Tuning of Explainable Artificial Intelligence (XAI) tools in the field of text analysis

Bachelor's Thesis of

Philipp Weinmann

at the Department of Informatics
Institute for Program Structures and Data Organization (IPD)

Reviewer:	Prof. Dr.-Ing. Klemens Böhm
Second reviewer:	Prof. Dr. Pascal Friederich
Advisor:	M.Sc. Clemens Müssener

15. January 2021 – 17. June 2021

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 15. June 2021

(Philipp Weinmann)

Abstract

In this bachelor thesis we analyse classification results using a 2017 published method called shap [1]. Explaining how an artificial neural network makes a decision is an interdisciplinary research subject combining computer science, mathematics and psychology. We have studied these explanations from a psychological standpoint and after presenting our findings, we will propose a method to improve the interpretability of text explanations using text-hierarchies, without losing much/any accuracy. In addition, the goal was to test out a framework developed to analyse a multitude of explanation methods. This Framework will be presented next to our findings and how to use it to create your own analysis. This Bachelor thesis is addressed at people familiar with artificial neural networks and other machine learning methods. Background knowledge about artificial intelligence explainers is not required.

Zusammenfassung

Das Ziel dieser Arbeit ist die Analyse von Klassifizierungen mithilfe der 2017 veröffentlichten Methode Shap [1]. Erklärungen, wie ein künstliches Neuronales Netz eine Entscheidung trifft, ist ein interdisziplinäres Forschungsgebiet, welches Informatik, Mathematik sowie Psychologie beinhaltet. Wir haben diese Erklärungen von einem psychologischen Standpunkt aus analysiert. Zudem schlagen wir eine Methode vor, um die Interpretierbarkeit von Textklassifizierern zu verbessern ohne signifikant an Genauigkeit zu verlieren. Darüber hinaus war das Ziel dieser Bachelorarbeit ein Framework zu entwickeln um eine vielzahl von Erklärmethoden zu analysieren. Dieses Framework wird neben unseren Resultaten präsentiert. Wir erklären wie es genutzt werden kann um eigene Analysen durchzuführen. Diese These ist an Personen gerichtet, die mit künstlichen Neuronalen Netzen sowie anderen Machine Learning Methoden vertraut sind. Kenntnisse über Erklärer sind nicht erforderlich.

Contents

List of Figures

List of Tables

1 Introduction

> *AI is one of the most important things humanity is working on. It is more profound than [...] electricity or fire [2]*
>
> (Sundar Pichai)

1.1 Motivation

1.1.1 Need for explainable artificial intelligence in general

Artificial Intelligence (AI) has been revolutionizing technology since computers were invented [3]. For most of that time the limitation had been processing power, limiting the complexity of artificial intelligence. Therefore, except for the rare exception, artificial intelligence which was being run could be understood by reading the underlying code that defined it.

Past decades have seen tremendous increase in processing power (see Figure 1.1) which has shifted the limitations of AI from computational to cognitive processing power.

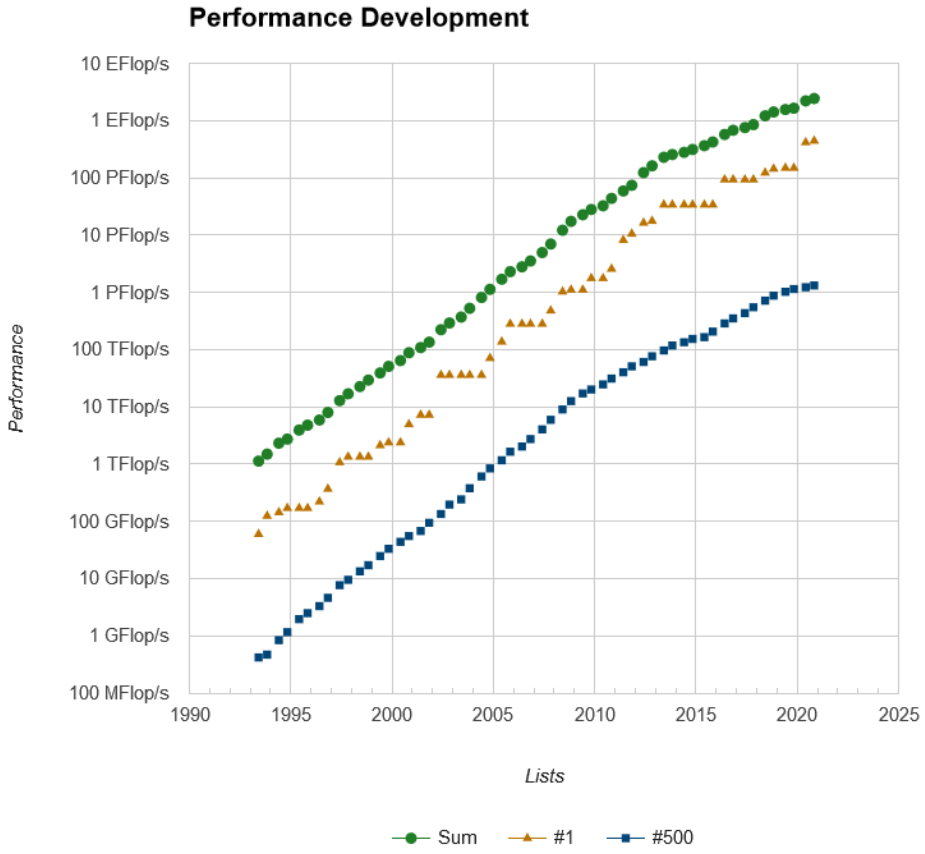

Figure 1.1: Performance of the top 500 supercomputers over time [4].

This has culminated in the rise of a new type of algorithms, artificial neuronal networks. These apply a fundamentally different approach to AI. The human doesn't define the algorithm anymore but feeds data to a complex algorithm which writes its own program to solve a specific problem. These approaches have led to breakthroughs in many fields [5].

While this step was arguably unavoidable for the advancement of AI, it poses its own challenges. Artificial neural networks can depend on millions of parameters [6] which cannot, in their entirety, be understood due to our limited cognitive abilities.

In Figure 1.2 I have tried to showcase the issue, modern day artificial neural networks are fundamentally more complex than any previous algorithms. 2013 is considered a breakthrough year in image recognition, with massive improvements in artificial neural network technology [7], but the exact moment the cognitive complexity barrier was breached is difficult to pinpoint.

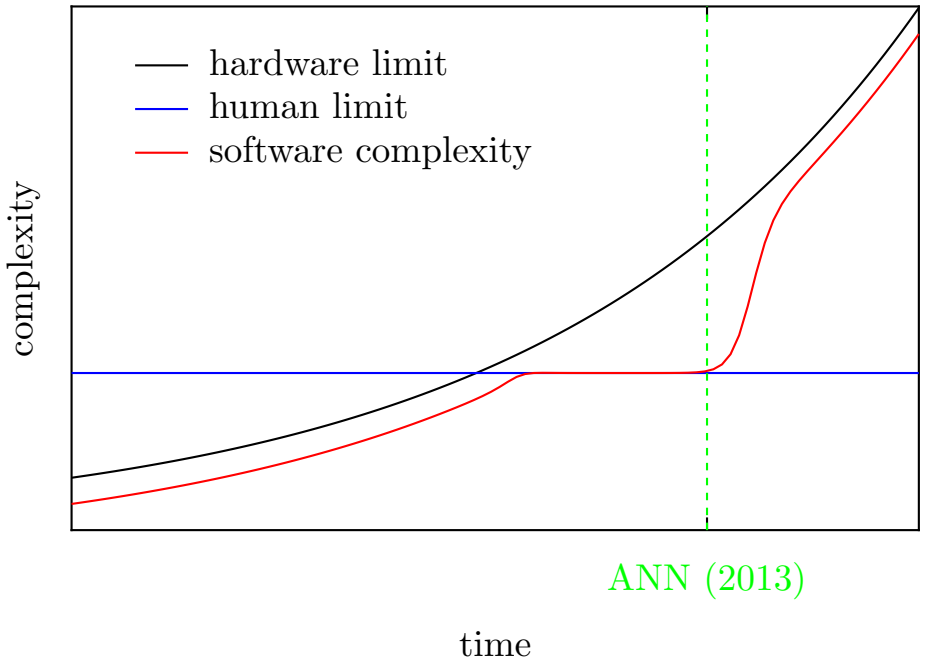

Figure 1.2: Breaking through the cognitive complexity barrier.

This inability to understand AI and algorithms in general poses a few fundamental problems.

- How do we debug/improve algorithms we do not understand?
- How do we enforce laws on algorithms we do not understand?
- Can/should we use such an algorithm on applications where fatal errors are not permitted *(ex: military applications)*

1.1.2 Need for tools to analyse Explainable AI

All of the above mentioned issues are being faced not only by scientists but more generally by software developers who do not have the luxury to spend months analysing one aspect of their software and require specialized tools to guide them. It can be argued that while neural networks where theorized decades ago, and could be implemented in hardware since the 2000s, it required frameworks and libraries like DistBelief, Tensorflow and more for their applications to become widespread and usable for most engineers and scientists. Theory and practicable tools need to go hand in hand to enable progress and in this context we wish to present our own framework to analyse explainable artificial intelligence tools.

1.2 Applications for explainable AI

The sceptical would argue that you cannot explain complex algorithms because they are by nature too complex to be understood in detail by a human. The argument would be, that if we were capable of explaining artificial neural networks, we could program them ourselves. While for certain cases this is true, it is not guaranteed. Multiple scenarios could happen:

1. Developers just need a hint on how to solve a problem and an even incomplete explanation of a Neural Network enables them to code it in a humanly understandable way.
2. We understand a fraction of how the artificial intelligence makes a decision but it is enough to disqualify it. This is especially important if some unwanted behaviour like biased *(racist/sexist/...)* decisions are revealed.
3. We understand a fraction of how the artificial intelligence makes a decision and the approach feels plausible to humans.

Note that scenario three does not guarantee that no unwanted or even illegal behaviour is occurring. Similar to the quote from Edsger W. Dijkstra: “Program testing can be used to show the presence of bugs, but never to show their absence”, an incomplete explanation will never guarantee the absence of all issues opaque algorithms bring with them. Nonetheless, XAI is fundamental if we wish to evaluate and compare algorithms with each other.

1.3 Key questions

In this bachelor thesis we will examine which approaches exist to understand artificial intelligence, meaning algorithms generated with the help of artificial neural networks, and we will explore one of these approaches in detail. In this context we will present a framework which has been developed by our team and myself to analyse XAI approaches and compare them with each other. Our key questions are which parameters are important for the usage of explainable artificial intelligence tools when applying them onto natural language in written form and how to tweak them to adjust unsatisfying explanations. Furthermore we will show how our framework can be used to create your own analysis.

2 State or the art

2.1 Approaches to Explainable Artificial Intelligence

There exist multiple approaches to XAI which each have their advantages and disadvantages. In this section we will briefly go over the most common ones [8]:

- Perturbation based approaches.
- Function based approaches
- Surrogate-/ Sampling based approaches
- Structure-Based approaches

Each one of these takes a pretrained ANN and analyses it and its behaviour by analysing how its output depends on variations of the input.

