizing a greater amount

of information from the model to generate explanations. In contrast, low translucency provides the advantage of a more portable explanation method, because they treat models as black boxes [12].

Portability describes how well the technique covers different models. Is the method limited to certain models like decision trees? Or can the method be applied to any kind of model?

The last important property is **Algorithmic Complexity**. If time is a relevant factor, you should consider explanation methods with a good algorithmic complexity.

2.2 Natural Language Processing

NLP is a multidisciplinary field that covers a wide range of techniques and methodologies that allow computers to effectively handle natural human language [17]. The main goal of NLP is to ensure that the meaning of a text is accurately represented. We want to keep the important information, while removing unnecessary noise. But there are some challenging problems which arise in text processing; to name some of them: nonstandard English, segmentation issues, homonyms, neologisms, and tricky entity names.

To transform text into a machine-readable format, one common approach is the Bag of Words (BOW) technique, where all the words in a text corpus are one-hot encoded, resulting in a large vector for each document. However, BOW has limitations, as it only provides a binary encoding indicating the presence of words and fails to capture their meaning. An improvement over BOW is the Term Frequency-Inverse Document Frequency (TF-IDF) method. In TF-IDF, the occurrence of a word in a document is weighted against its occurrence in the entire text corpus. This approach offers additional insight into the importance of each word in the vector, as it considers the relative frequency across the corpus. Despite these enhancements, both the BOW and the TF-IDF methods still do not fully capture the semantic meaning of each word. They focus mainly on encoding and frequency information rather than semantic understanding. Newer approaches like GloVe [18] and Word2Vec [19] utilize pre-trained word embedding to transform each word into a vector in the latent space which represents the meaning of each word. The latent space can be visualized by an example. Consider the vector representation of *King*, *Woman*, *Man*, by conducting vector arithmetic we can retrieve the vector corresponding to *Queen*.

$$King - Man + Woman = Queen$$

This exemplifies how word embeddings can reflect semantic relationships between words.

Pre-trained embeddings solve most of the problems we have with natural language by capturing the meaning of each word. But homonyms still pose a problem because we are not able to differentiate two words in different contexts. Because of this, ML models are not able to fully understand the meaning of a text. But with the rise of Transformer-based Models [20] and especially Encoder models like BERT [1] we are now able to represent a word in the context of a sentence.

2.2.1 Applications of NLP

In this thesis, we concentrate on two text classification domains in particular because they serve as examples of the broad range of NLP capabilities. These domains successfully highlight the essential conclusions of this thesis, which serve as surrogates.

Sentiment Analysis

Sentiment analysis is about figuring out the emotions in a text. The most straightforward approach involves categorizing emotions as negative, positive, or neutral. This involves inspecting the language and context to find the text's sentiment. The applications of sentiment analysis are diverse, such as studying markets, understanding customer feedback, and analyzing online conversations for arguments. To automate the classification of text into sentiment categories, machine learning algorithms are commonly used [21].

Argument Detection

Argument detection is about automatically identifying and classifying arguments within a text. It involves analyzing the text to recognize claims, evidence, and reasoning used to support or challenge a particular proposition. Recognizing and comprehending arguments are

crucial in different areas such as online discussions, debates, legal texts, and news articles. This task is highly relevant, as it helps to extract valuable insights from these sources of information [22]. Argument detection is a crucial component of the broader field known as Argument Mining, which seeks to fully understand arguments. An example of the application of Argument Mining is the creation of automatic debating systems such as IBM Debater [23].

2.3 Approaches to Text Classification

Text classification, which involves assigning predefined labels to text, is a fundamental task in numerous NLP applications [24]. It is suitable to differentiate between traditional machine learning models and Deep Learning methods [24]. In this context, traditional means statistic-based models such as decision trees, logistic regression, and support vector machines. In contrast, deep learning models consist of neural networks and have demonstrated improved performance compared to traditional models in areas such as speech recognition, image processing, and text comprehension [24].

2.3.1 Traditional Machine Learning Models

Traditional models are often inherently interpretable as they are simple statistical models. However, in text classification, we deal with many features (different tokens), making it difficult for humans to explain individual predictions just by looking at decision tree paths or function weights. In the following, we provide a concise introduction to two commonly used algorithms.

Decision Tree

A decision tree is a supervised learning model that builds a tree-like structure. Tree-based models divide the data into subsets by repeatedly applying thresholds to specific features. The final subsets are known as terminal or leaf nodes, whereas the intermediate subsets are known as internal or split nodes. These trees can be used for both classification and regression tasks

[12]. During the training phase, it uses the labeled data to build decision rules and paths. Once the decision tree is built, it can be used to classify unseen, unlabeled data by traversing the tree according to the feature values [24].

Logistic Regression

Logistic regression is a supervised learning algorithm primarily used for binary classification tasks and could additionally extend to multiclass classification [25]. In text classification, it can be used to predict class labels of textual documents by converting them into numerical representations, such as Bag-of-Words or TF-IDF, as discussed in the introduction 2.2. It employs the standard logistic function, which acts as a sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$ and maps the numerical representation of the text to a value between zero and one. In this thesis we use it for binary classification and in this case we can assign a class based on the output of this function, if the value is greater than 0.5, we classify the input as class 1, otherwise it belongs to class 0.

2.3.2 Transformer

Transformers comprise an encoder and a decoder, where the encoder builds a textual understanding from the input, and the decoder generates a new text. Initially developed to enhance text translation, this domain was previously dominated by Recurrent Neural Networks (RNNs). However, the revolutionary attention mechanism [20] demonstrated that attention is the only requirement that allows for significantly more parallelization and state-of-the-art results.

The **Encoder** consists of multiple identical layers and each layer is composed of two sublayers. The first sublayer is a Multi-Head Attention layer, and the second sublayer is a Feed-Forward network. The input of the Encoder is text, which is first transformed into input embedding and then with positional encoding passed to the six layers. The output is a precise and context-aware word embedding of the input text [20].

The **Decoder** is built similarly to the Encoder. It has three sub-layers; two from the Encoder, and additionally a multi-head attention layer over the output of the encoder. The objective

of the Decoder is to predict the next word. Therefore, the self-attention layer within the Decoder undergoes a modification to ensure that it does not attend to the following positions. This, combined with the offset of the output embeddings, helps to facilitate effective word prediction within the Decoder [20].

Attention Mechanism

The introduction of Attention, as initially proposed by [26], addresses the limitations of RNN Encoder-Decoder models in machine translation. This mechanism allows the model to dynamically search for relevant parts of the source sentence, eliminating the need for a compressed vector and improving the interaction between the encoder and decoder.

The Attention mechanism used in the transformer architecture [20] significantly transforms the encoder, decoder, and decoder-encoder interaction, effectively replacing the reliance on recurrent neural networks (RNNs). This paradigm shift is accomplished through three distinct components:

- **Self Attention**: A mechanism that assesses the significance of different positions within an input sequence by establishing connections among them.
- **Masked Self-Attention**: This is a variant of Self-attention that masks future words, this is used in the decoder.
- **Multi-Head Attention**: Every head pays focuses on different relations in the sentence and the results of all heads are combined, this enables the transformer to capture a wide range of information and learn complex relationships.

Transfer Learning by Pretraining

In the following, many models were presented based on the novel Transformer Architecture [2, 1, 27]. By leveraging specific parts of the Transformer, they revolutionized the world of Large Language Models (LLM). They are pre-trained on an arbitrary task on huge datasets with enormous computing capacity to gain language understanding and can be fine-tuned on domain-specific tasks like translation, QA, and classification. These models achieve in

a short training time [20] state-of-the-art results in numerous fields [2, 1, 27]. This is done using transfer learning, where the most significant computational load is carried out during the pretraining phase.

BERT

One of these models is BERT, the Bidirectional Encoder Representation Transformer. It uses only the Encoder of the Transformer. The Encoder does a good job of understanding language; therefore, the main objective of BERT is to understand language and produce meaningful text representations by chaining at least 12 Encoders ($\text{Bert}_{\text{BASE}}$) [1]. The ability of BERT to capture contextual information and generate high-quality representations of text has led to its widespread adoption in various fields. However, understanding and interpreting the predictions made by BERT models can be challenging due to their complex architecture.

BERT is pre-trained on 2 tasks. The first task is a Masked Language Model, where BERT's goal is to predict the masked word. The second task is "Next Sentence Prediction," where given a Sentence B, BERT has to decide if it immediately follows Sentence A. The intention is not to achieve high accuracy in the tasks; it is almost impossible to predict the masked word accurately. Instead, the intention is to tune the lower layers of the model in order to gain a rich understanding of language. One could say that language understanding is a byproduct of arbitrary training tasks [1]. After pre-training, the model is typically fine-tuned on a specific task. One such task is text classification.

