Technische Universität München

Department of Mathematics

Master Thesis

Explainability of Hate Speech Detection Models

Edoardo Mosca

Supervisor: Prof. Dr. Georg Groh

Advisor: M.Sc. Maximilian Wich

Submission Date: April 30, 2020

I hereby declare that this thesis is my own work and that no other sources have been used except those clearly indicated and referenced.

Garching,

Abstract

Online hate speech is a large scale phenomenon of great concern in modern society. A considerable amount of research leverages machine learning techniques to propose automatic and accurate detection approaches. While learning directly from the data is a major key for their success in terms of performance, methods are progressively becoming more complex and opaque to humans.
This work combines efforts from explainable artificial intelligence and hate speech detection. From increased social acceptance to compliance with future legislation, we identify interpretability as fundamental for the adoption of learning methods in real-life applications. Our focus lies on post-hoc local methods — we seek to explain non-transparent detection methods on a single prediction-level.
We thoroughly review approaches to explain text classifiers and discuss their theoretical foundation. This includes popular methods like LIME, LRP, and DeepLIFT. Based on our analysis, we encourage the usage of methods belonging to the SHAP interpretability framework.
DeepSHAP is employed to inspect a model trained on the Davidson benchmark for hateful and offensive language. In contrast to earlier analysis, we conclude that the task design itself is a significant cause of the models' limited detection capabilities. In addition, integrating context information, in the form of social features, leads to improved performance in detection models. Interpretability techniques empirically demonstrate that — and also illustrate *why* — social features are indeed the reason for performance gains.
We propose the usage of artificially crafted messages to examine the behaviour of models beyond the dataset they trained on. We show the technique to be effective in bringing to light unintended bias in specific scenarios.
Our results consistently encourage future research to set interpretability as a priority. In particular, we recommend tailoring explanations to the intended target audience to make the most of their potential.

Abstract

Online Hassreden sind ein weit verbreitetes Phänomen, das unsere moderne Gesellschaft vor neue Herausforderungen stellt. In der Forschung wird maschinelles Lernen immer öfter eingesetzt, um Hassreden automatisch und zuverlässig zu erkennen. Der Erfolg von maschinellem Lernen basiert zu einem großen Teil auf dem direkten Lernen anhand von Daten. Modelle werden dadurch jedoch auch zunehmend komplex und undurchschaubar für menschliche Nutzer.
Diese Arbeit kombiniert Erkenntnisse aus dem Bereich der erklärbaren künstlichen Intelligenz und der Erkennung von Hassreden. Wir identifizieren Interpretierbarkeit als unabdingbar, um Methoden des maschinellen Lernens im echten Leben einzusetzen und dabei Akzeptanz in der Gesellschaft und die Erfüllung zukünftiger Gesetze sicherzustellen. Unser Fokus liegt hier auf lokalen post-hoc Methoden — wir möchten intransparente Klassifizierungsmethoden auf dem Level einer Vorhersage für einen bestimmten Input erklären.
Wir besprechen bestehende Herangehensweisen zum Erklären von Textklassifizierungen, auch im Hinblick auf ihre theoretische Basis. Dazu gehören u.A. LIME, LRP und DeepLIFT. Basierend auf unserer Analyse empfehlen wir Methoden der Klasse SHAP.
Mit Hilfe der DeepSHAP Methode analysieren wir ein Modell, das wir anhand der Davidson Benchmark für hasserfüllte und beleidigende Sprache trainieren. Im Gegensatz zu früheren Analysen, ziehen wir den Schluss, dass das Design der Benchmark ein entscheidender Grund für die eingeschränkte Genauigkeit des Modells ist. Des Weiteren stellen wir fest, dass die Zuhilfenahme von Kontext in der Form von Informationen über das soziale Umfeld eines Users zur Verbesserung der Genauigkeit beiträgt. Empirisch lässt sich mithilfe von Interpretierbarkeit zeigen, dass und warum der soziale Kontext in der Tat der Grund für die verbesserte Genauigkeit ist.
Wir schlagen vor künstlich verfasste Textnachrichten zu benutzen, um das Verhalten von Modellen jenseits des Datensatzes, mit dem sie trainiert wurden, zu analysieren. Diese Technik stellt sich als effektives Werkzeug heraus, um impliziten Bias in spezifischen Szenarien aufzudecken.
Unsere Resultate legen nahe, dass die zukünftige Forschung Interpretierbarkeit als Priorität festsetzen sollte. Vor allem empfehlen wir Erklärungen an ihre Adressaten anzupassen, um das Potenzial der erklärbaren künstlichen Intelligenz vollends auszuschöpfen.

Contents

1 Introduction

In modern society, more and more people are connecting via the internet. As a result, online information and communication technologies are increasingly becoming an essential part of everyday life. Particularly in social media, people express themselves freely with few to no restrictions. While free speech and free expression are human rights that should be cultivated, often these media are used as a means to spread hateful content (MacAvaney et al., 2019).

Hate crimes have long existed. With modern technologies, however, abusive behaviour has become an omnipresent phenomenon rather than a sporadic occurrence (Duggan, 2017). According to a 2017 survey (Duggan, 2017), two-thirds of U.S. adults entered in contact with some form of online harassment. The effects on the targeted people can be very harmful: from high levels of anxiety (Duggan, 2017) to depression (Munro, 2011). Online harassment has even been linked to physical crime (Williams, Burnap, Javed, Liu, & Ozalp, 2020).

To counteract this problem, a considerable amount of research has focused on the detection of hate speech (Davidson, Warmsley, Macy, & Weber, 2017; Schmidt & Wiegand, 2017; Waseem, Davidson, Warmsley, & Weber, 2017) resulting in a large variety of recognition approaches (MacAvaney et al., 2019; Mishra, Tredici, Yannakoudakis, & Shutova, 2019a; Rizoiu, Wang, Ferraro, & Suominen, 2019). Many of them leverage the strength of *Artificial Intelligence* (AI) and *Machine Learning* (ML) techniques to automatically extract and learn features from the data itself. Despite current limitations in accuracy, ML-based approaches can drastically reduce the amount of manual effort necessary for monitoring online platforms. At the same time, recent methods have become very complex and opaque for humans (Arrieta et al., 2020). Most famously, *Deep Neural Network* (DNN) have gained popularity due to their success in practice, but behave like black-boxes (Arrieta et al., 2020).

As ML-based technologies are increasingly adopted in human society, more and more people are affected by their predictions. This understandably raises concern, as their predictions cannot be determined nor explained. A new research field, called *eXplainable Artificial Intelligence* (XAI), has emerged and seeks to explain the behaviour of complex and opaque models. From increasing social acceptance to ensuring fairness in the decision-making process, ML research would benefit immensely from interpretable model predictions (Molnar, 2019).

Hate speech recognition is no exception here. There is a clear need to interpret models that can classify content as hateful or not. This becomes even more evident if their predictions are used to implement countermeasures against supposedly hateful users. Despite the large amount of work done in detecting hate speech, interpretability is seldom taken into account. This is especially true for models that address current accuracy limitations by incorporating social network data. Current research shows a considerable increase in

performance when integrating these features as they provide the detection model with context (Mishra, Del Tredici, Yannakoudakis, & Shutova, 2018; Mishra et al., 2019a; Ribeiro, Calais, Santos, Almeida, & Meira Jr, 2018). To the best of our knowledge, the existing literature does not cover explainability for these methods at all. Hence, so far, the effect of social features cannot be quantified, interpreted, nor understood.
In this thesis, we focus on the explainability of hate speech detection. We do so to take a step towards detection methods suitable for real-life scenarios and applications. In this sense, recognition approaches have to be both accurate and interpretable. Hence, we pursue the following research objectives:

O1 Demonstrate the crucial role that interpretability can play in closing the gap between current machine learning research and real-life applications.

O2 Identify the most relevant tools for explaining hate speech detectors without sacrificing prediction performance.

O3 Explain the model's prediction also when social network data is exploited to construct more accurate models. Moreover, it should be clear how these features affect the outcome and the behaviour of the model.

Our work is structured into three main parts. To begin with, in chapter 2, we look at the two main fields that constitute the background of this work: XAI and hate speech detection. Regarding the first, we review the existing literature from multiple venues and examine interpretability concepts from different perspectives. In particular, we seek to answer some fundamental questions regarding XAI: *when is it necessary? How can we define interpretability and other XAI related terms? How can we evaluate an explanation?* We decide to focus more on *post-hoc local* methods — we seek to explain non-transparent models on a single prediction-level, as this is more in-line with our objectives. We then give an introduction to hate speech. We discuss its prevalence in society and why recognising it is a challenging task. Eventually, a brief review is given on existing detection strategies. Many hate speech detectors are text classifiers based on *Natural Language Processing* (NLP) approaches. Hence, in chapter 3, we take an in-depth look at applicable explainability methods (Bach et al., 2015; Lundberg & Lee, 2017; Ribeiro, Singh, & Guestrin, 2016). In particular, we focus on analysing the mathematical ideas and reasoning behind their functioning. We contrast explainability methods and highlight their strengths and weaknesses. Finally, based on our analysis, we provide the reader with a suggestion on which approaches to use.
Next, in chapter 4, we show the crucial role of interpretability in concrete applications. First, we utilise explainability techniques to analyse the Davidson benchmark for hate speech and offensive language (Davidson et al., 2017). Then, in 4.2, we observe how the inclusion of context, in the form of social features, is beneficial for hate speech detection. This is done in three steps. First, our experiments show a considerable leap in performance

when social network data are integrated into a model. Then, their usefulness is empirically proven by explainability methods that show the active role played by social features in the prediction. Finally, on a deeper level, explainability also sheds light on how exactly social features affect what a model has learnt.
The conclusion and final considerations of this work can be found in 5. We also evaluate our achievements in terms of our research objectives and highlight the limitations of our approaches.

2 Background

In this chapter, we discuss the two main research areas that constitute the background of this work: explainable artificial intelligence and hate speech detection. The first serves as a set of valuable tools and methods. The second is the application area to which we contribute. Both domains are relatively new, and their research methodologies have been substantially influenced by the recent technological revolution and the advent of learning algorithms.
To begin with, in 2.1, we discuss several questions regarding explainable artificial intelligence from multiple perspectives: *when is it necessary? How can we define interpretability and other XAI related terms? How can we evaluate an explanation?* We then seek a rigorous answer to each question and include findings from different fields such as philosophy and psychology. As a result, we put together a comprehensive scientific view of interpretability.
Furthermore, in 2.2, we expose the characteristics and severity of online hate speech. We also discuss why its detection is a challenging task and take a closer look at the work done so far in the field.

2.1 Explainable Artificial Intelligence

From autonomous driving to text-to-speech technology, ML-based solutions are increasingly being adopted in many human activities. Its usage allows the automation of a number of tasks in various domains including commerce (Oliver, 1996), law (Možina, Žabkar, Bench-Capon, & Bratko, 2005) and medicine (Kourou, Exarchos, Exarchos, Karamouzis, & Fotiadis, 2015). In several cases, ML techniques are able to perform nearly as good or even better than humans (Silver et al., 2018). This, combined with the increasing availability of adequate software/hardware architectures and experts in the field, has created the perfect environment for this technology to grow very fast.
Naturally, the increasing adoption of ML also corresponds to a growing presence in human society. The impact is especially noticeable in cases where an ML-based system supports decision-making processes. More and more people's lives are affected by predictions that are automatically generated by algorithms (Rudin & Ustun, 2018). This did not generate strong concerns as long as the algorithms used were deterministic and therefore expressed the intention and reasoning of the people who designed them directly. In other words, the behaviour of deterministic algorithms can be entirely controlled by their engineers. Therefore, their outcome incorporates and reflects the human logic of approaching a particular task.
With the advent of ML, the behaviour of algorithms is being learnt increasingly directly from data, and therefore, not wholly determined by their creators. This means that predictions computed by such algorithms might not reflect the intention behind the AI system. Thus, it is natural to wonder why a certain prediction was computed. The first

systems based on ML were quite simple and easy to interpret, e.g. logistic regression (Hosmer Jr, Lemeshow, & Sturdivant, 2013; Jaccard, 2001). However, in recent times more complex architectures such as DNNs have risen to prominence and are intrinsically black-boxes (Arrieta et al., 2020).
Due to the growing concern about the opacity of ML systems, the field of XAI has emerged to interpret the behaviour of learning algorithms. XAI is referred to under many different names such as *Explainability*, *Transparency*, *Comprehensibility* and *Interpretability*. These terms, very often used interchangeably, do not carry the same meaning (Lipton, 2018). For this reason, they will be the object of the discussion in 2.1.1.
In the following, we want to discuss the different aspects of XAI, including their related definitions, purposes, and taxonomies. In particular, based on the existing literature, we seek a rigorous answer to the following questions:

- Why is XAI fundamental for the future of ML in human society?
- In which cases is explainability necessary and when is it not?
- How can we define the terms related to XAI more precisely?
- What different types of explainability techniques exist? What are their characteristics?
- What makes a good explanation? How can we assess it?

In 2.1.1, we tackle the first two questions by analysing in which cases explanations are needed or not. We also discuss how the potential of machine learning is extremely limited if it is not supported by interpretability. In 2.1.2, we look at XAI's related terms. We try to construct rigorous definitions and provide a detailed taxonomy of methods based on these notions. Finally, in 2.1.3, we analyse what makes explanations good and how to evaluate them. For this part, we take the opportunity to include complementary results from other research fields, such as psychology and philosophy. These different venues have collected in the years a considerable amount of valuable work that is too often not taken into consideration.

2.1.1 Why and When is Explainable Artificial Intelligence Important?

In a prediction task, often a trade-off has to be made: are we more interested in **what** is predicted or **why** it is predicted? In some cases, only the first is important and therefore sufficient. However, in other cases also the second can be fundamental rather than just helpful. The necessity of addressing the why component of a prediction becomes more evident as soon as the task presents some particular traits, for example:

- The prediction affects a situation that involves some forms of risk, e.g. a medical decision.

- The field, the task, or the type of prediction are not yet profoundly understood, e.g. stock market forecast.

On the other hand, in a situation with no evident risk involved or in a well-studied context, predictions are usually accepted even without an explanation, e.g. a movie or shopping item recommendation.
At the same time, leaving out the why of a prediction goes against the nature of human curiosity and the classic goal of science: understanding. This occurs if problems are solved simply by using data sets and black-box algorithms. Interpretability, on the contrary, has the potential to transform models into a source of knowledge that can advance scientific research. When humans learn, they seek to understand the reasons behind certain events rather than passively recording what happens around them. When a person feels sick, they will wonder why. Making sense and finding meaning in what is happening lets us adapt our behaviour to the surrounding environment. If a person is sick every time they drink milk, then that person might change their behaviour towards milk. In this sense, when we compare ML predictions to the human perception of knowledge, having a predicted outcome without an interpretation solves the challenge only partially (Molnar, 2019).
Humans value explanations since an explainable decision-making process is more similar to their way of thinking. Therefore interpretability represents a fundamental tool for the integration of machines and algorithms into our daily lives and for their social acceptance (Molnar, 2019). Interestingly, a famous experiment has shown how humans attribute personal traits to the objects they interact with (Heider & Simmel, 1944). In case the personified entity is an ML decision-maker, the system is more likely to be socially acceptable if it gives reasons for its actions.
Social acceptance of decisions strongly relies also on trust, which can be negatively affected when the entity in charge of decision exhibits a bias. ML algorithms have shown to pick up biases from the data used for training (Dixon, Li, Sorensen, Thain, & Vasserman, 2018). Hence, they might operate in a discriminatory way against some groups of people (Bolukbasi, Chang, Zou, Saligrama, & Kalai, 2016). Interpreting models can expose this type of behaviour and serves as a bias detection tool to debug models and data sets. Thus, interpretability is a valuable tool both in the development phase and after deployment. In the former, for debugging, and in the latter to support its decisions with reasons while being operative.
The necessity of interpretability for the broader adoption of ML is not only about social acceptance and similarity to the human and scientific perspective of knowledge. So far, no law restricts the usage of ML-based technologies, but this is likely to change in the future. Some already claim that the European Union 2016 *General Data Protection Regulation* (GDPR) (Regulation, 2016) will not just have a significant impact on the usage of personal data. They also argue that it will also affect the applicability of ML algorithms by imposing a "*right to explanation*" (Goodman & Flaxman, 2017). Such right

would give the ability to users to ask for an explanation regarding an algorithmic decision that was made about them.

It is not yet perfectly clear whether this "*right to explanation*" does already exist and how AI systems should adapt to comply with it (Edwards & Veale, 2017; Wachter, Mittelstadt, & Russell, 2017a; Wachter, Mittelstadt, & Floridi, 2017b). Nonetheless, users' rights and restrictions on non-interpretable algorithmic decisions will be at the centre of future discussion and are likely to appear soon. This is motivated by an urgent need for trustworthiness and transparency of AI systems, especially to make sure they do not operate in a discriminatory way.

In essence: why can we not trust ML algorithms even when they perform well? The problem relies on the fact that performance metrics, such as F1 scores, are an incomplete description of most real-world tasks (Doshi-Velez & Kim, 2017; Molnar, 2019). Interpretability provides us with the tools to increase social acceptance and adoption of ML-based systems. At the same time, it can also enable the usage of models as a valuable source of knowledge. Some works (Ray, Yao, Kumar, Divakaran, & Burachas, 2019; Selvaraju et al., 2019; Strout, Zhang, & Mooney, 2019) already show significant benefits, including improved performance, when using a collaboration between ML models and human supervisors that exchange explanations during the learning process.

2.1.2 Defining Explainable Artificial Intelligence

The field of XAI is, at the same time, new and highly unstructured. Many different terms are used interchangeably as synonyms of *interpretability*. However, although they refer to similar concepts, they may carry a different meaning. This suggests that interpretability is quite far from being a monolithic concept (Lipton, 2018). Instead, it represents a variety of ideas and its intuitive meaning depends on the context. Figure 1 shows how different terms, although nowadays used very often as synonyms, have a different trend in time. Ergo, suggesting a difference in their conceptual purpose.

With no technical or formal definition for interpretability, some works try to define it simply as a requirement to trust a model (Kim, Rudin, & Shah, 2014). Yet, also the concept of trust is not a technically defined notion (Lipton, 2018). In practice, accuracy is often used as a trust measure for a model. This implies that only how often the model is wrong is valued. On the other hand, where it is wrong is neglected. Also trusting only models that make mistakes in regions where also humans do is not enough as a requirement. Although in this case it seems intuitive that models can perform and generalise like humans, their generalisation capacity is limited to the test set. In fact, they lack fundamental properties like transferability and robustness. This is consistently shown in the literature by challenges such as *adversarial examples* (Liang et al., 2018; Szegedy et al., 2014; Yuan, He, Zhu, & Li, 2019) and *domain adaptation* (Ganin & Lempitsky, 2015; Glorot, Bordes, & Bengio, 2011).

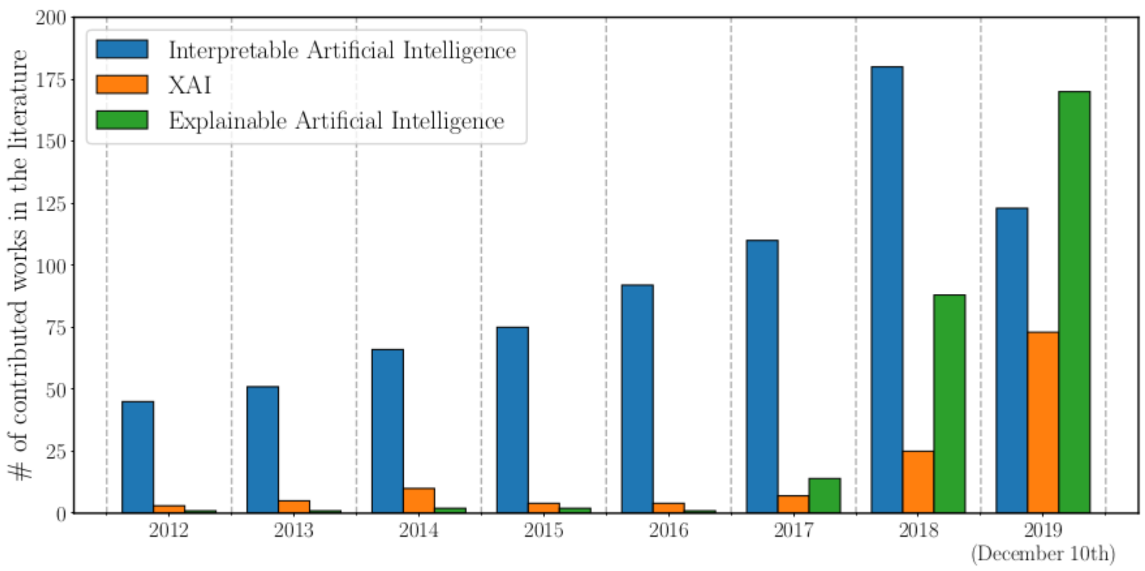


Figure 1: Total publications in recent years whose title, abstract, or keywords refer to the field of XAI. While the research on interpretable AI systems has been growing steadily over time, the term explainable AI became trendy only in recent years. A possible reason for this is the recent success of complex black-box models. Picture originally included in Arrieta et al. (2020, Figure 1).

A more viable approach is to look at the concept of interpretability as too broad to be technically defined. As an alternative, we can identify more specific sub-categories that can be described in detail. In fact, works like Murdoch, Singh, Kumbier, Abbasi-Asl, and Yu (2019) and Lipton (2018) distinguish between two kinds of systems: *transparent models* and *post-hoc explainability*. The first comprehends models that are intrinsically interpretable by design. The latter indicates ones that are explained by means of additional techniques. In this case, model *transparency* is the opposite of *opacity* and *blackbox-ness*. It indicates the presence of some sort of knowledge of how the model works. The level to which we understand the inner workings of the models determines a further classification breakdown:

- *Simulability*, strictly speaking, means that a person should be able to contemplate or even to simulate the entire model at once (Arrieta et al., 2020; Lipton, 2018). Only elementary models (Tibshirani, 1996) belong to this category.
- *Decomposability* stands for the property that each part of the model admits an intuitive explanation. In other words, all the components of the model fall in the previous category when considered in isolation (Arrieta et al., 2020; Lipton, 2018).

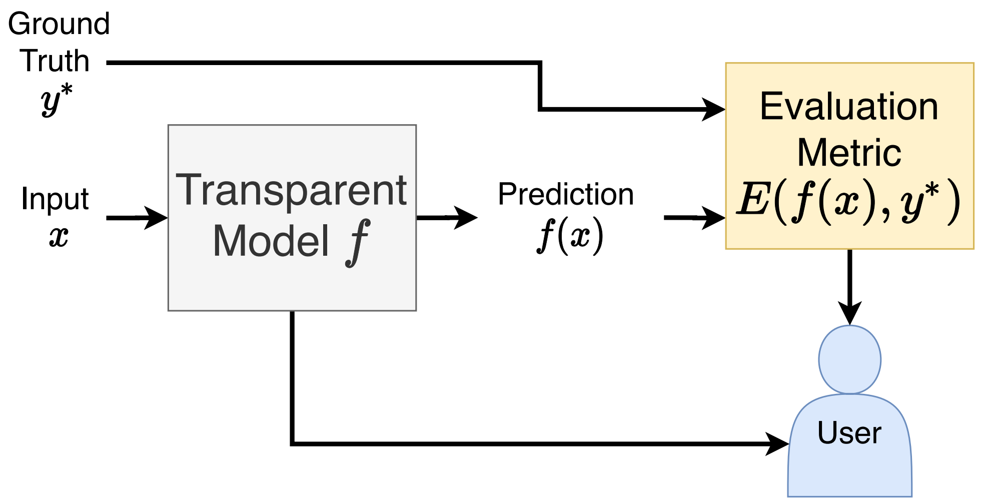


Figure 2: Information's flow when the model is transparent. The user can interpret the model directly and explain it.

- *Algorithmic Transparency* relates to the learning process of a model. Models in this category follow an understandable training process to produce a rule that connects input and output (Arrieta et al., 2020; Lipton, 2018). For example, linear models have a closed-form solution to their optimisation problem. Therefore, they are algorithmically transparent, while DNNs are not.

Although modern complex black-box algorithms do not belong to any of these categories, it is interesting to notice that neither humans do (Lipton, 2018). Figure 2 shows the information flow in a setting that uses a transparent model.
Post-hoc explainability, instead, is not a property of the model itself. It instead describes a setting in which an external method, often referred to as *explanation method*, targets the non-transparent model and extracts information. This information is then used to provide the user with valuable knowledge. Post-hoc methods do not describe the exact inner workings of a model. Instead, they usually confer useful additional information that is presented together with the model's prediction and enhances its interpretability. One of the main advantages of this concept is that it can be applied after-the-fact without sacrificing prediction performance (Lipton, 2018). Figure 3 graphically illustrates a conceptual flowchart of a system using a post-hoc explainability setting.
If we assume that humans are to some extent interpretable, their interpretability is of post-hoc type (Lipton, 2018). When a person makes a decision, they behave like a black-box model. In this sense, nobody can precisely know how the decision is being conceived in their mind. At the same time, the person can then explain their decision by providing some reasons.
The literature (Arrieta et al., 2020; Guidotti et al., 2019; Murdoch et al., 2019) categorises post-hoc approaches under different aspects, based on the type of models they can target,

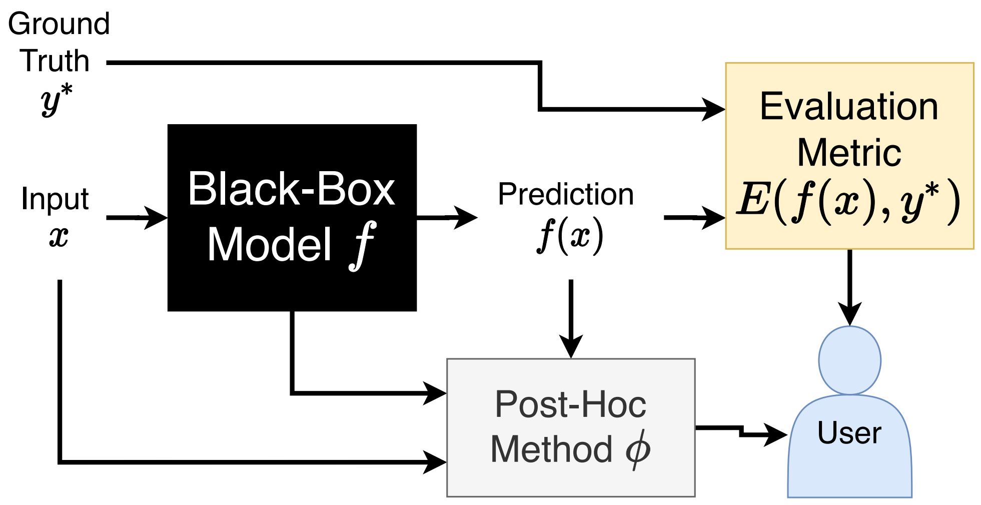


Figure 3: Information flow when the model is a black-box and hinders the user from interpreting it. An external explainability technique is used to find reasons for the model's prediction.

their focus, and the form in which explanations are provided:

- *Model-specific or model-agnostic:* methods belonging to the first category can only be applied to a specific class of methods, e.g. to decision trees or to a DNN. They usually make assumptions on the structure of the model to be interpreted and use particular properties derived from it. Model-agnostic approaches, instead, can provide explanations for any model without making any assumption on its structure. Usually, these approaches only need to perform several runs of the target model without needing access to its structure and parameters (Arrieta et al., 2020; Guidotti et al., 2019).