Most advanced tools including Shap [9] (the tool we used) combine several of these approaches trying to combine their respective advantages and compensating their respective disadvantages.

2.1.1 Perturbation based approaches

Perturbation based approaches excel in their simplicity. The core algorithmic logic is dependent on random or carefully chosen changes to features in the input data [10]. This approach is best visualized when applied to images.

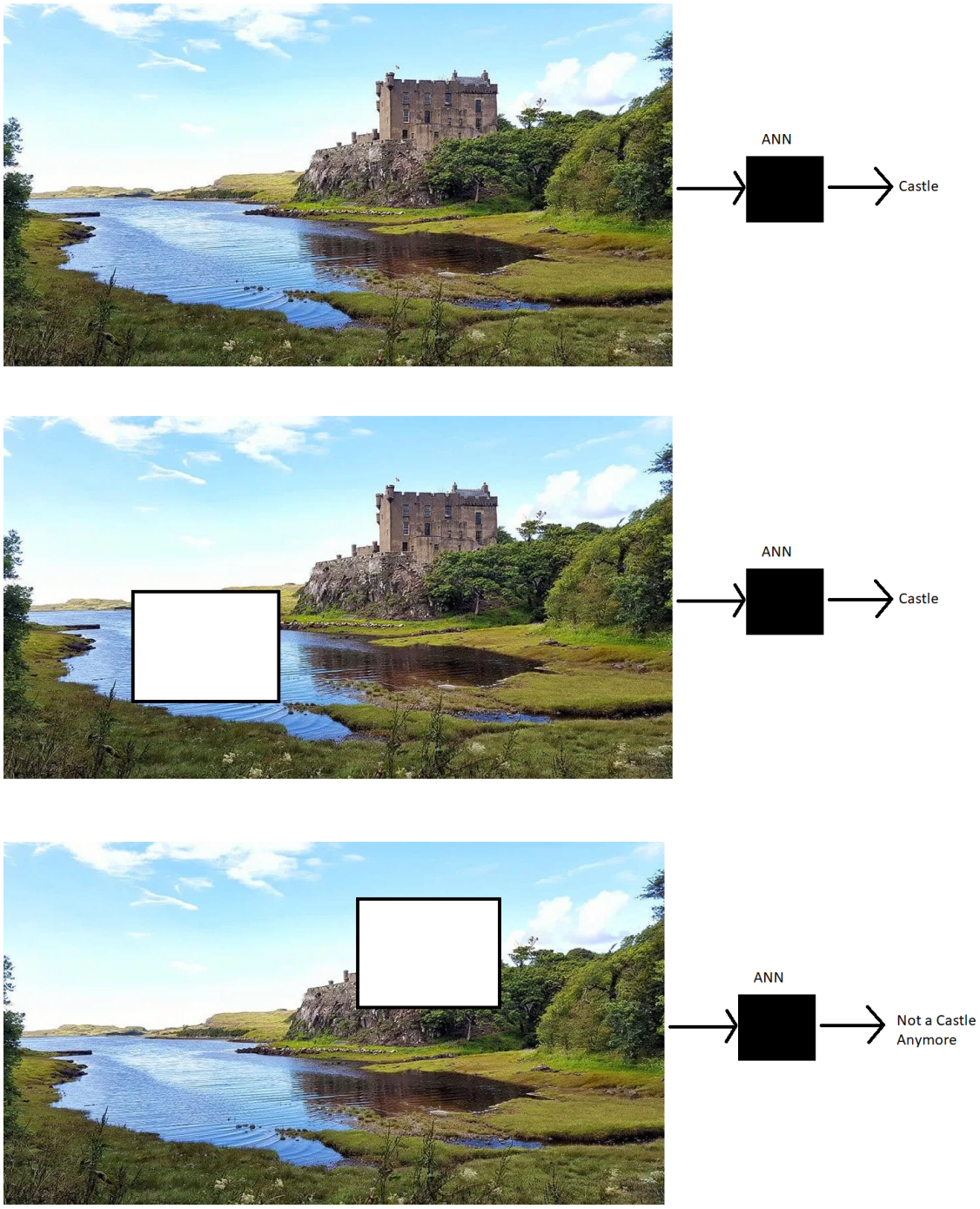

Figure 2.1: Perturbation based approach visualized on an ANN trained to detect castles [11]

In the example above: Figure 2.1 the perturbation approach XAI tool would conclude that the pixels in the first rectangle are not important for the classification while the

ones in the second one are. While seeming straight forward, this approach has two key disadvantages:

- Assumes locality, *meaning that it assumes that the output only depends upon the current input and the base rate of the model(the average output of the model).*
- Perturbations may introduce artefacts. *In the above example you could imagine that a big grey rectangle is recognized if not as a castle, then at least as a concrete building. This is a particular issue when explaining text. The classifier might respond to grammatically incorrect sentences in a certain way which would be an unwanted side effect of said removal.*

2.1.2 Function based approaches

In this section we will cover the well-known Gradient x Input approach.

We will quickly reiterate some basics and then explain how this approach can be used to create an explanation. Let's say we have a classification function f which takes a multidimensional input x. We now would like to know which direction/dimension was more important than the others. To find this dimension, we use following function: The *Gradient*.

$$\nabla f(x) = \frac{\delta f(x)}{\delta x} \tag{2.1}$$

If we apply that *Gradient* to x like done above, we get a vector of partial derivatives of $f(x)$ for every dimension of x. We now know which direction has the largest impact on the classification. We can now look at that direction and determine from it which values of x are the most responsible ones for the classification result. This can be done in the form of a *Gradient x Input* matrix called an attribution map. For most applications, this works well but still has some issues. The gradient only tells us the importance of a dimension if we just take a tiny step which is very local information. Furthermore it suffers from "noise" (the explanation map does not form a clear structure but shows seemingly random information), an issue this thesis tries to address. This method serves as a solid conceptual basis for more involved explanation methods including the one we implemented [12].

2.1.3 Surrogate-/ Sampling based approaches

Sampling's main goal is speed. You quickly approximate a solution without guaranteeing that the solution is correct. This is a valid approach since computation power is still the biggest limiting factor concerning artificial neural networks. We will quickly look at one of these approaches: **local interpretable model-agnostic explanations (LIME)** [13].

This method ignores the global underlying black box and only tries to provide an explanation for a single prediction. To achieve that goal, it generates a new dataset consisting of modified samples and runs them through the black box. Using this new dataset, lime then trains an interpretable model (we will come back to what an interpretable model is in chapter 3, for now just consider it a simple model which can be understood by just looking at its structure/code). Lime's model should be a good approximation of the original black box locally but does not have to be a good global approximation (see Figure 2.2). This kind of accuracy is called local fidelity [14].

Throughout this thesis we will call the approximated model the *explanation model* and the model we wish to explain the *original model*.

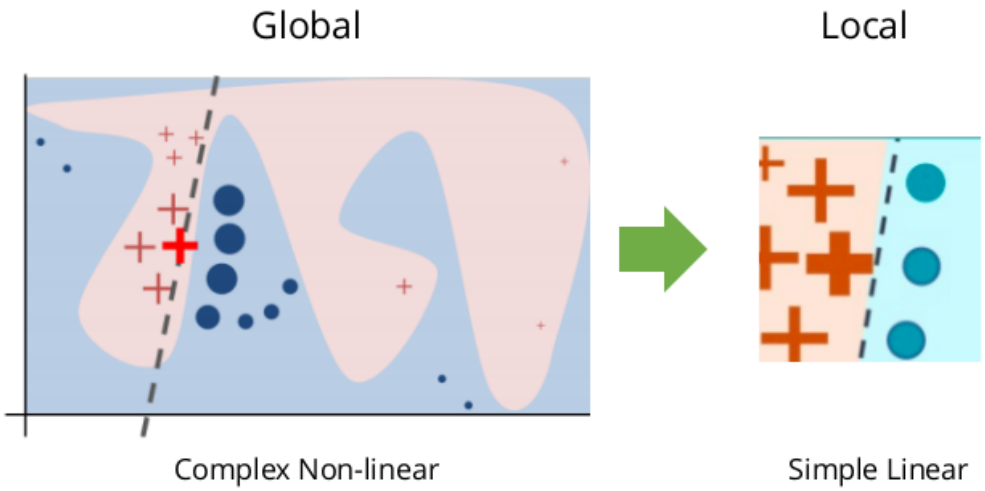

Figure 2.2: LIME localizes a problem and explains the model at that locality, rather than generating an explanation for the whole model [13].

The method we implemented, called shap, is actually a special implementation of Lime adding further rules to the algorithm making it more predictable but also less flexible. We will explain these differences in more detail in chapter 4.

2.1.4 Structure based methods

Structure based methods look at the structure of the ANN. We will briefly talk about the layer wise relevance propagation method. It tries to find subfunctions in the neural network. For example a part of the network which finds a certain pattern. We do this by starting from the output. Often, not always, we can detect a pattern, which part of the neural network is responsible to detect a certain output class. This can then be analyzed

to realize what part of the input is responsible for a certain classification.