BERT for Text Classification

The state of the art performance can be achieved for various tasks, including text classification, argument mining, and question answering, by fine-tuning the pre-trained BERT model with just a single additional output layer [1]. In the context of text classification, fine-tuning involves adjusting the additional output layer parameters to better fit the specific task. This is achieved by continuing the training process on the task-specific data. The additional output layer is commonly referred to as the classification head. This layer is typically a fully connected layer that takes the output of the BERT model and generates a final output that

corresponds to the classes in the classification task [28].

Chapter 3

Shapley Additive Explanations (SHAP)

To understand the applications of SHAP in NLP, we begin by looking at the evolution of SHAP and its previous uses. This provides a connection between the Shapley values derived from game theory and their role in achieving explainable AI for text-based ML models.

3.1 Evolution of SHAP

3.1.1 Shapley Values

In 1953, the game theorist Lloyd Shapley introduced the Shapley value, which serves as a solution to a common problem in cooperative games: how to fairly distribute the game's rewards among the players [29]. The idea behind this concept is to measure each player's contribution to the outcome of a game and fairly distribute a reward among a set of players. This is achieved by considering the impact of a player in every possible coalition on the result (*marginal contribution*). Imagine a group of pirates lifting a heavy stone with a treasure hidden below it, each pirate's Shapley value determines their contribution to lifting the stone. By evaluating how the group performs with and without a specific pirate, the Shapley value helps to distribute the treasure fairly among the pirates.

Although the Shapley value was originally introduced by Lloyd Shapley, we will explore a refined and more intuitive definition [30]. In the upcoming sections, we will present the fundamental building blocks of this definition, resulting in Shapley values.

To begin with, the framework where the players $N = \{1, 2, ..., n\}$ contribute to the outcome is a *cooperative game*. A *cooperative game* is a function:

$$v\colon 2^N \to \mathbb{R}$$

with $v(\emptyset) = 0$. The game v maps a group of players S to an output, which represents the total payoff $v(S)$ of that coalition [30].

To construct all possible groups of players, the most intuitive way is to define Π as the Set of all *permutations* on N. Let $\pi \in \Pi$ be one possible *permutation*. Each player i is only part of a coalition with the players preceding him in that particular order π. Therefore Let

$$p_\pi^i = \{j : \pi(i) > \pi(j)\}.$$

be the Set of players preceding player i in order π [30].

The **difference** between the total payoff of the coalition consisting of all players preceding i and i and the coalition p_π^i is called the *marginal contribution* of the player i with respect to that order π

$$v(p_\pi^i \cup i) - v(p_\pi^i).$$

Now, if *permutations* are randomly chosen from Π , with equal probability for each one of the $n!$ *permutations*, then the *average marginal contribution* of player i in the game v is

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(p_\pi^i \cup i) - v(p_\pi^i)]$$

Which is the Shapley value of player i. $\phi_i(v)$ is the i's player contribution to the outcome of the game [30].

Lloyd Shapley demanded some properties [29] that he said must be fulfilled for a fair distribution in a cooperative game. These properties are given by four axioms, and the Shapley values represent the singular solution that meets all four axioms [31].

1. **Efficiency** means that the sum of Shapley values should add up to the total payout. This implies that the Shapley values are in the same unit as the payout. Consider the pirate example; each of the pirate's payouts should sum up to the total value of the treasure.

2. **Symmetry** dictates that if two players are equal in their contributions, they must receive the same Shapley value. In the pirate example, two identically built pirates should receive the same share of the treasure.

3. **Dummy** states that a player who is not contributing at all to the game should receive a Shapley value of 0. A pirate with broken arms would not contribute to lifting the stone and therefore would receive no share of the treasure.

4. **Additivity** asserts that if you have two games v_1 and v_2 then you can sum these two and receive a new game with two value functions $\phi_i(v_1+v_2) = \phi_i(v_1)+\phi_i(v_2)$. Consider the scenario where the pirates find a second treasure. This axiom states that it is irrelevant if the individual payout is calculated after each treasure or calculated at once.

Following that, the Shapley values have established themselves as a fundamental concept within cooperative game theory, finding applications in diverse domains such as political science, economics, and computer science [32].

3.1.2 Shapley Values for Machine Learning

Within the context of machine learning models, players are represented by the input features, while the payout corresponds to the model output. Shapley values, in this scenario, assign an importance score to each feature of the input, indicating which feature contributed the most to that specific output. The **efficiency** axiom comes especially handy because it ensures that the Shapley values are in the same unit as the model's output.

One challenge associated with Shapley values is their exponential time complexity. As a

solution, various estimation methods have been developed to approximate Shapley values while maintaining the desired properties. The first approximation methods were introduced in 2010 and are about sampling-based approximation [33, 34].

3.1.3 Additive Feature Attribution Methods

Lundberg and Lee (2017) introduced a framework called SHAP (SHapley Additive exPlanations) [35]. One contribution of the SHAP paper is that they identified a new class of explanation models, the **additive feature attribution methods**, and associated properties.

Additive feature attribution methods are a class of explanation methods used to explain individual predictions of a machine learning model. They approximate the model's output by using a simpler model which is easy to interpret. These methods allocate an importance value ϕ_i to each feature i, indicating its contribution to the model prediction [35].

Definition 1 Additive feature attribution methods

$$g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x_i'$$

The components of the formula can be understood as follows: The output of $f(x)$ for a specific instance x is approximated by a simpler model $g(x')$. The input to g is x' it maps to the original input by $x = h_x(x')$, is a binary vector of length M and corresponds to the total number of features in the dataset. If a feature is null or simply not present in the instance x it is set to 0 otherwise it is set to 1 in x'. We approximate the model's output by summing over all feature attributions $\phi_i \in \mathbb{R}$. By multiplying ϕ_i with x_i' only the features that are present in the sample are considered. The base prediction or the average prediction across all instances of f is denoted as $\phi_0 \in \mathbb{R}$ [35, 12].

The following **Six** existing explanation methods utilize this explanation model:

- Local Interpretable Model-agnostic Explanations (LIME) [14]
- DeepLIFT [36, 37]

- Layer-Wise Relevance Propagation [38]
- Three classic Shapley Value estimation methods [34, 39, 40]

Besides the definition of the class, Lundberg and Lee also defined desirable properties which indicate the quality of the explanation technique [35].

Property 1 (Local Accuracy) The first property is local accuracy. It means that the output of the explanation model $g(x')$ should match the output of the model itself $f(x)$ [35].

$$f(x) = g\left(x'\right) = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

This property should sound familiar to you as its based on the **Efficiency** Property in game theory. The model's output corresponds to the total payout of the game and the explanation model corresponds to the sum of all Shapley values.

Property 2 (Missingness) The missingness property implies that a feature that is absent (represented by a value of 0) receives an attribution of zero. This prevents them from affecting the explanation [35].

$$x'_i = 0 \Longrightarrow \phi_i = 0$$

Although this property is not derived from game theory, it is needed because in additive feature attribution methods not all possible features are present in each sample. Unlike the cooperative games where all the players attend the game.

Property 3 (Consistency) Consistency ensures that if a model changes and, as a result, the contribution of a feature to the model increases or remains the same when other features are unchanged, its assigned Shapley value should never decrease, preserving a meaningful interpretation of feature importance during model changes [35]

Let $f_x(x') = f(h_x(x'))$ and $x' \backslash i$ indicates that feature i is absent in sample x, $x'_i = 0$. The contribution of a feature i can be formalized as

$$f_x\left(x'\right) - f_x\left(x'\backslash i\right)$$

For any two models f and f' if

$$f'_x\left(x'\right) - f'_x\left(x'\backslash i\right) \geq f_x\left(x'\right) - f_x\left(x'\backslash i\right)$$

for all inputs $x' \in \{0,1\}^M$, then $\phi_i\left(f',x\right) \geq \phi_i(f,x)$. This property satisfies **Dummy**, **Symmetrie**, and **Additivity** from game theory [35] .

3.1.4 SHAP a Unified Approach

The properties **Local Accuracy** and **Consistency** are based on the four axioms (3.1.1) in game theory. As demonstrated by [31], the Shapley values are the unique solution to achieve fair distribution in a cooperative game. As a result of this, Lundberg and Lee [35] have established that additive feature attribution methods, based on Shapley values, represent the only viable option. This results in a **unification** of all six explanation methods into one framework, SHAP (SHapley Additive exPlantions).