- *Local or Global:* the two categories refer to prediction-level and dataset-level explanations respectively. Approaches belonging to the first focus on providing reasons for a specific prediction, e.g. which input features contributed to the model's decision. Global methods instead focus on describing the overall model's behaviour and what it has learnt, e.g. in terms of relationships and patterns within the dataset (Arrieta et al., 2020; Guidotti et al., 2019).

- *Explanation Form:* it describes how the explanation is delivered to the user, for example as text or visualisation. Each of them comes with its own advantages and disadvantages. They can also be combined. Usually, the choice is almost exclusively dependent on the context and users' preferences (Arrieta et al., 2020; Murdoch et al., 2019).

This categorisation can be expressed as a taxonomy, summarised in figure 4, useful to better understand XAI related terms. This is done in Arrieta et al. (2020), where the following definitions are provided.

Definition 1 (Interpretability)**.** *"It is defined as the ability to explain or to provide the meaning in understandable terms to a human" (Arrieta et al., 2020, Page 5).*

Definition 2 (Explainability)**.** *"Explainability is associated with the notion of explanation as an interface between humans and a decision maker that is, at the same time, both an accurate proxy of the decision maker and comprehensible to humans" (Arrieta et al., 2020, Page 5).*

Definition 3 (Transparency)**.** *"A model is considered to be transparent if by itself it is understandable. Since a model can feature different degrees of understandability, transparent models [...] are divided into three categories: simulatable models, decomposable models and algorithmically transparent models" (Arrieta et al., 2020, Page 5).*

Although there are certainly many formulations that could be used as definitions of XAI related terms, these terms should not be used interchangeably. Instead, future research should consider their actual meaning and how they relate to a more rigorous taxonomy. For example, while the term *interpretability* appears to be more general, the concepts of *transparency* and *explainability* are closer to what we introduced as *model transparency* and *post-hoc explainabilty* respectively. In general, we also suggest to use a larger variety of terms and integrate more technical notions that belong to a rigorous taxonomy.
In the upcoming sections of this work, 3 and 4, we discuss and work with XAI techniques applied to the field of hate speech recognition. Our focus will be on local post-hoc explainability methods, as we now argue that they are more suitable for our intended application usage.

- **Why post-hoc approaches?** Although some works argue in favour of transparent models (Rudin, 2019), we shall not forget that they are usually tailored to a certain task. Therefore, each field of application and use-case will have different design requirements for interpretable models. Moreover, not knowing all the inner mechanisms of a model and instead providing some reasons for a certain prediction carries more similarities to the human explanation process (Lipton, 2018). Finally, post-hoc approaches do not have a strong impact on the target model's design. This does not discourage the usage of more complex models that potentially perform much better. In 3.2, we also show how the interpretation of transparent models can be seen as a special case of a post-hoc process.

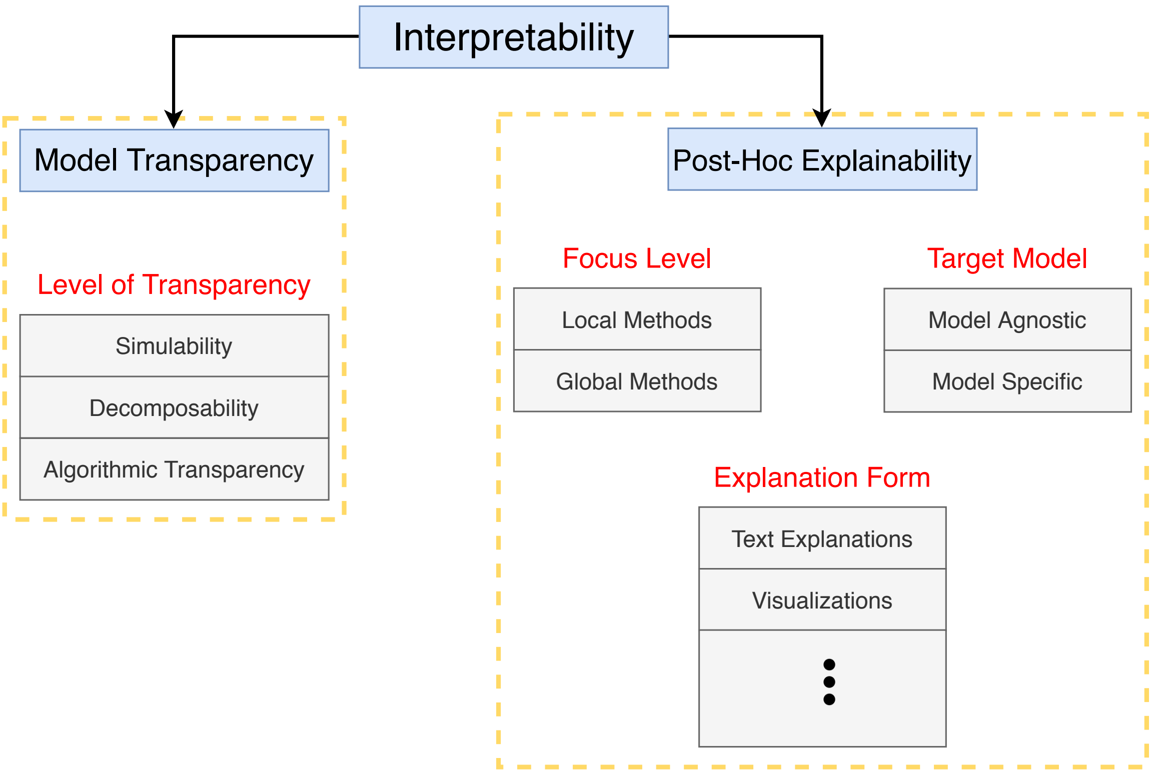


Figure 4: Taxonomy of XAI approaches and terms.

- **Why local methods?** Global methods can indeed gather more information about the whole model they target and eventually even lead to a deeper understanding of it. This type of knowledge is perhaps more useful for domain experts and researchers rather than end-users interfacing with the AI system. Local methods, on the other hand, seek to provide reasons for a single prediction level while leaving out more global connections and relationships that the target model has learnt. We argue that the latter, although perhaps less research-oriented, is more useful for real-world applications also once the model is deployed. For instance, a local method appears quite tailored to support an AI system that has to explain a decision on a user that has requested it, i.e. complying with the "*right to explanation*" (Goodman & Flaxman, 2017).

When it comes to the target model and form of explanations, we do not exclude any category a priori. In the upcoming chapters, we present both model-agnostic and model-specific approaches. Also, several forms of explanations will appear, e.g. a combination of visual artefacts, text, and feature relevance scores.

2.1.3 Good Explanations: Characteristics and Evaluation

So far, we discussed why and in which cases XAI can significantly favour the adoption of fairer ML-based technologies. At the same time, we also built definitions for XAI related concepts through a more rigorous approach. Yet, it remains to be clarified what explanations are and what they should look like in order to be helpful and human-friendly. Notoriously, XAI research has been dominated by those technical fields that also contribute to developing ML technologies, e.g. Computer and Information Science. Miller (2019) observes that most of the existing literature in XAI only relies on the researchers' intuition of what a "*good explanation*" is and are unaware of more rigorous findings coming from other fields. To overcome this, the author suggests that XAI should take advantage of the vast and relevant work that has been done in other fields such as philosophy, psychology, and cognitive science. In fact, there is a considerable amount of valuable research on how people define, present, and perceive explanations. Miller (2019) exhibits major findings from more than 250 publications in the social sciences. In the following, we list some of these findings that we believe should be considered more in XAI research when it comes to building human-friendly explanations.

- Explanations are *contrastive* (Lipton, 1990), i.e. they should expose the difference between two opposing scenarios: the current one and a hypothetical situation in which the predicted event would not have occurred. In other words, as also shown in example 1, people do not ask why a certain event happened, but rather why it happened instead of another one.

- Explanations are *selected*. Full and scientific explanations are usually less desired than short ones that contain only a handful of reasons (Mittelstadt, Russell, & Wachter, 2019). When presented with a long list of factors that influence a certain prediction, humans tend to select one or two and ignore the others. This simplification process follows our personal beliefs and is therefore affected by our cognitive bias (Miller, 2019).

- Explanations are *social*, i.e. they represent a specific case of transfer knowledge between two individuals within a social context (Miller, 2019). Therefore, in addition to the user's bias, explanations are also affected by the context in which they occur.

Example 1. *A person's loan is rejected based on a model that took into consideration some personal features. We expect that the person is not interested in the entire list of reasons. Preferably, this person wants to know which factors they would need to change for their loan to be accepted. At the same time, if a list of reasons for the rejection is shown to the person, they might ignore part of it. Namely, they might only focus on a few items depending on their cognitive bias and the current social context. For example. they could select 1-2 reasons that they find unfair.*

In general, the desirable explanation properties of an AI system (*trust, causality, fairness, privacy awareness* (Arrieta et al., 2020)) are highly dependent on the interacting entities and the social context. Hence, they are hard to define a priori. However, once

these requirements and goals are defined for a specific task, how can one evaluate the effectiveness of an explanation?
When considering an ML algorithm, assessing the performance can be done through several metrics on the test set. Common examples are accuracy, precision and F1 score. Although the discussion on which metric is the best is still open, different metrics share the ability to measure the performance with a numerical score. This allows objective comparisons between task methods and the creation of a benchmark for each task. For hate speech detections, popular examples are Waseem (2016) and Founta et al. (2018).
Unfortunately, for interpretability, there are no obvious performance metrics that quantify with a numerical score how good an explanation is. This restricts the options for objective comparison and may limit the choice of a certain framework to be based only on subjective preferences. To address this, recent works build more complex alternative ways, called explanation evaluation paradigms, which can attribute performance scores also to explanation methods (Mohseni & Ragan, 2018; Poerner, Schütze, & Roth, 2018). These new paradigms contributed to the foundation of a new field: explanation evaluation (Doshi-Velez & Kim, 2017).
For ML algorithms, evaluation metrics and methods can be found in organised taxonomies (Japkowicz & Shah, 2011). Every evaluation procedure should quantify the contribution of the framework that it evaluates. In the same spirit, Doshi-Velez and Kim (2017) builds an analogous taxonomy of evaluation approaches for interpretability, also summarised in figure 5:

- *Application-grounded (real humans, real tasks):* it is the direct application and evaluation of the method in its intended task. This often involves domain experts in concrete scenarios, e.g. doctors using an ML framework as a decision support tool (Herron, 2004). This type of approach is very well aligned with the goals of explainability, i.e. with verifying that the ML framework delivers w.r.t its intended purpose (Doshi-Velez & Kim, 2017).

- *Human-grounded (real humans, simplified tasks):* instead of applying the method directly to the target application, simpler human-subject experiments with the same character and goals are conducted (Doshi-Velez & Kim, 2017). One example is Mohseni and Ragan (2018), where humans ranked explanations for text and image classifiers. Naturally, evaluations of this kind cannot be as specific as the application-grounded ones. However, they permit to relax many of the constraints (availability of domain experts, costs, time) that reduce the direct applicability in a real and concrete scenario. On the other hand, preserving the end-goal when shifting to a more general and abstract experiment design can indeed be a challenge. Nevertheless, in some cases, it provides flexibility w.r.t some design factors such as task complexity (Kim, Chacha, & Shah, 2015).

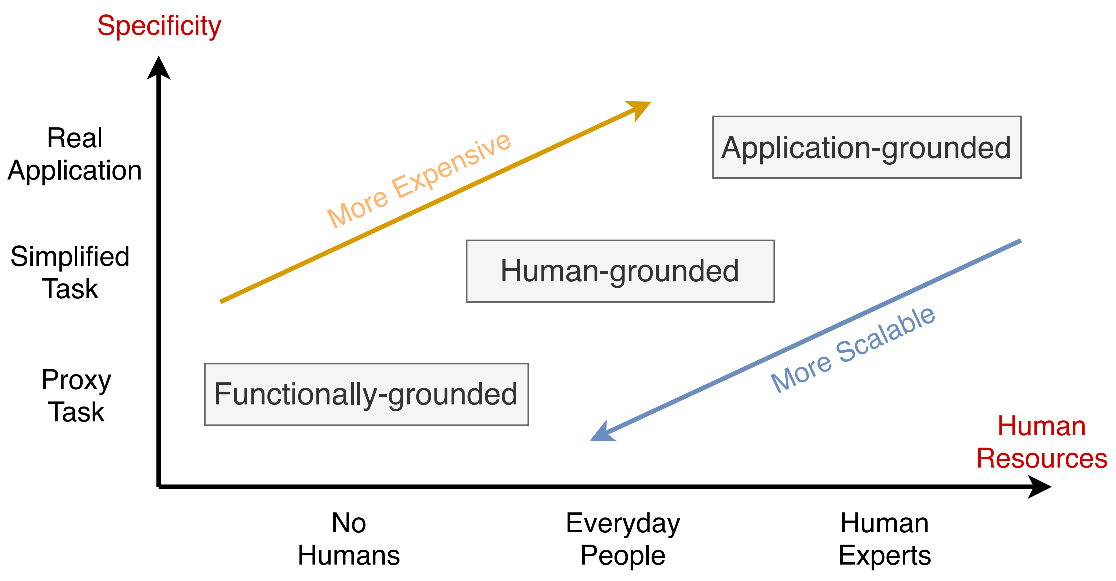


Figure 5: Three types of explanation evaluation techniques compared on different metrics. On the x-axis, the human resources necessary to implement the experiments. On the y-axis, the specificity of the task used for the evaluation.

- *Functionally-grounded (no humans, proxy tasks):* by using some formal definition of interpretability, these approaches build a proxy task that can assess explanation quality without involving human subjects at all. For instance, in Arras, Horn, Montavon, Müller, and Samek (2016), a proxy task is designed for evaluating explanations for NLP algorithms by altering the input documents (concretely deleting and inserting words). On the one hand, the absence of the need for a pool of human subjects removes all the related costs and makes functionally-grounded approaches extremely scalable. On the other hand, formally defining interpretability and explanation quality for a certain task can be very hard. Moreover, designing a proxy task that reflects these notions is a major challenge and often a limitation to these techniques (Doshi-Velez & Kim, 2017).

Evidently, the different types of evaluation paradigms differ substantially and imply completely different design processes for their experiment, some of which can be costly. Therefore, it is crucial to understand beforehand which setting applies more to a particular domain. So, how can we choose which one is more suitable?
Of course, as with most of the other questions in this thesis, there are no yes or no answers. Nonetheless, observing factors and limitations provides a good indication of what type of evaluation is more suitable or rather feasible. Some examples are *time constraints, availability of experts or a human subjects pool, financial costs, task generality, and opportunities for proxies/simplifications* (Doshi-Velez & Kim, 2017).
Application-grounded evaluations test if a certain AI system is in line with its intended purpose. However, they often require the design of very high-quality experimental settings

that can be very costly w.r.t. several resources: financial, human, time. Hence, although performance concerning real objectives certainly gives strong evidence of success (Doshi-Velez & Kim, 2017), cost factors play a huge role and cannot be underestimated.

Human-grounded evaluations enable the researcher to design experiments that can be carried out with humans with no particular expertise in the domain of application. This allows for both a bigger subject pool and fewer expenses since there are most likely no highly-trained domain experts to be compensated. Human-grounded evaluation is most appropriate when testing more general notions regarding the quality of explanations (Doshi-Velez & Kim, 2017).

Functionally-grounded evaluations are, with no doubts, the most inexpensive and scalable option. They are also appropriate or rather the only possible path when an experiment cannot involve human subjects because this would be dangerous or unethical. But at the same time, proxy tasks can reflect the real application's goal inaccurately.

To sum up, in the presence of strict cost constraints and needs for scalability, functionally-grounded approaches outperform the two alternatives and are usually preferred if viable. Nevertheless, the type of evaluation must match the claim being made by the research (Doshi-Velez & Kim, 2017). Hence, in some cases, involving humans in the process is not only costly and helpful, but also necessary. As we will see in the following section, hate speech detection is hard due to the different perceptions that people have of hate. Hence, when evaluating explanation algorithms in this application field, involving humans seems a more appropriate choice in most cases.

2.2 Hate Speech Detection

In recent times, with the advent of social media networks and other types of online platforms that incorporate user-generated content, online communities have witnessed an increasing phenomenon of abusive and hateful behaviour between their members. The effects can be devastating for the people targeted: high levels of anxiety and stress (Duggan, 2017) and, in some cases, depression (Munro, 2011). Williams et al. (2020) also links online harassment to physical crime.

A closer look at the statistics shows a concerning situation. It demonstrates that online hostile behaviour is more than a sporadic occurrence. In fact, 66% of U.S. adults have witnessed some form of harassment online (Duggan, 2017). Given the prevalence of the phenomenon, there is no doubt that manual controls and prevention is by no means an effective solution.

The urgent need for automated and scalable methods for tackling online abuse motivated intensive research in the field. Numerous works such as Cho et al. (2014), Djuric et al. (2015) and Mishra et al. (2018) have contributed to the existing literature giving birth to continuously evolving research in hate speech detection.

In 2.2.1, we discuss general aspects of hate speech, including notions and intuitions on why its detection is an arduous task. In 2.2.2, we review the main strategies that can be found in the existing literature for recognising hate speech. Furthermore, we organise the methods in a time-line and emphasise the milestones in the research.

2.2.1 General Aspects and Challenges

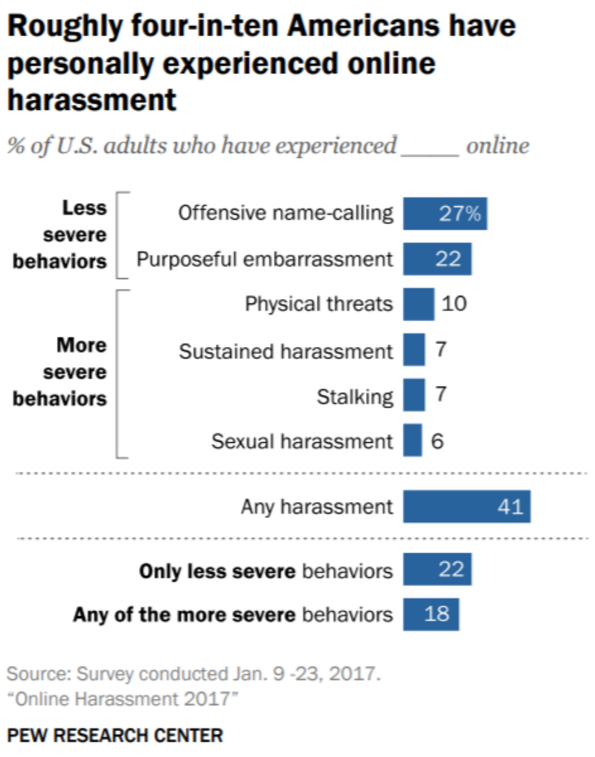
Several terms can be found in the existing research to indicate hate speech content: *abusive*, *hateful*, *flames*, *harassment*, *toxic*. Although some of them differ slightly in meaning, we do not discuss these differences as done in 2.1.2 for interpretability. We choose to do so since, unlike different notions within XAI, the difference between hate speech related terms does not correspond to any difference in the recognition task goal. Thus, in this work, we use these terms interchangeably. We refer to hate speech (and its synonyms) as any communication that denigrates a single person or a group of people based on some characteristics, e.g. gender, sexual orientation, nationality, religion (Schmidt & Wiegand, 2017).

A survey conducted by the Pew Research Center on 4,248 U.S. adults reveals the scale at which online hate speech occurs (Duggan, 2017). According to their results, 41% of Americans have personally experienced harassment. 18% has even been subjected to some severe form of it. If we also include people who witnessed online abuse without being the direct target, we observe that about two-in-three adults (66%) have experienced some form of hate speech. For completeness, we report the results from Duggan (2017) about harassment severity and pretext in the figures 6a and 6b. One can observe how the pretext and severity of abusive conduct vary considerably.

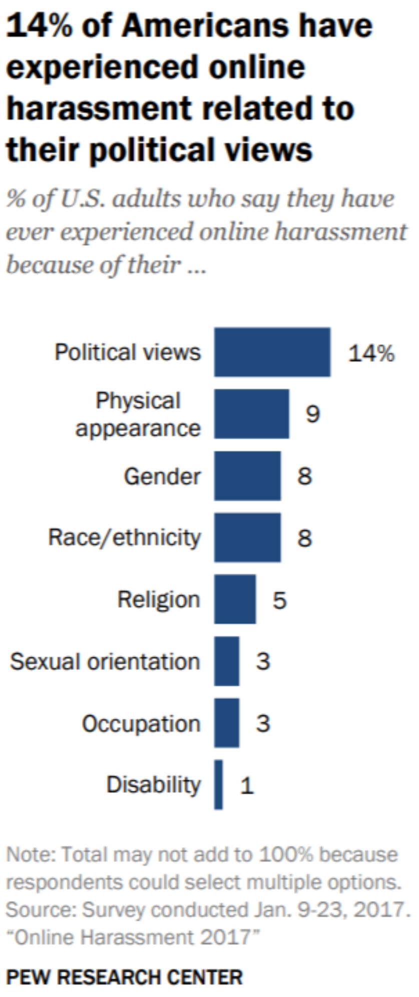
Despite the evidence for the large extension and effects of hate speech, it is often hard to agree on what content constitutes hate speech. Each community has diverse norms to establish what is abusive and what is not and may have a different perspective on the gravity of online harassment. For instance, 7 shows how different groups of people, in this case separated by gender, have a divergent perspective on hate speech online (Duggan, 2017).

Can we distinguish different forms of hate? Waseem et al. (2017) classifies hate based on *explicitness*:

- *Explicit:* usually straightforward to recognise, it uses offensive terms such as insults or threats (Waseem et al., 2017).
- *Implicit:* it comes in a more subtle form. In most cases does not contain explicit terms (Waseem et al., 2017). Instead, hate appears behind ambiguous terms and figures of speech such as metaphor and sarcasm (Mishra, Yannakoudakis, & Shutova, 2019b)



(a) Severity of Harassment



(b) Pretext for the Harassment

Figure 6: Pictures originally in Duggan (2017, Page 3, Page 17). Statistical results on severity (a) and pretext (b) of online harassment.

and on *directness*:

- *Directed:* abuse targets a specific individual and points at personal information and traits of the target (Waseem et al., 2017).

- *Generalised:* it abstracts from individuals and instead targets a larger group of people. Usually, the target group has a particular trait in common among its members, e.g. gender, ethnicity, or political orientation (Waseem et al., 2017).

Sometimes, generalised forms of hate speech have a specific name. For example, the term *sexism* is used to describe hate speech towards women and *racism* usually indicates abuse towards people with a non-western ethnicity.

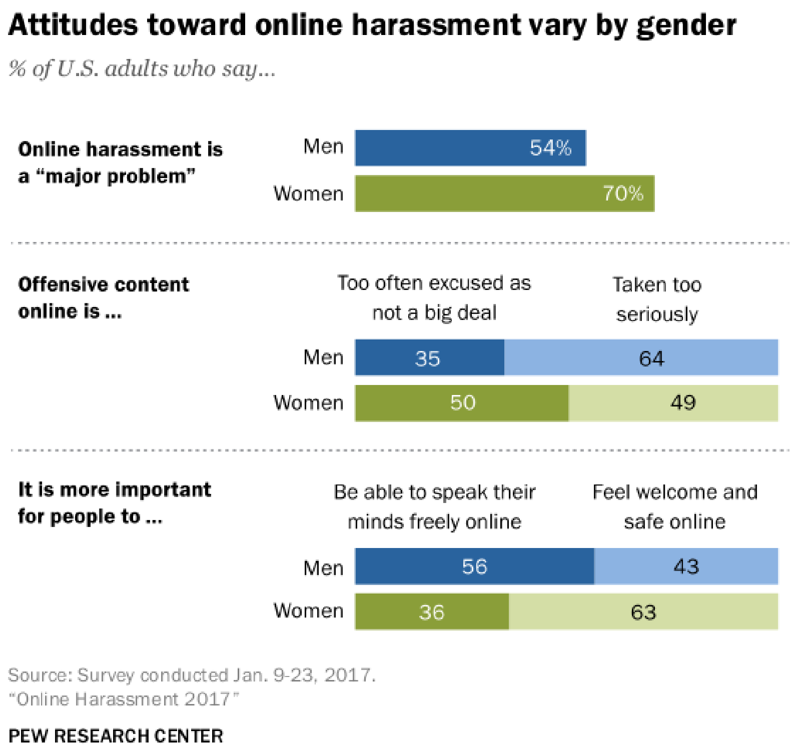


Figure 7: The perception of hate differs between two groups, e.g. genders. Figure originally in Duggan (2017, Page 7).

Needless to say, this categorisation already exposes the complexity and intricacy of capturing hate speech. When one attempts to recognise it, categories like explicit and directed are easier to detect. In contrast, implicit and generalised hate is far more challenging to identify (Waseem et al., 2017). The existing literature also shows how, depending on the type of hate, even the labelling process carries a substantial disagreement between human annotators. Dadvar, Trieschnigg, Ordelman, and de Jong (2013) reports a very high agreement on cyber-bullying content (direct and explicit). In contrast, Waseem (2016) reports a significant disagreement, especially when it comes to sexism (generalised and often implicit). Simply put, in the presence of a more subtle expression of hate, annotators tend to not agree with each other. As a consequence, the labelling process becomes complicated.

Even with an excellent automatic solution to detect abusive behaviour that enables online platforms to counteract it, there are still challenges to be considered. As argued in 2.1.1, the future will most likely see emerging laws that support and protect end-users that have been affected by an algorithmic decision, i.e. the "*right to explanation*". Hate speech detection, where an algorithm is employed by a social media platform like Twitter, will need to be interpretable if used. As soon as the platform applies a countermeasure against the supposedly hateful user (e.g. content deletion, account block), the service should be able to provide the affected user with an explanation of why their content was judged abusive. Therefore, hate speech recognition surely requires interpretability to be successfully applied in practice.

Besides potential law enforcement, the ability to explain a hate speech classifier also serves the purpose of assessing or rather confirming the correct functioning of the detection algorithm in use. We cannot exclude that the supposedly hateful user is, in reality, a victim of the model's bias. Within this respect, it is important to distinguish *unintended bias* (Dixon et al., 2018) from the more general concept of *bias/fairness* (Feldman, Friedler, Moeller, Scheidegger, & Venkatasubramanian, 2015). Bias, in general, is necessary for the model to recognise hate and to make decisions. Unintended bias, on the other hand, indicates a different behaviour of the model against some groups of people and should be avoided (Dixon et al., 2018). This problem is studied in-depth in Dixon et al. (2018), where solutions are proposed for future research. Concretely, the authors discuss a definition of unintended bias specifically for hate speech detection and present strategies for quantifying it and mitigating it in datasets and models.

2.2.2 Evolution of Detection Strategies

When we look at hate speech detection models, a transition of the *State Of The Art* (SOTA) has taken place in recent years. We went from methods that use manual feature extraction and hand-crafted rules (Spertus, 1997) to approaches that involve more automated feature extraction processes such as DNNs (Badjatiya, Gupta, Gupta, & Varma, 2017). Recent times have also witnessed the introduction of methods that do not rely solely on text input and instead use additional features such as user and network data (Mishra et al., 2018). In the following, we discuss some of the main contributions to hate speech detection, which in figure 8 are also shown compactly in a timeline.

The first documented detection approach for hateful messages dates back to 1997 (Spertus, 1997). The proposed method extracts features manually from the raw data using a rule-based heuristic approach. The extracted features are then fed into a Decision-Tree classifier that outputs whether the message is hostile or not. While the approach already reaches a high accuracy on non-abusive messages and reaches a recall of 98%, it performs poorly on the abusive ones, where it only achieves a 64% recall score. The performance leap is caused by the difference between the amount of hateful tweets, referred to as *flame*, and non-hateful ones, referred to as *okay*. Spertus (1997) also notes some specific limitations of their method, such as *sarcasm, mistakes, innuendo*. They consider these limitations as intrinsic of the task and not caused by the current NLP recognition technology. In particular, even human readers sometimes cannot recognise sarcastic expressions with confidence.