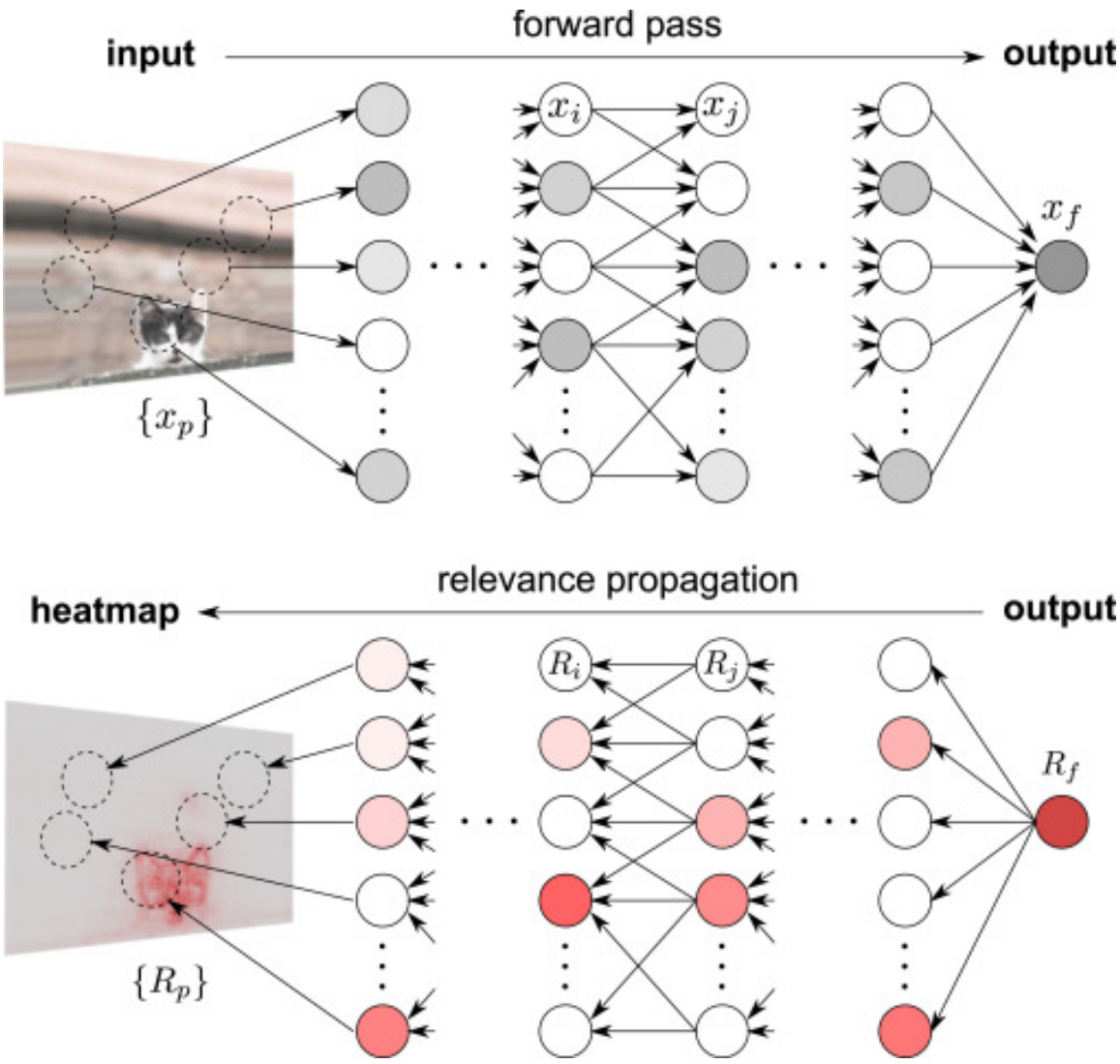

Figure 2.3: Layer wise relevance propagation [15]

In Figure 2.3 we see how the red inputs form the shape of a cat. We can then conclude, that the "activated" part of the ANN is a function detecting the shape of a cat.

This is a very powerful method with only one drawback, its implementation is very dependent on the shape of the neural network. Classifiers that do not use the traditional neural network shape cannot be analyzed using that method.

2.1.5 Shap

All of these approaches are valid, they each have their advantages and disadvantages. The method we will implement, shap, is actually a mixed approach picking tools from each basket to form a powerful explainer (chapter 4). Yet it suffers itself from a few disadvantages. We will go over in more details about those and how our work tries to address one of these issues (chapter 6).

3 Basic Concepts

In this chapter we will go over some of the more fundamental concepts which are necessary to understand how explainers work in general and their challenges. Expert readers may skip this chapter.

3.1 Interpretability

Interpretability is a rather vague concept which can be summarized in following statement: "can the average human understand it?". This simple statement has multiple issues. It is very unclear what that means in practice. A lot of work has been done to define that statement in the last few years [16]. The latter reference defines a few desiderata which models should have to be interpretable. We will go over some of them and explain what that means in our context of text classifiers.

- **Simulatability**: In opposition to the opacity of artificial neural networks, simulatability is a property interpretable models need to have. The idea is, that a model is simulatable, if a person can contemplate the entire model at once in a *reasonable* timeframe. This is important because it is possible to contemplate parts of artificial neural networks, and understand them because in itself they are very simple functions, but in such quantities, that it is impossible to contemplate them all at once. It also contradicts the idea that linear models are always interpretable. If complex enough, even a linear model is not humanly understandable.
- **Decomposability**: States that the model can be decomposed in parts which can be understood by themselves, without needing to understand the whole function. This notion of interpretability while being popular is controversial, because it excludes many models from being interpretable but I will make the argument that it is very much true. Humanly understandable means that one human can explain the model to another human. That is only possible if you can start somewhere, if you can put your explanation in logical following parts. This is how humans arguments, pick up information, a process called **active learning** [17]. While a model might seem simple enough to be understood by the human mind, it needs to be decomposable so that a human can acquire the knowledge about the model in a structured way.
- **Algorithmic transparency**: While it is possible to understand how an algorithm works, it can be much more difficult to accept that the algorithm will provide the desired result. This is however an important part of Interpretability, if the algorithm

cannot be trusted to work as intended, it is not really understood because the results are much more difficult to predict [16].

Let's now analyze what that means in the case of text classification using artificial neural networks. It is obvious that we cannot provide any of the three desiderata for modern neural networks. It is thus necessary to take a detour to try to explain the model in an interpretable way. To do this we create another much simpler function, the explanation model, which approximates the result the original model provides. As mentioned in chapter 1, it cannot be guaranteed that this approximation is correct. While that should be criticized, it is the best we have. To achieve that goal we use three methods: *Post-hoc analysis, local explanations* and *explanations by example.*

3.2 Post-hoc analysis

Post-hoc analysis is a simple concept, consisting of statistical analysis specified after the data is seen. This makes the analysis method model agnostic, it can analyse any function, artificial neuronal networks among them. While there are (like discussed in chapter 2) methods which analyse the structure of the neural network to create an interpretable model, the post-hoc analysis does not care about the structure of the neural network. One major drawback with post-hoc analysis is that it is hard to discern between causality and correlation (Figure 3.1). This issue is also present with neural networks, since those are trained following the same Post-hoc analysis principle, and any correlation which isn't causality is very likely to also be present in the original model, hinting that the model cannot be used to provide proper classifications.

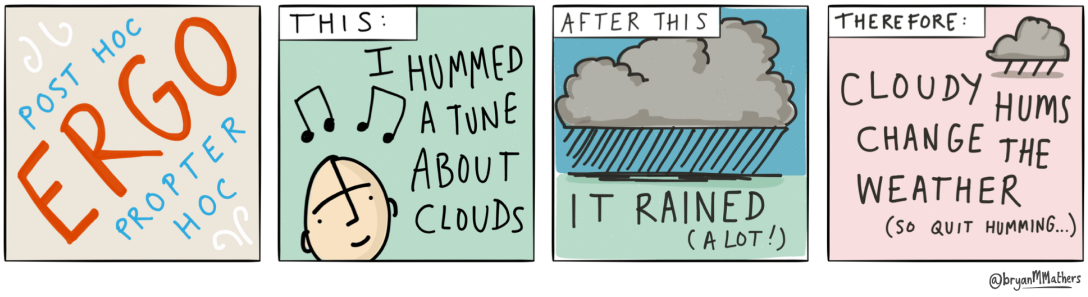

Figure 3.1: Post-Hoc causality vs correlation [18].

Again, what does this mean in our case of text analysis? Our inputs are the strings we wish to classify, the outputs are the classification scores. We can create an explanation model by altering the input and analysing its effects on the classification score. We have discussed one tool (lime) in chapter 2 which works in this way and we will further look at how shap does it.

3.3 Local Explanations & Explanation by example

The second method we are using is not trying to explain the entire original model, but only approximate it in a local context. This means, that our explanation model is only faithful to the original model for that one classification. This is necessary because the complexity of modern neural networks can greatly surpass anything which could be approximated with an interpretable function. This allows us to solve one of the big issues with neural networks, we can debug and notice possible mistakes. All three scenarios mentioned in chapter 1 can now happen:

1. Developers just need a hint on how to solve a problem and even an incomplete explanation of a neural network enables them to code it in a humanly understandable way.

2. We understand a fraction of how the artificial intelligence makes a decision but it is enough to disqualify it. This is especially important if some unwanted behaviour like biased *(racist/sexist/...)* decisions are revealed.

3. We understand a fraction of how the artificial intelligence makes a decision and its approach feels plausible to humans.

For text-classifiers, this means that we get an explanation as to why a specific text got its classification score. It proposes an explanation model which approximates the original model for this one text. It can and will substantially diverge from the actual model, but its results will be consistent for this one specific text. The hope is that this approximation picks the parts of the original model which are relevant for the classification of this one text, creates its own simpler model using them and ignores the rest of the original model [19].

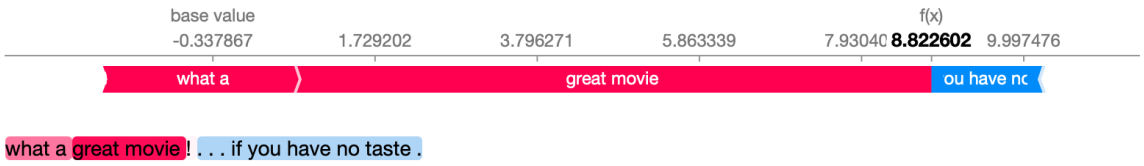

Figure 3.2: Linear approximation model [20]

Above we have an example of a linear explanation model, which provides the same result as the original model $f(x)$, but is much simpler to understand. The base value is the value you would get if you classified an empty text. The red text snippets are the words which move the classification score to a greater value, the blue ones to a lower value. At the end you get the same classification score as the classifier $f(x)$ produces.

The tool we implemented provides a post-hoc local explanation. We will now go into detail how it achieves that.

4 Shapley additive explanations (shap)

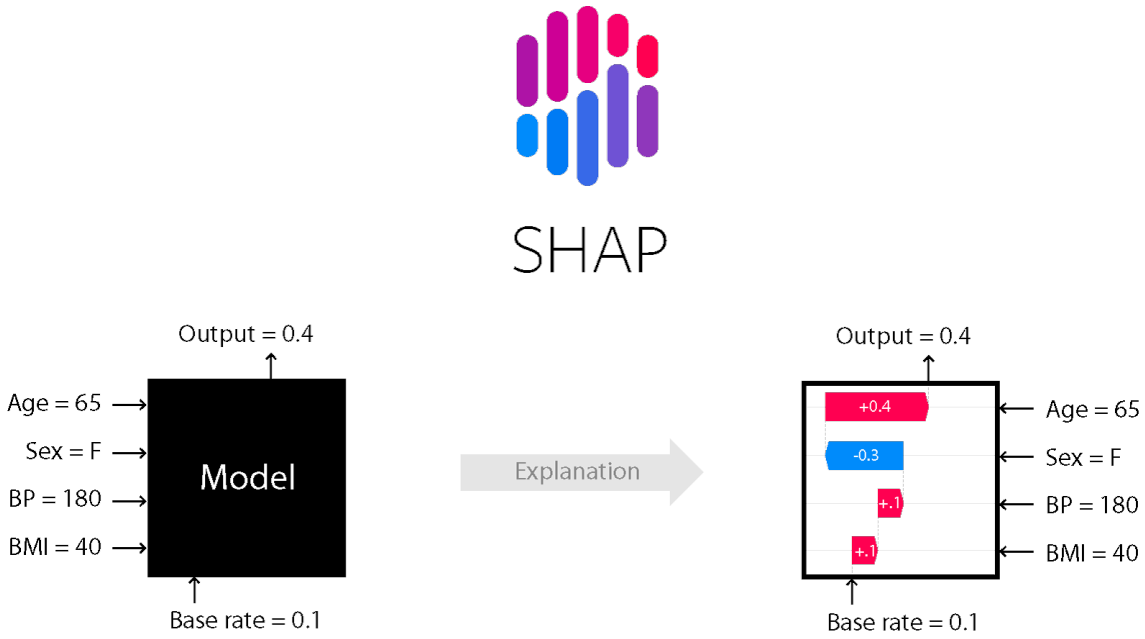

Figure 4.1: Visualization of a shap explanation [20]

4.1 General idea

The human mind is astonishingly good at detecting patterns. It achieves this at an incredible rate still unmatched by our most powerful supercomputers [21]. The issue is, that once a pattern is recognized, it gets rather hard to "forget" it even if it is a so called "false pattern". This false pattern recognition is called apophenia and is considered a massive roadblock to creating an explanation model, because it severely limits the ability to understand the explanation model when presented to the user. If the user thinks he understood the explanation model and then gets another example explanation which contradicts his understanding of the explanation model, psychologically this is very confusing.

