Autonomous Agents and Multi-Agent Systems (2023) 37:38
https://doi.org/10.1007/s10458-023-09619-4

A multi-scenario approach to continuously learn and understand norm violations

Thiago Freitas dos Santos[1,2] · **Nardine Osman**[1] · **Marco Schorlemmer**[1]

Accepted: 25 July 2023 / Published online: 16 August 2023

Abstract
Using norms to guide and coordinate interactions has gained tremendous attention in the multiagent community. However, new challenges arise as the interest moves towards dynamic socio-technical systems, where human and software agents interact, and interactions are required to adapt to changing human needs. For instance, different agents (human or software) might not have the same understanding of what it means to violate a norm (e.g., what characterizes hate speech), or their understanding of a norm might change over time (e.g., what constitutes an acceptable response time). The challenge is to address these issues by learning to detect norm violations from the limited interaction data and to explain the reasons for such violations. To do that, we propose a framework that combines Machine Learning (ML) models and incremental learning techniques. Our proposal is equipped to solve tasks in both tabular and text classification scenarios. Incremental learning is used to continuously update the base ML models as interactions unfold, ensemble learning is used to handle the imbalance class distribution of the interaction stream, Pre-trained Language Model (PLM) is used to learn from text sentences, and Integrated Gradients (IG) is the interpretability algorithm. We evaluate the proposed approach in the use case of Wikipedia article edits, where interactions revolve around editing articles, and the norm in question is prohibiting vandalism. Results show that the proposed framework can learn to detect norm violation in a setting with data imbalance and concept drift.

Keywords Norm violation · Incremental learning · Pre-trained language models · Interpretability · Online communities

Disclaimer: This article presents content (offensive language) that may be disturbing to different audiences.

✉ Thiago Freitas dos Santos
thiago@iiia.csic.es

Nardine Osman
nardine@iiia.csic.es

Marco Schorlemmer
marco@iiia.csic.es

[1] Artificial Intelligence Research Institute (IIIA), CSIC, Barcelona, Catalonia, Spain

[2] Universitat Autònoma de Barcelona, Barcelona, Catalonia, Spain

1 Introduction

The ability to continuously learn what constitutes norm violation, as understood by a given community, and detect when such violations happen is essential for any normative system that intends to regulate the behavior of its interacting agents (in this work, referred to as community members). This is especially critical considering that discrimination, hate speech, and cyberbullying can cause significant harm to individuals and negatively impact the community experience in online platforms [38, 60, 81]. Thus, our work aims to address two main challenges. First, to continuously learn what an online community understands as norm violation by using examples of behaviors depicted as such, gathered either as text sentences or formalized as a set of features. Second, explain to community members the parts of their actions associated with norm-violating behavior. To do that, we are interested in finding and adapting the definition of norm violation as interactions unfold. It is important to note that not only online communities stand to benefit from this research, as the challenges we tackle are also of interest to fields in which detecting misbehavior can prevent infractions (e.g., credit card fraud, personal information leakage, and network infiltration).

Some interesting approaches to detect norm violation in online communities have already been proposed, with applications to Wikipedia [8, 104], Software Engineering (SE) communities [22, 23], Reddit [19] and other communities [46, 64, 106]. However, these approaches can not continuously update the system used to classify an action as a norm violation. Consequently, they can not handle the evolution of the community's view about what constitutes such violations. This characteristic is fundamental in our work since we argue that the understanding of a norm violation is dynamic, e.g., what is considered hate speech may change rapidly as new members are incorporated and interactions unfold. For Instance, the word "nigger" may be viewed differently as more African Americans join the community and begin to salute each other using this term. In this context, a normative system deployed to govern these interactions must adapt to the current view of the community. We address this issue by proposing a framework that handles the interactions of an online community as a stream of actions with an imbalanced class distribution and the presence of concept drift. In other words, a stream of actions that contain more elements related to regular behavior than to violation behavior, aggregating to that, changes in how the community members understand norm violation. Thus, unlike existing approaches, our framework handles the dynamic nature of norm violations in online communities by incorporating community members' feedback as the ground truth to continuously adapt to changes in the meaning[1] of norm-violating behavior over time.

Furthermore, previous works in the realm of norms and normative systems have addressed different challenges that arise in the field, with a series of proposals to handle mechanisms for norm conflict detection [4, 32], norm synthesis [67, 71] and norm emergence [58, 68, 88]. Additionally, several domains have benefited from this field, applying the concepts of norms and normative systems to the prevention of discrimination by Machine Learning (ML) models [25], to the formalization of contracts and laws [36, 82], and to handling ethical dilemmas and moral values [5, 91]. In this work, we are particularly interested in supporting normative systems with mechanisms for learning from interactions and agents' feedback (human or artificial) to help decide what

[1] In this work, the parts of an action (i.e., set of features or words of a sentence) are the elements that indicate the meaning of a violation.