The usage of lexicon-based features appears only 13 years later in Razavi, Inkpen, Uritsky, and Matwin (2010). The proposed method extracts features at different conceptual levels by building a dictionary of words and phrases in which every entry is weighted with

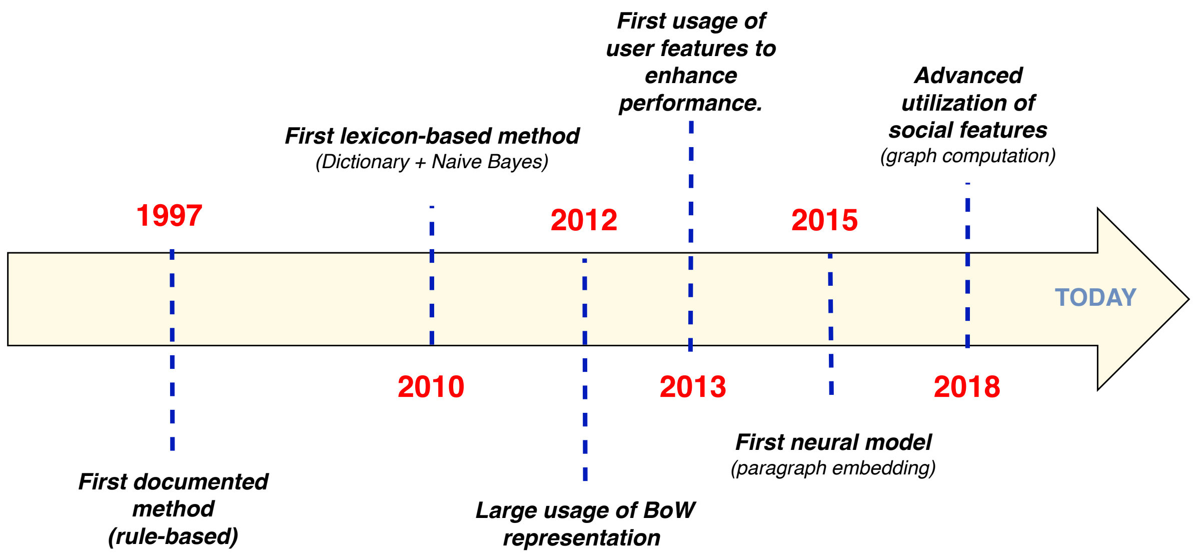

First usage of user features to enhance performance.
Advanced utilization of social features
(graph computation)
First lexicon-based method
(Dictionary + Naive Bayes)
1997
2012
2015
TODAY
2010
2013
2018
First neural model
(paragraph embedding)
First documented method
(rule-based)
Large usage of BoW representation

Figure 8: Milestones in the field of hate speech detection in chronological order. Recent times, in particular from 2012 onwards, have seen a substantial increment in the number of recognition approaches. For reference, Facebook and Twitter were launched in 2004 and 2006 respectively.

an abusive score. Every level is then fed to an independent Naive Bayes classifier (Complement (Witten & Frank, 2002) for the first level, Multinomial Updatable (Witten & Frank, 2002) for the second and Decision Tree/Naive Bayes hybrid (Hall & Frank, 2008) for the last). Similar approaches, such as Gitari, Zuping, Damien, and Long (2015), also emerged in the following years and kept improving in terms of performance. Nevertheless, the direct usage of a lexicon tends to work well only when the type of abuse/hate is explicit and expressed through specific words (Mishra et al., 2019b).

For any text classification task, it is quite a popular choice to use *Bag of Words* (BoW) for representing text. In fact, from 2012, this representation appears very often in hate speech models (Chen, Zhou, Zhu, & Xu, 2012; Sood, Antin, & Churchill, 2012; Warner & Hirschberg, 2012; Xu, Jun, Zhu, & Bellmore, 2012). While using BoW features usually yields a good classification performance, the effectiveness of this representation highly depends on whether predictive words appear both in training and test set. Moreover, data features represented as BoW may face a data sparsity problem due to the restricted length of text inputs for hate speech detection tasks (Schmidt & Wiegand, 2017).

More recently, with the large adoption of DNN algorithms, a technique called *word embedding* (Mikolov, Chen, Corrado, & Dean, 2013), has been proposed as an improved representation for words and phrases. For each word (or sentence, e.g. *paragraph embedding* (Le & Mikolov, 2014)) a numerical vector representation is computed by a DNN. This representation has the advantage that different words with similar semantics may also end up having similar embedded vectors. Djuric et al. (2015) were the first to adopt a technique of this type for abusive language recognition. In their case, a paragraph embedding was used to compute intermediate features which were then classified by a logistic regressor, outperforming previous BoW-based methods. As expected, in the following years, approaches based on neural models became strong contenders as SOTA methods (Badjatiya et al., 2017).

Other works, starting with Dadvar et al. (2013), took a different path and considered including social features and personal traits alongside text. Some examples are the user's gender (Waseem, 2016), or the geo-location and the profile's language (Galán-García, Puerta, Gómez, Santos, & Bringas, 2016). In recent literature, one can find works that also take into account social communities and treat them as graph input to enhance prediction performance. For instance, Mishra et al. (2018) and Mishra et al. (2019a) create a community graph that encodes the social environment of users. This graph is then either embedded as a matrix or fed directly to a novel DNN technique: *Graph Convolutional Neural Networks* (Hamilton, Ying, & Leskovec, 2017).

Although a considerable amount of work has been done in the realm of hate speech detection, the same cannot be said about interpretability for hate speech detection. For now, only a few contributions to the literature discuss interpretable solutions to hate speech recognition as more than a side note.

For instance, MacAvaney et al. (2019) proposes a transparent method for hate speech detection that uses a combination of simple classifiers. Concretely, their *multi-view Support Vector Machine* model combines several linear support vector machine classifiers. The model achieves nearly SOTA performance and, at the same time, can be directly interpreted. In fact, one can check which one of the different individual classifiers contributes the most to the model's prediction. Rizoiu et al. (2019), instead, builds a 2-dimensional representation of tweets from the model's latent space called *the map of hate*. The map is used to understand the effect of transfer learning on a model. However, the technique is then not used to explain any prediction.

The small amount of available interpretable tools for hate speech detection motivates our work. In particular, in 4, we also discuss how interpretability can be combined with techniques that include social features.

3 Local Explainability for Text Hate Speech Classifiers: a Mathematical Analysis

In this chapter, we explore methods for local explanations that are applicable to NLP or, more specifically, to hate speech detection models.
Our aim is not only to gain insights into the theoretical aspects and implications of existing methods. We also seek an answer to the question of which explanation approaches are more relevant for the hate speech detection use-case. As we will see, some modifications of the methods will be necessary to adapt them to different possible representations for text data. Two new concepts, *local interpretable representation* and *localised linear model*, are introduced to facilitate such adaptation and to exhibit some interesting properties that further clarify the existing work in the field.
The approaches that we consider come from several sources and therefore present a substantially different notation. Hence, it becomes harder to compare existing methods and, as a consequence, prefer one over the others. For this reason, we analyse existing methods with a common and consistent notation that we define a priori. Also, through specific examples, we examine how they relate to our use-case context.
To begin with, we summarise the main symbols that build up our consistent notation. In 3.1, we introduce the basic notions and definitions that characterise our analysis setting. In 3.2, we look at the simplest possible case: the model to be explained is already interpretable. To start with explanation methods, in 3.3, we describe input perturbation methods (Kádár, Chrupała, & Alishahi, 2017; Li, Monroe, & Jurafsky, 2016). These procedures systematically analyse the response of a model to a specific alteration of the input. Then, in 3.4, we describe backpropagation methods. Such approaches explain a model's prediction by propagating a quantity called relevance from the output back to the input. Some of them, called gradient methods (Denil, Demiraj, & De Freitas, 2014; Simonyan, Vedaldi, & Zisserman, 2014; Sundararajan, Taly, & Yan, 2017), achieve this by computing partial derivatives. Others, like LRP (Bach et al., 2015) and DeepLIFT (Shrikumar, Greenside, & Kundaje, 2017), are more specific for explaining neural networks and define a set of rules that take advantage of their layer structure. Afterwards, we explore how complex models can be approximated globally or locally with simpler ones in 3.5. We also argue in favour of local approaches and present a popular method: LIME (Ribeiro et al., 2016).
Considering the high number and variety of explanation methods, in 3.6, we present a unified view of existing approaches that goes under the name SHAP (Lundberg & Lee, 2017). This new perspective notably simplifies the space of existing approaches. Therefore, it provides us with a tool to support our recommendation to the reader. Moreover, it supports the existence of a general theory for explanation methods. Finally, in 3.7, we draw our conclusion on which approaches to prefer.
Figure 9 summarises conceptually the methods that we will explore in the chapter.

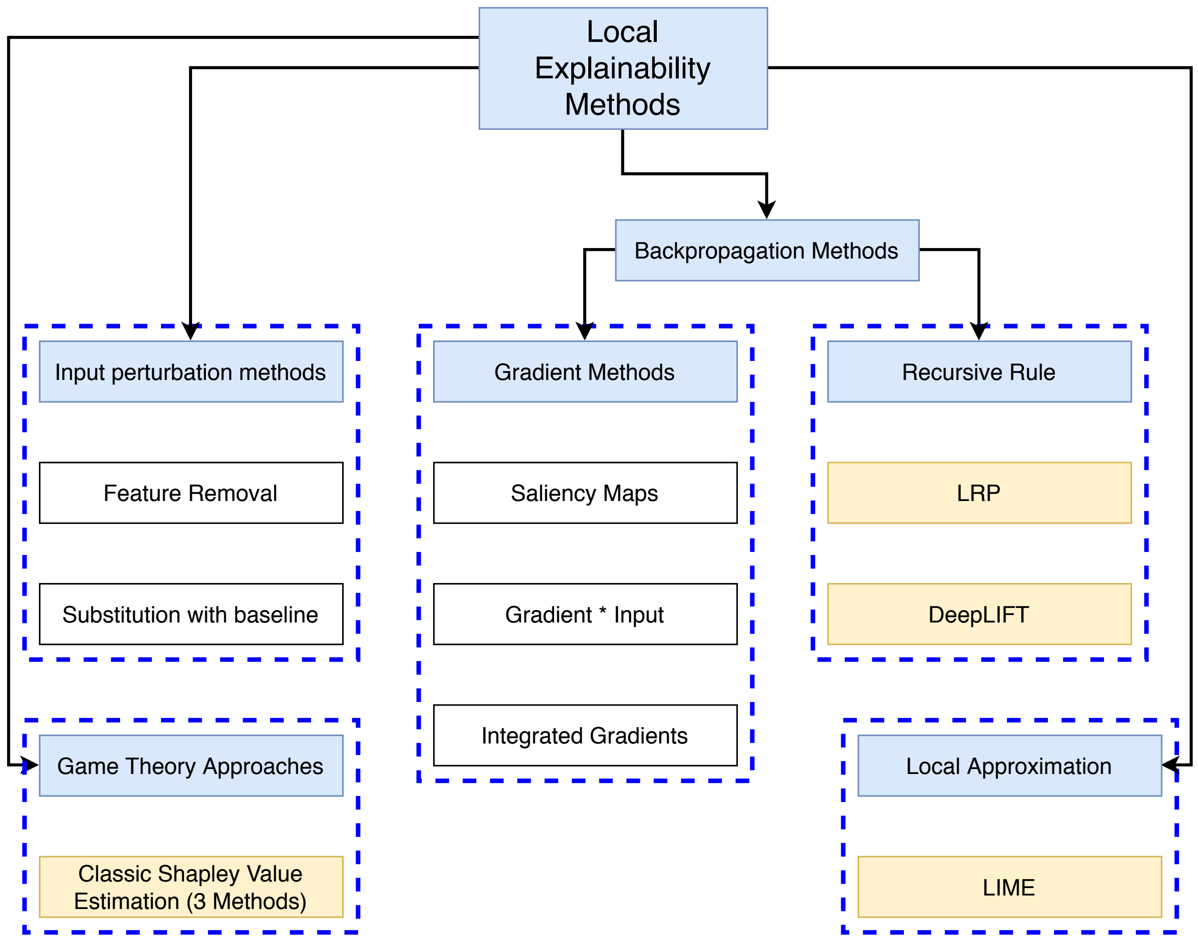


Figure 9: Conceptual map of the explanation methods considered in this chapter. In yellow, methods that will be proven to be equivalent in 3.6. Methods that are built for very specific architectures or activation functions, e.g Grad-CAM (Selvaraju et al., 2017), Deconvolutional Network (Zeiler & Fergus, 2014), Guided Backpropagation (Springenberg, Dosovitskiy, Brox, & Riedmiller, 2015), and Cell Decomposition for Gated RNNs (Murdoch & Szlam, 2017), are out of the scope of this analysis.

Used Symbols

Here are listed all the main symbols that will be defined and used in the chapter. A very brief description accompanies every symbol. A more complete definition is given the first time that the symbol in question appears in the chapter.

x	Input sentence.
x'	Binary interpretable data representation of x
$\|x\|$	Length of the sentence x.
X	Space of input sentences.
t	Position in a sentence.
x_t	Word in position t in sentence x.
V	Vocabulary, i.e. set of available words.
c	Speech class.
C	Set of speech classes in the classification task.
c^*	Predicted class.
f	Task method, text classifier.
F	Space of classifier functions.
g	Interpretable classifier.
G	Space of interpretable classifiers.
ϕ_{name}	Explanation method, accompanied by the method's *name*
$\phi(t, c, x)$	Relevance score w.r.t. class c of word x_t.
z (or z')	Fake data sample obtained by perturbing x (or x')
ϕ_t	Shapley value of word/feature x_t.
l	Layer in neural network task method.
u	Unit in a layer of a neural network.
a_u	Activation of unit u.
w_{uv}	Weight between the units u and v.
σ	Activation function, e.g. sigmoid.
$R_u^{(l)}$	Relevance for unit u in layer l.

3.1 General Definitions

Let us look at a sentence x as an ordered sequence of words (or more in general an ordered sequence of tokens) x_t, that is:

$$x = \begin{bmatrix} x_1, & x_2, & \cdots & x_n \end{bmatrix}.$$

We say that x_t is in position t within x and, if we let $|x|$ denote the length of x, then we have that $1 \leq t \leq |x|$. X will indicate the space of all sentences and V the vocabulary, i.e. the set of all available words.
Since hate speech detection is a task that adopts the classification setting, we also require an output notation for our classes. Let us then write $C = \{c_1, .., c_{|C|}\}$ for the set of considered classes as which input sentences, i.e. elements of X, can be classified.

Definition 4. *We call a function $f : X \longrightarrow \mathbb{R}^{|C|}$ a* task method *or* classifier *if, taken $x \in X$, $f(x)$ is a $|C|$-dimensional vector containing the predicted scores $f(x)_c$ for every $c \in C$.*

Remark 1. *With $f(x)_c$ we mean the normalised class scores, i.e. the class probabilities $\mathbb{P}(c \mid x)$. They can simply be obtained from the unnormalised scores $S_c(x)$:*

$$f(x)_c = \mathbb{P}(c \mid x) = \frac{exp(S_c(x))}{\sum_{c' \in C} exp(S_{c'}(x))}.$$

In case the task of the classifier is hate speech detection, we can also refer to f as *hate speech detection model.* Examples of commonly used task methods in NLP are *Long Short-Term Memory* (LSTM) (Hochreiter & Schmidhuber, 1997), *Gated Recurrent Unit* (GRU) (Cho et al., 2014), and *attention models* (Vaswani et al., 2017).

Definition 5. *Given a task method f and an input sentence $x \in X$. We say that $c^* \in C$ is the* predicted class*, or more simply* prediction*, if*

$$c^* = \arg\max_{c \in C} f(x)_c$$

In practice, to allow computation of symbolic input such as text, words and sentences are often encoded in numerical vectors. This can be achieved in several manners. One of the most common ones is to use a BoW representation $x \in \{0,1\}^{|V|}$. Alternatively, words can be encoded as word embeddings, i.e. as dense vectors $x_t \in \mathbb{R}^m$ (Mikolov et al., 2013; Pennington, Socher, & Manning, 2014). The embedding dimension m is chosen arbitrarily and, to represent a sentence x, encoded words can be stacked together in a matrix, i.e. $x \in \mathbb{R}^{|x| \times m}$.
The way text is encoded does not play a minor role. On the contrary, it can be crucial for the success of a task model. For instance, Dhingra, Liu, Salakhutdinov, and Cohen (2017) shows how initialisation details in the encoding can be more relevant w.r.t. performance than the classification model itself.

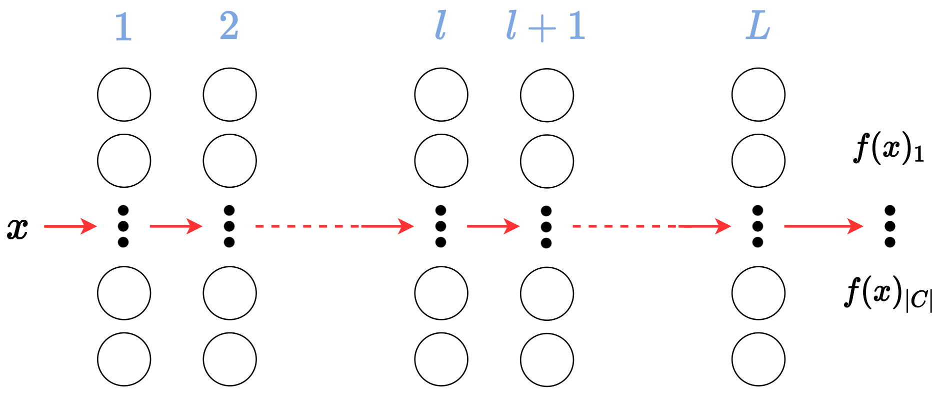


Figure 10: Notation introduced for layers in a DNN.

We will use the symbol x_t to denote a word independently of the used encoding. In the case in which no particular notation is specified, words can be assumed to be encoded through an m-dimensional embedding, i.e. $x_t \in \mathbb{R}^m$. Later, in 3.2, more will follow on this.

Definition 6. *Given a task method f, an input sentence $x \in X$ and a class $c \in C$, an* explanation method $\phi : \{1, \dots, |x|\} \times C \times X \to \mathbb{R}$ *produces word relevance scores $\phi(t, c, x)$ for every word position t. That is, how relevant the word in position t would be for f to classify x as belonging to class c.*

In real-world applications, as given class c, it would often make sense to pick the prediction c^* of the method f. This way, ϕ can be used to give reasons for the decisions taken by f in an automated process. Here follows an example that should further make the introduced notation clear.

Example 2. *Let us consider the sexist sentence $x =$* "women should always stay in the kitchen" *and a classifier f that predicts $c^* =$* "hate speech"*, we might expect for instance that our explanation method ϕ will consider the words $x_1 =$* "women" *and $x_7 =$* "kitchen" *more relevant than the word $x_4 =$* "stay"*. With the introduced notation, this translates to $\phi(1, c^*, x), \phi(7, c^*, x) > \phi(4, c^*, x)$.*

Due to the recent success of DNNs and their intrinsic black-box behaviour, some explanation methods are specifically designed to treat them (Bach et al., 2015; Shrikumar et al., 2017). Hence, w.r.t. our mathematical analysis, it raises the need to adopt additional symbols for the DNNs' components that we are going to work with. This extension in the notation aims to simplify the detailed description of a DNN f.

Definition 7. *Let f be a DNN task method. We describe f as a composition of* layers, *each of which is composed by a set of* units. *We use l to indicate a layer and u for a unit. We can also write $u \in l$ if the unit u belongs to the layer l. Then, with a_u and w_{uv} we will respectively indicate the* activation *output for a unit u and the* weight *between two units u and v. Finally, σ will denote a generic activation function.*

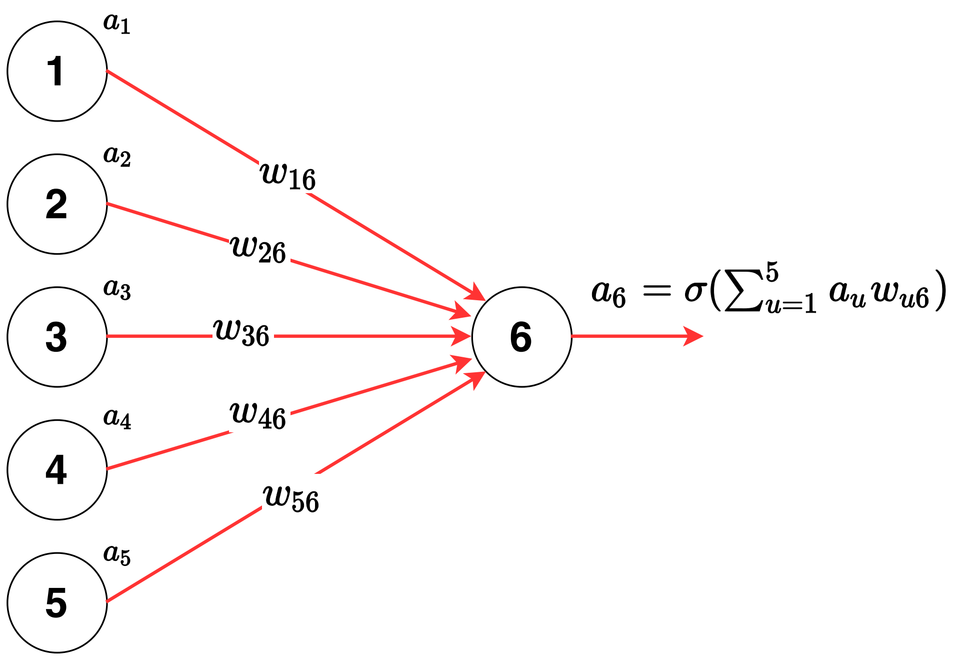


Figure 11: Notation introduced for units in a DNN.

Figures 10 and 11 sketch how the notation introduced by definition 7 works for layers and units in a DNN respectively.
With the introduced notions, we are now ready to explore and discuss the various explanation methods. For each one of them, the main mathematical idea is discussed together with advantages and disadvantages. To pursue a more consistent notation, we follow Ancona, Ceolini, Öztireli, and Gross (2017), Molnar (2019) and Poerner et al. (2018) in addition to the original works of each explanation framework.

3.2 Direct Interpretation

As discussed in 2, some task models can be considered to be transparent. If f is transparent and no other explanation technique is necessary, then the classifier can be directly interpreted. This is usually done by extracting specific information from the model, e.g. a set of weights or attention scores. Given the space of task methods F, interpretable models can then be expressed as subset $G \subset F$. If $f \in G$, then we will denote it with g instead. The following example shows how direct interpretation can be applied and how it relates to the concept of word relevance $\phi(t, c, x)$.

Example 3. *Let $\{w_0, w_1, \ldots, w_{|x|}\}$ be a set of weights and let g be a transparent linear classifier*

$$g(x)_c = w_0 + \sum_{t=1}^{|x|} w_t x_t \tag{1}$$

that computes the score for a class c. Suppose now that we are interested in explaining the output $g(x)_c$. It is intuitive to take the coefficients/weights as indicators of the relevance of their respective feature. This procedure, translated into our notation, is equivalent to assigning the weights as relevance, i.e. $\phi(t, c, x) = w_t$.

Although this example appears to be trivial, a clarification about x's possible representations is necessary and is presented in the following.

Remark 2. *Suppose that x is encoded in a BoW notation, i.e. $x \in \{0, 1\}^{|V|}$. Then, a linear model g can be expressed in the form*

$$g(x)_c = w_0 + \sum_{v \in V} w_v x_v.$$

Our definition of relevance scores $\phi(t, c, x)$ requires both the model and the representation for x to preserve the word order. For this reason, we modify both formulations before attributing relevance scores. x becomes a vector of 1s of the size of the original sentence. g is transformed and uses a different set of weights $w_t \in \mathbb{R}$ for $t = 1, \dots, |x|$. Namely, each w_t is extracted from the BoW model's weight w_v corresponding to the vocabulary entry $v \in V$ for the word x_t. Thus, g can now be expressed, w.r.t. the specific sentence x, in the same form as (1).
Thanks to this reformulation process, also shown for clarity in figure 12 as an example, word order is preserved. Thus, we can proceed with the attribution of the scores $\phi(t, c, x)$. Note that the formulation of g depends on the specific instance x. The definition is not ill-posed for our analysis since we focus on local explanations.

This representation obtained for x and g will be quite relevant in upcoming sections and useful to recall.

Definition 8. *We call x and g* interpretable local representation *and* localised linear model *respectively with the representations built in remark 2.*

Remark 3. *If x is instead encoded with a word embedding, i.e. $x \in \mathbb{R}^{|x| \times m}$, then the word order is already preserved. But, in this case each word x_t is an m-dimensional vector and, as a consequence, also each weight w_t has the same form. When extracting the relevance as described in example 3, we need to reduce the vectors to a score. This can be done simply by taking the L_2 norm. Hence, $\phi(t, c, x) = \|w_t\|_2$.*

Although some more complex clarifications were necessary, direct interpretation is probably the most straightforward and intuitive way of explaining a classifier. It leverages the fact that the classifier itself already contains, in some form, interpretable information that can be recovered. Unfortunately, limiting ourselves to use transparent task models would be quite restrictive as we would not leverage the strength of more complex ML algorithms.

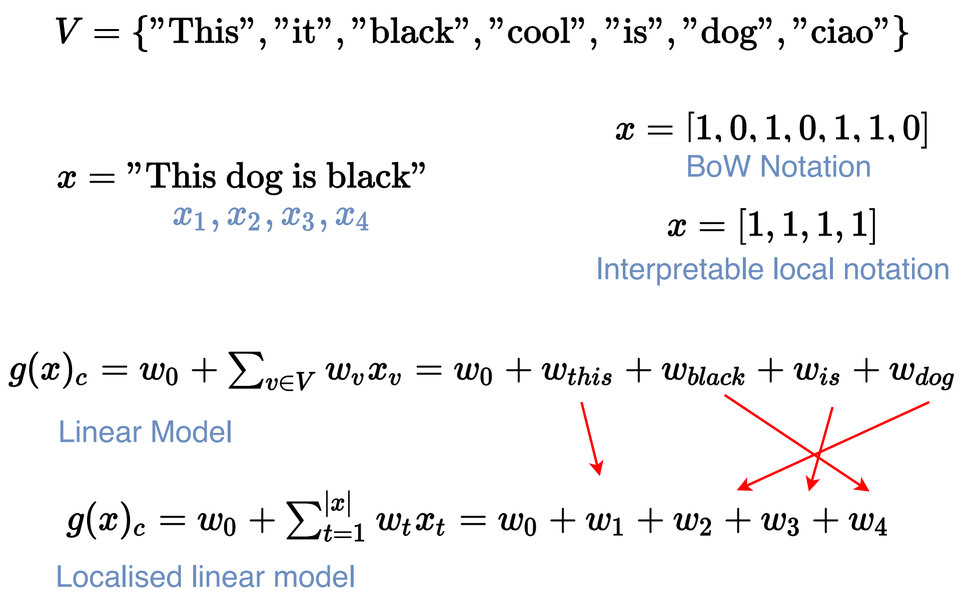

$V = \{"This", "it", "black", "cool", "is", "dog", "ciao"\}$
$x = [1, 0, 1, 0, 1, 1, 0]$
BoW Notation
$x =$ "This dog is black"
x_1, x_2, x_3, x_4
$x = [1, 1, 1, 1]$
Interpretable local notation
$g(x)_c = w_0 + \sum_{v \in V} w_v x_v = w_0 + w_{this} + w_{black} + w_{is} + w_{dog}$
Linear Model
$g(x)_c = w_0 + \sum_{t=1}^{|x|} w_t x_t = w_0 + w_1 + w_2 + w_3 + w_4$
Localised linear model

Figure 12: Example of transformation from BoW model to localised linear model. For instance, since $x_2 =$ "*dog*", then we have the assignment $w_2 = w_{dog}$

3.3 Input Perturbation Methods

Input perturbation methods were among the first approaches to be proposed for visualising and exploring the local behaviour of black-box task models. Most of the methods belonging to this class entered the NLP field by adapting analogous existing approaches in computer vision (Zeiler & Fergus, 2014). They are based on the assumption that removing important parts of the input x will significantly change the output of the classifier f. Following this intuition, these approaches systematically alter a part of the input, run a forward pass with the modified new input, and then compare the new prediction scores to the ones obtained from the original input. While in computer vision this translates to perturbing a block of adjacent pixels in the input image (Zeiler & Fergus, 2014), in our case, it means to perturb one or more words $x_t \in x$ at a time. This is done, for example, by either removing the words (Kádár et al., 2017) or by substituting them with a baseline (Li et al., 2016).
Following the two described ways of perturbing x, we can then define two explanation models, $\phi_{occlude}$ and $\phi_{substitute}$. The relevance scores $\phi(t, c, x)$ assigned by both methods are higher for words x_t that caused more significant changes in the predictions of f. The next example wraps up the procedure in the hate speech detection context.