A simple solution to this problem is to aid the human pattern recognition by defining rules that cannot be broken inside the approximation model. We will go over detail about these rules and why they make sense. This obviously doesn't eliminate the risk of apophenia but should decrease it significantly.

Please note that the psychological aspect was not explored in detail by the original shap paper. This is my own interpretation and argumentation as to why their work is a revolutionary approach to balance accuracy and interpretability in the context of neural network explanations. This is done because my work tries to improve the interpretability of the explanations by tackling above mentioned psychological barriers.

4.2 Competitive Game Theory

Game theory is a simple concept, where you study mathematical models of the interplay between rational decision makers. The most known example is the prisoners dilemma shown in Figure 4.2.

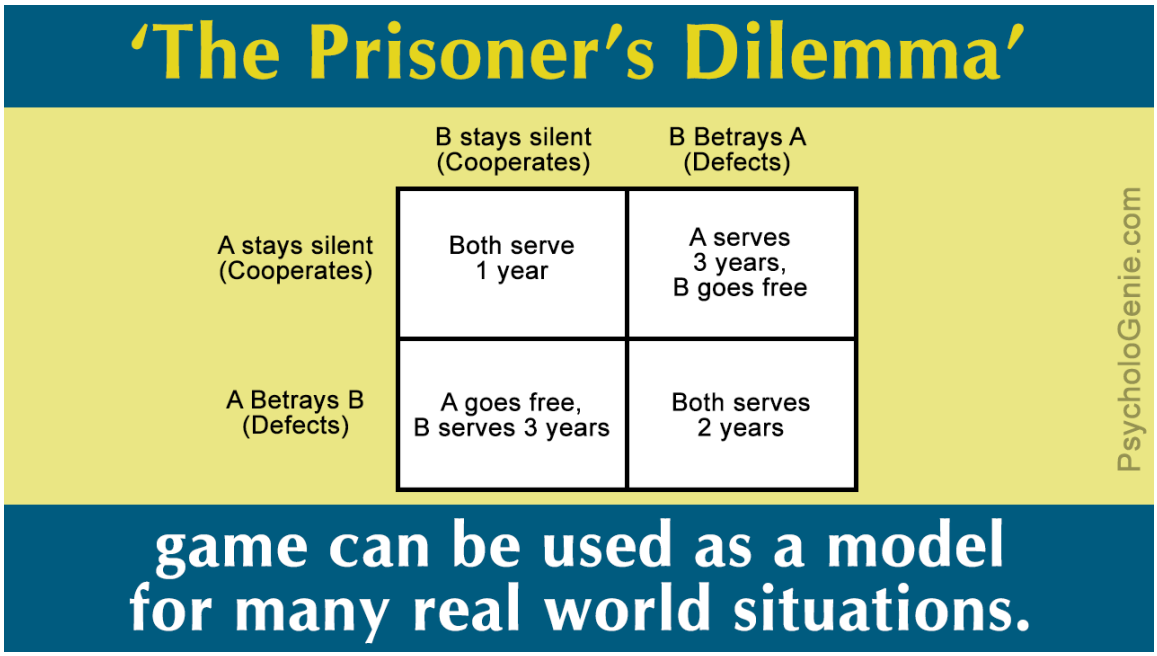

Figure 4.2: Prisoners Dilemma [22]

Game theory looks at how everyone should behave individually for the best outcome, and how a group of people should behave for the best outcome. The differences can then be discussed and often yield interesting properties. Competitive game theory is a subcategory of game theory, where a positive outcome for one "player" yields a negative outcome for another.

In our text classifications, this concept is used to determine the influence text-snippets have on the classification. Shap looks at the *individual* influence of a single word, and then at how its context influences its meaning. For example the following text: "Jesus word is the word of God" will most likely be classified as being nearly 100% christian, *instead of being atheist.* If we remove the word Jesus from the text, it becomes as follows: "word is the

word of God" which will also most likely be classified as nearly 100% christian. Therefore individually, the word *Jesus* does not influence the classification, but we intuitively know that it very much does or at least should. To find a more accurate value, to take the context into account, Shap removes multiple words in different orders and then combines the classification score differences using competitive game theory to find a more accurate value for that text-snippet.

4.3 Shapley Properties

The values which determine the influence of each element introduced into the original model are called **shapley values**. They satisfy three very important properties meant to improve the interpretability of the results:

- **Local Accuracy:** For the original input, the approximation model exactly matches the output of the original model.
- **Missingness:** This property is very misleading and should not be considered important. As described in the original shap paper [1]: If the missing of a feature in the original model does not change the result of the original model, then it should not have any impact in the approximation model either. This seems to contradict the use of competitive game theory and it does. I will directly quote the author of the shap paper Scott M. Lundberg when asked about this:

  "The missingness property is really just a minor book-keeping property to close a loop hole. It is required since local accuracy is specified as a linear model and x' [The input of the approximated model] could in theory have some zero entries (meaning the input is already missing [in the original input]). These zero entries mean that local accuracy would still hold no matter what phi values correspond to those entries, so to have a unique solution we need to constrain them to be 0 (which is what we want since they are missing already and so have no impact). In practice for SHAP we will never consider a feature to already be perfectly missing unless that feature's value is constant over the whole background dataset [23]."

  To sum it up, it means that if a feature, meaning an indivisible input, is not present in the original input, it shouldn't play any role in the explanation.

- **Consistency:** if the importance of a feature increases because the original model has changed, then its importance should not decrease in the explanation model. This property makes the results much more interpretable, because consistency is an important part of interpretability.

4.4 Examples for explanations of text based classifiers

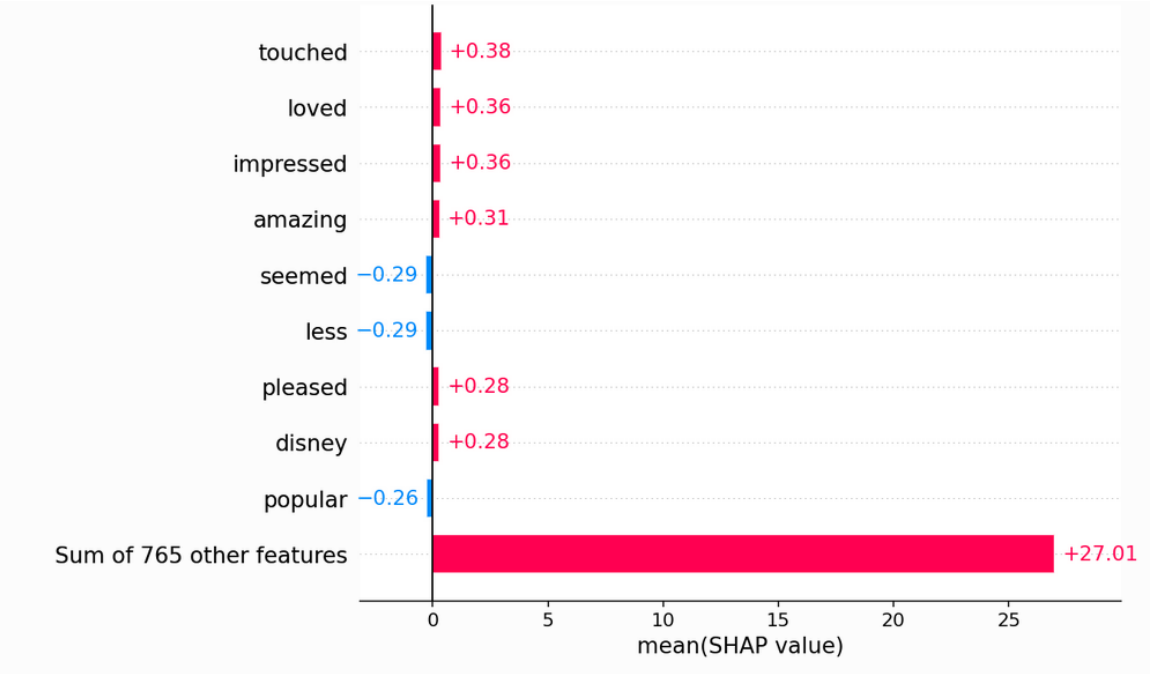

Figure 4.3: Shap explanation example [20]

In Figure 4.3 we see the top 9 features which have the largest influence on the classification with their respective shapley values. These values are getting dwarfed by the sum of all other features, but it helps to give an idea about how the classifier works. If the top features should not be part of the classification process, we can suspect that something is wrong with the classification.

This can be rather difficult, especially, if words which only show their importance in their respective context are part of the top 10 most important features. This issue is one we are going to try to approach in chapter 6 and suggest how to improve the interpretability of the explanation by using different text hierarchies.

4.4.1 Example where the explanation is difficult to understand because the context is missing.

The following email has been classified as being 99% christian:

From: jenk@microsoft.com (Jen Kilmer)
Subject: Re: Homosexuality issues in Christianity
Organization: Microsoft Corporation

Lines: 27

In article <May.11.02.36.59.1993.28108@athos.rutgers.edu>
dps@nasa.kodak.com writes:
>In article 15441@geneva.rutgers.edu, loisc@microsoft.com (Lois
Christiansen) writes:

>|>he can, especially homosexuality. Let's reach the homosexuals for
Christ.
>|>Let's not try to change them, just need to bring them to Christ. If He
>|>doesn't want them to be gay, He can change that. [....]

>don't hate the people. I don't. I don't hate my kids when they do
>wrong either. But I tell them what is right, and if they lie or don't
>admit they are wrong, or just don't make an effort to improve or
>repent, they get punished. I think this is quite appropriate.

Note the difference here. One is saying, if *Christ* disagrees with
a Christian being gay, *Christ* can change that.

The other is saying, if *I* think being gay is wrong, that a Christian
cannot be gay, *I* need to tell them to change.

As Lois said, and as before her Paul wrote to the believers in Rome,
WHO ARE YOU TO JUDGE ANOTHER'S SERVANT?