SHAP values are a measure of feature importance. They are Shapley values of a conditional expectation function of the original model [35]. SHAP values attribute contributions to features, revealing the impact on the model's prediction as each feature is considered. These values clarify the transition from the base value, denoted as $E[f(z)]$ previously described as ϕ_0, which is the average prediction across all instances of f, to the actual output, $f(x)$. This process is illustrated in Figure 3.1 and Figure 3.2

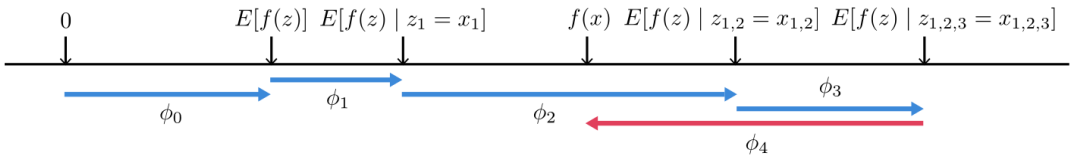


Figure 3.1: SHAP explanation for a single sample x visualized $E[f(z)]$ is the base value of the model $f(x)$ is the prediction of sample x [1]

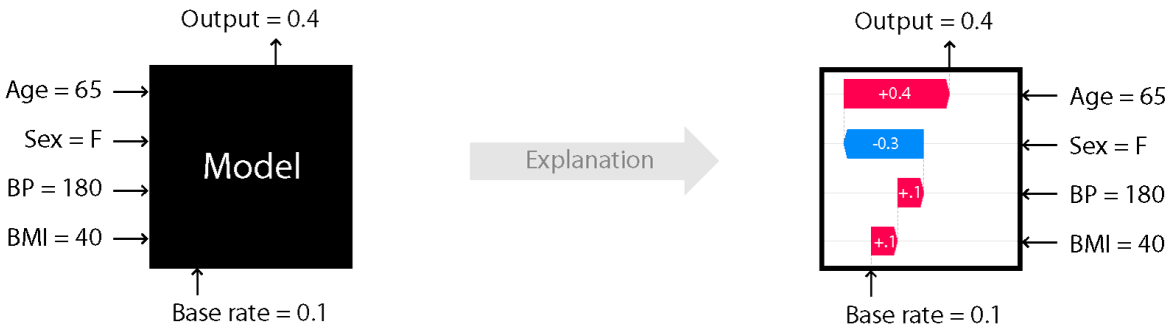


Figure 3.2: SHAP treats the model as a black box and produces explanation by visualizing the SHAP values in the form of plots (in this case *waterfall plot*)[2]. A feature has a red bar if the SHAP value is positive, indicating a positive influence on the model output. The blue bar indicates the opposite scenario. The importance of a feature for the prediction increases with the length of its corresponding SHAP value bar.

The paper not only unifies six existing explanation methods under SHAP but also addresses the primary challenge of Shapley values, the algorithmic complexity. It introduces a diverse range of estimation methods known as Explainers [35] that offer various approaches to understanding the behavior of the model.

On the one hand, we have model-agnostic Explainers, such as the Kernel Explainer, which is a kernel-based estimation method for SHAP values inspired by local surrogate models like LIME [14]. This Explainer enables the estimation of SHAP values for any model, making it highly versatile in its applicability. On the other hand, there are also model-specific explainers like the Linear Explainer, which leverage knowledge about the underlying model to calculate SHAP values more efficiently and accurately.

The authors of SHAP published an open source Python library called shap [3], which offers a wide range of Explainers and plots that enable the visualization of local and global explanations. When you use the shap library, you don't have to worry about Explainers. The default setting is 'auto', which smartly picks the best option for you, making the process easy and efficient. These and other reasons, such as the theoretical basis, contributed to the huge popularity of the SHAP framework up to the present time [12].

[1]Image source: [35]
[2]Image source: (https://shap.readthedocs.io/en/latest/index.html)
[3]https://github.com/shap/shap

To avoid confusion on this topic, we have provided you with a list of naming conventions established in the community and that we will adhere to throughout this thesis [41].

- **Shapley values**: Game theory method to fairly distribute the reward of the payout among the players based on their performance.
- **SHAP**: Using Shapley values for interpreting ML predictions.
- **SHAP values**: Values returned by SHAP.
- **shap**: Implementation of SHAP in Python.

Although SHAP has gained widespread usage for explaining tabular data, its potential application in NLP has not been comprehensively explored. In the following, we want to demonstrate how you can use SHAP to gain insight into NLP models.

3.2 SHAP in Natural Language Processing

SHAP values are the contribution of features to a model outcome. However, text datasets pose a unique challenge, as text is typically transformed into a high-dimensional vector representation. This raises the question: What exactly are the features in a text dataset? And is there a special explanation for text data?

3.2.1 Partition Explainer

The problem with high-dimensional data, like text, is that it becomes highly computationally intensive to calculate all possible coalitions. Therefore, different estimation methods have been developed over time to make the calculation of SHAP values more efficient. The default estimator that SHAP uses for NLP is the **Partition Explainer**. The Partition Explainer is an efficient method used for text-based models to calculate Owen values [42], which are similar to Shapley values but consider the formation of coalitions among players (features). Unlike other estimators like Kernel-Explainer or Sampling-Explainer, the Partition-Explainer exhibits a quadratic exact runtime with a balanced partition tree, making it computationally

more tractable for high-dimensional data. This is achieved by grouping the features using hierarchical clustering [43, 44]. By adopting this approach, we can significantly reduce the number of computations required, focusing only on a smaller subset of the original coalition.

3.2.2 Masker

Text-based models rely on tokens as input, which can vary in form, such as vector representation through Tf-IDF or learned embeddings (as discussed in Section 2.2). These tokens are created by tokenization, a preprocessing step in NLP. With simple models, it is beneficial, for the analysis with SHAP, to define a pipeline that concludes the tokenization and the model. The entire pipeline would be a black box. When calculating SHAP values, we focus solely on the input (in this case, the text input) and the corresponding model output. In game theory language, the input text is considered the *team* and the model output represents the *payout*. The players in this team can be individual characters, words, or sentences, offering flexibility to adjust the granularity as needed. This can be done with the **Masker**. The **Masker** has its own **Tokenizer**, which is applied before the text is passed to the black box. This allows the **Masker** to operate independently of the underlying model and provides a valuable explanation on a custom level of granularity.

If the underlying model is a Transformer like BERT then, by default, the **Tokenizer** is set to the same granularity as the tokenizer used in the Transformer, resulting in a player definition on the granularity defined by BERT. The BERT tokenizer splits words into subwords, leading to fewer tokens needed to represent a corpus, a word like *surfing* would be split into *surf* and *##ing*. Although this provides additional valuable information, it is important to keep in mind that we are not restricted by this and can still change the **Tokenizer** to our preferred granularity, because in most cases a word-based player definition is sufficient 3.1.

To handle the absence of players in the text setting, the **Masker** plays a vital role. Suppose that you define the granularity by word. In that case, the output of the model is calculated for all necessary coalitions of players. Although these coalitions are still represented as text, players not in a specific coalition are excluded, allowing for various exclusion methods [45]. The model interacts with the Masker as follows: The raw text is fed to the **Explainer**, for example: "I like action movies". Initially, the **Explainer** generates a set of masked versions

of the input text. Each of the versions now has a different subset of masked words. In this case, the masker splits the text by whitespace, resulting in a masked text like "... ... action movies". The **Explainer** then passes the text to the model to make predictions for each of the masked versions of the input text. This process is illustrated in Figure 3.3 and in Listing 3.1.

```
# Masker splits text by whitespace
masker = shap.maskers.Text(tokenizer=r"\W+")
explainer = shap.Explainer(model, masker=masker)
shap_values = explainer(["I like action movies"])
# visualize the first prediction's explanation
shap.plots.waterfall(shap_values[0, :, "POSITIVE"])
```

Listing 3.1: An example code demonstrating the use of the shap library to calculate SHAP values for a text sample and visualizing them with a plot.

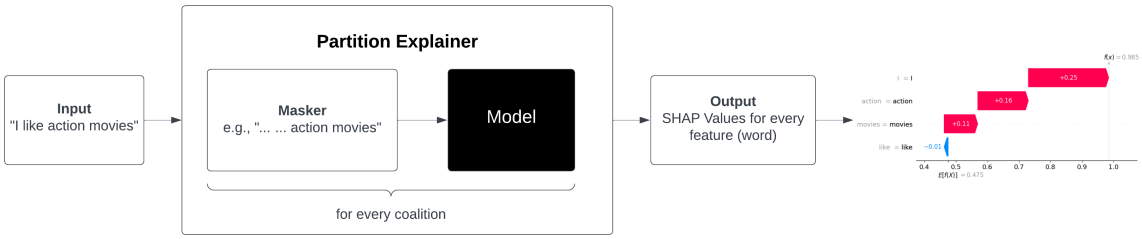


Figure 3.3: The process of extracting SHAP values from text involves passing the text to the Partition Explainer. The resulting SHAP values for every feature can be visualized using a plot.

3.2.3 Explanations

SHAP is a framework for model agnostic post hoc local interpretability [35]. Thus, the objective is to explain certain predictions. If you are interested in why a prediction performed well or poorly or if you are simply interested in the focus of the model, SHAP can help. However, it cannot provide a global explanation of how the model works internally. But, by accumulating a certain subset of local explanations, you can investigate certain trends of

the model. For text classification, you can examine the most important words or the least important words to get a glimpse of what the model's focus is.