Example 4. *Let us take a sentence x that is classified as $c =$ "hate speech" with probability 0.9. After altering the word x_3, the predicted probability drops to 0.8 while it drops to 0.5 if x_8 is altered instead. This tells us that x_8 has a much bigger impact in f's decision. We could then assign the two relevance scores, $\phi(3, c, x) = 0.1$ and $\phi(8, c, x) = 0.4$, that are equal to the drop in the predicted probability. This is done for all $t = 1, \dots, |x|$.*

On the one hand, input perturbation methods have the significant advantage of a straightforward implementation and a simple intuition. They are also completely model-agnostic as they only need to run several predictions for the task model f without really taking the architecture of f into account. On the other hand, the scores $\phi(t, c, x)$ have to be computed one by one for every $t = 1, \ldots, |x|$. This makes the computation very costly, especially for longer input texts. In the field of hate speech detection, this problem is mitigated because often input sentences are rather short, e.g. tweets.

3.4 Backpropagation Methods

Unlike input perturbation methods, backpropagation-based methods are able to compute the relevance for all input features in one single forward and backward pass (Bach et al., 2015; Shrikumar, Greenside, Shcherbina, & Kundaje, 2016; Simonyan et al., 2014). To be more precise, sometimes multiple passes are necessary, but still a constant number of them, i.e. not depending on the input size.
This category of methods contains more sophisticated approaches that use substantially different ideas. Some of them, called gradient methods, utilise gradients w.r.t. the input to backpropagate the relevance. Others, instead, are based on recursive formulas derived from intuitive principles and do not involve the computation of any derivative. As we will see, in many cases, new methods are developed to solve pitfalls encountered by previous ones.

3.4.1 Gradient methods

Simonyan et al. (2014) proposes a first approach, called *saliency map*, that uses a gradient to attribute a relevance score to a feature input x_t starting from an output score $f(x)_c$. Simply put, the computed relevance is the partial derivative of the first w.r.t. to the second, i.e.

$$grad(t, c, x) = \frac{\partial f(x)_c}{\partial x_t}.$$

Complex derivatives can be computed by using the chain rule. Figure 13 shows how such information, in DNNs, can be computed exactly like backpropagation.
The gradient $grad(t, c, x)$ has the same shape as x_t, e.g. a vector with the dimensionality of the word embedding space. However, in our case we want to assign a numerical score to the position t. Hence, one possible step to reduce the gradient to a single numerical score is to take the L_2 norm (Bansal, Belanger, & McCallum, 2016). Combining these two steps gives us the explanation method:

$$\phi_{grad^{L2}}(t, c, x) = \|grad(t, c, x)\|_2 = \left\| \frac{\partial f(x)_c}{\partial x_t} \right\|_2 . \tag{2}$$

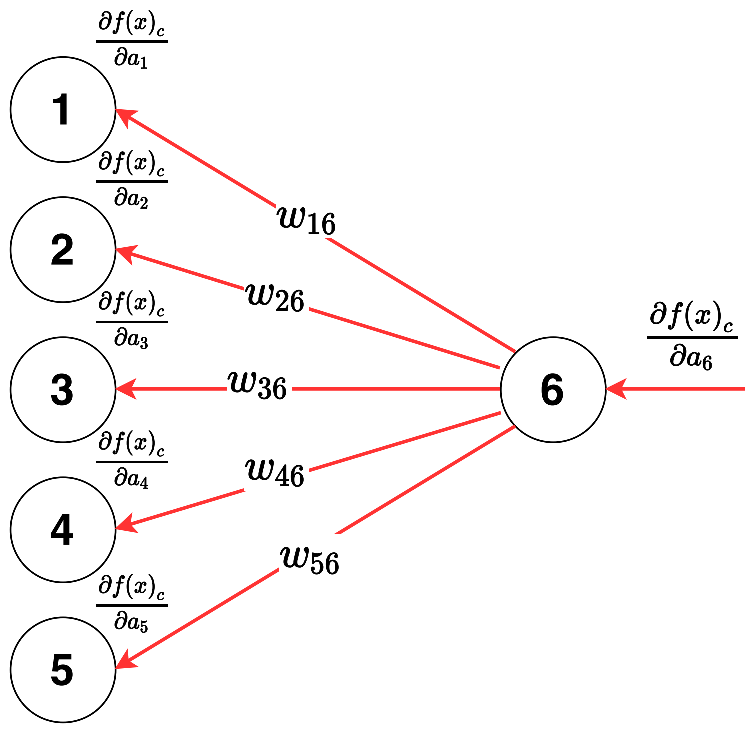

$\frac{\partial f(x)_c}{\partial a_1}$
$\frac{\partial f(x)_c}{\partial a_2}$
$\frac{\partial f(x)_c}{\partial a_3}$
$\frac{\partial f(x)_c}{\partial a_4}$
$\frac{\partial f(x)_c}{\partial a_5}$
$\frac{\partial f(x)_c}{\partial a_6}$
1
2
3
4
5
6
w_{16}
w_{26}
w_{36}
w_{46}
w_{56}

Figure 13: Chain rule applied to the case of figure 11. In this example, for $u = 1, 2, 3, 4, 5$, we have

$$\frac{\partial f(x)_c}{\partial a_u} = \frac{\partial f(x)_c}{\partial a_6}\frac{\partial a_6}{\partial a_u} = \frac{\partial f(x)_c}{\partial a_6}\frac{\partial \sigma(x)}{\partial x}w_{u6}$$

Intuitively, the L_2 norm of the gradient w.r.t. x_t represents how much a perturbation applied to the input feature x_t affects $f(x)_c$. Therefore, the biggest relevance score $\phi_{grad^{L2}}(t, c, x)$ for some t indicates the strongest sensitivity of the output w.r.t changes in x_t, with no regards for the direction of this change. This is coherent with the notion of x_t being relevant for the final prediction score. Nevertheless, applying the L_2 norm prevents the detection of positive and negative relevance since we ignore everything about the gradient's direction. Moreover, in practice, saliency maps appear to be generally noisy (Samek, Binder, Montavon, Lapuschkin, & Müller, 2016).
Denil et al. (2014) uses a different approach to reduce the gradient to a score. Instead of taking the L_2 norm, they compute the dot product between the signed gradient $grad(t, c, x)$ and the embedded input feature x_t itself. Evidently, the modification addresses the obstacles encountered by saliency maps. The change in the reduction step "sharpens" the previous explanation method to

$$\phi_{grad^{dot}}(t, c, x) = x_t \cdot grad(t, c, x) = x_t \cdot \frac{\partial f(x)_c}{\partial x_t}. \tag{3}$$

The approach does not only take into account the magnitude of the gradient but also its

direction. Hence, it becomes possible to distinguish positive and negative contribution from a feature x_t.
As pointed out by Shrikumar et al. (2017), both (2) and (3) underestimate the relevance of input features that saturates an activation function and can introduce discontinuities in the relevance attribution. The first problem occurs when f uses, for instance, tanh or sigmoids as nonlinearities, while the second happens in the presence of ReLU activations. We will discuss these two pitfalls more thoroughly in 3.4.3.
Sundararajan et al. (2017) addresses both fallacies by using a baseline input $\tilde{x}$. Instead of computing only a single partial derivative w.r.t. x_t, it averages its value along a linear interpolation between x and $\tilde{x}$. Thus, given such interpolation as $\tilde{x} + \alpha x$ with $\alpha \in [0, 1]$, we can build the integrated path of gradients:

$$grad_{\int}(t, c, x) = \int_0^1 \frac{\partial f(\tilde{x} + \alpha x)_c}{\partial x_t} \partial \alpha.$$

The choice for the baseline is arbitrary. For instance, we could choose $\tilde{x}_t$ as the embedded vector with all zero entries. As for the computation of the integral, we have to rely on approximations such as:

$$grad_{\int}(t, c, x) \approx \frac{1}{M} \sum_{m=0}^{M} \frac{\partial f(\tilde{x} + \frac{m}{M} x)_c}{\partial x_t}.$$

By using the approximation of this integrated path, we can craft two new methods. One that uses the L_2 norm reduction as (2) and one that instead uses the dot product with x_t as (3). Formally, we add

$$\phi_{grad_{\int}^{L2}}(t, c, x) = \left\| grad_{\int}(t, c, x) \right\|_2 = \left\| \int \frac{\partial f(\tilde{x} + \alpha x)_c}{\partial x_t} \partial \alpha \right\|_2 \tag{4}$$

and

$$\phi_{grad_{\int}^{dot}}(t, c, x) = x_t \cdot grad_{\int}(t, c, x) = x_t \cdot \int \frac{\partial f(\tilde{x} + \alpha x)_c}{\partial x_t} \partial \alpha \tag{5}$$

to our collection of explainability approaches.
So far, we obtained a succession of methods that arise from solving the problem of the original saliency map approach (Simonyan et al., 2014). To be more precise, we obtained four from two possible choices of reduction method and two ways to compute the gradient: in a single point or interpolating with a baseline. Yet, although the integrated gradient method $\phi_{grad_{\int}^{dot}}$ tackles all the pitfalls exposed in this section, approximating gradients well enough is computationally costly (Shrikumar et al., 2017).

3.4.2 Layer-wise Relevance Propagation (LRP)

If we assume a priori that the task model f is a DNN, then different explanation methods that can take advantage of its layer-structure are applicable. One of them, called

Layer-wise Relevance Propagation (LRP), was initially proposed in Bach et al. (2015) for *Fully Connected Networks* (FCNs) and *Convolutional Neural Networks* (CNNs) and then extended in Arras, Montavon, Müller, and Samek (2017) also for *Recurrent Neural Networks* (RNNs). LRP became quite popular as an alternative to gradient methods. In fact, it uses a rule-based backward pass on the task model instead of computing derivatives.
Before describing the method, we first need to understand the concept of relevance and what principles should apply for its computation in a DNN's architecture. These principles are expressed in the form of mathematical constraints and are used to build the LRP rules (Bach et al., 2015).
We can imagine relevance as a quantity that flows from the output to the input. As we will see later, extending the concept of relevance score for an input feature x_t to a general unit u in f allows us to take advantage of the layer-structure of f. Hence, we let $R_u^{(l)}$ denote the relevance score of a unit u in the layer l. By using these quantities, we can decompose a target class output $f(x)_c$ into relevance per input feature x_t, or equivalently, per unit u belonging to the first layer 1. That is

$$f(x)_c \approx \sum_{t=1}^{|x|} \phi(t, c, x) = \sum_{u \in 1} R_u^{(1)}.$$

Simply put, the constraints is equivalent to requiring that the whole output score can be broken down in input features contributions.
Although every entry of $f(x)$ is in $[0, 1]$, by subtracting 0.5 we can translate it to the range $[-0.5, 0.5]$. This change does not affect anything mathematically. Yet, it is quite a useful trick that allows us to distinguish features with positive and negative contribution more clearly. Namely, we say that u contributed to the prediction $f(x)_c$ if its relevance $R_u^{(l)}$ is positive. Otherwise, if $R_u^{(l)}$ is negative, then u contributed against the prediction (and perhaps in favour of another class c').
With the extended concept of relevance, also the notion of decomposing the output score into input relevance can be extended to a more general concept of relevance preservation that applies to every layer. This can be expressed as:

$$f(x) \approx \sum_{u \in L} R_u^{(L)} = \cdots = \sum_{u \in (l+1)} R_u^{(l+1)} = \sum_{u \in l} R_u^{(l)} = \cdots = \sum_{u \in 1} R_u^{(1)} \tag{6}$$

where L indicates the DNN's last layer before the output. Thus, (6) enforces that the output $f(x)_c$ can also be broken down into relevance per feature at any given layer l and not only into relevance per input feature x_t. In other words, the features in any layer should be able to "explain" the final outcome. Note that this principle is not merely artificially crafted. On the contrary, it is consistent with the idea that the information contained in any layer is computed from the input layer and is sufficient to compute the final output.

It is quite clear that only respecting (6) is not enough. The equation has multiple solutions, some of which are not even meaningful explanations. For instance, Bach et al. (2015) builds a simple and intuitive example method that satisfies (6) and that always backpropagates a positive amount of relevance to every feature independently of the input. This scenario is unrealistic in a classification setting. We would expect some input features having a negative contribution in most cases.

As an additional constraint, we restrict the backpropagation to be between connected units. That is, when backpropagating the relevance from an unit u in layer l, we would expect only to backpropagate to the features in layer $l-1$ that contributed to u. Moreover, the amount of relevance that backpropagates should be proportional to the contribution, i.e. relate to the weight w_{vu} for every $v \in (l-1)$.

By putting these constraints together, Bach et al. (2015) builds the LRP recursive algorithm. The method starts from the last layer and uses the target's neuron output itself as relevance while setting the relevance of all other neurons to zero. This can be expressed with

$$R_u^{(L)} = \begin{cases} f(x)_c, & \text{if } u \text{ is the output unit for the target class } c \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Then, LRP proceeds backwards one layer at a time and assigns the relevance scores until the input layer is reached. As redistribution rule from one layer to the preceding one, the following recursive formula is used:

$$R_u^{(l)} = \sum_{v \in (l+1)} R_v^{(l+1)} \frac{a_u w_{uv}}{a_v + \varepsilon \cdot sign(a_v).} \tag{8}$$

Here, as defined already in 3.1, a_u and w_{uv} are used respectively to indicate u's activation and the weight between u and v. The following example shows the applications of LRP on a very small and simple DNN.

Example 5. *Let us consider the DNN in figure 14. If for instance we want to compute the contribution of unit* 3 *for the prediction* a_7*, we have*

$$R_3^{(1)} = \sum_{v=4,5,6} R_v^{(2)} \frac{a_3 w_{3v}}{a_v + \varepsilon \cdot sign(a_v)} =$$

$$= R_5^{(2)} \frac{a_3 w_{35}}{a_5 + \varepsilon \cdot sign(a_5)} =$$

$$= \frac{a_5 w_{57}}{a_7 + \varepsilon \cdot sign(a_7)} \cdot \frac{a_3 w_{35}}{a_5 + \varepsilon \cdot sign(a_5)}.$$

Here, we used that 3 does not contribute to 4 and that 6 has zero relevance because it does not contribute to the target 7.

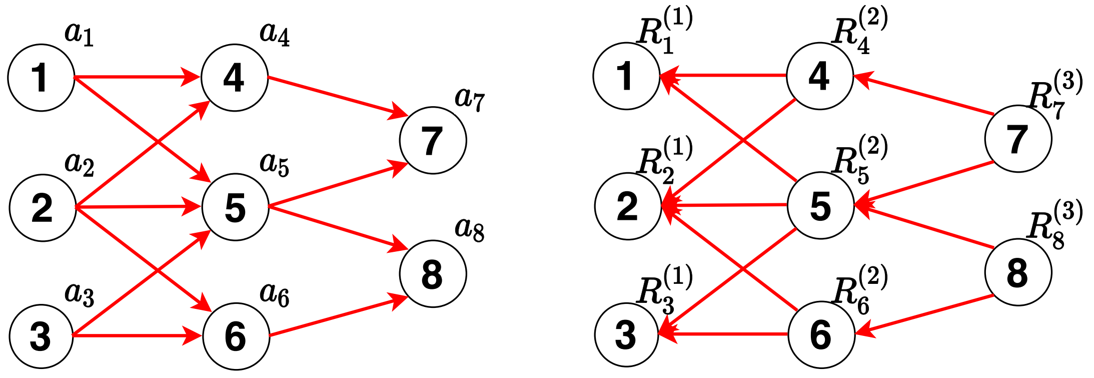


Figure 14: LRP applied to a simple network with 3 layers and 8 units. Forward pass prediction (left) and computation of relevance scores (right) in a layer-wise backpropagation fashion.

The reason for adding the small quantity $\varepsilon \cdot sign(a_v)$ is to avoid a null denominator. In fact, if a_v is positive (respectively negative) and very close to 0, this could create numerical instabilities.
Once the input layer 1 is reached, we consider the relevance of all the units relative to an input feature x_t, i.e.

$$\{R_u^{(1)}\}_{u \in x_t}.$$

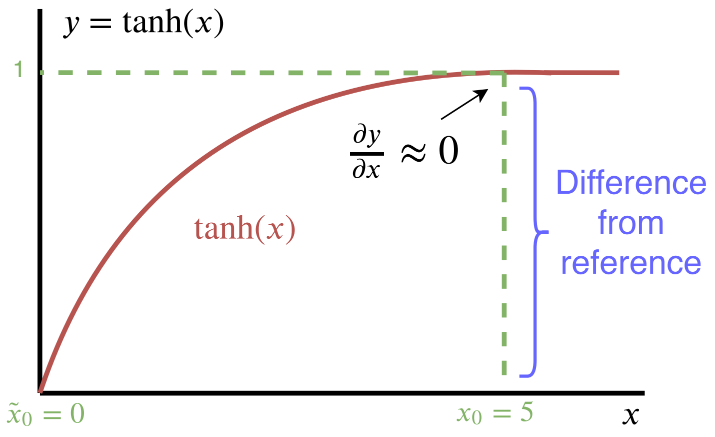
For instance, if x_t is a dense vector resulting from a word embedding, then also its corresponding relevance forms a vector of the same dimension. As already done for gradient methods, we need to reduce the relevance vector to a single score. This can be done by taking the sum over the entries and leads directly to the LRP overall formulation:

$$\phi_{LRP}(t, c, x) = \sum_{u \in (x_t \cap 1)} R_u^{(1)}$$

where, for each $u \in 1$, $R_u^{(1)}$ is computed by following (7) and (8).

3.4.3 Deep Learning Important FeaTures (DeepLIFT)

Now we examine another approach that, like LRP, is designed explicitly for DNNs. For this reason, as we did for analysing LRP, we assume that f belongs to the DNN class. The method, called *Deep Learning Important FeaTures* (DeepLIFT), was initially proposed in Shrikumar et al. (2016). However, after exciting results from Lundberg and Lee (2017) followed, an extended version of DeepLIFT was released in a second publication (Shrikumar et al., 2017).
In some cases, previously examined methods such as $\phi_{grad^{L2}}$, $\phi_{grad^{dot}}$, ϕ_{LRP} still produce misleading explanations. This can happen as they are unable to handle saturation of activation functions and discontinuities in the backpropagation gradients. DeepLIFT



$y = \tanh(x)$
1
$\frac{\partial y}{\partial x} \approx 0$
$\tanh(x)$
Difference from reference
$\tilde{x}_0 = 0$
$x_0 = 5$
x

Figure 15: High values of x saturate a tanh activation function. Backpropagation via a gradient method therefore fails. On the contrary, a difference-from-reference $\tanh(x) - \tanh(\tilde{x})$ is not zero also when the gradient is.

combines ideas coming from LRP (Bach et al., 2015) and integrated gradients (Sundararajan et al., 2017) to address the two pitfalls. Before exploring the approach, a couple of examples are presented to show how previous methods fail in handling certain scenarios.

Example 6. *Let u be a unit in our task method f with a* tanh *activation, i.e.*

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

We know that σ saturates since $\lim_{x\to\infty} \sigma(x) = 1$. If for instance $x > 5$, then we already have

$$\sigma(x) \approx 1 \ , \ \frac{\partial \sigma(x)}{\partial x} \approx 0.$$

In this case, as shown in figure 15, methods like $\phi_{grad^{L2}}$ (saliency maps), $\phi_{grad^{dot}}$ (gradient $\times$ input) are unable to backpropagate the relevance as the gradient is 0.

Example 7. *Let u be a neuron unit inside f with a ReLU activation and bias of $-b$, i.e.*

$$\sigma(x) = \max(x - b, 0).$$

Then, as shown in figure 16, $\phi_{grad^{L2}}$ (saliency maps) and $\phi_{grad^{dot}}$ (gradient $\times$ input) present a discontinuity at $x = b$. Thus, even a very small change in x can cause enormous changes in the assigned contribution.

We have already seen some countermeasures against the scenarios described in the two examples. In fact, as already seen in 3.3, integrated gradients methods $\phi_{grad^{L2}_{\int}}$ and $\phi_{grad^{dot}_{\int}}$ address this problems by integrating between the original input x and a reference input

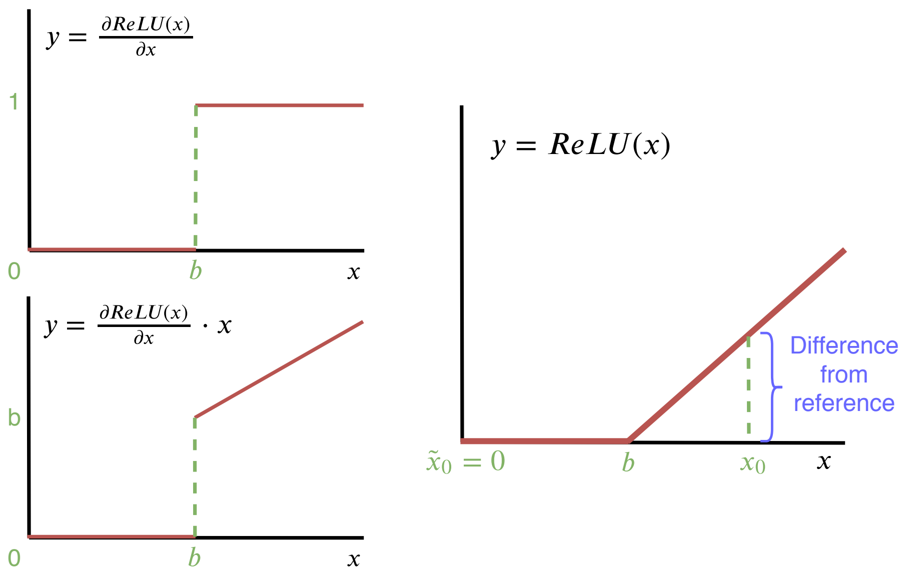


Figure 16: The validity of the relevance scores can be affected by discontinuities in the gradients. Here is the activation of ReLU with a bias of $-b$. Both ϕ^{L2}_{grad} (saliency maps, top left) and ϕ^{dot}_{grad} (gradient $\times$ input, bottom left) present a discontinuity at $x = b$ in the assigned contribution. On the contrary, using a reference (right) $\tilde{x}$ provides a continuous contribution. Figure inspired by Shrikumar, Greenside, and Kundaje (2017, Figure 2).

$\tilde{x}_t$. Nevertheless, a good numerical approximation of integrals requires many sample evaluations and therefore implies a big computational overhead.

DeepLIFT uses a difference-from-reference approach combined with a backpropagation in the LRP fashion to propagate an importance signal. This combination represents the strength of the method since it computes quantities that are non-zero even in situations where the gradient is zero. In this way, as shown in figures 15 and 16, artifacts caused by saturations and discontinuities can be avoided. At the same time, unlike the integrated gradients method, the approach is not computationally costly. With all of this in mind, we now examine the method in more detail.

Let us consider a target class c and its corresponding output neuron $f(x)_c$. By using the baseline input $\tilde{x}$ and its corresponding output $f(\tilde{x})_c$, we define the difference-from-reference input for the feature t as $\Delta x_t = x_t - \tilde{x}_t$ and the relative difference-from-reference output $\Delta f_c = f(x)_c - f(\tilde{x})_c$. The goal of DeepLIFT is to assign relevance scores $R_{\Delta x_t}$ to the difference-from-reference input Δx_t in such a way that the *summation-to-delta* property holds (Shrikumar et al., 2017, Equation 1):

$$\sum_{t=1}^{|x|} R_{\Delta x_t} = \Delta f_c.$$

This property is conceptually very similar to the preservation constraint (6) of LRP.

However, instead of the standard input and output, we use their difference-from-reference versions. Hence, the entire amount of difference in the output can be attributed to the difference in the input. In other words, every change in the output w.r.t. a baseline has to be completely "justified" by a change in the input. By involving a reference value, the relevance $R_{\Delta x_t}$ can be non-zero also when the gradient is zero and levels out discontinuities. In the original work (Shrikumar et al., 2017), the formulation of the method is quite complex. It relies on building rules for every operation inside a DNN. In this thesis, for the sake of a more comfortable and consistent notation, we report a mathematical description that is closer to the one given in Ancona et al. (2017). The authors express DeepLIFT in a recursive formulation that is very similar to the one given for LRP in 3.4.2. Nevertheless, instead of assigning an absolute relevance, DeepLIFT propagates a relative relevance. i.e., proportional to the difference between the activation a_u with the original input and the activation $\tilde{a}_u$ when using a baseline $\tilde{x}$.
More formally , (7) and (8) change respectively to:

$$R_u^{(L)} = \begin{cases} f(x)_c - f(\tilde{x})_c, & \text{if } u \text{ is the output unit for the target class } c \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

and

$$R_u^{(l)} = \sum_v R_v^{(l+1)} \frac{a_u w_{uv} - \tilde{a}_u w_{uv}}{a_v - \tilde{a}_v + \varepsilon \cdot sign(a_v - \tilde{a}_v)} \tag{10}$$

Once the input layer is reached, we can follow the same sub-procedure used for LRP to reduce a vector of scores relative to x_t to a single numerical value. Hence, similarly to ϕ_{LRP}, we have

$$\phi_{DeepLIFT}(t, c, x) = \sum_{u \in (x_t \cap 1)} R_u^{(1)}$$

where, for each $u \in x_t$, $R_u^{(1)}$ is computed by (9) and (10).
With this recursive formulation, on a less strict mathematical level, DeepLIFT can be seen as LRP with a relative input, i.e.

$$\phi_{DeepLIFT}(t, c, x) \approx \phi_{LRP}(t, c, x - \tilde{x}).$$

3.5 Global and Local Surrogates

In ML, some task models are inherently more transparent than others. For instance, as seen in 3.2, a simple linear classifier can be explained by looking at the coefficients of the input features. On the other hand, the interpretation of methods like DNNs becomes unfeasible due to the large number of parameters or, more in general, because they are too complex.