-jen

– [24]

The classification is obviously correct, we will now run shap trying to find out why that classification score was reached:

Feature name	Feature importance
rutgers	0.040224
athos	0.036232
geneva	0.030274
1993	0.025009
christ	0.022898
article	0.021479
writes	0.019735
com	0.019473
paul	0.016807
don't	0.014403
nntp-posting-host	0.010084
nntp-posting-host	0.010084
atheism	0.008166
christian	0.00862
christianity	0.008166
christians	0.00797

Table 4.1: Example where it is difficult to interpret what the classifier does.

Here we find out, that the classification is being done by looking at the header, the email provider. If studied in more detail, it would become clear, that the training dataset is flawed, all christian emails come from the same email provider. The classifier is not doing what it should, it reaches the correct classification, but if the header is missing, the classification result would become mostly random. Coming to this conclusion takes a lot of work though, because if we didn't provide the original text, you would be confused about the presence of words like "rutgers", "athos" or "don't". These are so dependent on their context, that by themselves they do not provide any information at all. This problem is particularly apparent for text-classifiers and are the reason we chose them. In chapter 6 we will suggest a method to remedy that particular issue.

We will now present the framework which was developed to help analyse multiple explainers.

5 Framework evaluation

In this chapter we will look at an evaluation framework developed at the institute for program structures and data organization. We will go over its goal, a basic overview of its components and how it is meant to be used. Finally we will talk about the challenges of implementing an analysis using shap withing this framework.

5.1 Goal of the framework

As discussed in chapter 1, tools are essential for the mass adoption of new technologies. XAI is still met with a lot of scepticism and rightfully so, every new technology should be tested extensively before it is widely adopted. Our evaluations took over a year to produce and we believe that by using the same framework we developed other scientists can create further evaluations in much less time.

5.2 Overview of the framework

The following description of our framework is a simplification, please refer to the documentation or actual code for a truthful representation. This is especially true for the overview diagram Figure 5.1. This was done for overview purposes.

The framework can be divided into 5 distinct parts which in theory should be exchangeable without having to modify any of the other 4 parts.

1. Resources, *data set, embedding, preprocessing*
2. Classifiers, *abstract classifier class, fit function, classification function*
3. Explainers, *abstract explainer class, explain function*
4. Analyses
5. Output, *functions to save the analysis data in csv files for further processing.*

In Figure 5.1 I provide the overview over the five components and their interfaces.

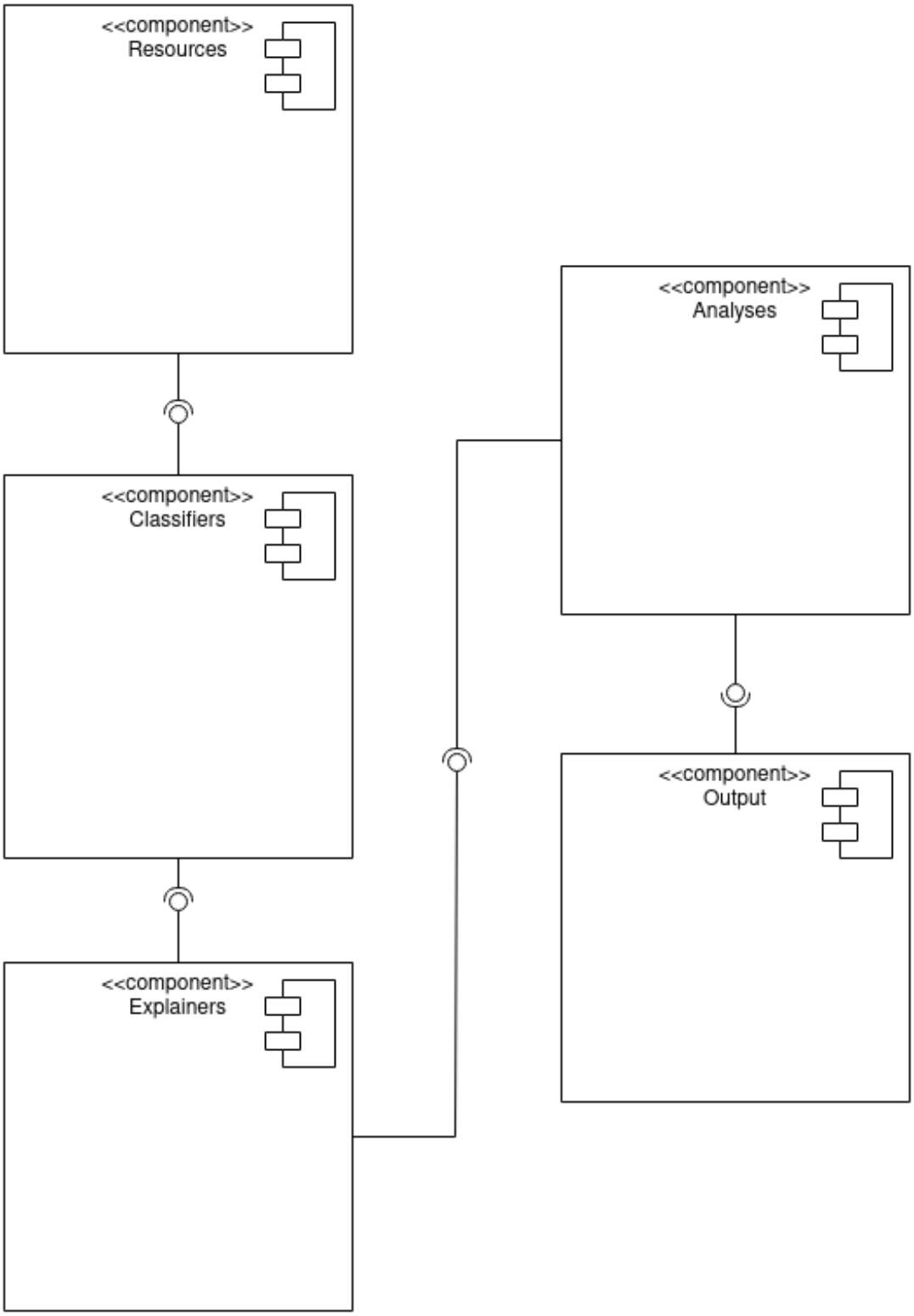

Figure 5.1: Component diagram, framework overview

5.2.1 Component 1: Resources

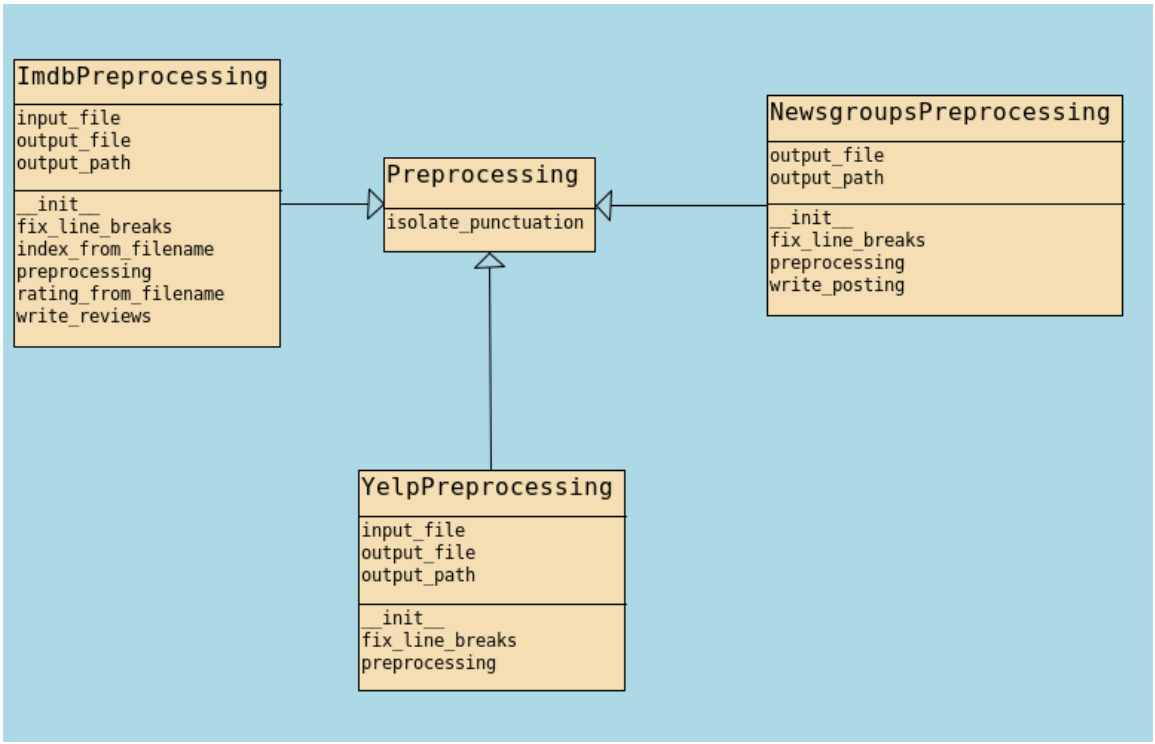

Figure 5.2: Preprocessing

The *Resource* component includes the data set and any preprocessing functions which might be needed to prepare/modify the given dataset. If you decide to use/include your own dataset, and you need to preprocess your data, you should make a child class to the preprocessing class.