3.2.4 Visualization

The SHAP framework not only calculates feature importance but also provides a class of plots to visualize the values, enhancing interpretability. There are two categories of plots offered by the framework: local plots and global plots. Local plots offer insights into specific predictions. The local *bar plot* starts at 0 and illustrates how each word contributes to a particular prediction. Another similar plot is the *waterfall plot*, which begins with the base prediction (prediction for empty string) of the model and demonstrates how the features (words) of the current sample influence the direction of the prediction 3.4. A text-specific local plot is the *text plot* 3.4. It combines a *force plot* where all negative values are on the right-hand side and positive values are on the left-hand side, with the point of contact representing the model's prediction. Below the *force plot*, the raw text is highlighted with colors corresponding to the SHAP value. Red indicates positive values, while blue indicates negative values. This plot provides an overview of all SHAP values for the text and allows for in-depth analysis of specific words. However, the global plots provided by SHAP are very useful for tabular data, but are not particularly useful for text data.

While the visualizations provided by SHAP are great, it is also possible to create custom plots if you want even more insight into the model. In this thesis, we demonstrate a custom global graph to visualize the words with the highest SHAP values inside a box plot to gather even more information, instead of just showing the mean of each of the top words 5.6. Additionally, we implemented a local plot. We have found that when you have longer texts, it is difficult to immediately explain a prediction with the text plot because there is too much information. Therefore, our local plot highlights the top words in a sentence 5.2.

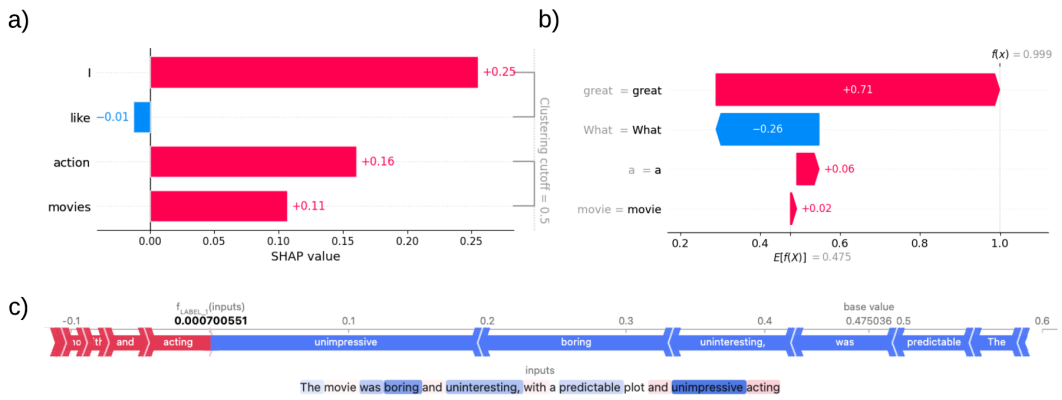


Figure 3.4: a) Bar plot b) Waterfall plot c) Text plot

3.2.5 Usage of SHAP for NLP Models: Literature Review

Since the publication of the SHAP framework in 2017, it has been widely used in NLP models, providing valuable insights. During the Covid-19 pandemic, researchers used SHAP to enhance model interpretability, resulting in higher trust and increased sharing of COVID-19 information with SHAP explanations, which could combat misinformation and build public trust [46]. Another study explored arguments on Twitter, using SHAP claim-specific keywords to accurately identify tweets that support or oppose specific claims [47]. Furthermore, SHAP is used to compare different models and investigate the reasons behind variations in model performance [48]. Moreover, SHAP helps us to understand the "personality" of models and aids in debugging them with greater depth [49].

Additionally, numerous SHAP-based variants have emerged [50]. Many of these variants are applicable to NLP and extend or enhance SHAP in various ways, such as improved efficiency, different scopes, different models, or different inputs. One notable variant is Neuron Shapley [51], a new framework based on SHAP that quantifies the impact of individual neurons on the prediction of a deep network. This framework specifically focuses on deep neural networks (DNNs) and provides a different type of explanation, no longer limited to local feature attribution. Another method called HEDGE [52], as mentioned in the review, is particularly

suitable for long text explanations. It uses a hierarchical explanation, which contains more information compared to non-hierarchical SHAP [50].

As you can see, SHAP is widely used in NLP, and the impact of SHAP extends beyond the framework itself. It paves the way for many SHAP-based approaches with the same goal of improving the interpretability of machine learning models.

Chapter 4

Implementation

In this section, we trained three text-based models on two distinct datasets. The primary motivation behind this was not to achieve state-of-the-art results within the models but to cover a wide range of different NLP models. These are used to showcase the applicability of SHAP in the next chapter, where we interpret these models.

4.1 Datasets

Within the scope of this thesis, we leveraged two datasets for our analysis. The first is the IMDb dataset, providing a foundation for sentiment analysis. The second dataset the IBM Debater: Evidence Sentences, is utilized for detecting evidence and supporting arguments.

4.1.1 IMDb

To perform sentiment analysis, we used the Large Movie Review Dataset, also known as the IMDb dataset. This dataset contains 50,000 reviews, which are evenly split into training and testing sets. Each review consists of text and a label, indicating whether it is positive or negative. Additionally, it was labeled using a heuristic regarding the stars of the review. A positive review has 7 or more stars out of 10, while a negative review has 4 or fewer stars out of 10. The next chapter contains, among others, the question of whether this heuristic is

a good choice or not. Furthermore, the dataset includes no more than 30 reviews per movie [53].

4.1.2 IBM Debater: Evidence Sentences

The Project Debater developed an automatic debating system [23]. To accomplish this task, several datasets regarding argument mining were labeled, one of which is the *Evidence Sentences* dataset. This dataset contains 5785 annotated sentences from Wikipedia, each assigned a label of *evidence* or *no-evidence* The annotation process involved 10 crowd-sourced labelers assigning labels to topic-sentence pairs based on majority agreement with a 50/50 tie considered *no-evidence*. To receive a positive label, each sentence had to meet three criteria: support or contest the topic, be coherent and capable of standing mostly on its own, and be convincing with evidence to influence someone's stance on the topic [54].

4.2 Preprocessing

We performed several preprocessing steps on the **IMDb dataset** to prepare it for analysis. Initially, we removed HTML tags from the text. Next, we converted the text to lowercase to maintain consistency throughout the dataset. Following this, we resolved contractions. Contractions are shortened words like *you're*, *i'm*, and *yall* which we resolved to their full form like *you are*, *i am*, and *you all*. After that, we removed all punctuation from the text. Afterward, we addressed instances of multiple consecutive spaces. As a result, the preprocessing yielded a unified text corpus that retains the original context and reduces the dimension of the vocabulary.

The **IBM Debater** dataset already includes preprocessing steps. Each sentence underwent a cleaning process to remove unnecessary formatting, except for footnote markers. The footnote markers were then replaced with a common token [REF] to indicate the source of the information. The main concept or topic of each sentence was hidden. This step was taken to enable the training of systems capable of identifying arguments regardless of the topic. The words in the sentence linked to the topic were masked using a common token TOPIC_CONCEPT. Finally, the sentences were subjected to several filters before being used

for annotation. They were required to have a length between 7 and 50 words, and additional manual rules were applied to ensure sentence coherence [54].

4.3 Model Building

For both NLP tasks, we constructed three models. Two of these are Decision Tree and Logistic Regression. These models were implemented using the scikit-learn libraries: *DecisionTreeClassifier* and *LogisticRegression*. Each model was incorporated into a pipeline which takes the input text and produces the corresponding predictions. To avoid code duplication, we developed a universal Pipeline Wrapper class for all simple models. This Wrapper offers a standardized and simplified interface, contributing to reduced code redundancy. The last model was a BERT model. The process of training the BERT model for the tasks proved to be more complex. Initially, we employed the pre-trained model "distilbert-base-uncased" [55], a smaller and faster variant of BERT, pre-trained on the same corpus through self-supervision, with the BERT base model serving as the teacher. Fine-tuning of the model was accomplished using the straightforward HuggingFace Trainer Interface [1].

4.4 Evaluation

We used the F1 score to receive the model's accuracy. As shown in Table 4.1, the Decision Tree model has the lowest F1 score, followed by the Logistic Regression model. The fine-tuned BERT model achieved the highest F1 score, indicating superior performance. Interestingly, we observed that sentiment classification outperformed Argument Classification overall.

[1] https://huggingface.co/docs/transformers/main$_c lasses/trainer$

	Dataset	
	Sentiment	Argument
Decision Tree	0.72	0.62
Logistic Regression	0.89	0.67
Finetuned BERT	0.93	0.70

Table 4.1: Comparing Accuracy of Decision Tree, Logistic Regression, and Finetuned BERT on sentiment and argument task

We now know which model performs best, but we do not have any information about the decision-making process of the models. In the next chapter, we will provide detailed insight into the decision-making process of these models with the help of SHAP.

Chapter 5

Interpreting NLP Models with SHAP

To gain a deeper insight into the models and better understand why a model makes certain predictions, we will analyze the models at various levels of granularity with the help of SHAP. This analysis will provide an overview of the possibilities of using SHAP in NLP tasks.

1. Local interpretation
2. Global interpretation
3. Investigating differences in model predictions

This analysis will help us understand why certain misclassifications occur and can contribute to improving future model classifications. Furthermore, we can provide a detailed explanation for each prediction, indicating how much each word contributed to the outcome. This will be useful for debugging the model.