It would be ideal if any model, independently of complexity, could be simplified and made interpretable. Unfortunately, the direct simplification of a model does not seem yet to be a reasonable choice. It is not clear how to simplify a model without intrinsically changing it.

Instead of directly simplifying a task model f, one could try to build a copy $g \in G$ that behaves like f but is easy to interpret. This "simpler copy" can also be referred to as *global surrogate* (Molnar, 2019). Naturally, building an exact copy that is also simpler is not possible. However, an approximation of f could be built by using a simple model g and then training it on the same data as f. To train the approximating surrogate, the predictions of f are used as labels instead of the original ground truth. Once an interpretable surrogate is built, it can be used to explain the output of a singular instance.

The global surrogate approach is very flexible as it does not impose any restriction on g. Therefore, we could pick any interpretable task model to build our approximating surrogate. At the same time, the implementation is quite simple and straightforward. Metrics to measure how well the surrogate approximates the original model exist (Cameron & Windmeijer, 1997; Molnar, 2019). However, it is generally not clear what threshold guarantees that g is close enough to f (Molnar, 2019).

Approximating a complex model globally with something simple seems quite a hard task. Not only do we not know, even in theory, if a simple model can in practice approximate well a behaviour that is orders of magnitude more complex. But also, explaining a single output is much more accessible than describing a task method f as a whole. With this motivation, other approaches, instead of seeking to find a global surrogate for f, try to do it at least locally. These methods build an interpretable model g, called *local surrogate*, that approximates the classifier to be explained only around a specific instance. In the next section, we discuss one popular method that belongs to this category.

3.5.1 Local Interpretable Model-agnostic Explanations (LIME)

One of the most successful explanation methods, called *Local Interpretable Model-agnostic Explanations* (LIME) (Ribeiro et al., 2016), directly implements the idea of a local surrogate. In fact, it defines an interpretable model g that is *locally faithful* (Ribeiro et al., 2016) to the classifier to be explained.

Before we examine the mathematical details of LIME, we shall sketch the steps of the method with no formality. We do this with the support of figure 17.

1. Pick a classifier, an input instance, and the corresponding output that we are interested in explaining.

2. Generate points by perturbing the selected instance and, for each new point, obtain the prediction from the classifier.

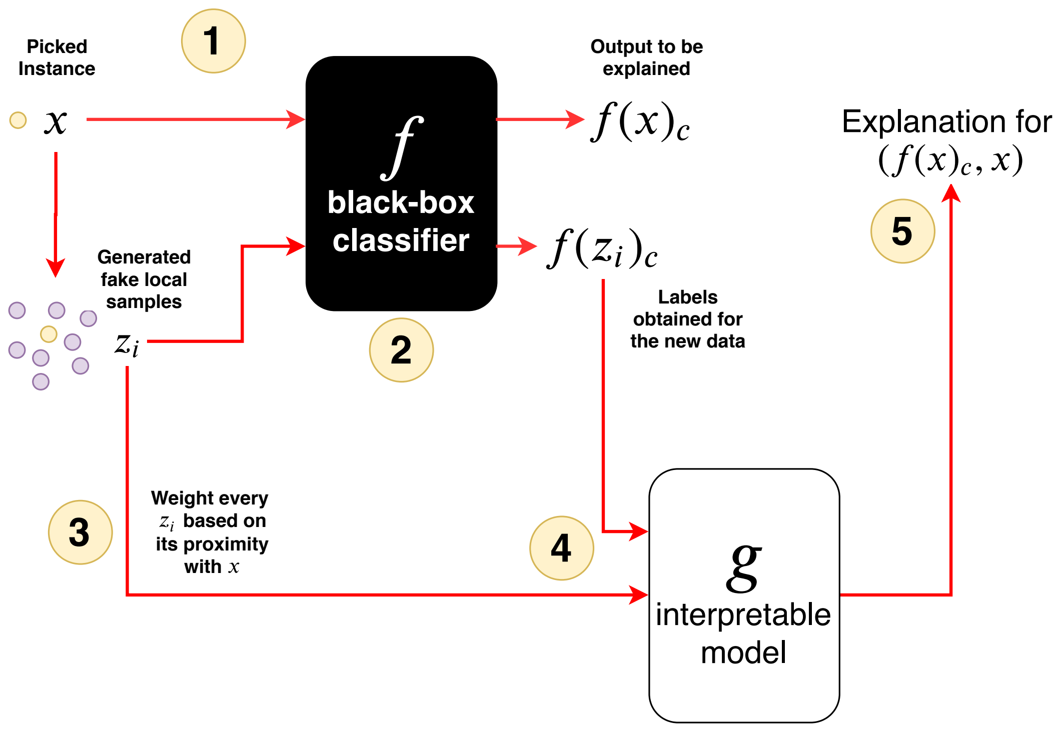

1
Picked Instance
x
f
black-box classifier
2
Output to be explained
$f(x)_c$
Generated fake local samples
z_i
$f(z_i)_c$
Labels obtained for the new data
Explanation for $(f(x)_c, x)$
5
3
Weight every z_i based on its proximity with x
4
g
interpretable model

Figure 17: Flow of the LIME approach to explain the prediction of a classifier f on an instance x.

3. Weight the newly generated samples based on the proximity to the originally selected instance.

4. Pick an interpretable model and train it on the weighted generated dataset of variations and the labels obtained from step 2.

5. Explain the original prediction by interpreting the simple local model.

It is quite easy to see how the locality of the approach is enforced with the creation of a fake local dataset. We now continue to examine the method from a more formal perspective.
LIME, to perform step 2-5, uses an additional *interpretable data representation* $x' \in \{0,1\}^M$ that differs substantially from the original input x used by the classifier f (Ribeiro et al., 2016). This additional representation consists of binary features which have a different meaning depending on the data type, e.g. text or images. The following example shows a choice of interpretable data representation that applies to our hate speech detection use-case.

Example 8. *Let us consider a hate speech detection model f. Its input features for a sentence $x \in X$, most likely word embeddings, are quite complex and hard to interpret. We shall consider now the sentence $x =$* "Chinese people are a big minority in this city"*.*

A word embedding with dimension m will map x into $\mathbb{R}^{|x|\times m}$. In this case, it is a matrix in $\mathbb{R}^{9\times m}$ to be passed to f to classify the sentence.
As a simpler data representation, we can for example represent a sentence by a binary vector x'. To preserve the word order, instead of using a classic BoW as proposed in Ribeiro et al. (2016), we can use the interpretable local representation already defined in 8. This way, we have a binary representation $x' \in \{0,1\}^{|x|}$. Note that this is possible thanks to the fact that the LIME method is formulated around an instance x.

We now consider the already mentioned families, F and G, of classifiers and interpretable classifiers respectively. Let $\Omega : F \to \mathbb{R}$ be a measure of *complexity*, meant in this case as opposite of intepretability, and let π_x be a proximity measure. That is, $\Omega(f)$ expresses how complex the classifier f is and, given another input z, $\pi_x(z)$ measures the proximity of z to x. Finally, given $f \in F$ and $g \in G$, let $\mathcal{L}(f, g, \pi_x)$ indicate the *local fidelity* of g to f, i.e. how good g is as a local approximator.
With the above notation, LIME finds an interpretable model for a sample x as result of the minimisation

$$g^* = \arg\min_{g\in G} \mathcal{L}(f, g, \pi_x) + \Omega(g).$$

To build LIME to be model-agnostic, we want to minimise $\mathcal{L}(f, g, \pi_x)$ regardless of f. i.e. we want to learn the local behaviour of f around a specific instance x with no assumption on the classifier itself. To do so, we draw instances z' around x' by choosing non-zero entries of x' uniformly at random. Then we recover their original representation z and weight them by π_x. Finally, we retrieve the predictions $f(z)_c$. These are then used as labels to train g on the perturbed samples z' and therefore minimise the local fidelity $\mathcal{L}$. Once the simple local model has been trained, we can explain $f(x)_c$ by interpreting $g(x')_c$ on the instance x that we initially picked.
The abstract notation maintains the description quite general. Thanks to this, the formulation holds for different families G, fidelity function $\mathcal{L}$, complexity measures Ω and proximity kernels π_x. At this point, it remains to put the abstract notions into our hate speech detection context. This includes taking choices w.r.t. the different measures to use within the method. To this end, we construct the following example.

Example 9. *For an input sentence x, we use the intepretable local representation x' from definition 8. In the same spirit as example 8, as family of interpretable models G we pick linear models localised on x, also defined in definition 8.*
Differently from what is proposed in Ribeiro et al. (2016), we do not need an explicit complexity constraint $\Omega(g)$ to make sure that g is interpretable enough. In Ribeiro et al. (2016) a limit K is set on the number of words considered in a BoW model, i.e.

$$\Omega(g) = \begin{cases} +\infty, & \text{if } \|w_g\|_0 > K,\ w_g \text{ weights of } g \\ 0, & \text{otherwise} \end{cases}.$$

In our case instead, a linear interpretable model g is kept from being complex thanks to the localisation process described in 3. Therefore, we can simply set $\Omega(g) = 0$.
As local fidelity, we choose a square loss weighted based on a proximity kernel π_x. That is

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z'\,sampled} \pi_x(z)(f(z) - g(z'))^2.$$

For a hate speech detection model, we need to pick a proximity kernel suitable for text input data. This can be done, following the same logic as similarity in word embeddings, by looking at the "direction" of two different sentences in the original representation space $\mathbb{R}^{|x| \times m}$ (Mikolov et al., 2013). The cosine distance computes the distance direction-wise and can be wrapped in an exponential kernel

$$\pi_x(z) = exp\left(-\frac{\cos(x,z)^2}{\sigma^2}\right) \quad where \quad \cos(x,z) = 1 - \frac{x \cdot z}{\|x\|_2\|z\|_2}$$

Note that the choices taken could vary depending on what the interpretability needs are. For instance, π_x is normally chosen based on the input type and the transparent model family g is at the user's discretion. We can define the LIME explanation method ϕ_{LIME} as coming from the direct interpretation of g^*.
In Poerner et al. (2018), based on our same needs, another variation of LIME is built for task methods that respect word order. This approach, called *Local Interpretable Model-agnostic Substring-based Explanations* (LIMSSE), uses a randomly extracted substring instead of a BoW (or our localised version) to build the interpretable representation.

3.6 SHapley Additive exPlanations (SHAP)

Despite the fast growth of the space of existing methods, many explanation approaches seem to be closely related (Lundberg & Lee, 2017). One might wonder if there is a more general theory that could describe explanations from a more abstract perspective. Such a theory, perhaps missing due to the novelty of the explainability field, could find a more formal link between existing methods and provide the tools to build new ones.
Lundberg and Lee (2017) presents a unified approach that connects some of the most popular methods. This novel perspective, besides simplifying the set of existing approaches, strongly indicates that a unifying theory might exist.
Before we describe formally the new theoretical results introduced in Lundberg and Lee (2017) and their implications, let us list them in a more informal way.

- A new class of explainability methods is defined, called *additive feature attribution methods*. The elements of this class provide explanations by building a simpler transparent model that locally approximates the original model to be explained.
- Six existing methods are proven to belong to this class. This includes the already studied LRP, DeepLIFT and LIME together with three older methods from the field

of game theory (Datta, Sen, & Zick, 2016; Lipovetsky & Conklin, 2001; Štrumbelj & Kononenko, 2014).

- For additive feature attribution methods, three desirable properties for good explanations are formally introduced.
- It is proven that, within the additive feature attribution methods class, there is a unique local model that has to be built to fulfil all desirable properties.
- Existing methods in this class need to be adapted to build this optimal local model. The improved versions of these methods are grouped together under the name of *SHapley Additive exPlanations* (SHAP).

For a simple and interpretable model, the best explanation is the model itself. For more complex models, this does not hold. As we have already discussed in 3.5, trying to simplify or to globally approximate it with a simple one has considerable drawbacks. Local methods like LIME instead approximate f only around a specific instance x. This follows the assumption that even complex models are locally simple. Thus, considering models only locally reduces the gap between a complex model f and a simple model g. As also mentioned in 3.5.1, the interpretable model g uses a a simplified representation x' which can map to the original input x through the mapping $x = h_x(x')$. Once more, local approaches have to ensure local fidelity, i.e. $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$.

Definition 9. *(Lundberg & Lee, 2017, Definition 1)* Additive feature attribution *methods have an* explanation model $g \in G$ *that is a linear function of binary variables:*

$$g(x') = \phi_0 + \sum_{t=1}^{M} \phi_t x'_t$$

where $x' \in \{0,1\}^M$, M *is the number of simplified features and* $\phi_t \in \mathbb{R}\ \forall t$.

In other words, methods using an interpretable model g that matches definition 9 attribute an effect ϕ_t to each feature plus a default effect ϕ_0. The latter indicates the bias of g, i.e. the expected average output, also called *base value* or *baseline*.
Already in older literature, in particular in the field of cooperative game theory, one can find explanation methods (Datta et al., 2016; Lipovetsky & Conklin, 2001; Štrumbelj & Kononenko, 2014) that have been proven to belong to the class of definition 9 (Lundberg & Lee, 2017). At their foundation, there is a concept called *Shapley values*. This was introduced by Shapley (1953) more than 60 years ago for cooperative games. The original purpose of Shapley values is to explain how to fairly distribute a payout among the players. In our domain, each feature can be thought of as a player and the payout as the feature's relevance/effect. In the following, we explore the meaning of this quantity more in detail. Following the game theory definition, the Shapley value ϕ_t of a feature x_t is the *average marginal contribution of its value across all possible coalitions* (Molnar, 2019; Shapley,

1953). This notion may appear quite confusing in our context. Hence, as done in Molnar (2019), we reformulate it in a more formal and perhaps clearer way.

Definition 10. *Let us consider the output $f(x)_c$ to be explained and let S be a subset of the features of x, i.e. $S \subset \{x_t\}_{t=1}^M$. Let x_S be the input x where the features not in S are set to 0. Then, for $t = 1, \dots, M$, the* Shapley value *ϕ_t of x_t is*

$$\phi_t = \sum_{S \subset \{x_i\}_{i=1}^M \setminus \{x_t\}} \frac{|S|!(M - |S| - 1)!}{M!} (f(x_{S \cup \{x_t\}}) - f(x_S))$$

and, for $t = 0$, we have the default value $\phi_0 = f(x_\emptyset)$.

Note that $\frac{|S|!(M-|S|-1)!}{M!}$ is simply the number of possible choices for the subset S and works as a normalisation term.
Interestingly, Shapley values ϕ_t and our concept of feature relevance $\phi(t, c, x)$ are equivalent when we perform direct interpretation on a localised linear model as $g \in G$. We prove this in the following theorem. In other words, when we extract some weights from a localised linear model g to explain the output, we are actually looking directly at the Shapley values.

Theorem 1. *Let g be a localised linear model on an instance x, then the following claims about g hold:*

1. *Direct interpretation of g is an additive feature attribution method.*

2. *Given a class c, the Shapley values ϕ_t and the features' relevance $\phi(t, c, x)$ match for every feature x_t. Therefore, using the same notation $\phi(t, c, x)$ for both is indeed consistent.*

Proof. From definition 8, let us consider $g \in G$ to be a localised linear model on an instance x. Ergo, g is an interpretable model that uses the input in its interpretable local representation $x' \in \{0, 1\}^{|x|}$. We denote g by using the set of weights $\{w_0, w_1, \dots, w_{|x|}\}$, i.e.

$$g(x') = w_0 + \sum_{t=1}^{|x|} w_t x'_t.$$

Regarding the first claim, it is immediate to see that the direct interpretation of g respects definition 9 by using itself as explanation model with $M = |x|$. Now it is left to prove that $\forall t = 1, \dots, |x|$ we have the equivalence between the Shapley value ϕ_t and the features' relevance $\phi(t, c, x)$. The latter quantity, since g is a simple linear model and can be interpreted by looking at the weights, is equivalent to the weight w_t.
If $t = 0$, then by definition $\phi_t = g(x'_\emptyset) = w_0$ where the last equivalence holds because $S = \emptyset \implies x'_t = 0 \ \forall t > 0$.

If $t > 0$, then by definition

$$\phi_t = \sum_{S \subset \{x'_i\}_{i=1}^{|x|} \setminus \{x'_t\}} \frac{|S|!(|x| - |S| - 1)!}{|x|!} (g(x'_{S \cup \{x'_t\}}) - g(x'_S))$$

Due to the simplicity of g, in this case it is possible to explicitly compute its output for a feature subset S. In fact, we have

$$g(x'_S) = w_0 + \sum_{t \in S} w_t x'_t$$

$$g(x'_{S \cup \{x'_t\}}) = w_0 + w_t x'_t + \sum_{t \in S} w_t x'_t$$

and therefore

$$g(x'_{S \cup \{x'_t\}}) - g(x'_S) = w_t x'_t = w_t.$$

The last equality holds as $x'_t = 1$ when considering $S \cup \{x'_t\}$.
By plugging the result, the following holds:

$$\begin{aligned} \phi_t &= \sum_{S \subset \{x'_i\}_{i=1}^{|x|} \setminus \{x'_t\}} \frac{|S|!(|x| - |S| - 1)!}{|x|!} w_t = \\ &= w_t \sum_{S \subset \{x'_i\}_{i=1}^{|x|} \setminus \{x'_t\}} \frac{|S|!(|x| - |S| - 1)!}{|x|!} = \\ &= w_t = \phi(t, c, x). \end{aligned}$$

□

Remark 4. *Thanks to this proof, by setting $\phi(0, c, x) = \phi_0$, the definition of localised linear model can be rewritten using directly the Shapley values:*

$$g(x)_c = \phi(0, c, x) + \sum_{t=1}^{|x|} \phi(t, c, x) x_t = \phi_0 + \sum_{t=1}^{|x|} \phi_t x_t.$$

So far, we know that Shapley values are a very useful tool for some explanation algorithms inspired by game theory that fall in the class of additive feature attribution models. At the same time, we have proven how they match precisely the features' relevance in case we use a localised linear model as an explanation model g. But how do they relate to the popular methods we examined in the previous sections? To answer this, Lundberg and Lee (2017) connects several existing approaches to the class that we introduced. For better comprehension and consistency, the following theorem groups these results from Lundberg and Lee (2017, Sections 2.1, 2.2, 2.3) all together.

Theorem 2. *LIME (Ribeiro et al., 2016), DeepLIFT (Shrikumar et al., 2017) and LRP (Bach et al., 2015) belong to the class of additive feature attribution methods.*

Proof. The definition of LIME naturally adheres to our class of methods. Indeed, LIME builds an interpretable model g that uses a simplified input $x' \in \{0,1\}^M$ and locally approximates f. How x' is built and how it can be mapped back to the original input $x = h_x(x')$ depends on the task. Nevertheless, it is always consistent with definition 9 if we use linear models as family G. All the details on how g is built, and consequently on how the $\phi_{LIME}(t, c, x)$ are computed, have been discussed in 3.5.1. It has also be shown in example 9 that LIME can use a localised linear model as g.
DeepLIFT instead, attributes to each input x_t a value $R_{\Delta x_t}$ that represents the relative effect of x_t when compared to a reference value $\tilde{x}_t$. In a simplified binary representation $x' \in \{0,1\}^M$, we can think of the 1s as representing the usage of the original value and the 0s for the reference value. This implicitly defines an intepretable model g, in which $x = h_x(x')$ converts a binary input to the original input. Therefore, by letting $\phi_0 = g(\tilde{x}_t)$ and $\phi_t = R_{\Delta x_t}$ for $t > 0$, we have that the model g built by $\phi_{DeepLIFT}$ is an additive feature attribution method.
Finally, as seen in 3.4.3, LRP is equivalent to DeepLIFT with reference input $\tilde{x}$ set to 0. Therefore, with the same argumentation as for $\phi_{DeepLIFT}$, we have that also ϕ_{LRP} matches definition 9. □

This result shows us how methods that can differ a lot in appearance are actually strongly linked by the concept of Shapley values. In this case, LIME, DeepLIFT, and LRP appear to be a different version of the same type of method. More specifically, they only differ in the meaning assigned to x' and in the way g is built.
It seems natural now to wonder if other methods belong to this class, perhaps even some among the ones we already explored. One way to answer this could be to manually inspect how other methods implicitly build g and x' and from there derive better ways of doing it. Instead, Lundberg and Lee (2017) takes a more indirect, but more effective strategy, that we present in the following. The authors ask themselves what desirable properties an ideal additive feature attribution method should have and then understand how to build a method that fulfils them.
Hence, as done in Lundberg and Lee (2017), let us now define three properties that are intuitively helpful for obtaining good explanations. The first property, called *local accuracy*, enforces that an excellent local approximation should at least match f on the instance x.

Definition 11. *(Lundberg & Lee, 2017, Property 1) Suppose that g is built to locally approximate f at x and $x = h_x(x')$. We say that we have* local accuracy *if*

$$f(x) = g(x') = \phi_0 + \sum_{t=1}^{M} \phi_t x'_t$$

Another desirable property, named *missingness*, states that missing features should not impact the explanation.

Definition 12. *(Lundberg & Lee, 2017, Property 2) Let $g \in G$ the intepretable model, then we say that we have* missingness *if*

$$x'_t = 0 \implies \phi_t = 0$$

Finally, a slightly more complicated property is introduced: *consistency*. When a feature $x'_t \in \{0, 1\}$ has a more significant impact on a model f' than a model f regardless of other inputs, then that input's attribution ϕ_t should also be more prominent.
In the following, we indicate with $x' \setminus t$ the input x' if we set $x'_t = 0$.

Definition 13. *(Lundberg & Lee, 2017, Property 3) We say that our explanation method has* consistency *if the following holds. For any two models f, f', if*

$$f'(h_x(x')) - f'(h_x(x' \setminus t)) \geq f(h_x(x')) - f(h_x(x' \setminus t)) \quad \forall x' \in \{0, 1\}^M$$

then, also ϕ_t is bigger for f' than for f.

LIME, DeepLIFT and LRP have missingness by default, but do they respect the other two properties? The following result proofs that only methods adopting the Shapley values respect all three properties at the same time. Therefore, if one of these methods operates without them, then it lacks either local accuracy or consistency.

Theorem 3. *(Lundberg & Lee, 2017, Theorem 1) Let f be a task method to be explained. If one uses an additive feature attribution method, only one possible explanation model g satisfies definitions 11, 12 and 13:*

$$\phi_t = \sum_{z' \subset x'} \frac{|z'|!(M - |z'| - 1)!}{M!} (f(h_x(z')) - f(h_x(z' \setminus t)))$$

where $|z'|$ is the number of non-zero entries in z', and $z' \subset x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

Proof. See Lundberg and Lee (2017) and Young (1985). □

This fundamental result shows that the usage of Shapley values in existing explanation models is required to have the desirable properties detailed above. A specific choice can ensure this in some of the method's components. For example, in LIME, a suitable choice for the measure Ω should be made.
These specific choices do not imply any conceptual change in the previously examined methods. Therefore, we do not discuss them. For further details, we refer to Lundberg and Lee (2017, Section 4). The combination of Shapley values and existing methods led to the implementation of two new methods: *KernelSHAP* (LIME + Shapley values) and *DeepSHAP* (DeepLIFT + Shapley values) (Lundberg & Lee, 2017).
It is interesting to notice how, since localised linear methods are intrinsically using Shapley values by theorem 1, they also respect local accuracy, missingness, and consistency.

3.7 Evaluating and Choosing an Explanation Method

Explainability of ML methods is gradually becoming as important as performance. The available frameworks for explaining black-box models are already many and increasing in number (Guidotti et al., 2019). With all these options offering to give good explanations, it becomes hard to understand which one to prefer over the others. At the same time, we have seen how works like Lundberg and Lee (2017) manage to simplify the choice by creating a unified view of different existing methods. Yet, our question on which method to pick is still to be answered.

Unfortunately, as already discussed in 2.1.3, there are no obvious performance metrics for explanations. Hence, there is no simple way of assessing how good an explainability approach is with a numerical score. On the other hand, we know that comparisons can be made via some evaluation techniques: *application-grounded, human-grounded, and functionally-grounded* (Doshi-Velez & Kim, 2017).

For our particular case, many functionally-grounded evaluation studies can be found (Ancona et al., 2017; Arras et al., 2016; Poerner et al., 2018) for NLP classifiers. However, their evaluation results differ substantially from each other. This supports the hypothesis that evaluation results are highly dependent on the chosen evaluation paradigm. For instance, Ancona et al. (2017) shows ϕ_{occ}, $\phi_{grad^{dot}}$, and $\phi_{grad^{dot}_{\int}}$ tied for the best performance while Poerner et al. (2018) reports better results for LRP, DeepLIFT, and LIME.

Nevertheless, Poerner et al. (2018) carries out a comprehensive review and, unlike the other works, does not uses syntactically broken inputs that can introduce artefacts (Sundararajan et al., 2017). Also, their results are more coherent with what we found to be the methods with the most reliable theoretical background. Hence, we pick LRP, DeepLIFT, and LIME as potential candidates of our choice.

Since LRP is a special case of DeepLIFT that theoretically still has the pitfalls discussed in 3.4.3, we exclude it from our list of preferred methods. At the same time, Poerner et al. (2018) does not include yet the results introduced by SHAP (Lundberg & Lee, 2017). The advantages of using Shapley values, as we have seen, are strongly backed up by theoretical results. Moreover, SHAP's explanations have been empirically found to be more coherent with human intuition (Lundberg & Lee, 2017).

From a practical perspective, it is worth mentioning that the implementation available for DeepLIFT [1] only supports (at the moment) a few types of DNN operations. In contrast, SHAP [2] and LIME [3] repositories support a wider variety of task methods.

We take into account the theoretical results, the different results from the most relevant experimental comparisons, and the available implementations. In conclusion, we encourage the usage of LIME and DeepLIFT in their SHAP variants. That is, the method

[1]https://github.com/kundajelab/deeplift

[2]https://github.com/slundberg/shap

[3]https://github.com/marcotcr/lime

KernelSHAP (Lundberg & Lee, 2017) in case one is interested in a model-agnostic explanation method and *DeepSHAP* (Lundberg & Lee, 2017) in case f is a DNN.
Ultimately, in this work, we tried to back up these suggestions using multiple perspectives. Nonetheless, we should not forget that there is no definitive and absolute answer to the question of which explanation method should be used. In general, no approach seems to be intrinsically superior to the others. Thus, as already discussed in 2.1.3, possible choices should still take the use-case and the explainability goals into consideration.

4 Towards Explainable Hate Speech Detection

Automatic hate speech detection is challenging for several reasons such as subtleties in the language and limitations in available data. At the same time, recent methods can be hard to interpret and therefore, as discussed in 2.2.1, not suitable for real-life applications. As already discussed in 2.2.2, interpretability for hate speech detection is only in its early stages and only a few contributions can be found (MacAvaney et al., 2019; Rizoiu et al., 2019; Wang, 2018). This is especially true for detection methods that make use of social features such as user information and relationships. To the best of our knowledge, this is the first work that explains the behaviour of such models.

In this chapter, we explore new possible interpretability techniques for hate speech detection. We also show practical applications of the proposed approaches and how the existing research can benefit from them. As we will see, our explainability techniques facilitate crucial observations about datasets and models. Hence, it gives us more insights and provides us with tools to improve future methods.

To begin with, in 4.1, we closely analyse a hate speech detection benchmark on which current methods achieve poor performance in one of the classes. We review the findings from an earlier analysis and we compare it to ours, conducted with more recent explainability methods. In 4.2, we explain detection models that incorporate context, in the form of social features. Experiments show the benefits of considering social traits in terms of accuracy. Moreover, we are able to clearly illustrate how such features are used by the model to enhance prediction performance. We also discuss how our techniques can be combined with artificially crafted tweets for examining specific scenarios. Finally, we point out the importance of the target audience and observe how our techniques can be adapted for different types of final users.