5.2.2 Component 2: Classifiers

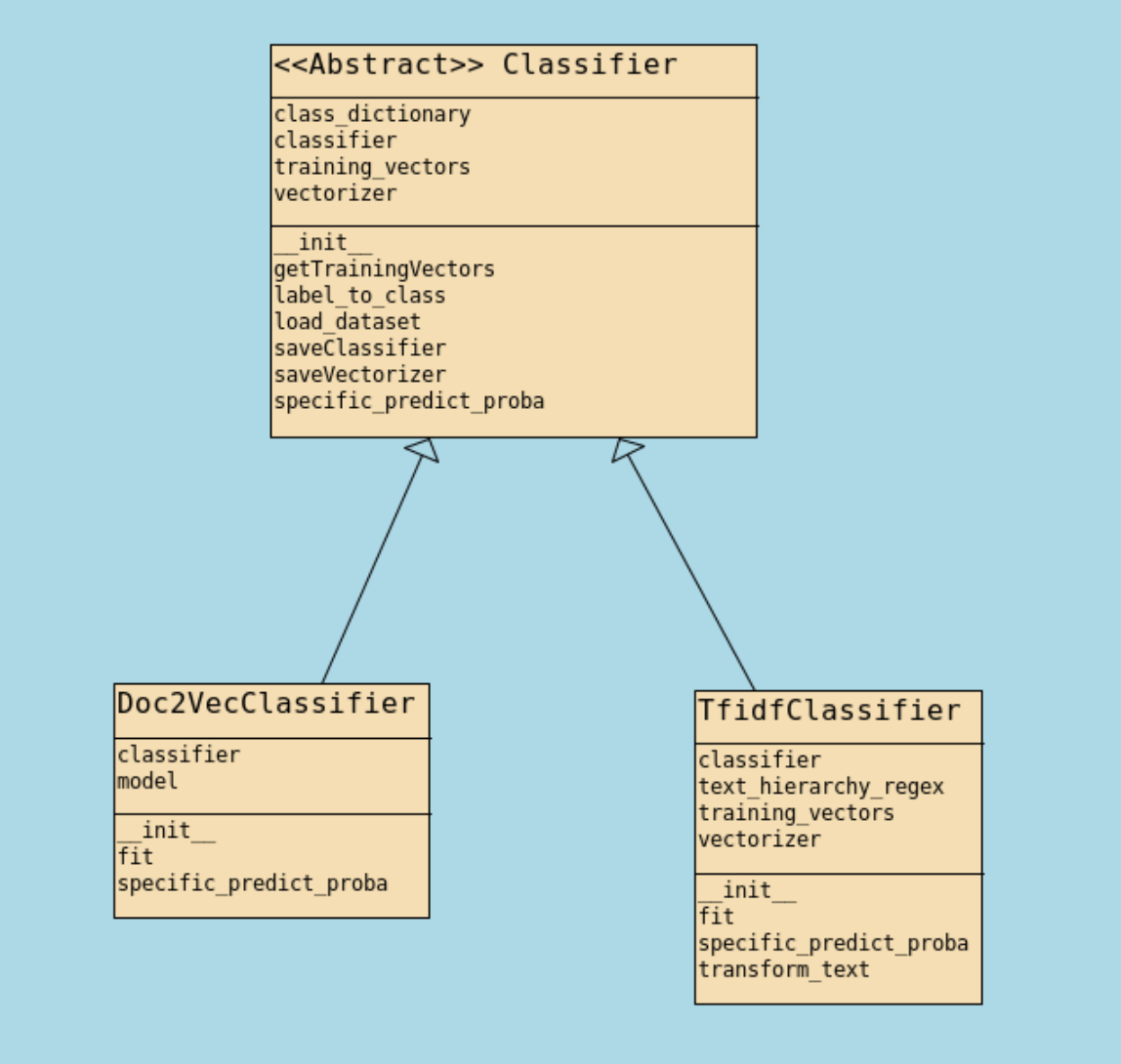

Figure 5.3: Classifiers

The *Classification* component includes the classifiers, with functions to train or load them. Pretrained classifiers are not supported yet, but you can save a classifier after training it. A new classifier should be a child to the abstract classifier class.

5.2.3 Component 3: Explainers

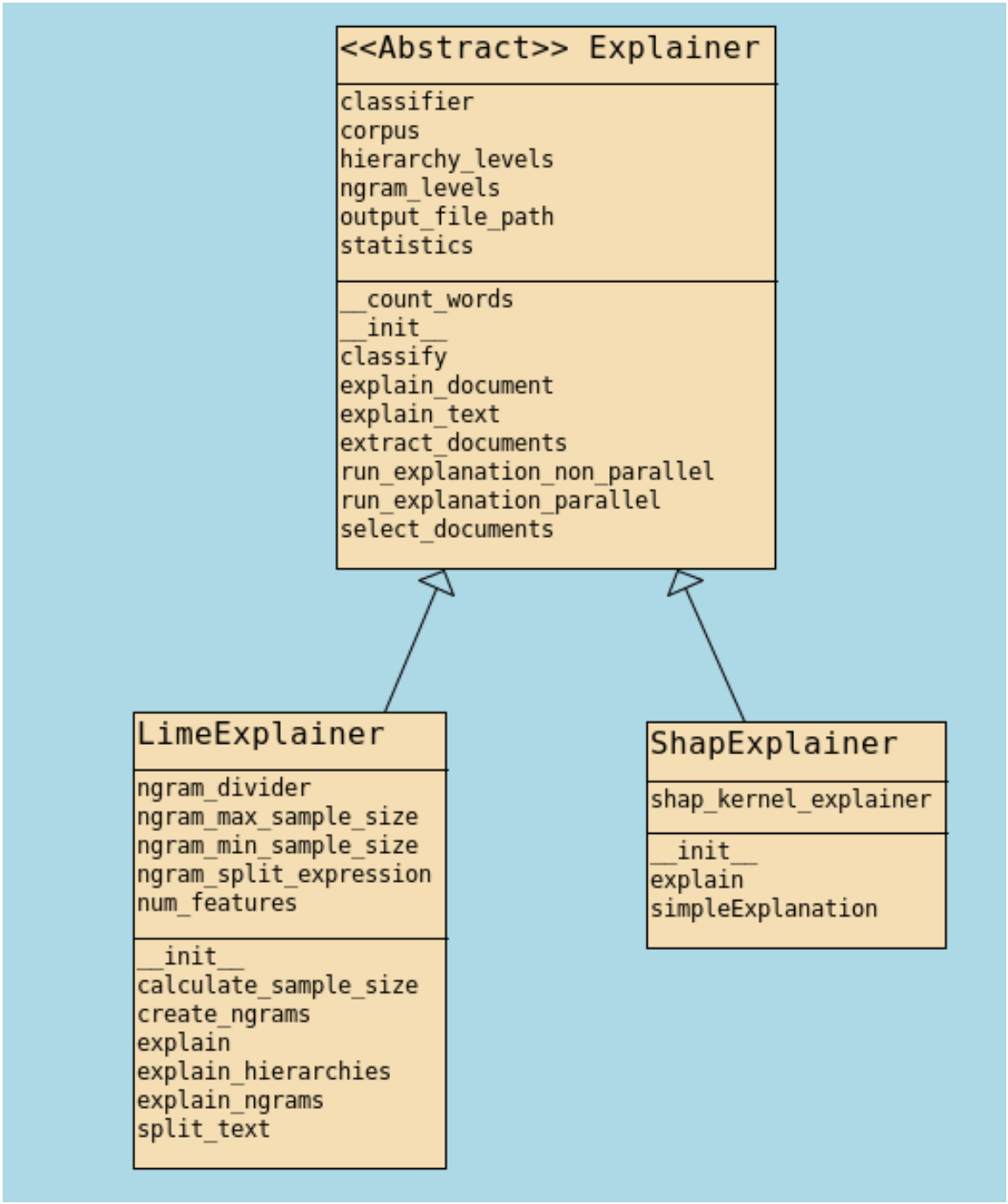

Figure 5.4: Explainers

The *Explainer* component includes the explainers and functions to explain texts or documents. If you wish to add another explainer, it should be a child class of the abstract

explainer class. This can be challenging, because the output format of the explanation must follow a certain format.

5.2.4 Component 4: Analyses

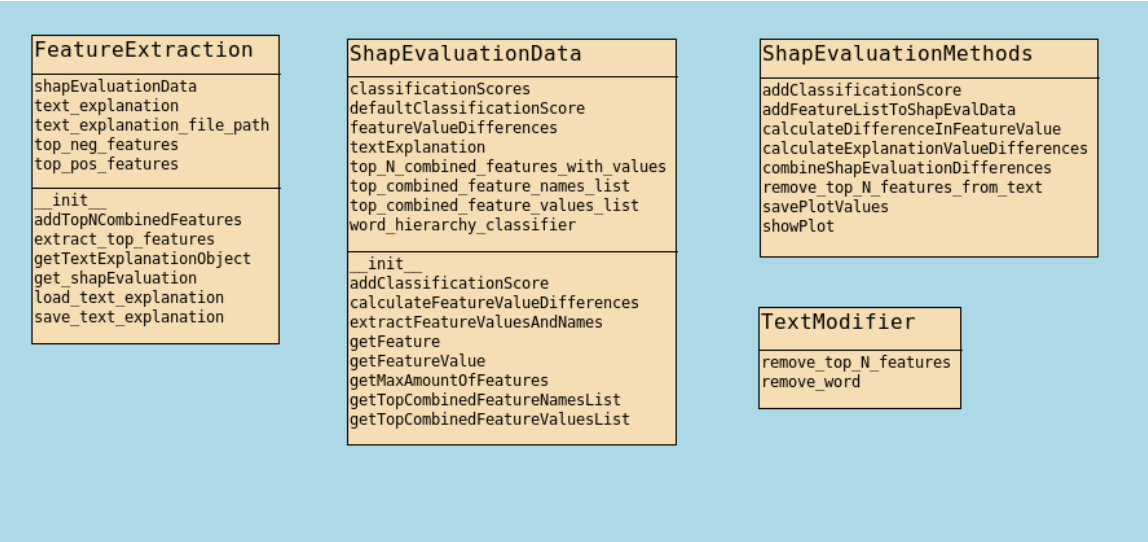

Figure 5.5: Analyses

The *Analyses* component is much more loosely structured. The user should be able to freely carry out his own analyses.

5.2.5 Component 5: Output

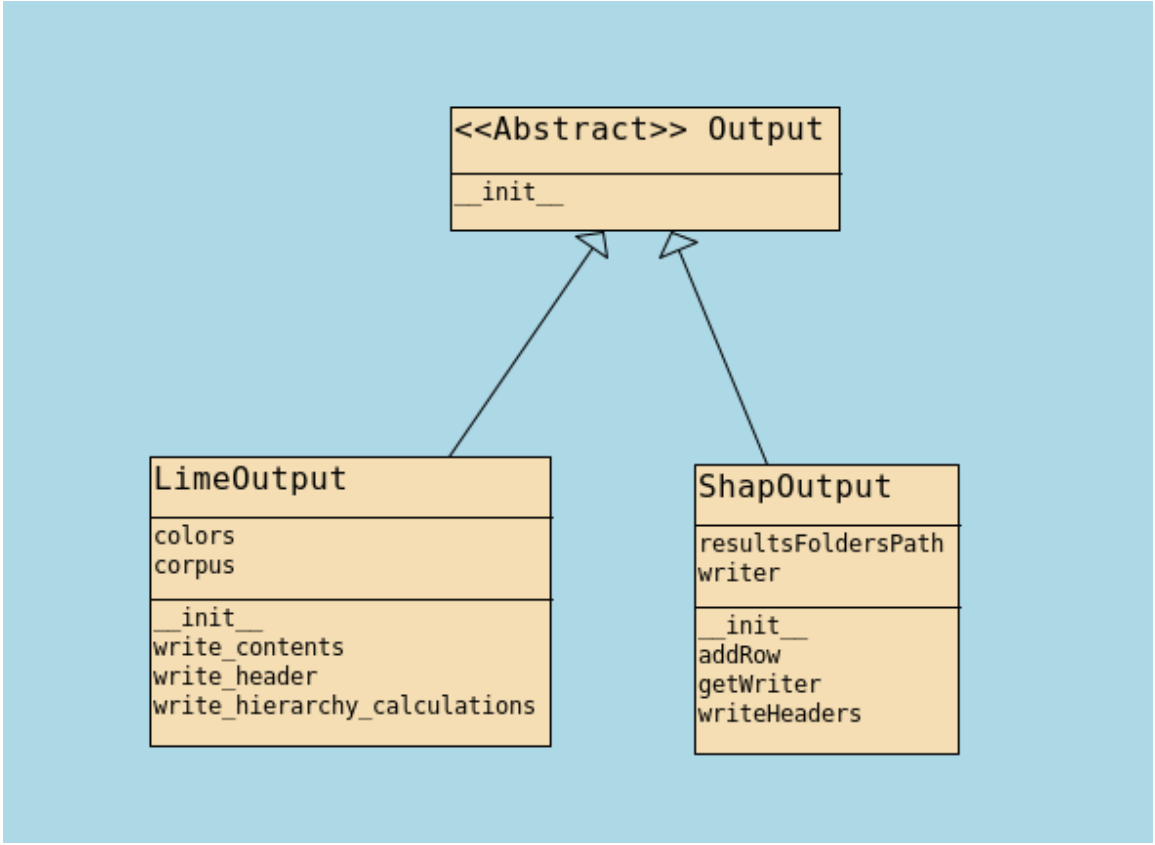

Figure 5.6: Output

The *Output* component defines how the analysis data is saved. Here we break the modularity of the framework. If you add another analysis, it is very likely, that you will have to write your own output child class. We are working on improving this, hopefully this will not be necessary in the future.