5.1 Local Interpretation

You can utilize the local interpretability ability of SHAP to gain a deeper understanding of why certain classifications occur. This is especially interesting in cases of misclassification, as in our binary classification scenario. It is useful to investigate the most significant misclassification instances for each class to gain a rough understanding of why the model fails in certain situations.

5.1.1 Sentiment Analysis

To demonstrate this, we use a sample misclassified by logistic regression for Sentiment Analysis. The original label of the sample was *Positive*, but the logistic regression classified it as *Negative* with a probability of 0.98. Figure 5.1 shows the waterfall plot of the SHAP values. It is evident that of the top 9 SHAP values, 7 are negative. This observation alone indicates that the model prioritizes negative words and does not comprehend the true meaning of the review. Furthermore, as shown in Figure 5.2, you can observe the most significant positive and negative words for the model. In particular, one of the crucial parts of the review, *after the slow beginning and some of the soap opera antics i started liking it the plot was different than anything i had ever seen now*, went unrecognized. The shortcomings of this model become apparent when it misinterprets words like *bad*. In this instance, the term does not refer to a negative movie quality but rather expresses disappointment about the film's lack of funding. Additionally, the model should prioritize positive words such as *liking* and *entertaining*. This issue could stem from an excessive occurrence of the term *bad* in negative reviews, revealing a potential overrepresentation that undermines the model's understanding of varied contexts. Further investigation of all classifications involving the term *bad* revealed that only 0.5% of negative-labeled reviews containing the word *bad* were misclassified by the model. However, the model misclassified 25% positive-labeled reviews with the term *bad* supporting our initial assumption.

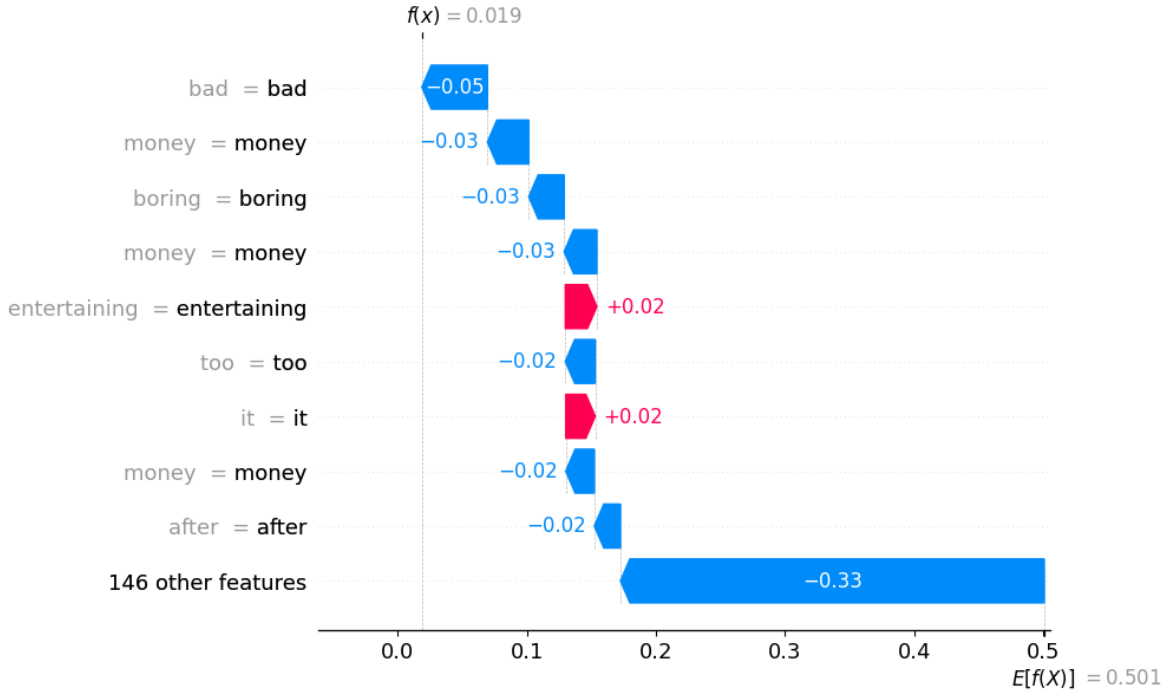


Figure 5.1: Waterfall plot, which begins at the Base value and concludes at the model prediction. This plot illustrates the impact of each feature on the model's prediction, showing whether each feature pulls the prediction in a positive or negative direction.

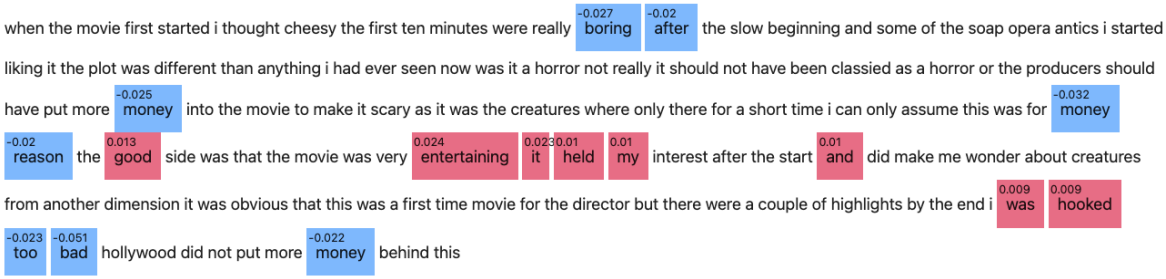


Figure 5.2: Custom highlight plot visualizing the most important negative and positive words based on SHAP values

The second sample was misclassified by BERT. The real label was *Positive*, but BERT classified it as *Negative* with a certainty of 0.99. This sample is especially interesting because BERT has such high accuracy but fails to classify it correctly. The waterfall plot in Figure 5.3 indicates that negative words, especially *whimper* and *worst*, dominate the classification. In

combination with the text plot Figure 5.4, we can see that the model correctly understood the negative and positive words, indicating a highly negative review. However, the original label is *Positive*. An interpretation could be that this review is from a fan who was disappointed by Laurel and Hardy's last movie but did not give a bad review (7/10). The underlying problem behind this misclassification is not the poor accuracy of the model; it is the poorly labeled dataset. As described before, the dataset was labeled using a heuristic where reviews above 6 were considered positive sentiments, and those below 5 were considered negative sentiments. However, in this case, the heuristic fails because overly critical fans may still rate a movie highly even if the review is full of negative sentiment words.

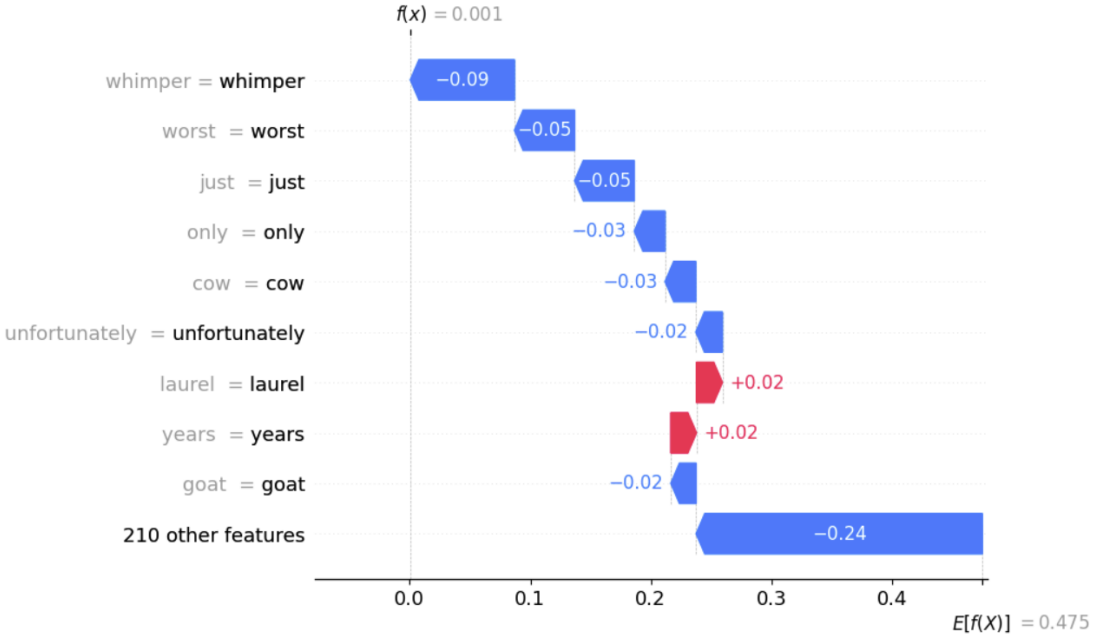


Figure 5.3: Waterfall plot highlighting the most important words *whimper* and *worst*

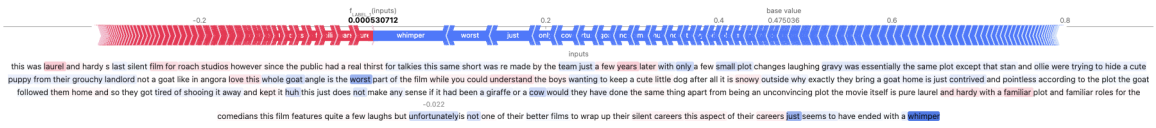


Figure 5.4: A text plot that colors each word based on its SHAP value. Darker colors show higher or lower SHAP values, making it easy to spot standout words.