4.1 Explainability as a Dataset Analysis Tool

In this section, we make use of an explainability approach to understand why models perform poorly on the hate detection benchmark introduced by Davidson et al. (2017). The employment of explainability reveals itself extremely useful as it exposes otherwise hidden patterns in the classifier's behaviour. Our analysis shows that the dataset's construction is a major cause of the models limited detection capabilities. As a result, we recommend future benchmarks to use a different task formulation from the one adopted in Davidson et al. (2017).

To begin with, in 4.1.1, we briefly introduce the benchmark from Davidson et al. (2017) and remark the authors' motivation to construct such a detection task. In 4.1.2, we discuss an analysis previously carried out by Wang (2018) on the benchmark and review their results. Finally, in 4.1.3, we extend the analysis with more recent explainability tools. Concretely, we build and train a detection method on the benchmark. The model is then explained via DeepSHAP (Lundberg & Lee, 2017). The resulting explanations help us draw conclusions about the model and the data.

4.1.1 The Davidson Benchmark for Hateful and Offensive Language

Motivated by the missing distinction between hate speech and offensive language in previous works, Davidson et al. (2017) creates a dataset in which they separate the two in different classes. Throughout, we refer to this dataset as the Davidson Benchmark. Davidson et al. (2017) implicitly defines offensive language as the usage of offensive words and argue that not all instances of offensive language are hate speech. The authors explain how the same word might or might not be hateful, depending on the context. For instance, the word "*nigga*" is also used by black people to address each other in everyday life and thus not only found as a hate term. The authors conclude that future work should also make use of more fine-grained labels and account for context and heterogeneity in hate speech usage.

Davidson et al. (2017) collects, via the Twitter API, over 85 million tweets containing hate speech lexicon from the time-line of 33,558 users. Then, they randomly sample a subset of 25,000 tweets, which are labelled manually via crowdsourcing as one of three classes: *hate, offensive, neither.* Annotators were explicitly asked to think not just about the words contained in a particular tweet but also about the context in which they were used. The resulting dataset is distributed as 5% hate speech, 76% offensive language, and 16.6% neither (Davidson et al., 2017). After discarding tweets with no majority in the annotators' votes, the authors put together a final collection of 24,802 labelled tweets.

Furthermore, Davidson et al. (2017) trains and tests several detection methods on the dataset. Among the models achieving the highest performance (0.90 F1 score), an L2-regularised Logistic Regression is picked as the final framework. The authors report poor performance on the hate class and observe that most of the misclassification occurs between the hate and offensive classes. Concretely, a lot of hateful tweets are wrongly classified as offensive. The authors claim this effect is caused by the presence of offensive terms that can eventually distract the classifier from more hateful slurs.

Other works that benchmarked on the dataset introduced in Davidson et al. (2017) also achieve a meagre performance on the hate class. For example, Rizoiu et al. (2019) builds two models that achieve 46% and 53% accuracy on hate messages respectively, while performing on average well above 80% on other classes. As we will see in the next sections, this big leap in performance among different classes motivated analysis on the benchmark. The big leap in performance among different classes motivates a thorough analysis of the benchmark, which we cover in the upcoming sections.

4.1.2 Previous Analysis of the Davidson Benchmark

Wang (2018) implements a SOTA method for detection based on a *convolutional*-GRU (Zhang, Robinson, & Tepper, 2018) and tests it on the Davidson Benchmark. The model easily surpasses 80% as overall accuracy while reaching only 31% on hateful tweets. To further understand these results, they revisit some simple explainability techniques from computer vision and adapt them to the hate speech recognition use-case. These techniques focus on visualising sensitive regions of the input, i.e. words in a sentence, or identifying the features learnt by individual neurons. Input sensitivity is computed via perturbation of the initial sentence, in the same way as we described in 3.3. Features learnt by a unit in a DNN can be extracted by finding the features that maximise the unit's activation (Wang, 2018).
As a result of their experiments, Wang (2018) identifies several weak points of the target model when trying to recognise hate in tweets:

- *Overlocalisation:* this indicates the hyper-sensitivity of the classifier's decision w.r.t. a particular part of the input (e.g. a word or two) rather than the entire context. This type of error appears in long sequences, which often are misclassified as offensive (Wang, 2018).

- *Lack of sensitivity:* the model does not show sensitivity w.r.t any part of the input. This occurs in long sentences, often part of the hate class, that do not present particularly offensive terms (Wang, 2018).

- *Unintuitive features:* in some cases, the model shows sensitivity w.r.t. input parts that are not intuitively correlated with the output. This behaviour appears in the misclassified samples of each class (Wang, 2018).

- *Classes discretisation:* hateful samples are often misclassified as offensive. This is caused by the discretisation between the hate and offensive classes, which naturally intersect. In other words, there is an overall tendency of the model to classify sentences that contain insults as offensive and miss the underlying presence of hate (Wang, 2018).

Wang (2018) argues that the first three pitfalls can be addressed by changing the model's architecture or, more in general, the detection approach. The last one, instead, is inherited from the data itself. In this case, the authors claim that misclassified samples do not indicate a faulty model, but rather express the challenge resulting from the complex linguistic similarity between hate speech and offensive language. Ergo, the detection performance is inevitably affected by the insufficient class separability.

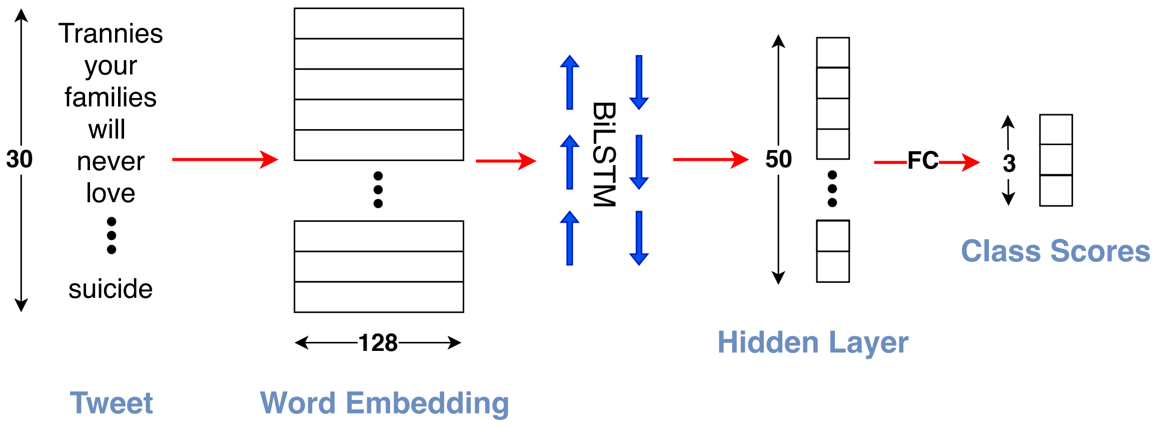


Figure 18: Architecture of the BiLSTM detection network utilised for our analysis.

4.1.3 A SHAP-Driven Benchmark Analysis

We now make use of a different explainability technique based on the results of our mathematical analysis in 3. We build a simple detection model and analyse some of its predictions. Based on our experimental results, we will argue that the benchmark and detection models would benefit from a change in the task design. In other words, we suggest that the detection benchmark fabricated in Davidson et al. (2017) should undergo some modifications. We also conjecture that a larger share of the pitfalls mentioned by Wang (2018) can be attributed to the data itself rather than the detection model.

To support these claims, we build on the experimentation in Wang (2018). Concretely, we build a detection model that we then explain by means of more recent explainability techniques. For this purpose, we design a detection model based on a *Bidirectional LSTM* (BiLSTM). The model, as also sketched in figure 18, has a word embedding step followed by a BiLSTM layer as the main body. The output is then flattened and fed to a *Fully Connected* (FC) layer to compute class probabilities.

Tweets are preprocessed before being fed to the model. First of all, we annotate terms belonging to categories like *url, email, percent, number, user,* and *time*. For instance, "*341*" is replaced by "*<number>*". Then, we perform word segmentation and spell correction based on Twitter word statistics. Both methods and statistics were provided by the *ekphrasis* [4] text preprocessing tool (Baziotis, Pelekis, & Doulkeridis, 2017).

As for the model's implementation, we use Keras 2.3 (Chollet, 2015) with Tensorflow back-end 1.14 (Abadi et al., 2016). Individual words are embedded as 128-dimensional vectors. For training and evaluation, we use 60% of the dataset as training set and equally split the remaining samples in validation and test set, i.e. both 20% of the initial dataset. The performance of the model, although not the focus of this work, is comparable to the one achieved by recent methods (Davidson & Weber, 2019; Founta et al., 2019). The

[4]https://github.com/cbaziotis/ekphrasis

Speech Class	Model Results (F1 Scores)		
	Our Model	Davidson and Weber (2019)	Founta et al. (2019)
Hate Speech	0.30	0.4	-
Offensive Language	0.95	0.92	-
Neither	0.84	0.87	-
Overall	0.90	-	0.89

Table 1: Per class and overall F1 scores of our detection method in comparison to Davidson and Weber (2019) and Founta et al. (2019). We use - to indicate missing entries from the other works. Our model's scores are averaged over 5 runs with different train/validation/test sets.

results are laid out in table 1 in terms of per class and overall F1 score. As already mentioned for other methods benchmarking on Davidson et al. (2017), one can notice how also our model performs poorly on the hate class. For completeness, we report that additional BiLSTM and fully connected layers did not lead to any improvement in terms of performance.

To interpret our recognition method, we want to utilise a local post-hoc approach that provides feature attribution scores. Therefore, following our argumentation and suggestions from 3.7, we pick the *DeepSHAP* method (Lundberg & Lee, 2017).

As a short recap, DeepSHAP is a method tailored to DNNs that combines DeepLIFT and Shapley values. It explains a model's prediction by computing the input features' contributions w.r.t. a baseline. The baseline, which we denoted in 3.6 by ϕ_0, is computed by averaging the model's output across several samples and measures its background bias. Following the implementation guidelines from Lundberg and Lee (2017), we generate our baseline with a large amount of randomly selected tweets (in our case, 200). We then use DeepSHAP to analyse a few tweets from the hate class misclassified as offensive. In figures 19, 20 and 21 we report some concrete examples. The explanations are shown in the form of force plots, where every feature can be thought of as a positive or negative force that moves the predicted score, starting from a base value. In our case, the features' forces represent the words' contribution, and the base value is the classifier's bias ϕ_0. The force plots are w.r.t. the predicted class, i.e. offensive language.

Our technique constitutes a significant improvement over the approach in Wang (2018). The explainability method used in Wang (2018) looks at model sensitivity w.r.t. input words but neglects the sign of the contribution. In other words, their method cannot show which words contributed positively and which one negatively to the predicted class. Our analysis, on the other hand, lets us see in which direction words contribute to the prediction and not just the overall importance of a word. In each example, we see the model's prediction for every class on top together with the ground truth label. Below that, the preprocessed tweet is shown together with its corresponding explanation in the

```
TWEET:
 <user> you do not know shit about me faggot bitch

REAL CLASS: Hate
PREDICTED CLASS: Offensive

FORCE PLOT FOR THE Offensive CLASS:
```

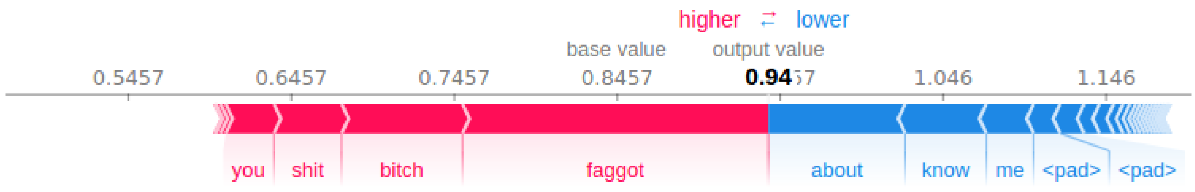


Figure 19: Directed hate misclassified as offensive language with very high confidence (94%). The two words "*bitch*" and "*faggot*" are the two main positive contributors to the score. Although the two words are indeed offensive, they misdirect the classifier which misses the clear hate emerging from the tweet.

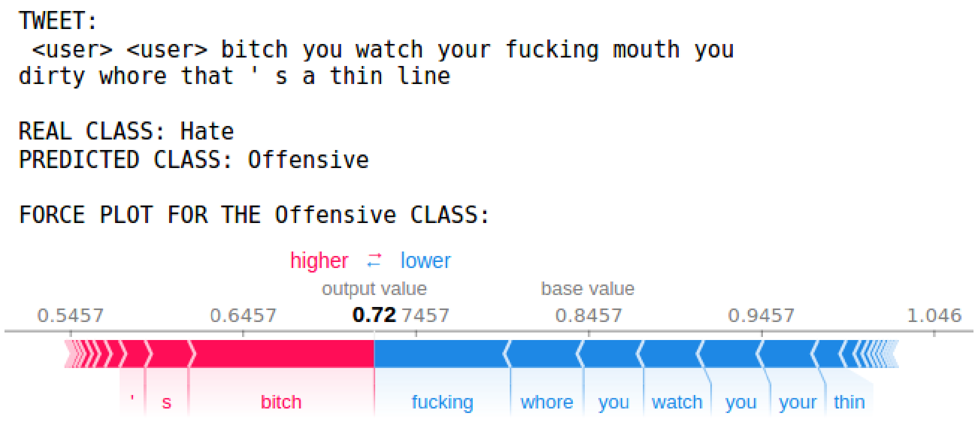
form of a force plot.

At first sight, one can immediately notice how the base value for the offensive language class is very high. Indeed, ϕ_0 is almost 85%, while one would expect something around 33% in a classification setting with three classes. The value expresses how the model is strongly biased towards classifying tweets as offensive. This confirms or rather explains the high F1 score in table 1 for the offensive class. It seems quite clear that the bias originates from the large portion of offensive tweets in Davidson et al. (2017) (more than 75%).

From the examples, it clearly emerges how offensive words like "*bitch*" or "*nigga*" push in favour of the offensive class. Although this is in principle correct, this leads to the classifier misclassifying sentences as offensive also when offensive words are used in a hateful context. Hence, samples are overall rarely classified as hate. This explains the low F1 score for hate in detection models that benchmark on Davidson et al. (2017), including ours.

Due to the design of the classification task in Davidson et al. (2017), the hate and offensive classes necessarily exclude each other. Offensive words and slurs are found on both sides of the force plot, supporting one of the two possible prediction outcomes. Although we agree with Davidson et al. (2017) that a further distinction between offensive and hate speech is necessary, we argue that classes' "competition" is not beneficial for the task. The classifier has to decide on whether a tweet is either hateful or offensive, where one choice excludes the other by design. Our explainability technique shows how this lead to offensive words contributing to one single outcome. As a consequence, most of them support the offensive class due to its bigger size within the dataset.

Our interpretation of this classifier suggests that the task should be adjusted to eliminate



```
TWEET:
 <user> <user> bitch you watch your fucking mouth you
dirty whore that ' s a thin line

REAL CLASS: Hate
PREDICTED CLASS: Offensive

FORCE PLOT FOR THE Offensive CLASS:
```

Figure 20: Another tweet with directed hate misclassified as offensive language, but this time with lower confidence. While the word "*bitch*" is found again as a positive contributor, the term "*fucking*" and the sexist slur "*whore*" keep the score from increasing. Overall, the output score is much lower than the base value, indicating how some feature induced the classifier to consider the tweet as hateful. This is unfortunately not enough given the high bias of the model towards offensive language.

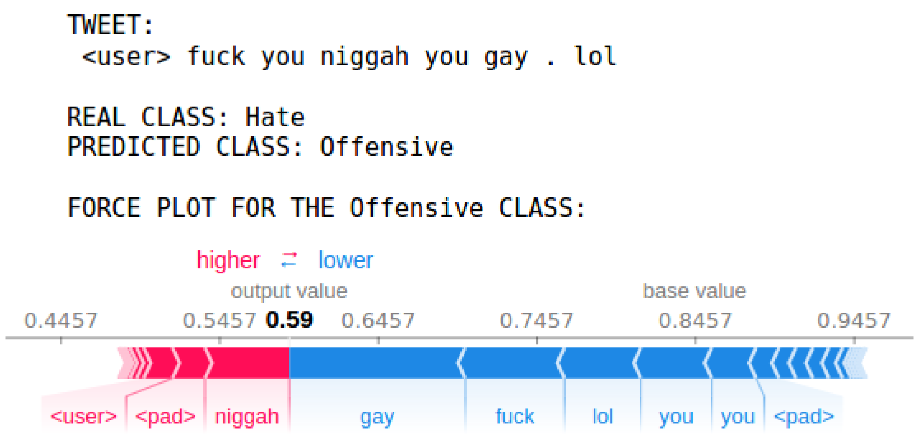
the competition between the two classes that intersect. One possible solution would be to switch to a multi-label classification task, where multiple classes are allowed as a predicted outcome. In this way, the two scores for offensive and hate are not competing and words can contribute to the detection of both.

We conjecture that also other problems such as overlocalisation, lack of sensitivity, and unintuitive features (Wang, 2018) would be less severe in a multi-label setting. The model could now learn to detect hate as a subset of offensive language and does not require words to stand for a single class.

On a more general note, we wonder if hate speech detectors purely based on text are ideal for the task or whether they suffer from a lack of context. Nevertheless, they are forced to learn a rule that connects some words in the input to a specific class of speech. The following example shows how some sentences are intrinsically not classifiable as hate or neutral without taking into account the context.

Example 10. *Let us consider the sentence* "oh my nigga you should be careful"*. This sentence can be hateful or not, depending on who says it, or more in general, depending on the context. The word* "nigga" *is a commonly used slang word in the black community. On the other hand, the same sentence sounds like a threat when said by any non-black person, especially if the author has a history of racist behaviour.*

This example shows how the exact same sentence should be classified differently depending on the context. Unfortunately, pure text classifiers are intrinsically not context-aware. This motivated a part of the existing research to include user traits and other kinds of

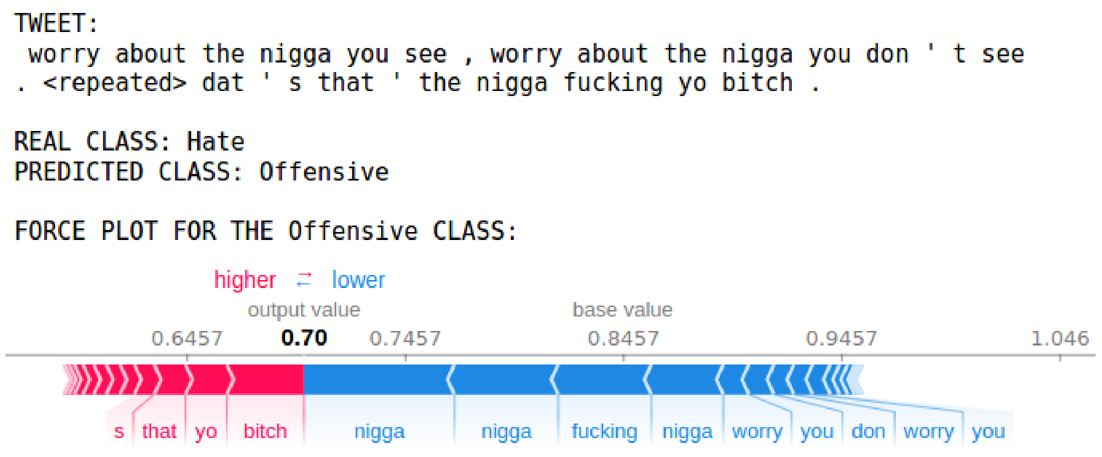


```
TWEET:
 <user> fuck you niggah you gay . lol

REAL CLASS: Hate
PREDICTED CLASS: Offensive

FORCE PLOT FOR THE Offensive CLASS:
```

(a) Directed racism/homophobia



```
TWEET:
 worry about the nigga you see , worry about the nigga you don ' t see
. <repeated> dat ' s that ' the nigga fucking yo bitch .

REAL CLASS: Hate
PREDICTED CLASS: Offensive

FORCE PLOT FOR THE Offensive CLASS:
```

(b) Generalised racism

Figure 21: Two examples of racist tweets. Both sentences contain racial slurs like "*nigga*". In one tweet the slur is directed at one user (a) while in the second one it has a more general target (b). Both tweets are classified as offensive but also report a relatively high hate score. The two slurs "*niggah*" and "*nigga*", although synonyms, contribute in opposite ways to the prediction.

social features in hate speech classification and will be the object of discussion in the following section.

4.2 Explainability and Social Features

Earlier in this work, the importance of context has emerged. Especially regarding hateful content detection, as mentioned in the previous section, solely relying on text seems a limitation. Social media platforms like Facebook and Twitter possess a colossal amount of user data, that we refer to as *social features*. This information might contain many useful features that could be employed to improve the detection of abusive behaviour.
Believing in the potential of social features, several works used them successfully to improve recognition performance. As already mentioned in 2.2.2, the literature is populated by a large variety of methods. For instance, some include the user's number of followers or friends (Chatzakou et al., 2017; Unsvåg & Gambäck, 2018). Others model social interactions with a graph and measure properties such as betweenness and eigenvector centrality (Ribeiro et al., 2018).
Despite the large number of detection frameworks that include social features, interpretability has not been yet considered in this setting. To the best of our knowledge, this is the first work that also focuses on explaining the behaviour of hate speech detectors that include social information.
In this section, we make a first step towards interpreting the effect of social features such as user traits and relationships in detection models. In 4.2.1, we describe the setting of our experiments by presenting the utilised data and detection models. In particular, we depict what social features we make use of and report the models' performance results. Following, in 4.2.2, we discuss the interpretability of social features. We show that their contribution to the output can still be calculated via Shapley values. Also, our experimentation empirically proves their usefulness. Then, in 4.2.3, we use explainability to seek a more in-depth understanding of why models benefit from having social information. We do so via exploring what the models have learnt via visualising their latent space. Afterwards, in 4.2.4, we observe the behaviour of our models on an artificially crafted tweet, i.e. not belonging to the dataset. We show how effective this type of analysis can be, particularly when combined with the techniques introduced earlier. Finally, in 4.2.5, we discuss the importance of considering the target audience of our explanations. Based on this discussion, we also look at how our techniques can be adapted depending on the final user.

4.2.1 Experimental Setup and Performance Results

In our experiments, we compare two models, one purely based on text and one that also takes into account additional information about the users and their interactions. As in 4.1 we exposed fallacies in the construction of the Davidson dataset (Davidson et al., 2017),

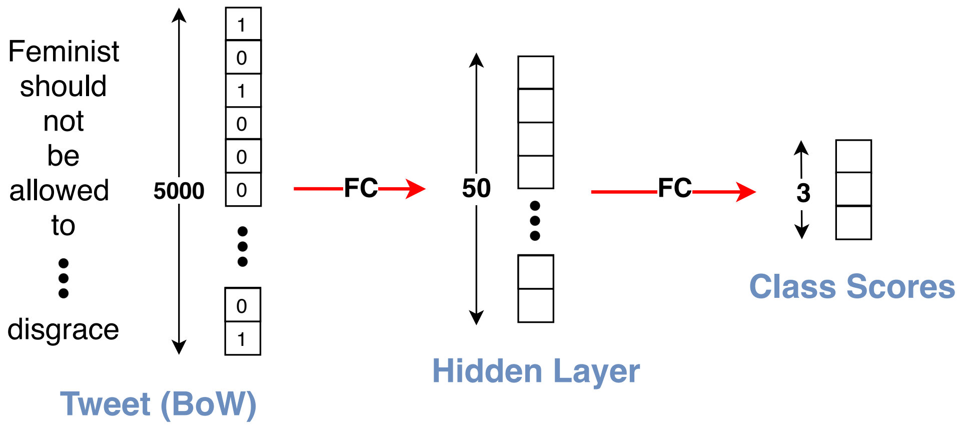


Figure 22: The architecture of our text model.

we do not use it further to benchmark our models. Instead, we pick another popular detection benchmark from Waseem and Hovy (2016). This dataset contains labels for different types of hate: sexism and racism. As we will see, this leads to insightful results that characterise how detection models perceive different types of hate.

The collection was originally composed of 16,914 labelled tweets (Waseem & Hovy, 2016). We were only able to retrieve 16,849 of them as some are not available anymore via the Twitter API. The tweets are divided into three classes: 3,378 in *sexism*, 1,970 *racism*, 11,501 *neither*. As already mentioned in 2.2.1, most of the disagreement between annotators was w.r.t. the sexism class, where class membership seemed not always explicit and straightforward to identify. Besides the tweets, we also want to model the relationship of their authors. To this end, we also create an undirected unlabelled *community graph* where nodes represent users, and the edges are the connections between them (Mishra et al., 2018). Namely, two nodes, i.e. users, are connected if one of them follows the other one. Eventually, we have a total of 2,031 nodes and 5,514 edges. The average degree of a node is 2.714 and 824 users are not connected to anyone else. Overall, the graph's density

$$D = \frac{2 \cdot \#\text{edges}}{\#\text{nodes} \cdot (\#\text{nodes} - 1)}$$

is 0.002674, thus indicating a very sparse network.

For a fair comparison, the two models use the same BoW representation for the input tweets. We limit the representation to the 5000 most common words due to memory constraints. As for implementation and preprocessing, we apply the same procedure as we did in 4.1 for the Davidson benchmark (Davidson et al., 2017).

The first model, shown in figure 22, purely utilises the tweets as input. The text is processed by two FC layers, which output probabilities for the three speech categories. We refer to this method as *text model*.

The second model instead has multiple input sources: the tweet, the user's vocabulary, and the follower network. The first is the same input as our text model. The second is

Speech Class	Model Results (F1 Scores)	
	Text Model	Social Model
Racism	0.711	0.735
Sexism	0.703	0.832
Neither	0.881	0.907
Overall	0.829	0.872

Table 2: Comparison between our text model and our social model. We report, for each, the overall and per class F1 scores. Again, all scores are averaged over 5 runs with randomly picked train/validation/test sets.

constructed from all the tweets of the author and represents their overall writing style. Concretely, we merge the tweets in their BoW representation, i.e. we apply a logical-OR to their corresponding vectors. The result is a BoW representation for the user, containing all the words used by the user in their tweets. The third, instead, is the follower network of the tweet's author and describes their surrounding community. On a more technical note, this can be extracted as a row from the adjacency matrix of our community graph. Both types of social features, i.e. the second and the third input, can also be found in SOTA models on the same benchmark (Mishra et al., 2018; Mishra et al., 2019a).

Regarding the architecture, the different input sources are processed in separate branches at first, each with an FC layer. Then, after being concatenated together, they are fed to two more FC layers as done for our text model. Ultimately, we sketch the architecture in figure 23. We refer to this model as *social model*.

We train both our models using 60% of the available dataset. We split the rest of the tweets equally into a validation and a test set. The results obtained, in terms of F1 score, are presented in table 2.

Our model with social features considerably outperforms (by 4.3%) our text model. Moreover, the improvement is visible in every single class.

Although pure performance is not the objective of our work, these results suggest that social data contains useful features for detection. Interestingly, our text model performs better on racist tweets than on sexist ones. We find this quite surprising since the sexism class is almost twice as big. Our finding suggests that sexism is, at least in this case, somewhat harder to detect in text. On the other hand, our social model has an impressive improvement in sexist tweets (almost 13%), suggesting the presence of detectable patterns in sexist users and their social interactions.

Evidently, our social model is superior to our text model in terms of performance. However, are social features interpretable? If yes, how? This will be the object of discussion in the next section.