Table 1 Examples of tasks that benefit from solutions that handle featurized and/or text datasets

Task	
Set of features	Text
Fraud discovering [9, 50, 95]	Hate speech detection [6, 23, 81]
Misbehaving detection [43, 52]	Fake news detection [45, 69, 98]
Identify deception writing styles [3, 93]	Adversarial attacks [41]
Fake reviews detection [13, 63]	Identify AI-generated reviews [1, 62]
Identify authors of violations [75, 94]	Style change detection [96, 110]

Specifically, we focus on violating behavior in online interactions

is considered a norm violation. In online communities, interactions are defined by the actions executed by community members that affect the whole community. For example, consider editing Wikipedia articles. In this scenario, interactions are the article edits performed by some members and the subsequent reading of those by others.

Table 1 highlights different scenarios where the formalization of online interactions may vary depending on the type of violation-detection task. Specifically, while some tasks only require a set of features to describe an action, others must handle the raw action input as it takes place due to the complexity of the domain. Thus, we are interested in building a framework employed in multi-scenario settings, namely tabular and text-based domains, since these cover a variety of use cases (including detecting violations in Wikipedia article editing). For instance, fraud discovering and misbehaving detection formalize an action as a set of features like user engagement and user-user interaction. While identifying deception writing styles might require mapping the input to linguistic-based features. These tasks usually benefit from tabular-related approaches. However, domains like hate speech detection, tackling the spread of fake news over social media, and adversarial attacks, need solutions that handle natural language sentences directly.

To create our solution, we investigate incremental learning in a framework that can learn the meaning of norm violation, adapt to changes in community view, and incorporate feedback from community members in the learning procedure. We depict the relationship between the components of our framework in Fig. 1. Specifically, this approach offers the following advantages in our context. First, it continuously updates the base classifiers using data blocks that arrive sequentially (Sect. 2.2). This embodies in our framework the ability to adapt to changes in the community view (concept drift), using only the most recent data to learn the meaning of norm violation and discarding the need to treat and maintain past information. Second, it facilitates incorporating feedback from community members as the ground truth about a norm violation. This aligns with our view that a system's understanding of norm violation needs to adapt to its users (in our case, community members).

Additionally, our framework incorporates an ensemble of Feed-forward Neural Networks (FNNs) with the incremental learning approach to handle class distribution imbalance in our tabular task context (Sect. 2.1). This is particularly useful when learning norm violation since this behavior usually happens less frequently than regular behavior. As for text-related scenarios, our approach incorporates Pre-trained Language Models (PLMs) [47] (Sect. 2.3). In contrast to FNNs, PLMs do not require an ensemble

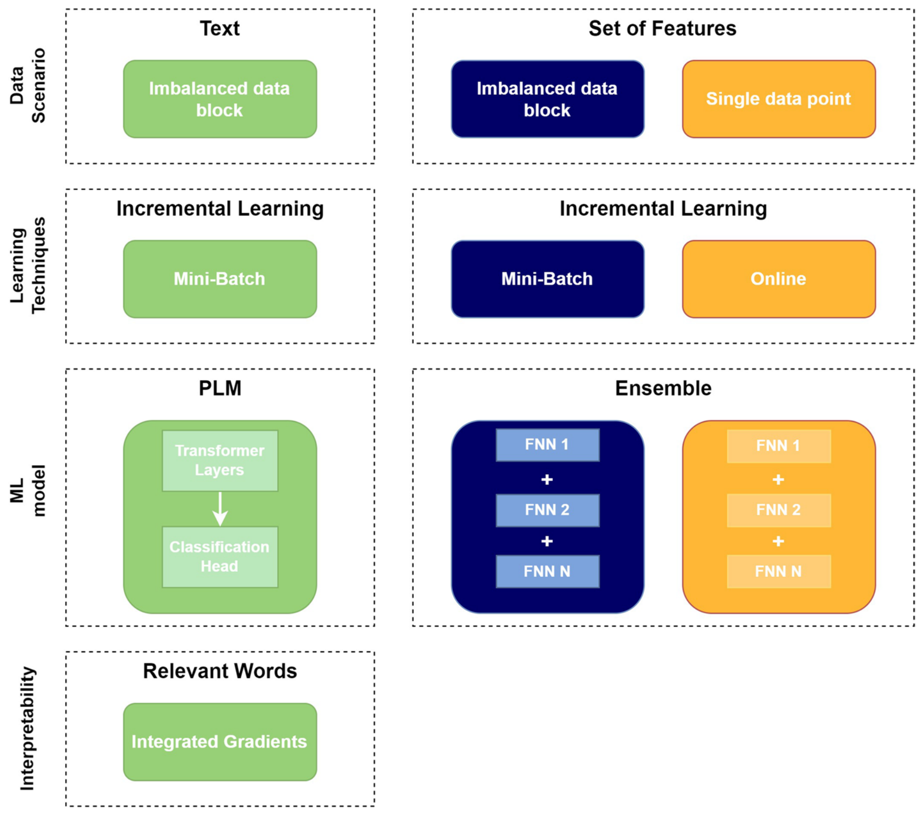


Fig. 1 A conceptual framework of our multi-scenario approach. In the ensemble, *N* indicates the number of classifiers

of classifiers. Instead, they can handle imbalanced datasets by encoding language structures (due to their size) and undersampling the majority class. Furthermore, to learn specific classification tasks, PLMs fine-tune only the classification head, which is integrated on top of the pre-trained layers of the model.