5.1.2 Argument Mining

To illustrate the functionality of SHAP applied to argument mining, we selected a sample misclassified by the BERT model. Through this example, we aim to demonstrate how SHAP helps in understanding the model's decision-making process and identifying areas of improvement in argument classification.

death warns against such misplaced values and condemns the practice of censorship as well as demonstrating there can be value in a show often dismissed as juvenile and immature like south park or terrance and phillip ref

This sentence fulfills the criteria for a claim. Firstly, it takes a clear stance by stating that *death warns against such misplaced values and condemns the practice of censorship*. It indicates a viewpoint that opposes censorship and highlights the importance of avoiding misplaced values. Secondly, it is coherent and can stand on its own. It presents a complete idea related to death's warning and condemnation of censorship. And lastly, the sentence is convincing and attempts to sway someone's stance on the topic. By highlighting that even in shows considered juvenile, there can be value, it makes an argument that goes beyond a simple statement or claim. Based on these criteria, the sentence was labeled correctly. However, when investigating this classification using SHAP in Figure 5.5, it appears that the words indicating that it is a claim are not well recognized. For example, *condemns* and *practice* only receive SHAP values of 0.04, which might not be sufficient to confidently label the sentence as a claim. In summary, BERT currently fails to attribute the correct importance to the words that indicate a claim in this sentence. Additional training and optimization of the model's performance are required to address this issue, as its current performance is not up to the desired standard.

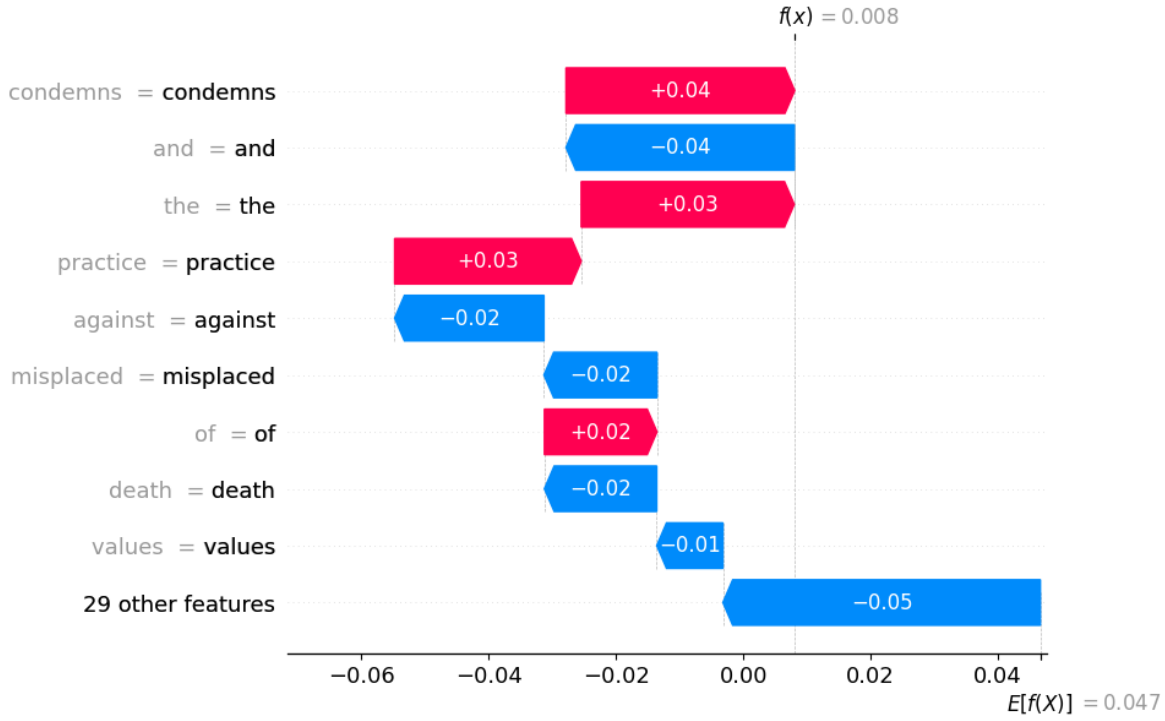


Figure 5.5: Text Plot

5.2 Global Interpretation

In addition to analyzing the local interpretation, a global analysis of the model can be performed by calculating the mean SHAP value for each word and presenting the top n words. This approach offers a concise summary of the most significant words that contribute to positive or negative classifications. In Figure 5.6, a custom boxplot was implemented to display the distribution of SHAP values. The boxplots on the left-hand side of the red dotted line represent words with negative SHAP values, indicating a contribution to negative sentiment. In contrast, the box plots on the right-hand side represent positive SHAP values, contributing to positive sentiment. On observing the plot, it becomes evident that negative words rarely cross the red dotted line. This indicates that the model consistently assigns negative SHAP values to these words whenever they appear. The same observation holds for positive words as well. This plot can be particularly useful for model-to-model comparison, providing a general understanding of how values are assigned to the top contributing words.

Moreover, the ability to generate plots that specifically target chosen words for a more detailed exploration is demonstrated in Figure 5.7. One word of particular interest is *that*, which is expected to serve as a valuable indicator for argument detection. Delving into how well the model assigns positive SHAP values to this word becomes intriguing. When analyzing the figure, it becomes evident that the model assigns a significantly high positive SHAP value to the word *that*. This finding suggests that *that* plays a crucial role as a strong indicator for claim sentences. Such insights contribute to a deeper understanding of real-world applications for argument detection, affirming the significance of this word as highlighted in the paper [56].

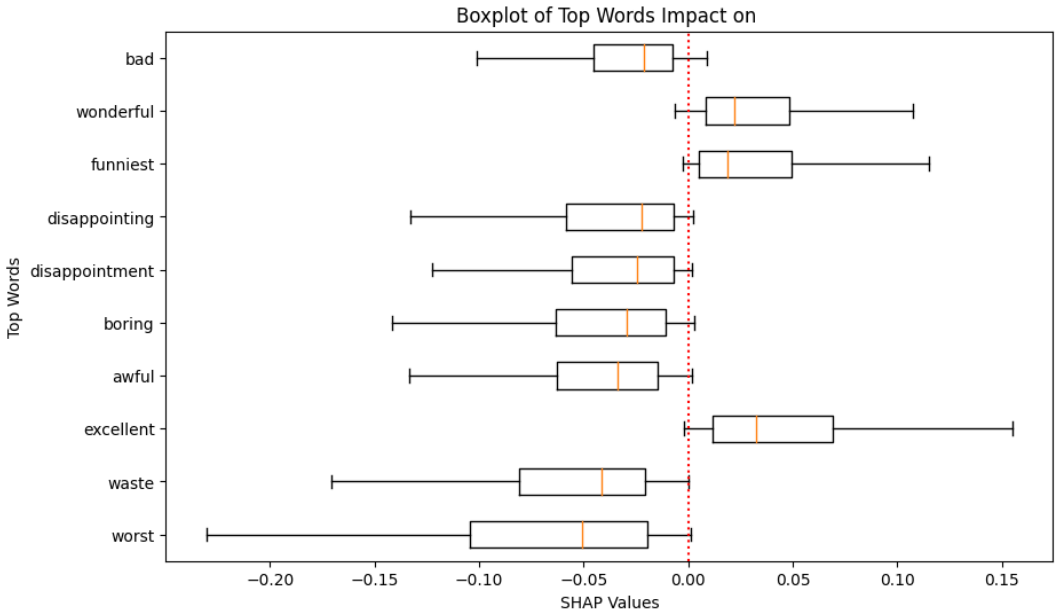


Figure 5.6: Words with the highest total mean SHAP value in sentiment analysis with logistic regression

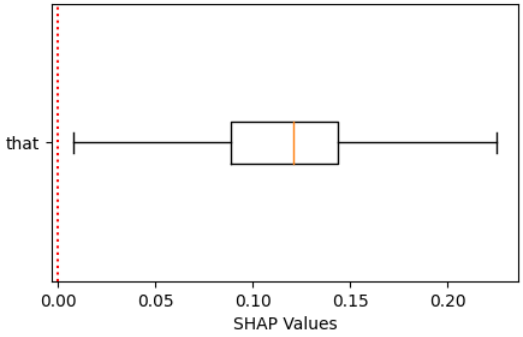


Figure 5.7: Boxplot of *that* in argument mining with Logistic Regression

5.3 Investigating Differences in Model Predictions

In many NLP tasks, we implement multiple models and compare them on the basis of their classification accuracy, as we have done in this paper. To further investigate the differences among the models, it could be helpful to select meaningful samples and analyze the SHAP values produced by the models. To find a meaningful sample, we sorted all predictions by difference among the models and found the sample in Table 5.1.