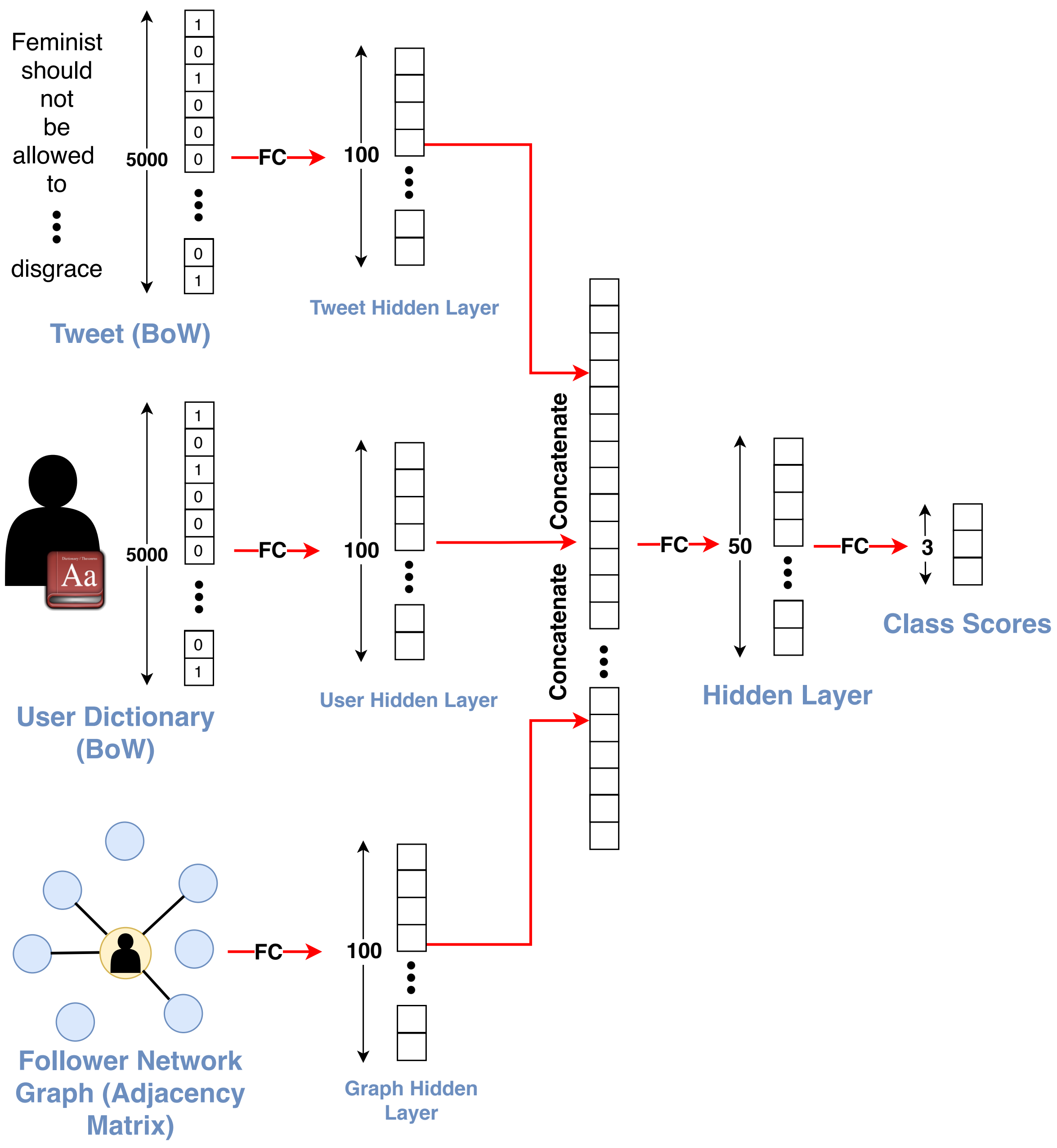


Figure 23: The architecture of our social model.

4.2.2 Interpretation of Social Features via Shapley Values

In 4.1 and in chapter 3, we attributed relevance scores to words as part of a text. Now, we face the challenge of also considering different kinds of input features, e.g. networks. There is not one unique rule on how to interpret particular inputs and the interpretation depends on the meaning that they carry. The following example shows the meaning carried by our follower network (as a graph) w.r.t. the users' relationships.

Example 11. *Let v be a user that has a relationship (e.g. being friends or followers, exchange messages, mention) with other users $v_1, v_2, \dots, v_d$ on a social media platform. If we model the social environment with a graph (V, E), then $(v, v_i) \in E$ if user v and v_i are related, e.g. one of them follows the other one. When a detection model f uses (V, E) as input, as does our social model, then edges can be modified to simulate a change in the social situation. For example, the graph $(V, E - \{(v, v_i)\})$ represents a hypothetical scenario in which v and v_i were not related.*

In the same way as words in a text, single edges in our follower network can be considered as other input features x_t. Hence, as we will see in the following, we can explain their effect with existing methods.
To estimate the contribution of an input feature x_t, we implement a simple method to approximate its corresponding Shapley value ϕ_t (Shapley, 1953), defined in 10. A similar estimation technique is used in Štrumbelj and Kononenko (2014). The algorithm, shown in the following, computes the Shapley values of the features of an input instance x.

Algorithm 1: Shapley Value Approximation

```
Result: Shapley value $\{\phi_t\}_{t=1}^M$ for every feature $\{x_t\}_{t=1}^M$
input $p$ sample probability, $x$ instance, $f$ model, $I$ number of iterations;
initialize $\phi_t = 0\ \forall t$;
for $i$ = 1,...,$I$ do
    for $t$ = 1,...,$M$ do
        sample a Bernoulli vector $P = \{0,1\}^M$ with probability $p$;
        pick $S$ a subset of the features $\{x_t\}_{t=1}^M \setminus \{x_t\}$ according to $P$;
        build $x_S$ alteration of $x$ with only features in $S$;
        $\phi_t \leftarrow \phi_t \frac{i-1}{i} + \frac{f(x_{S\cup\{x_t\}})-f(x_S)}{i}$
    end
end
```

The number of cycles I in the first for loop is arbitrary and only represents the number of runs performed. The higher this value is, the more accurate the approximation of ϕ_t will be. In our experiments, we found that $I = 10$ was already sufficient for convergence.

As sample probability, we arbitrarily chose $p = 0.7$. Interestingly enough, if we set $I = 1$ and $p = 1$ we obtain a simple occlusion method like described in 3.3.

We can notice that the algorithm requires to run the target model f with a perturbed input x_S. For our social model's first and second input sources, we can do this by removing words from the corresponding BoW representation. For the third input, i.e. the follower network, we alter the input as just described in example 11.

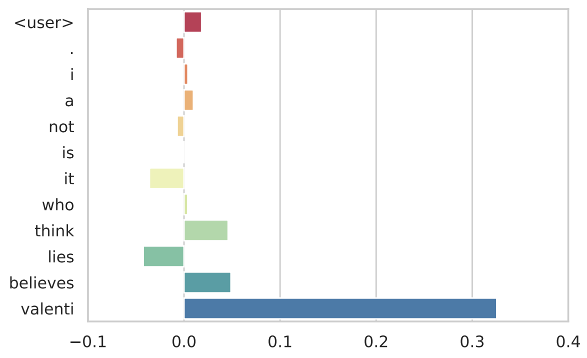
Note that in case graphs are not fed directly to the model as adjacency matrix and are instead embedded, then there is no need to occlude edges as described in 11. For instance, Mishra et al. (2018) embeds the follower network as dense vectors with *node2vec* (Grover & Leskovec, 2016), one of the many available techniques for embedding graphs. In this case, it would be sufficient to substitute the embedded nodes, i.e. numerical vectors, with a baseline.

Figure 24 illustrates the explanations for two tweets fed to our text and social models. For simplicity, we run the algorithm considering the user's vocabulary and the follower network as single features. As one can see, in 24c, the decision is strongly affected by the user's behaviour rather than by the tweet itself. Instead, in 24d, words still play the biggest role in determining the outcome. In both cases, we notice how adding context reduces the impact of the words that are the strongest contributors. For instance, in 24c, the contribution of the word "*valenti*" (largest contributor in 24a) shrinks by more than 0.1. The same can be observed for the words with the biggest impact in the second example (figures 24b and 24d).

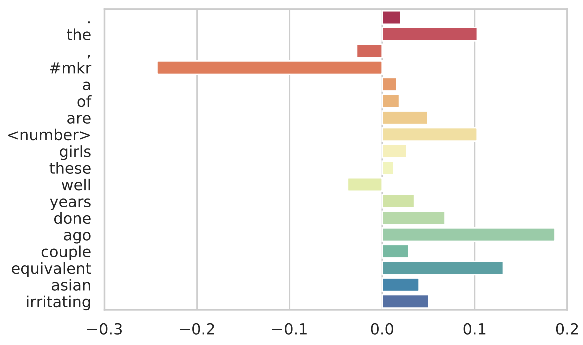
By explaining several other instances, we observed that the graph component of our input, i.e. the follower network, does not affect the decision as much as the user's vocabulary. Most likely, the cause is the graph's high sparsity, due to the limited amount of available data. We believe that future work, perhaps with the availability of a larger amount of data, will also benefit from explaining the effect of single edges or a small group of them.

Besides explaining the prediction, here explainability also shows that social features play a substantial role in the model's decision. Hence, it proves their usefulness to a more considerable extent than simply observing improved performance. Indeed, a higher F1 score could have been a result of the difference in the text and social models' architectures. However, thanks to explainability, we can be sure that using social features is the cause of the improved detection capability shown by our social model.

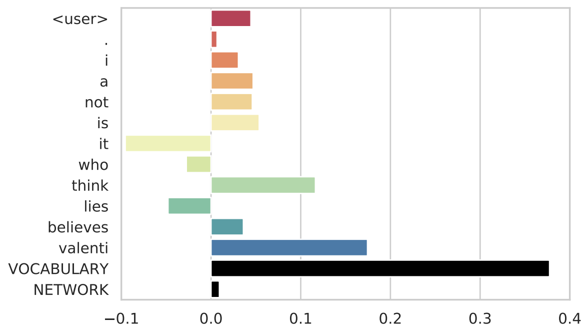
We conclude that this type of feature is indeed useful. Yet, we do not really know why. Can we also get a more in-depth understanding of why models benefit from having social information? As we will see in the next section, explainability comes in again as a powerful tool to tackle this question.



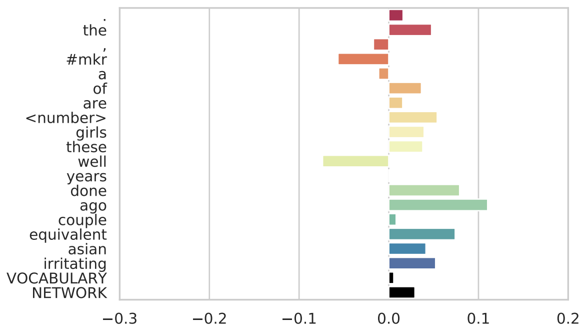
(a) Sexism, Text Model



(b) Racism, Text Model



(c) Sexism, Social Model



(d) Racism, Social Model

Figure 24: Example of features contribution, computed via Shapley value approximation, for our text and social models. In (a) and (c) we use as input the tweet "*<user> I think Arquette is a dummy who believes it. Not a Valenti who knowingly lies.*". The sexist tweet refers to the actress Patricia Arquette, who spoke in favour of gender equality, and the feminist writer Jessica Valenti. Some words are missing in the plot as our BoW dimension is limited during preprocessing. In (b) and (d), we use the racist tweet "*These girls are the equivalent of the irritating Asian girls a couple of years ago. Well done, 7. #MKR*". The hashtag refers to the Australian cooking show "*My Kitchen Rules*".

4.2.3 A more in-depth Perspective: Latent Space Visualisation

In this section, we focus on understanding *why* social features are beneficial for detection models. Unfortunately, Shapley values are not a viable option for this purpose as they only quantify *how much* single features contribute to the outcome. Hence, we need to use an explainability technique that explores the model more in-depth: *latent space exploration.* As we will see in the following, this approach is inherently global, and we treat it as such within this section. In the next section 4.2.4, we will convert this technique to be local, i.e. on a single instance. At the same time, we will use it in combination with Shapley values to provide more complete explanations.

Now, how do social features affect what models learn? We shall rephrase this question to a more general one: what have our models learnt? To answer this, we can take a look

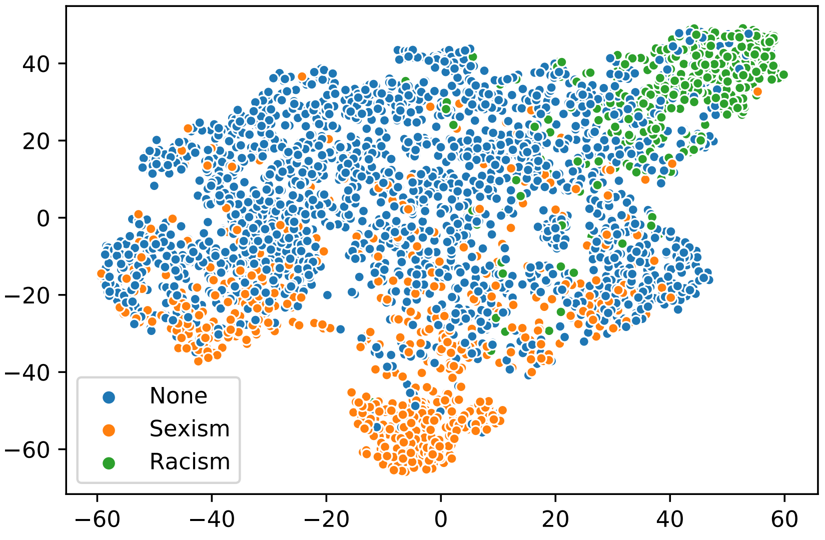


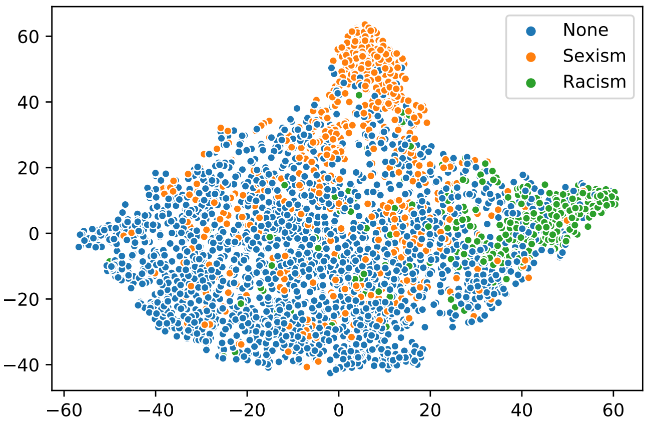
Figure 25: Tweets, colored by label, in the latent space of our text model.

at our tweets in the latent feature space learnt by the detection models. For instance, we consider our text model without the last FC operation. The tweets' embedding in the latent space can be obtained by feeding the tweets to this partial model and looking at the corresponding 50-dimensional output. To visualise high dimensional data, we employ a popular probabilistic technique, called *t-Distributed Stochastic Neighbor Embedding* (t-SNE) (Maaten & Hinton, 2008), which assigns a 2-dimensional location to each sample based on their similarity. A similar technique is used in Rizoiu et al. (2019) to understand the effect of transfer learning on a model.

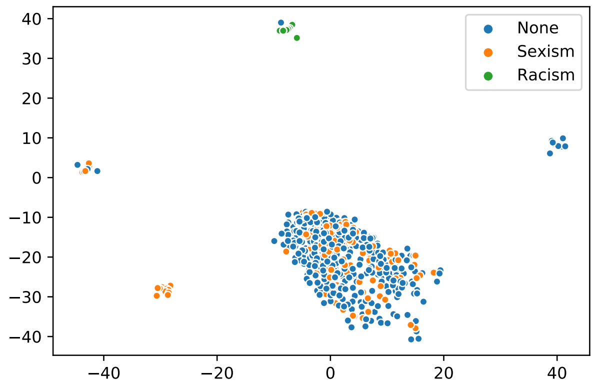
If we plot the resulting 2-dimensional map, shown in figure 25, we notice how all the tweets are in one single cluster. The racist tweets, in green, can be found in a very dense group. In contrast, the sexist ones, in orange, are more sparsely distributed and more hidden among the non-abusive samples. The plot suggests once again a more subtle nature of sexist language within this dataset. This is already the third indicator that we have, after the lower annotators' agreement and the poor text model's F1 score on the sexism class.

For completeness, we specify that visualisations with t-SNE can look different (e.g. slightly rotated) at each run of the algorithm. This can happen as t-SNE is not completely deterministic. However, the overall shape of the latent space plot and the relationship between the classes are not affected. For a concrete example, the reader can compare the figures 25 and 28, obtained from the same model.

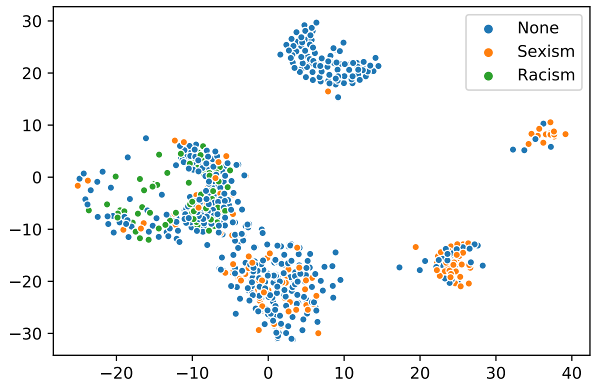
In light of the useful insights coming from this type of visualisation, we apply the same technique to our social model. At first, we analyse the single input features' hidden layers, i.e. the hidden layers right before the concatenation stage. The results, shown in figure 26, show the behaviour of the model in the different branches (compare 23). The tweet branch (figure 26a), not surprisingly, is very similar to the space learnt by our text model.



(a) Tweet Branch



(b) User's Vocabulary Branch



(c) Follower Network Branch

Figure 26: Latent space visualisation of our social model, coloured by label. The features are extracted from the single branches before the concatenation: tweet (a), user's vocabulary (b), follower network (c).

The user's vocabulary branch (figure 26b) shows the samples distributed in well-separated clusters. In particular, racist samples can all be found in a single and small group. The follower network branch (figure 26c) is similar to the one for the user's vocabulary, but groups are not as well-separated. This suggests once more that the follower network does not affect the decision as much as the user's vocabulary.

Overall, we see again racism concentrating in a small area of the plot and, thus, well-identified by the model. Sexism, on the other hand, appears in multiple clusters and is overall more mixed with regular tweets. The result validates, to some extent, the notion of *homophily* among racist users. In other words, racist users tend to cluster together or have a similar behaviour (Mathew, Dutt, Goyal, & Mukherjee, 2019).

Intuitively, being able to divide users into different clusters based on their behaviour could be helpful for classification at later layers. The higher performance of our social model suggests the same thing, but this is still not enough to understand how user information

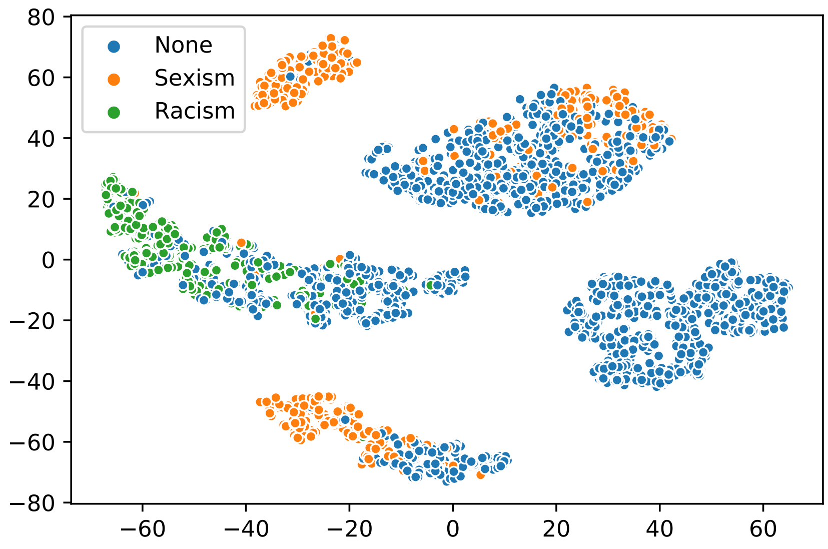


Figure 27: Tweets, coloured by label, in the latent space of our social model.

is combined with the tweet processing. Hence, we look at the latent space before the class scores of our social model. Here, the dimensionality has also been kept to 50 to have a better comparison with our text model latent space plot. The visualisation, which can be seen in figure 27, confirms our intuition. The tweets, which in our text model appeared in a single cluster, are now in well-separated groups. We can find racist posts almost exclusively in one single group. We also observe other pure or almost-pure clusters (meant as containing only or almost-only tweets with the same label). Once again, sexist tweets are more sparse and mixed with the other classes.

Qualitatively, we observe how the similarity metric learnt by the social model is substantially different from the text model. By no means do we know for sure whether this metric is intrinsically better or not. Nevertheless, it results in higher detection performance. As an intriguing observation, the last visualisation (figure 27) is to some extent a combination between what the text model has learnt (figure 25) and the information that could be extracted from the social data branches (figures 26b and 26c).

We have seen how social features can improve detection performance of hate speech by providing context. We also believe that even more refined social features exist and will be found in future research. Nonetheless, already the specific social features that we employed have a profound impact as shown by the big leap in performance and the explanations.

In the previous section 4.2.2, contribution scores (i.e. Shapley values) provided valuable information about how every input plays a role in the model's prediction. Here, instead, we explored the latent space learnt by a model and analysed the differences between our text and social model. While Shapley values proved the usefulness of social features, visualising the latent space explicitly illustrates *why* they are useful for a classifier.

So far, we studied and observed the behaviour of our models on a specific dataset. What happens when the model encounters a new tweet? In the next chapter, we show the benefits of explaining detection models on an artificially crafted tweet. Furthermore, we show how our visualisation techniques for exploring the latent space can be employed locally and in combination with Shapley values explanations.

4.2.4 Practical Analysis Beyond the Dataset: Explaining a Novel Tweet

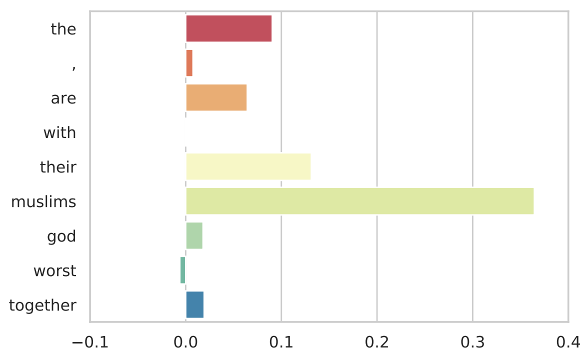
Often, we observe the behaviour of a model on a specific dataset: the one on which the model trained. But in practice, for the model to be applicable, it should be able to generalise far beyond the test set. For hate speech detection, this can be easily checked by feeding the model a novel tweet, i.e. one made from scratch. This is possible as everybody can quickly generate any kind of tweet. The same would not be possible, or at least not easy, in another domain like computer vision.

What if we are interested in checking how our detection method reacts to specific sub-types of hate? For instance, we want to check our model's behaviour when fed with an anti-Islamic tweet. To this end, we consider the racist sentence: "*muslims are the worst, together with their god*". We can feed it to our text model and look at the resulting prediction. The sentence is classified as racist with a 75% confidence.

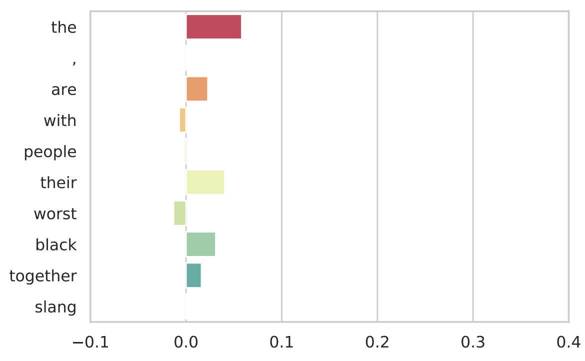
Furthermore, we can apply one of our explainability tools and check how single words play a role in the decision (figure 28a). The word "*muslims*" is the major contributor to this decision.

We can also inspect how this tweet compares to others already seen by our text model. The comparison can be made by adapting one of the tools from the previous section. In fact, we can convert our latent space analysis tool into a local explainability method by using it to project a new instance onto the latent space. In other words, we can use the method to plot where the new given tweet would be positioned in the space learnt by a detection model. This approach can give a more precise idea of how the model perceives the new tweet w.r.t. the dataset on which it has been trained. In figure 28c, we can see how the islamophobic sentence has been embedded among the other racist tweets, explaining how the model perceives it and thus confirming the original prediction.

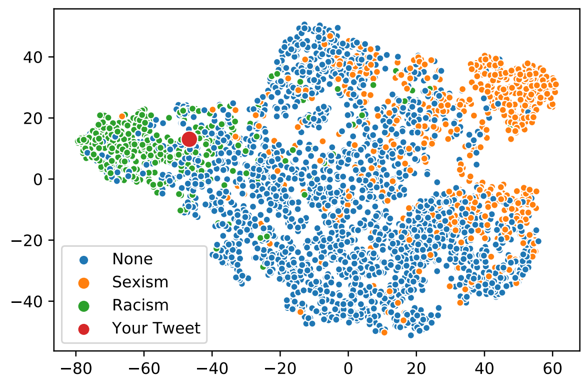
If we now change the target of our racist tweet, how does the model act in response? We compose the following hypothetical anti-black tweet: "*black people are the worst, together with their slang*". After feeding the fake tweet to our text model, we notice considerably different prediction probabilities. The text, judged before as racist when directed to Muslim people, is now judged with relatively high confidence as admissible (73% confidence). When we look at the explanations for such an outcome, our technique reveals that none of the words contributes significantly to the racism class. In particular, the term "*black*" did not affect the prediction as much as the word "*muslims*" did earlier. In the latent space, visualised in figure 28d, one can notice how the tweet location has moved slightly away from the racism cluster. This indicates the different perception that



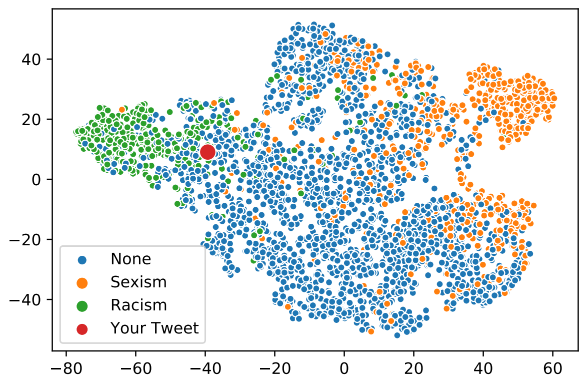
(a) Anti-Islam, Shapley Values



(b) Anti-Black, Shapley Values



(c) Anti-Islam, Embedding in Latent Space



(d) Anti-Black, Embedding in Latent Space

Figure 28: Features contribution (Shapley values w.r.t. the racism class) and embedding in the text model's latent space of an islamophobic and a anti-black racist tweets. The two sentences had, according to our text model, the 75% and 24% probability of being racist respectively.

the model has of anti-black tweets.

Both the prediction and explanations via Shapley values and latent space projection strongly suggest the presence of bias in our text model. On the one hand, the model is biased against black people as it does not detect hate towards them. On the other hand, the term "*muslims*" has a very strong negative impact, although it is by itself not a negative word. This could result in judging too harshly content that contains the term but is not overall harmful.