5.3 Intended use

We do not expect you to be satisfied with the provided components, depending on the scope of your work, you might want to exchange multiple components. Nonetheless, you might still find our framework useful. A major difficulty in setting up an XAI analysis, is that it can take months of coding to get an analysis to work properly. We suggest you build it step by step, working on one component at a time and using our provided ones to check for mistakes and provide structure to your workflow.

5.4 Shap Implementation using our framework

There are many different implementations of the shap algorithm, notably *gradientexplainer, deepexplainer, linearexplainer* and *kernelexplainer*. Except for *kernelexplainer*, each one is specialized for a special type of neural network. Since our framework is intended to have modular, interchangeable components, I tried to use *kernelexplainer* in our analysis.

During development, I encountered following challenges:

- Kernelshap's poor performance
- Kernelshap's resource requirements
- Missing required function in the library

Let's go over them in detail:

5.4.1 Poor performance

Kernelshap is model agnostic (meaning it can work with every type of artificial neural network) which meant that it could not be optimized according to the shape of the respective neural network. Because of that, generating an explanation for a single text took between 5 to 10 minutes on my workstation and the institutes server alike. This is very hindersome during development. While impossible to completely solve, I took some measures to cache as many steps as possible.

```
    def saveClassifier(self, classifier_out_file):
        print("saving classifier: ", classifier_out_file)
        dump(self.classifier, classifier_out_file)

    def saveVectorizer(self, vectorizer_out_file):
        print("saving vectorizer:", vectorizer_out_file)
        dump(self.vectorizer, vectorizer_out_file)
```

5.4.2 Resource requirements

Kernelshap uses a lot of memory. You need around 10 GB of ram to run an explanation for a single text. Since this is hard-coded into the shap library, not many optimizations were possible. I upgraded my hardware and added parameters to limit the amount of texts being explained in parallel.

```
    if RUNNING_ON_SERVER:
        # how many cpus are used during parallelization. With more than
    2 my pc crashes
        AMOUNT_OF_CPU_CORES = 10
    else
        # how many cpus are used during parallelization. With more than
    2 my pc crashes
```

```
6           AMOUNT_OF_CPU_CORES = 2
```

```
1       pool = mp.Pool(shap_constants.AMOUNT_OF_CPU_CORES)
2       # the actual explanation is now saved inside of this array of
        textExplanations
3       text_explanations = pool.starmap(self.explain_document,
        explain_documents_arguments_iterable_tuple)
4
5       pool.close()
```

5.4.3 Missing functions in library

Shap is a young explainer in active development. A few API functions which were necessary for my analysis weren't provided, so I had to write them myself. One example is that it was not possible to get a list of the top N features which have the most impact on the classification score. It was especially challenging to write this function, since the documentation on how to extract these values is nonexistent. It was thus necessary, to analyse the library's code line by line.

```
1 def get_top_features_of_specific_class(index: int, amount_of_features:
      int, shap_values, feature_names, pos_or_neg_val: str):
2       '''
3       :param index: 0 for first class, 1 for second class, doesn't matter
        since the pos ones for one class are the neg
4       ones for the other but we encourage to only use class 1 to avoid
        confusion.
5       :param amount_of_features: how many features should be presented in
        the explanation
6       :param shap_values: I believe those to be of the same format as the
        model output: in our case the model outputs
7       2 values so the shap_values are a list with two arrays. first array:
         shapley_values[0] holds the features which are
8       believed to nudge the explanation towards class 0,
        shapley_values[1] hold the values which are believed to
9       nudge the explanation towards class 1.
10      :param feature_names: the model takes text transformed in vector
        form, the explanation should output text again.
11      feature names map the vector values back to text.
12      :return: top <amount_of_features> for class <class_number> as a
        table.
13      '''
14      # extract top features either positive or negative
15      if pos_or_neg_val == "pos":
16          shap_class_values = shap_values[index].mean(axis=0)
17      elif pos_or_neg_val == "neg":
18          shap_class_values = - shap_values[index].mean(axis=0)
19      elif pos_or_neg_val == "combined":
20          shap_class_values = np.abs(shap_values[index]).mean(axis=0)
```

```
21
22      importance_df = pd.DataFrame([feature_names, shap_class_values.
        tolist()]).T
23      importance_df.columns = ['feature-name', 'shap_importance']
24      importance_df = importance_df.sort_values('shap_importance',
        ascending=False)
25      return importance_df.head(amount_of_features)
```

5.5 Framework outlook

The Framework is still a work in progress. Most notably it lacks documentation and overview to reduce the entry barrier. That being said, once its architecture adheres to Figure 5.1 and some basic documentation is provided, I believe that it has the potential to contribute in a significant way to future XAI analyses.

6 Results

6.1 Challenges evaluating XAI Explanations

As we've seen in the previous chapter, shap is a very specific recently proposed method to generate an explanation. Its algorithm follows rules and gives a result which exactly matches the output from the classifier [9].

> When approximating the original model f for a specific input x, local accuracy requires the explanation model to at least match the output of f for the [*original input x*]
>
> (Scott M. Lundberg, Su-In Lee [1])

Even if, using *kernelexplainer*, the result is approximated, this approximation error is not directly accessible, because competitive game theory produces a feature value which when all feature values are summed up, exactly matches the classification result [1, 25]. Nonetheless there are multiple possibilities to evaluate the quality of an explanation.

6.2 Dimensions of the explanation quality

I propose to categorize explanation qualities into three categories: **Accuracy, Interpretability** and **Resource-requirements.** These are not independent, we expect accuracy to increase with more allocated computational resource and vice-versa. Interpretability, *meaning the ease with which the result can be understood* is expected to be negatively correlated with the accuracy. The more features that are analyzed and presented to the data analyst, the more difficult it gets to interpret the explanation.

It is important to keep those interactions in mind when tweaking parameters of the explanations. A definitive advice on how to set those parameters will not be possible because it is unclear how important each dimension is for the user.

In this thesis we will not be able to tackle all of these dimensions and will focus on the accuracy and interpretability ones.

6.3 Necessary characteristics to analyse XAI results

The principle is simple, we've seen above that to evaluate an explanation we need to compare it to another explanation to get useful results. Following properties need to be fulfilled for an evaluation method to be of worth:

1. Be able to generate a theoretically unlimited amount of explanations for each parameter tweak.

2. Be able to pinpoint the influence of a single parameter (or a clear defined combination of multiple parameters).

3. Be able to theorize about the results on a **global** level (See Figure 4.3), *meaning that the influence of tweaking one or multiple parameters can be logically understood or at least theorized about by looking at a multitude of explanations. If the tweaking of a parameter results in chaotic changes which are not understood, tweaking that parameter is pointless.*

4. *(optional but desirable)* Be able to theorize about the results on a **local** level, *by looking at a tuple of explanations of a single corpus with different parameters.*

While the first three points are rather obvious, the last one might be a bit controversial and I will try to explain my reasoning. The goal of text-explanations is mostly to debug classifiers, to understand where they are failing and to fix issues in the underlying data. Examples like biased data, inadvertently pre-classified data (see subsection 4.4.1) come to mind. These issues are not always visible on a global scale. A lot of times the classifier uses shortcuts like detecting email-providers only on certain data entries which will not be visible when looking at a multitude of explanations. Some issues will not be detectable on a global approach, so explanations should not only focus on the entirety of the dataset. Since this is the expected use case for someone implementing/using shap, tweaking parameters of his implementation to learn more about a classification issue in a single classification is an important use case and it would be desirable that tweaking a parameter has an immediate effect on this single classification. This will be discussed in more detail in chapter 7.

6.4 Parameter: Text hierarchy

One important parameter we will evaluate is the text-hierarchy, defining which entries shap considers atomic entities *(which cannot be divided)*. This is important, because the user can only process a limited amount of information at once [26], but the length of that information plays a secondary role, as long as each chunk of information is perceived as one [27].

6.4.1 Dataset

The dataset used to generate these results, is a collection of christian/atheism themed emails provided by the Newsgroups20 [24] dataset. It is composed of 717 emails. We fitted a binary tfidf classifier for each text hierarchy supposedly classifying the emails in atheist and christian categories. Using the 80/20 rule (80% training data, 20% test data) our classifiers never dipped below 90% accuracy on any of our analyses (they are refitted on each run). The dataset and our approach is not ideal though, and can rightfully be criticized which we try to do in subsection 6.4.3.

6.4.2 Context importance by text hierarchy

Text-Hierarchy	Regex
Word	`\b[\w|'|-]+\b`
Sentence	`[^(.|?|!|\n)]+`
Paragraph	`((?:[^\n][\n]?)+)`
2-gram	`\b[\w|'|-]+\b\W+\b[\w|'|-]+\b`
...	...

Table 6.1: Text Hierarchies

We will now try to evaluate the explanation results by evaluating the Newsgroup20 [24] dataset with different text-hierarchies. For that purpose we used following formula:

$$\text{contextInfluence}(w') = \\ |(\text{classificationScore}(W) - \text{classificationScore}(W \backslash w')) - \text{shapFeatureImportance}(w')| \tag{6.1}$$

w': The feature which had the most impact in the classification according to shap
$contextInfluence(w')$: The importance of the context of the word w' according to shap
$classificationScore(W)$: The classification score the classifier gives the text W
$classificationScore(W \backslash w')$: The value the classifier gives the text W without the word w'
$shapFeatureImportance(w')$: The feature importance shap gives the word w'

In Equation 6.1 we compare the change of the classification score when we remove the feature w' with the feature importance shap gives w'. We averaged the context influence

over the entire corpus. We repeated this but instead of only remove a single feature we removed multiple features (in descending feature importance order) at once and plotted these results in Figure 6.1.