Model	BERT	Logistic Regression	Decision Tree
Class 0	0.0043	0.944	0.542
Class 1	0.996	0.056	0.458

Table 5.1: A representative instance displaying significant variations across the models. The accurate classification label for this sample is Class 1.

We visualized the different SHAP values in Table 5.2 The worst classification for this sample was made by logistic regression. This model does not understand that the reviewer is talking about the movie. Instead, it pays attention to positive words such as *best* and *pleasure* On the other hand, the best classification was achieved by BERT. It is evident that BERT understands that the rating of 7, in combination with 10, indicates a positive review. The SHAP value for the rating of 7 is 0.54. This is indeed a trash movie full of bad lines, but the reviewer liked the movie in the end, and BERT was the only model that successfully understood this.

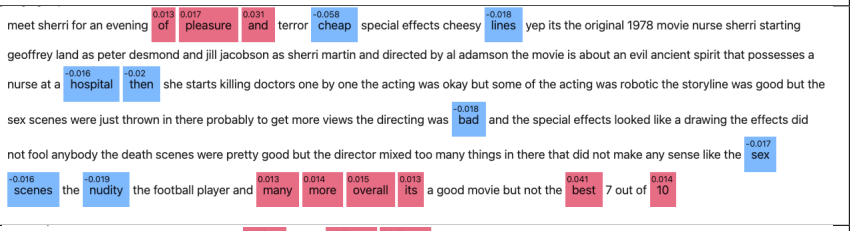
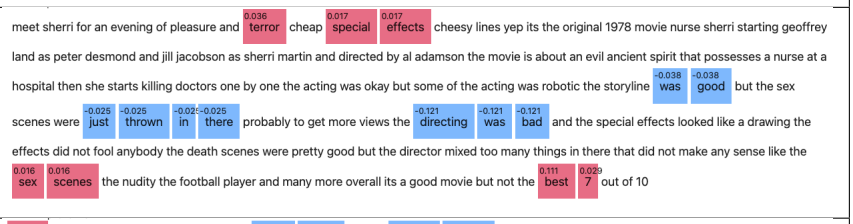
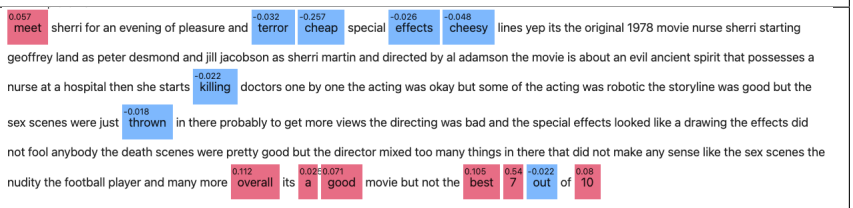
Highlight Plot	
Logistic Regression	
Decision Tree	
BERT	

Table 5.2: Comparison of classification results among Decision Tree, Logistic Regression, and BERT, illustrated through a custom highlight plot showcasing the most important SHAP values.

5.4 Conclusion

In summary, SHAP offers numerous possibilities to explore text-based models at various levels of granularity. We can delve into misclassifications and correct classifications, identifying the most influential words on the basis of their SHAP values. This allows us to further investigate the models' behavior based on these words. Moreover, we have the flexibility to investigate specific words of interest, comparing them with the expectations set by recent studies. Furthermore, by comparing the SHAP values of the same input across different models, we gain valuable insights into their strengths and weaknesses. This analysis sheds light on the focal points of various models, helping us to identify the underlying reasoning behind their outputs. Such investigations serve multiple purposes. On the one hand, they provide us with invaluable tools for debugging the model, enabling us to identify potential areas for improvement. On the other hand, the explanations derived from the SHAP values allow us to understand why the model arrived at a specific output, providing a solid foundation for

retraining the model or optimizing the training dataset.

Chapter 6

Advantages and Challenges of SHAP

In this chapter, we provide guidelines for future researchers on whether to use SHAP for text-based models or not. Following this objective, we will briefly summarize the advantages of SHAP and discuss the challenges and considerations that researchers should consider.

6.1 Advantages

One of the biggest advantages of SHAP is the theoretical basis in game theory, which allows us to generate highly valuable explanations that fulfill the properties of additive feature attribution methods 3.1.3. As discussed in 2.1.1, the primary purpose of explanation techniques is to help humans understand the reasoning behind the output of a model. It achieves this goal primarily by producing *contrastive* explanations. SHAP provides a clear and intuitive understanding of each feature's contribution to the model's output, by visualizing the effect of each token on the output. Moreover, SHAP offers a high degree of *expressiveness* by providing a wide range of plots and allowing customization of plots to suit the evaluator's specific needs. Additionally, the Partition Explainer is an efficient method to approximate SHAP values without being limited by the exponential runtime of the exact computation. The utilization of faster estimation methods allows SHAP to calculate multiple explanations efficiently, providing a comprehensive global explanation of the model's functioning. As demonstrated in 5, SHAP demonstrates remarkable *portability*, capable of being applied to various NLP tasks and working seamlessly with different machine learning models. By only perturbing the input features and watching the changes in the output, it is able to treat every

model as a black-box. This flexibility minimizes any extra overhead when switching between tasks and models, making SHAP truly model and task-agnostic, facilitating fast and straightforward explanations. As demonstrated in this thesis, SHAP's versatility and effectiveness make it a valuable tool for interpreting NLP models. Additionally, there are many SHAP-based techniques for NLP that evolve from the original framework, either extending the framework or creating a new model or task-specific explanations [50] and SHAP is a foundation for many of the recently proposed explanation techniques [52, 57, 58]. Furthermore, the python library *shap*[1], is open source, and during the time of writing this thesis, the repository exhibits a substantial number of contributions and active development, which is a positive indication for the future of SHAP.

6.2 Challenges and Considerations

The main challenge with using SHAP is the resource-intensive computation of SHAP values. Although the computation complexity was reduced by the introduction of the Partition Explainer, long text and complex models like Transformer still cause a problem for fast computation of SHAP values. We experienced this issue in particular when calculating the SHAP values for the Sentiment Analysis task using the BERT model. While the computation for one prediction by BERT is already slower than that of simpler models, it gets even worse when calculating SHAP values for a sample. Here we need to compute BERT's output for many coalitions; even if the total number of coalitions is reduced by the Partition Explainer, it still causes a problem. This problem becomes even worse with increasing input length. We experienced that this is negligible when you generate an explanation for one sample. However, if you are limited by time and require generating a global explanation, this is an unbearable problem. Another important aspect to consider is that SHAP values cannot be used for causal inference. Instead, they reveal how each feature impacts a model's prediction without directly showing its effect on the target variable. It is crucial to keep in mind that a model may not always precisely represent real-world phenomena, thus SHAP values do not provide direct insight into causation [59, 60]. This leads to another concern, where evaluators might misinterpret explanations and draw incorrect conclusions. This can result from confirmation bias or even malicious intent, making it a disadvantage for recipients of the explanations, as they cannot be entirely certain of the accuracy of the explanation [12, 60].

[1] https://github.com/shap/shap

Additionally, there are criticisms about existing SHAP-based methods that overlook uncertainties in explanations, cause unexpected changes in Shapley values, and attribute positive impacts to features that do not significantly affect the machine learning model [61]. Furthermore, it is essential to note that SHAP might not always accurately capture certain model behaviors due to its low translucency, lacking insight into the model's weights. In such cases, additional model-specific explanation techniques with higher translucency may be needed to complement SHAP explanations.

Chapter 7

Conclusion

This thesis has provided a comprehensive exploration of SHAP in the domain of NLP. By delving into the fundamentals of IML, NLP, and Shapley values, we have established a strong foundation for our research. Through the implementation of SHAP on two datasets with three different models, we gained valuable insights into the interpretability of NLP models. We demonstrated the model and task-agnostic nature of the SHAP framework, easily adapting it to interpret all models with minimal adjustments. Additionally, we showcased how SHAP can be extended to meet the specific needs of evaluators by creating custom plots and images for optimal explanations, proving that we are not confined to the boundaries of the framework. Our custom boxplot provided detailed insights into the distribution of the top words, while the highlight plot showed the possibility of creating concise and intuitively understandable plots, prioritizing the most important words based on SHAP values. This reduced cognitive overload, making the models understandable even to individuals with limited knowledge of NLP or IML. Moreover, we thoroughly discussed SHAP's strengths and weaknesses, allowing readers to understand its suitable scenarios and how to handle SHAP explanations effectively. This thesis serves as a bridge between SHAP and NLP, motivating further research and development in this area. As we move forward, future research could focus on novel explanation techniques for text data, improving computational complexity, introducing new plots, or enhancing existing ones to make explanations even more human-friendly. As the influence of AI continues to grow, we hope that this research inspires further exploration into IML, promoting the adoption of more transparent and reliable AI systems. During this thesis, I had the privilege of being a beta reader for the renowned expert in IML, Christoph Molnar. It was an honor to be mentioned in his new book [45].