In general, one or two tweets are not enough to jump to conclusions. Surely, whether a model has a bias or not has to be confirmed by examining more cases. Nevertheless, these two techniques can be used complementary as useful indicators of potential biases. Shapley values can identify non-hateful terms that regularly play a substantial role in the outcome. In contrast, the latent space expresses the overall model's perception of a tweet w.r.t. to the dataset used for training. If changing the target of the hate changes the prediction and the model's perception, then the model/dataset might be biased against that target.

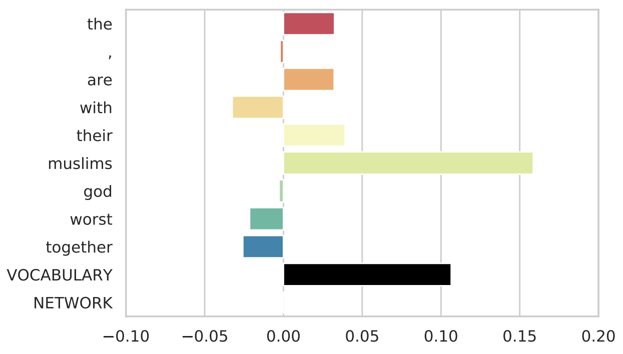
We can now run similar experiments with our social model. As we will see, the results will expose the nature of the model and how it differs from the text one. To feed our social model with the racist fake tweet, we also need to feed some author's features. For a fair comparison, we pick one random user with other racist tweets, one random user with other sexist tweets and one random user with no hateful tweets in the dataset. We refer to these users as racist, sexist and regular users respectively.

The islamophobic tweet is classified as racist when coming from a racist user (64%). However, it is instead judged non-hateful in both the other cases (12% and 19% for a sexist and user with no hate background respectively). Evidently, racist tweets also need some contribution from the social features to be judged as racist. In other words, it seems that racist tweets are predicted as such only if coming from racist users. But why is this the case?

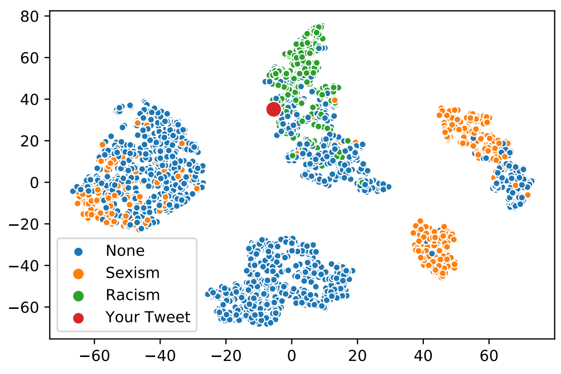
A very informative explanation comes again from both the Shapley values and latent space exploration (figure 29). On the left side, we can see the Shapley value for each case, i.e. for each type of user. All the words have a similar contribution to the racism class in all cases. However, the difference in the authors plays a substantial role in the decision. Only the racist user positively contributes to the racism class. On the right side of 29, we can see the embedding in the latent space for each case. Three different input authors cause the tweet to be embedded in three different clusters. Only in the first one the model actually considers the possibility of the tweet being racist.

Hence, both explanation techniques confirm that racist tweets need to come from racist authors to be recognised by our social model (or at least from a user in the same cluster). Hence, we have found another strong indicator of model's bias. Specifically, the model relies too much on the overall behaviour of users and judges as admissible hateful tweet when they come from a usually non-hateful source. In this sense, the model could favour excessively people with an overall non-hateful content. Perhaps, our model believes that racism comes almost exclusively from a small group of people that cluster together. This bias might have been picked up from the data as all the racist tweets in Waseem and Hovy (2016) were written by only nine users.

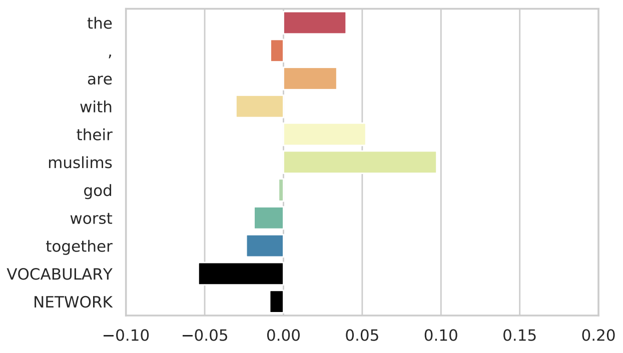
Ultimately, it becomes apparent how Shapley values and embedding in the latent space convey different kinds of information. The first one quantifies how every single feature plays a role, but does not really tell us what is happening in the background. The second one illustrates the model's perception of the tweet but does not provide any quantitative information for the prediction. Furthermore, we have seen that artificially crafting and modifying a tweet can be useful to examine the models' behaviour in some particular scenarios. In concrete examples, the two approaches worked as bias detectors both with only text and with also social features.



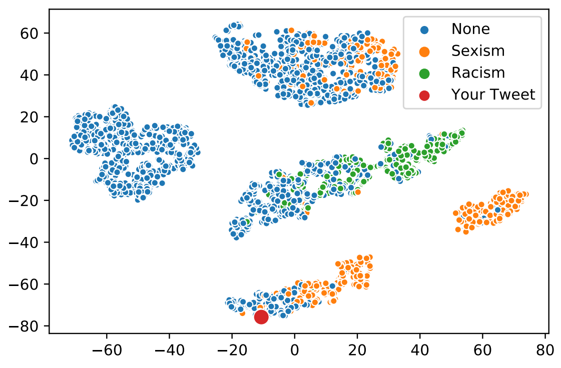
(a) Racist User, Shapley Values



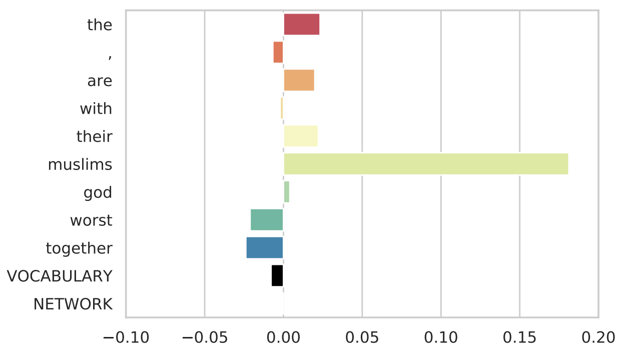
(b) Racist User, Embedding in Latent Space



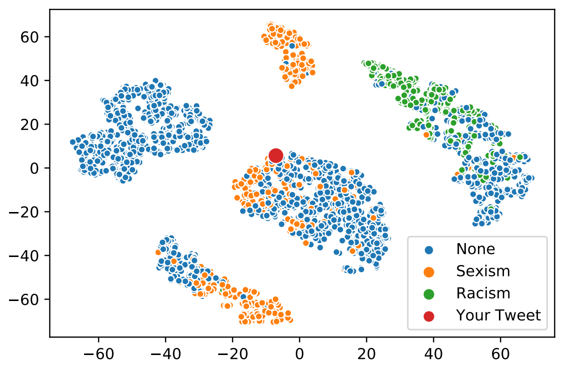
(c) Sexist User, Shapley Values



(d) Sexist User, Embedding in Latent Space



(e) Regular User, Shapley Values



(f) Regular User, Embedding in Latent Space

Figure 29: Features contribution (w.r.t. racism class) and embeddings of the islamophobic tweet in the social model's latent space. The three pairs of plots are w.r.t. to three predictions done with different users as input: a racist one (a,b, 64%), a sexist one (c,d, 12%), and a regular one (e,f, 19%).

4.2.5 Targeting the Right Audience

So far, we explored techniques that allow us to interpret the behaviour of a decision model. Previously, in 2.1.3, we also discussed what makes a good explanation and described several evaluation criteria. However, we have not discussed yet who is the target of our explanations. In other words, we should ask ourselves: *to whom* is this model interpretable? (Tomsett, Braines, Harborne, Preece, & Chakraborty, 2018).
Some works (Arrieta et al., 2020; Tomsett et al., 2018) already emphasise the importance of the audience for explanations. They also discuss and define possible roles that a human can have within the *machine learning ecosystem*. Each role, e.g. *creators, data-subjects, decision-subjects*, has a different conception of the ML system. Hence, they also have different expectations in terms of interpretability.
Embracing the concept of the audience's importance, we need to identify who is the final target of our explanations. For simplicity, we consider two major groups.

- *Developer:* we use this term to indicate anyone who is, at any stage, involved in the model's development. This includes all kinds of contributions, from early concept design to deployment and maintenance.

- *End User:* anyone who interfaces the ML algorithm and might be affected by its decisions.

Surely, in concrete cases, a more fine-grained categorisation can also be considered for designing explanations. The following example puts the two groups into a hate speech detection context.

Example 12. *Let us suppose that a social media service provider develops a new interpretable method for detecting hateful content among users on their platform. Anyone who develops the model and works on its correct functioning is a developer. If the service provider decides to apply some countermeasure on the content which is judged hateful, then every user who interfaces the effects of the detector is an end-user. Of course, the two types of target expect different explanations when interacting with the recognition algorithm.*

On the one hand, our first technique based on Shapley values attributes a contribution score to every input component: words and social features. We believe that this approach provides explanations suitable for end-users. The explanation is a list of reasons that is reasonably short and easy to understand. At the same time, the end-user can also edit the tweet and see how the model's reaction changes. This makes the explanations *contrastive* (see section 2.1.3) as the explanation's target, i.e., the user, can compare two scenarios with different decision outcomes (Lipton, 1990).
On the other hand, our second technique to explore the model's latent space and project a new tweet onto it seems more adapted to developers. We assume it is quite unlikely

that an end-user has the knowledge to interpret a latent space plot. Also, in most cases, only the developers have access to the detection model. Furthermore, combining the tool with the possibility of feeding an original tweet allows the developers to test the correct functioning in specific cases. For instance, our text model's behaviour on a racist tweet is a strong indicator of a model bias against black people that developers should identify and address.

Regarding our social model, we see how projecting a novel tweet onto the latent space can provide us with in-depth information on the model's functioning. Developers should be aware of the substantial importance that the model attributes to the user traits when making a decision.

For a regular user of an online social platform potentially affected by the hate speech recognition system, exploring the latent space is not an informative explanation. Among the tools we built, we certainly would recommend a feature contribution approach more as it is understandable and straightforward. To target people with more expertise, more complex information such as our latent space exploration can be included.

In this section, for simplicity, we treated two possible potential types of audience. We have also seen how our technique should adapt to meet their different level of expertise. In case a more fine-grained type of audience can be identified, it becomes easier to successfully meet the final user's needs.

In conclusion, the design of explanations should take the target audience into account as a priority. Explanations are especially informative when tailored to their intended final user.

5 Conclusion

Automatic detection of hate speech is urgently needed as online hateful behaviour continues to be a societal problem on a large scale. We have seen how detecting hate speech is an extremely challenging task. One of the main causes is the variety of forms in which hate appears: *implicit* or *explicit*, *directed* or *generalised*. Moreover, language subtleties like sarcasm and figures of speech represent additional obstacles that current research has to overcome.

Nonetheless, recent times have witnessed the proposal of a large variety of recognition methods and a steady growth in detection performance. Several strategies can be found in the existing literature, from lexicon-based algorithms to context-aware approaches. The latter category addresses current limitations in performance by including social features such as user traits and relationships.

ML technologies are widely used in recognition approaches. Indubitably, this is one of the key factors to their success in terms of detection performance and generalisation capabilities. However, they often behave as black-boxes which are not understandable to humans.

Our work focuses on explainable artificial intelligence for online hateful content recognition algorithms. In particular, as stated in 1, we pursued the following research objectives:

O1 Demonstrate the crucial role that interpretability can play in closing the gap between current machine learning research and real-life applications.

O2 Identify the most relevant tools for explaining hate speech detectors without sacrificing prediction performance.

O3 Explain the model's prediction also when social network data is exploited to construct more accurate models. Moreover, it should be clear how these features affect the outcome and the behaviour of the model.

We identify the inclusion of interpretability for learning methods as fundamental for the adoption of ML-based methods in real-life applications (**O1**). The reasons go from societal acceptance and deployment of unbiased decision-makers to benefits for scientific research and compliance with future legislation (**O1**). We decided to focus on post-hoc local methods. That is, we explain single predictions of black-box models, regardless of their complexity (**O2**).

We have examined existing methods for explaining text classifiers in-depth and analysed their theoretical foundations. This allowed us to understand the strengths and weaknesses of all methods and make a comparison between them (**O2**). As a result of our analysis, we recommend the usage of the methods *KernelSHAP* for model-agnostic explanations and *DeepSHAP* for explaining deep neural network predictions (**O2**). Furthermore, our literature review suggests that a more general theory of explanations is likely to emerge from future research.

We worked towards making hate speech detection explainable. For this purpose, we discussed and implemented various explainability techniques that provide crucial information about detection models. We showed how *DeepSHAP* can expose the fallacies in the Davidson benchmark construction (**O1**). In contrast to earlier analysis, we concluded that the task design is a major cause of the models' limited detection capabilities. More specifically, the task design induces competition between classes — this is detrimental to detection when two classes (such as offensive and hate) intersect. For future research, we also suggest considering settings that eliminate the issue of class competition, e.g. multi-label classification.

We have shown the importance of context in detecting hate speech. This was first suggested by observing that a model with social features has an improved performance (+4.3% F1 score) compared to one solely relying on text. The importance of context was then confirmed by two explainability techniques: *features contributions via Shapley values* and *latent space exploration* (**O3**). The former attests that social features play a role in the model's decision. The latter, instead, illustrates why models benefit from the incorporation of social features, as they can better separate speech classes.

We have also seen how these techniques can be combined together and applied to an artificially crafted tweet. This allows the user to examine the behaviour of the model in specific circumstances. Our techniques were applied to concrete examples. As a result, we were able to identify potential hidden biases in our models, both in the presence and absence of social features (**O3**). Shapley values might expose non-hateful words that consistently contribute positively or negatively to the hate class. Latent space exploration can illustrate the different model's perception of a tweet when changing the hate target or the author.

Our approaches also face some limitations. For instance, we are not able to explain the contribution of groups of words and entire sentences. We can quantify the role played by single words via Shapley values, but we do not yet understand how these words actually interact. Also, when explaining the effect of social context, we coarsely grouped social features. With the availability of more data, future approaches could explain the role played by more fine-grained social features, e.g. single relationships or small groups of them. Addressing these limitations is undoubtedly a promising direction for future research to close further the gap between machine learning research and real-life applications.

In real-life scenarios, besides accuracy, other requirements such as fairness and trust are fundamental for detection models. In this sense, we see the potential in interpretability to close the gap between accurate methods and methods that can be deployed in practice. Hence, we emphasise that future research should set interpretability as a priority when designing new detection frameworks. As a good design practice, we also encourage to identify the target user of the model's explainability. Explanations are especially informative when tailored to their intended final user.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467.*

Ancona, M., Ceolini, E., Öztireli, C., & Gross, M. (2017). A unified view of gradient-based attribution methods for deep neural networks. In *Nips 2017-workshop on interpreting, explaining and visualizing deep learning.*

Arras, L., Horn, F., Montavon, G., Müller, K.-R., & Samek, W. (2016). Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st workshop on representation learning for NLP* (pp. 1–7).

Arras, L., Montavon, G., Müller, K.-R., & Samek, W. (2017). Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th workshop on computational approaches to subjectivity, sentiment and social media analysis* (pp. 159–168).

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., … Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, *58*, 82–115.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), 130–140.

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep learning for hate speech detection in tweets. In *Proceedings of the 26th international conference on world wide web companion* (pp. 759–760).

Bansal, T., Belanger, D., & McCallum, A. (2016). Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th acm conference on recommender systems* (pp. 107–114).

Baziotis, C., Pelekis, N., & Doulkeridis, C. (2017). Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th international workshop on semantic evaluation (semeval-2017)* (pp. 747–754).

Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems* (pp. 4349–4357).

Cameron, A. C., & Windmeijer, F. A. (1997). An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics*, *77*(2), 329–342.

Chatzakou, D., Kourtellis, N., Blackburn, J., De Cristofaro, E., Stringhini, G., & Vakali, A. (2017). Mean birds: Detecting aggression and bullying on twitter. In *Proceedings of the 2017 acm on web science conference* (pp. 13–22).

Chen, Y., Zhou, Y., Zhu, S., & Xu, H. (2012). Detecting offensive language in social media to protect adolescent online safety. In *2012 international conference on privacy, security, risk and trust and 2012 international confernece on social computing* (pp. 71–80).

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1724–1734).

Chollet, F. (2015). Keras. Retrieved from https://github.com/fchollet/keras

Dadvar, M., Trieschnigg, D., Ordelman, R., & de Jong, F. (2013). Improving cyberbullying detection with user context. In *European conference on information retrieval* (pp. 693–696). Springer.

Datta, A., Sen, S., & Zick, Y. (2016). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 ieee symposium on security and privacy (sp)* (pp. 598–617).

Davidson, D., Thomas Bhattacharya, & Weber, I. (2019). Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the third workshop on abusive language online.*

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media.*

Denil, M., Demiraj, A., & De Freitas, N. (2014). Extraction of salient sentences from labelled documents. *arXiv preprint arXiv:1412.6815.*

Dhingra, B., Liu, H., Salakhutdinov, R., & Cohen, W. W. (2017). A comparative study of word embeddings for reading comprehension. *arXiv preprint arXiv:1703.00993.*

Dixon, L., Li, J., Sorensen, J., Thain, N., & Vasserman, L. (2018). Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 aaai/acm conference on ai, ethics, and society* (pp. 67–73).

Djuric, N., Zhou, J., Morris, R., Grbovic, M., Radosavljevic, V., & Bhamidipati, N. (2015). Hate speech detection with comment embeddings. In *Proceedings of the 24th international conference on world wide web* (pp. 29–30).

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608.*

Duggan, M. (2017). *Online harassment 2017.* Pew Research Center.

Edwards, L., & Veale, M. (2017). Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for. *Duke L. & Tech. Rev. 16*, 18.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining* (pp. 259–268).

Founta, A. M., Djouvas, C., Chatzakou, D., Leontiadis, I., Blackburn, J., Stringhini, G., ... Kourtellis, N. (2018). Large scale crowdsourcing and characterization of twitter abusive behavior. In *Twelfth international aaai conference on web and social media.*

Founta, A. M., Chatzakou, D., Kourtellis, N., Blackburn, J., Vakali, A., & Leontiadis, I. (2019). A unified deep learning architecture for abuse detection. In *Proceedings of the 10th acm conference on web science* (pp. 105–114).

Galán-García, P., Puerta, J. G. d. l., Gómez, C. L., Santos, I., & Bringas, P. G. (2016). Supervised machine learning for the detection of troll profiles in twitter social network: Application to a real case of cyberbullying. *Logic Journal of the IGPL*, *24*(1), 42–53.

Ganin, Y., & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd international conference on international conference on machine learning - volume 37* (1180–1189).

Gitari, N. D., Zuping, Z., Damien, H., & Long, J. (2015). A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, *10*(4), 215–230.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on international conference on machine learning* (513–520).

Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision-making and a “right to explanation”. *AI magazine*, *38*(3), 50–57.

Grover, A., & Leskovec, J. (2016). Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 855–864).

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., & Pedreschi, D. (2019). A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, *51*(5), 93.

Hall, M. A., & Frank, E. (2008). Combining naive bayes and decision tables. In *Flairs conference* (Vol. 2118, pp. 318–319).

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems* (pp. 1024–1034).

Heider, F., & Simmel, M. (1944). An experimental study of apparent behavior. *The American journal of psychology*, *57*(2), 243–259.

Herron, P. (2004). Machine learning for medical decision support: Evaluating diagnostic performance of machine learning classification algorithms. *Data Mining: Spring*, *24*.

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

Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression.* John Wiley & Sons.

Jaccard, J. (2001). *Interaction effects in logistic regression.* SAGE Publications, Incorporated.

Japkowicz, N., & Shah, M. (2011). *Evaluating learning algorithms: A classification perspective.* Cambridge University Press.

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

Kim, B., Rudin, C., & Shah, J. A. (2014). The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in neural information processing systems* (pp. 1952–1960).

Kim, B., Chacha, C. M., & Shah, J. A. (2015). Inferring team task plans from human meetings: A generative modeling approach with logic-based prior. *Journal of Artificial Intelligence Research*, *52*, 361–398.

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, *13*, 8–17.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196).

Li, J., Monroe, W., & Jurafsky, D. (2016). Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220.*

Liang, B., Li, H., Su, M., Li, X., Shi, W., & Wang, X. (2018). Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing.*

Lipovetsky, S., & Conklin, M. (2001). Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, *17*(4), 319–330.

Lipton, P. (1990). Contrastive explanation. *Royal Institute of Philosophy Supplements*, *27*, 247–266.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, *16*(3), 31–57.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems* (pp. 4765–4774).

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N., & Frieder, O. (2019). Hate speech detection: Challenges and solutions. *PloS one*, *14*(8).

Mathew, B., Dutt, R., Goyal, P., & Mukherjee, A. (2019). Spread of hate speech in online social media. In *Proceedings of the 10th acm conference on web science* (pp. 173–182).

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st international conference on learning representations, ICLR 2013*.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, *267*, 1–38.

Mishra, P., Del Tredici, M., Yannakoudakis, H., & Shutova, E. (2018). Author profiling for abuse detection. In *Proceedings of the 27th international conference on computational linguistics* (pp. 1088–1098).

Mishra, P., Tredici, M. D., Yannakoudakis, H., & Shutova, E. (2019a). Abusive language detection with graph convolutional networks. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2019* (pp. 2145–2150).

Mishra, P., Yannakoudakis, H., & Shutova, E. (2019b). Tackling online abuse: A survey of automated abuse detection methods. *arXiv preprint arXiv:1908.06024*.

Mittelstadt, B., Russell, C., & Wachter, S. (2019). Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency* (pp. 279–288).

Mohseni, S., & Ragan, E. D. (2018). A human-grounded evaluation benchmark for local explanations of machine learning. *arXiv preprint arXiv:1801.05075*.

Molnar, C. (2019). *Interpretable machine learning: A guide for making black box models explainable*. https://christophm.github.io/interpretable-ml-book/.

Možina, M., Žabkar, J., Bench-Capon, T., & Bratko, I. (2005). Argument based machine learning applied to law. *Artificial Intelligence and Law*, *13*(1), 53–73.

Munro, E. R. (2011). The protection of children online: A brief scoping review to identify vulnerable groups. *Childhood Wellbeing Research Centre*.

Murdoch, W. J., & Szlam, A. (2017). Automatic rule extraction from long short term memory networks. *arXiv preprint arXiv:1702.02540*.

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, *116*(44), 22071–22080.

Oliver, J. R. (1996). A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of management information systems*, *13*(3), 83–112.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp)* (pp. 1532–1543).

Poerner, N., Schütze, H., & Roth, B. (2018). Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 340–350).

Ray, A., Yao, Y., Kumar, R., Divakaran, A., & Burachas, G. (2019). Can you explain that? lucid explanations help human-ai collaborative image retrieval. In *Proceedings of the aaai conference on human computation and crowdsourcing* (Vol. 7, *1*, pp. 153–161).

Razavi, A. H., Inkpen, D., Uritsky, S., & Matwin, S. (2010). Offensive language detection using multi-level classification. In *Canadian conference on artificial intelligence* (pp. 16–27). Springer.

Regulation, G. D. P. (2016). Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. *Official Journal of the European Union (OJ)*, *59*(1-88), 294.

Ribeiro, M. H., Calais, P. H., Santos, Y. A., Almeida, V. A., & Meira Jr, W. (2018). Characterizing and detecting hateful users on twitter. In *Twelfth international aaai conference on web and social media.*

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1135–1144).

Rizoiu, M.-A., Wang, T., Ferraro, G., & Suominen, H. (2019). Transfer learning for hate speech detection in social media. *arXiv preprint arXiv:1906.03829.*

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215.

Rudin, C., & Ustun, B. (2018). Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. *Interfaces*, *48*(5), 449–466.

Samek, W., Binder, A., Montavon, G., Lapuschkin, S., & Müller, K.-R. (2016). Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, *28*(11), 2660–2673.

Schmidt, A., & Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media* (pp. 1–10).

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the ieee international conference on computer vision* (pp. 618–626).

Selvaraju, R. R., Lee, S., Shen, Y., Jin, H., Ghosh, S., Heck, L., … Parikh, D. (2019). Taking a hint: Leveraging explanations to make vision and language models more grounded. In *Proceedings of the ieee international conference on computer vision* (pp. 2591–2600).

Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, *2*(28), 307–317.

Shrikumar, A., Greenside, P., Shcherbina, A., & Kundaje, A. (2016). Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the 34th international conference on machine learning-volume 70* (pp. 3145–3153).

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *2nd international conference on learning representations, ICLR 2014*.

Sood, S. O., Antin, J., & Churchill, E. (2012). Using crowdsourcing to improve profanity detection. In *2012 aaai spring symposium series*.

Spertus, E. (1997). Smokey: Automatic recognition of hostile messages. In *Aaai/iaai* (pp. 1058–1065).

Springenberg, J., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2015). Striving for simplicity: The all convolutional net. In *Iclr (workshop track)*.

Strout, J., Zhang, Y., & Mooney, R. J. (2019). Do human rationales improve machine explanations? In *Proceedings of the second blackboxnlp workshop at acl* (pp. 56–62).

Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, *41*(3), 647–665.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th international conference on machine learning-volume 70* (pp. 3319–3328). JMLR. org.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. 2nd International Conference on Learning Representations, ICLR 2014.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Tomsett, R., Braines, D., Harborne, D., Preece, A., & Chakraborty, S. (2018). Interpretable to whom? a role-based model for analyzing interpretable machine learning systems. *arXiv preprint arXiv:1806.07552*.

Unsvåg, E. F., & Gambäck, B. (2018). The effects of user features on twitter hate speech detection. In *Proceedings of the 2nd workshop on abusive language online (alw2)* (pp. 75–85).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Wachter, S., Mittelstadt, B., & Russell, C. (2017a). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech. 31*, 841.

Wachter, S., Mittelstadt, B., & Floridi, L. (2017b). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, *7*(2), 76–99.

Wang, C. (2018). Interpreting neural network hate speech classifiers. In *Proceedings of the 2nd workshop on abusive language online (alw2)* (pp. 86–92).

Warner, W., & Hirschberg, J. (2012). Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media* (pp. 19–26). Association for Computational Linguistics.

Waseem, Z. (2016). Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter. In *Proceedings of the first workshop on nlp and computational social science* (pp. 138–142).

Waseem, Z., & Hovy, D. (2016). Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the naacl student research workshop* (pp. 88–93).

Waseem, Z., Davidson, T., Warmsley, D., & Weber, I. (2017). Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the first workshop on abusive language online* (pp. 78–84).

Williams, M. L., Burnap, P., Javed, A., Liu, H., & Ozalp, S. (2020). Hate in the machine: Anti-black and anti-muslim social media posts as predictors of offline racially and religiously aggravated crime. *The British Journal of Criminology*, *60*(1), 93–117.

Witten, I. H., & Frank, E. (2002). Data mining: Practical machine learning tools and techniques with java implementations. *Acm Sigmod Record*, *31*(1), 76–77.

Xu, J.-M., Jun, K.-S., Zhu, X., & Bellmore, A. (2012). Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 656–666). Association for Computational Linguistics.

Young, H. P. (1985). Monotonic solutions of cooperative games. *International Journal of Game Theory*, *14*(2), 65–72.

Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, *30*(9), 2805–2824.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer.

Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference* (pp. 745–760). Springer.