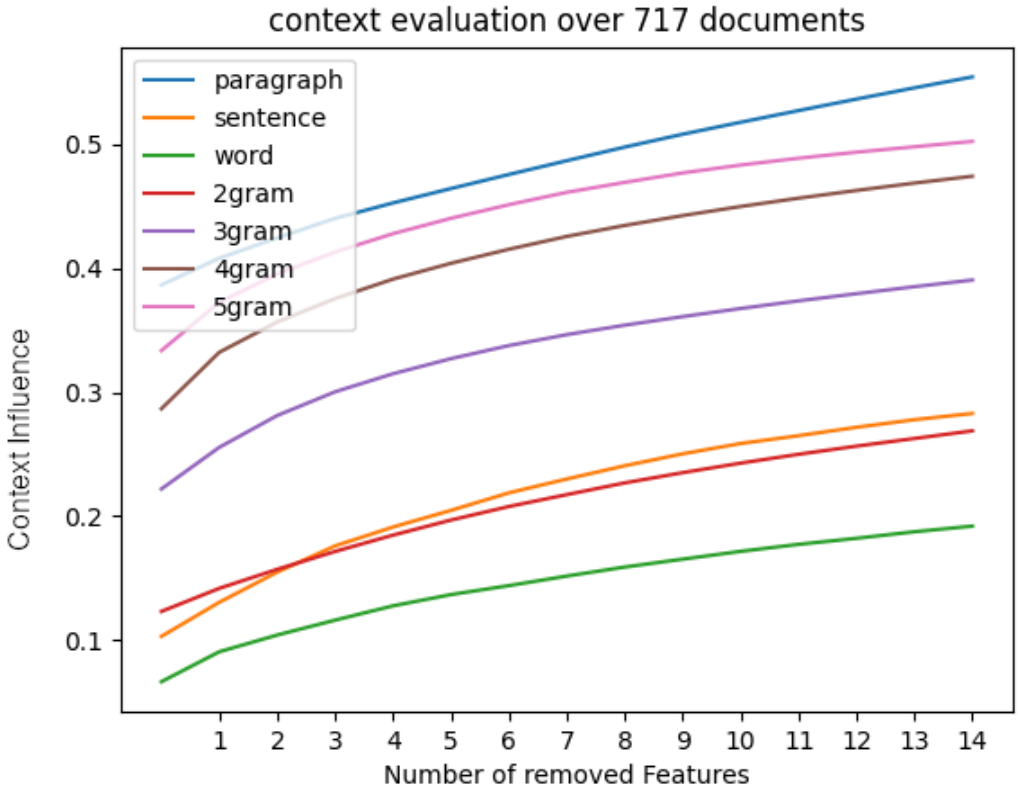

Figure 6.1: Shap Evaluation Results.

Now let's analyze what we did here. Calculating the importance of a feature by re-classifying the text without said feature is a simple perturbation based approach used before frameworks like *lime* and *shap* were developed. It does not take in account the context of a feature, an issue most recent approaches try to fix. The shap algorithm uses that method as well but adds more steps to it with the goal to **include the importance of the context** (section 4.2), and to a smaller degree, make the explanation more intuitive (section 4.3). So what **we are evaluating is how much weight the context of features has depending on the text hierarchy.**

With that information, let's analyze the produced plot:

- As expected, generally, the bigger the group of text taken out of the original email, the more dependent it is of its respective context. This is what we would expect because each word can interact with other words in the text.
- This rule is however broken for sentences. A sentence has 14 words on average. Since we are working with emails, a sentence might be shorter. Nonetheless we would expect the contect influence to hover above the 5gram curve.
- The same could be said about paragraphs, meaning they hold multiple sentences and are just slightly more dependent on the context than 5grams. I would be careful though, the original dataset being emails, did not provide very structured paragraphs. Furthermore, I consider emails to be too short to draw any decisive conclusion about paragraphs based on our results.

The fact that sentences are less dependent on their context than any other random n-gram makes sense, they are, from a grammatical standpoint, a self contained part of the text. On average, it is much less dependent on its context than a string of words of the same length cutting in half two sentences (as shown in our analysis). Even though our findings are unspectacular, they allow us to substantially improve the quality of XAI text explainers. The goal of an explainer is to provide as much information as possible without overwhelming the user. **We can, instead of providing words to the user, provide entire sentences**, which massively increases the quality of the explanation (because of the reduced context importance) without overwhelming him with to much information. As a reminder, chunking allows the human brain to process sentences with n words much faster than n independent words [28].

This is especially useful, if the previous explanation was not useful because the words provided were too dependent on their context (like shown in subsection 4.4.1). Our data shows that the per word influence of the context is reduced by 89,33 percent when comparing the context influence of sentences to the context influence of words.

6.4.3 Critical analysis

As mentioned before, there are a few points which could be improved in our approach, I will list them below:

1. The emails are to short to analyse paragraphs properly. *This is absolutely correct, in our analysis, we remove up to 15 features, and none of the emails in the dataset include over 15 paragraphs. The paragraph plot should not be used to make any conclusion and we only formulated a suggestion to use sentences, we did not make any statement about paragraphs.*
2. Why not use the same classifier for each text hierarchy? *Due to technical limitations, it was not possible to use the same classifier. Kernelexplainer does not support text classifiers properly, during my time working on this thesis, the support for kernelexplainer was dropped due to a multitude of reasons. I can only support that decision from*

the shap development team, kernelexplainer was very difficult to work with, I do not recommend using it to anyone.

3. The dataset is too small. *I agree with this statement, this was sadly due to time and technical limitations running the analysis on the institutes server takes multiple days. Substantially increasing the dataset size would have demanded code optimizations which would most likely have broken the modularity of the framework.*

4. Where is the error bar? *I would have liked to add an error bar to my work, but was asked to prioritize onto the framework. If you wish to verify my work, I invite you to write me an email, I will gladly provide the research data necessary. To get access to the code repository, please contact the Institute for Program Structures and Data Organization (IPD).*

7 Conclusions and Outlook

A lot of work is still ahead of us, but scientists are starting to tackle artificial intelligence with a more open approach, providing fast progress in the field. It is essential to stop trying to analyse and improve machine learning algorithms from a purely mathematical standpoint if we somehow want to stay in control of our systems or want to be able to debug them in the future.

7.1 Interactivity

Human psychology is a very tricky subject to approach, because it can be extremely subjective. While some properties are common to everyone, others are dependent on your culture or experience. I believe that it is imperative, that if we use psychological findings to set parameters in our tools, we need to allow the person using those tools to adapt those parameters to fit not the average human but himself. Like scissors don't come in the average human hand size, but in multiple ones, so should our explainable artificial intelligence not provide one explanation for all of humanity but multiple ones. There have been attempts to implement interactivity in XAI, but they are in their infancy. Based on our findings, the text hierarchy used to provide an explanation for text can be a very useful tool and should be a parameter the XAI user can tweak.

7.2 The future of XAI

XAI will keep becoming more and more of an essential tool. The stronger our computational capacities become, the more complex algorithms can be run. These will impact our lives to unimaginable extent and can only be developed and controlled by developing XAI alongside. While that might seem challenging, I am confident that if we start taking into account interdisciplinary ideas like psychology and philosophy, XAI can make important leaps forward to catch up to where it needs to be.

Bibliography

[1] Scott Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *CoRR* abs/1705.07874 (2017). arXiv: 1705.07874. URL: http://arxiv.org/abs/1705.07874.

[2] CEO of Google Sundai Pichai. *Quote about AI from Google CEO Sundar Pichai.* URL: https://www.cnbc.com/2018/02/01/google-ceo-sundar-pichai-ai-is-more-important-than-fire-electricity.html (visited on 11/24/2020).

[3] *Timeline of AI.* URL: https://en.wikipedia.org/wiki/Timeline_of_artificial_intelligence?oldformat=true (visited on 11/24/2020).

[4] *Performance of the top 500 Supercomputers over time.* URL: https://www.top500.org/statistics/perfdevel/ (visited on 11/24/2020).

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

[6] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015, pp. 1–9.

[7] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[8] Murat Durmus. *5 Methods for XAI.* URL: https://www.aisoma.de/5-methods-for-explainable-ai-xai/ (visited on 11/24/2020).

[9] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30.* Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[10] Arun Das and Paul Rad. *Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey.* 2020. arXiv: 2006.11371 [cs.CV].

[11] Anja. *Isle of Skye: Dunvegan Castle & Gardens.* URL: https://bullitour.com/isle-of-skye-dunvegan-castle-tipps/.

[12] Eugen Lindwurm. *Basics: Gradient*Input as Explanation | by Eugen Lindwurm | Feb, 2021 | Towards Data Science.* URL: https://towardsdatascience.com/basics-gradient-input-as-explanation-bca79bb80de0 (visited on 04/16/2021).

[13] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *CoRR* abs/1602.04938 (2016). arXiv: 1602.04938. URL: http://arxiv.org/abs/1602.04938.

[14] Christoph Molnar. *Interpretable Machine Learning.* Apr. 2021. URL: https://christophm.github.io/interpretable-ml-book/simple.html#simple.

[15] Dan Shiebler. *Understanding Neural Networks with Layerwise Relevance Propagation and Deep Taylor Series*. 2017. URL: http://danshiebler.com/2017-04-16-deep-taylor-lrp/ (visited on 04/18/2021).

[16] Zachary C. Lipton. *The Mythos of Model Interpretability*. 2017. arXiv: 1606.03490 [cs.LG].

[17] *Active learning*. URL: https://www.wikiwand.com/en/Active_learning.

[18] Bryan Mathers. *Post hoc ergo propter hoc*. Jan. 2017. URL: https://medium.com/@BryanMMathers/post-hoc-ergo-propter-hoc-812c39e526fc.

[19] Christoph Molnar. *Interpretable Machine Learning*. May 2021. URL: https://christophm.github.io/interpretable-ml-book/example-based.html.

[20] Scott M. Lundberg. *Welcome to the SHAP documentation*. URL: https://shap.readthedocs.io/en/latest/index.html.

[21] Wang Shugen. "Framework of pattern recognition model based on the cognitive psychology". In: *Geo-spatial Information Science* 5.2 (2002), pp. 74–78. DOI: 10.1007/BF02833890. eprint: https://doi.org/10.1007/BF02833890. URL: https://doi.org/10.1007/BF02833890.

[22] *Meaning of Prisoner's Dilemma With Real-life Examples*. Dec. 2014. URL: https://psychologenie.com/meaning-of-prisoners-dilemma-with-real-life-examples.

[23] Scott M. Lundberg. *Question about missingness · Issue 175 · slundberg/shap*. URL: https://github.com/slundberg/shap/issues/175.

[24] empty. *20 Newsgroups Dataset*. Ed. by empty. 2019. URL: http://people.csail.mit.edu/jrennie/20Newsgroups/.

[25] Adam Hayes. *How Game Theory Works*. Aug. 2020. URL: https://www.investopedia.com/terms/g/gametheory.asp.

[26] Ronald M. Baecker. *Readings in Human-Computer Interaction - Toward the Year 2000*. Elsevier, 2014. ISBN: 978-0-080-51574-8.

[27] Stephen B. Fountain and Karen E. Doyle. "Learning by Chunking". In: *Encyclopedia of the Sciences of Learning*. Ed. by Norbert M. Seel. Boston, MA: Springer US, 2012, pp. 1814–1817. ISBN: 978-1-4419-1428-6. DOI: 10.1007/978-1-4419-1428-6_1042. URL: https://doi.org/10.1007/978-1-4419-1428-6_1042.

[28] *Chunking*. URL: https://www.wikiwand.com/de/Chunking.