Bibliography

[1] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[2] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[3] Ludovica Marinucci, Claudia Mazzuca, and Aldo Gangemi. “Exposing implicit biases and stereotypes in human and artificial intelligence: state of the art and challenges with a focus on gender”. In: *AI & SOCIETY* 38.2 (2023), pp. 747–761.

[4] Yuren Yang et al. “Interpretability analysis for thermal sensation machine learning models: An exploration based on the SHAP approach”. In: *Indoor air* 32.2 (2022), e12984.

[5] Karim El Mokhtari, Ben Peachey Higdon, and Ayşe Başar. “Interpreting financial time series with SHAP values”. In: *Proceedings of the 29th annual international conference on computer science and software engineering*. 2019, pp. 166–172.

[6] Julian Tritscher et al. “Evaluation of post-hoc XAI approaches through synthetic tabular data”. In: *Foundations of Intelligent Systems: 25th International Symposium, ISMIS 2020, Graz, Austria, September 23–25, 2020, Proceedings*. Springer. 2020, pp. 422–430.

[7] W James Murdoch et al. “Definitions, methods, and applications in interpretable machine learning”. In: *Proceedings of the National Academy of Sciences* 116.44 (2019), pp. 22071–22080.

[8] Nick Bostrom, Eliezer Yudkowsky, and K Frankish. “The Cambridge handbook of artificial intelligence”. In: *The ethics of artificial intelligence. Frankish, K., Ramsey, WM (eds.), Cambridge University Press, Cambridge, UK* 316 (2014), p. 334.

[9] Clemens Otte. “Safe and interpretable machine learning: A methodological review”. In: *Computational intelligence in intelligent data analysis* (2013), pp. 111–122.

[10] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a “right to explanation””. In: *AI magazine* 38.3 (2017), pp. 50–57.

[11] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[12] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. URL: https://christophm.github.io/interpretable-ml-book.

[13] Marina Danilevsky et al. “A survey of the state of explainable AI for natural language processing”. In: *arXiv preprint arXiv:2010.00711* (2020).

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “" Why should i trust you?" Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[15] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial intelligence* 267 (2019), pp. 1–38.

[16] Marko Robnik-Šikonja and Marko Bohanec. “Perturbation-based explanations of prediction models”. In: *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent* (2018), pp. 159–175.

[17] Ekaterine Kochmar. *Getting started with Natural Language Processing: A friendly introduction using Python*. Manning Publications, 2022.

[18] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[19] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[20] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[21] Basant Agarwal et al. “Sentiment analysis using common-sense and context information”. In: *Computational intelligence and neuroscience* 2015 (2015), pp. 30–30.

[22] Ruty Rinott et al. “Show me your evidence-an automatic method for context dependent evidence detection”. In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 440–450.

[23] Noam Slonim et al. “An autonomous debating system”. In: *Nature* 591.7850 (2021), pp. 379–384.

[24] Qian Li et al. “A survey on text classification: From traditional to deep learning”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13.2 (2022), pp. 1–41.

[25] Maher Maalouf. “Logistic regression in data analysis: an overview”. In: *International Journal of Data Analysis Techniques and Strategies* 3.3 (2011), pp. 281–299.

[26] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[27] Mike Lewis et al. “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension”. In: *arXiv preprint arXiv:1910.13461* (2019).

[28] Chi Sun et al. “How to fine-tune bert for text classification?” In: *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*. Springer. 2019, pp. 194–206.

[29] Lloyd S Shapley et al. “A value for n-person games”. In: (1953).

[30] Eyal Winter. “The shapley value”. In: *Handbook of game theory with economic applications* 3 (2002), pp. 2025–2054.

[31] H Peyton Young. “Monotonic solutions of cooperative games”. In: *International Journal of Game Theory* 14 (1985), pp. 65–72.

[32] Stefano Moretti and Fioravante Patrone. “Transversality of the Shapley value”. In: *Top* 16 (2008), pp. 1–41.

[33] Erik Strumbelj and Igor Kononenko. “An efficient explanation of individual classifications using game theory”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 1–18.

[34] Erik Štrumbelj and Igor Kononenko. “Explaining prediction models and individual predictions with feature contributions”. In: *Knowledge and information systems* 41 (2014), pp. 647–665.

[35] Scott M Lundberg and Su-In Lee. “A unified approach to interpreting model predictions”. In: *Advances in neural information processing systems* 30 (2017).

[36] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *International conference on machine learning*. PMLR. 2017, pp. 3145–3153.

[37] Avanti Shrikumar et al. "Not just a black box: Learning important features through propagating activation differences". In: *arXiv preprint arXiv:1605.01713* (2016).

[38] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[39] Stan Lipovetsky and Michael Conklin. "Analysis of regression in game theory approach". In: *Applied Stochastic Models in Business and Industry* 17.4 (2001), pp. 319–330.

[40] Anupam Datta, Shayak Sen, and Yair Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems". In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 598–617.

[41] Christoph Molnar. *Interpreting Machine Learning Models With SHAP. A Guide With Python Examples And Theory On Shapley Values*. 1st ed. 2023, pp. 12–13.

[42] Guilliermo Owen. "Values of games with a priori unions". In: *Mathematical economics and game theory: Essays in honor of Oskar Morgenstern*. Springer. 1977, pp. 76–88.

[43] URL: https://shap.readthedocs.io/en/latest/generated/shap.explainers.Partition.html.

[44] Lilo Wagner. *Shap's Partition Explainer for Language Models*. https://towardsdatascien com/shaps-partition-explainer-for-language-models-ec2e7a6c1b77. Accessed on 24th July 2023 at 17:00. 2023.

[45] Christoph Molnar. *Interpreting Machine Learning Models With SHAP. A Guide With Python Examples And Theory On Shapley Values*. 1st ed. 2023.

[46] Jackie Ayoub, X Jessie Yang, and Feng Zhou. "Combat COVID-19 infodemic using explainable natural language processing models". In: *Information Processing & Management* 58.4 (2021), p. 102569.

[47] Muhammad Mahad Afzal Bhatti, Ahsan Suheer Ahmad, and Joonsuk Park. "Argument Mining on Twitter: A Case Study on the Planned Parenthood Debate". In: *Proceedings of the 8th Workshop on Argument Mining*. 2021, pp. 1–11.

[48] Rosario Catelli, Serena Pelosi, and Massimo Esposito. "Lexicon-based vs. Bert-based sentiment analysis: A comparative study in Italian". In: *Electronics* 11.3 (2022), p. 374.

[49] Nadhila Nurdin et al. "Explainable Artificial Intelligence (Xai) Towards Model Personality in Nlp Task". In: *IPTEK The Journal of Engineering* 7.1 (2021), pp. 11–15.

[50] Edoardo Mosca et al. "SHAP-based explanation methods: a review for NLP interpretability". In: *Proceedings of the 29th International Conference on Computational Linguistics*. 2022, pp. 4593–4603.

[51] Amirata Ghorbani and James Y Zou. "Neuron shapley: Discovering the responsible neurons". In: *Advances in neural information processing systems* 33 (2020), pp. 5922–5932.

[52] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. "Generating hierarchical explanations on text classification via feature interaction detection". In: *arXiv preprint arXiv:2004.02015* (2020).

[53] Andrew L. Maas et al. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 2011, pp. 142–150. URL: http://www.aclweb.org/anthology/P11-1015.

[54] Eyal Shnarch et al. "Will it blend? blending weak and strong labeled data in a neural network for argumentation mining". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2018, pp. 599–605.

[55] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).

[56] Ran Levy et al. "Unsupervised corpus–wide claim detection". In: *Proceedings of the 4th Workshop on Argument Mining*. 2017, pp. 79–84.

[57] Enja Kokalj et al. "BERT meets shapley: Extending SHAP explanations to transformer-based classifiers". In: *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*. 2021, pp. 16–21.

[58] Edoardo Mosca et al. "GrammarSHAP: An Efficient Model-Agnostic and Structure-Aware NLP Explainer". In: *ACL Workshop on Learning with Natural Language Supervision*. 2022.

[59] Scott Lundberg. *Be careful when interpreting predictive models in search of causal insights*. 2021. URL: https://towardsdatascience.com/be-careful-when-interpreting-predictive-models-in-search-of-causal-insights-e68626e664b6.

[60] Conor O'Sullivan. *The Limitations of SHAP*. https://towardsdatascience.com/the-limitations-of-shap-703f34061d86. Accessed on 24th July 2023 at 17:00. 2023.

[61] Luke Merrick and Ankur Taly. "The explanation game: Explaining machine learning models using shapley values". In: *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*. Springer. 2020, pp. 17–38.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 16. August 2023 Johannes Widera