Besides detecting a norm violation, deployed systems must also provide information on the reasons behind a decision. Therefore, we investigate interpretability to obtain explanations about an ML model's inner workings. Here, we are interested in understanding the words in a text sentence usually associated with norm violation. Since we use PLMs to solve these tasks, our framework employs the Integrated Gradients (IG) algorithm [97] to explain the behavior of our transformer-based models (Sect. 2.4). IG provides relevant words related to norm violation, enabling our framework to tackle two issues. First, it allows us to adhere to the principles of Responsible AI [10] since we enhance people's understanding of our models. Second, it prepares our approach for a future argumentation process, as information provided by the interpretation step assists users in deliberating and collaboratively agreeing on the definitions of norm violation.[2]

[2] We envision people to be in control of defining the meaning of their community norms and expect them to collectively agree on those norms through deliberation and argumentation mechanisms.

The experiments (Sect. 4) describe the implementation of two incremental learning techniques to train the base classifiers: mini-batch learning and online learning. In this work, FNNs are the models in the ensemble for tabular tasks. At the same time, we evaluate text scenarios by comparing two PLMs, DistilBERT and RoBERTa. The use case is the editing of Wikipedia articles. In this scenario, we detect norm violation either by formalizing an action (article edit) as a set of features provided by the community (e.g., number of profane words, occurrences of alphanumeric characters, etc. [34]) or by treating the text sentence input directly. An example of a violation is the sentence *"the big lipped,hairbraned,egotistical dirty nigger often defecated"*. Results show that the proposed approaches can learn the meaning of norm violation in an online community with imbalanced class distribution (only around 7% of the data correspond to edits with violation) and in the presence of concept drift (changes in the community view). To formalize an article edit, we define the tuple (X, y), in which X is the set of features of an action and $y \in \{0, 1\}$ is its class label, 0 denotes regular behavior, while 1 denotes norm-violating behavior.

This research extends our previous work [33] by (1) incorporating a mechanism to handle violations expressed as text sentences; (2) learning a multi-label task through the identification of different classes of violating behavior; (3) understanding the words usually associated with norm violation, specifically determining their relevance to different violation cases; and (4) comparing two PLMs to analyze their ability to learn in this scenario and how their architecture impacts the understanding of violating behavior.

The remainder of this paper is divided as follows. Section 2 presents the basic mechanisms used by our proposed framework, described in Sect. 3. Section 4 shows its application to the use case of Wikipedia article edits, and Sect. 5 discusses the results. Related literature is presented in Sect. 6, and we give our conclusions and propose our future work in Sect. 7.

2 Background

This section presents the base concepts upon which this work is built. First, we start by presenting an ensemble strategy to deal with the imbalanced nature of the dataset when handling a tabular task. Second, we describe the incremental learning approach used to continuously train the ML models considered in this work. Third, we introduce the concept of the Pre-trained Language Model (PLM), which is responsible for handling actions as natural language sentences. Lastly, we describe explainability and its application to understanding the classification output of PLMs.

2.1 Ensemble learning

Dealing with the detection of norm-violating behavior usually leads to cases of imbalanced datasets. This happens because regular (or expected) behavior is more common than violations. Thus, solutions that deal with domains in these settings must be equipped to handle class distribution imbalance. Otherwise, the solutions tend to be biased towards the class that describes regular behavior (the majority class). To tackle this issue, we use ensemble learning, which can be defined as the generation and combination of different ML models (e.g., neural networks, random forest, and logistic regression) to solve a predictive task [83]. The main idea of this technique is that by combining multiple ML models using a voting scheme, the errors of a single model will be compensated by the others. Thus the overall performance of the ensemble would be better than the performance of a single component [29].

Different ensemble methods can be used to build a classification system. Dong et al. [29] highlight some important ones, such as Bagging, AdaBoost, and Random Forest. Bagging is an interesting method to deal with the challenge of imbalanced datasets investigated in this work. This technique finds a solution by training different base classifiers in different subsets of the initial dataset. Then, the ensemble uses majority voting to decide the final output. As an example, in a binary tabular classification task with an imbalanced dataset D, it is possible to divide D into two subsets, majority class subset M and minority class subset P (the number of instances in these sets is represented by $|M|$ and $|P|$, respectively). In this context, the main goal is to train an ensemble E with n number of balanced datasets $B = \{B_1, ..., B_n\}$. Each $B_i \in B$ is a dataset with a similar class distribution, and $n = |M| \div |P|$. In this manner, because the number of instances in $P \subseteq D$ is smaller than the number of instances in $M \subseteq D$, subsets in B have size $2 \times |P|$ and are created with $|P|$ non-overlapping instances from M, while all instances of P are replicated to each subset.

The bagging method, as described above, can be applied to train ML models offline or in a mini-batch manner. However, this method cannot be used in an online setting (in which training happens one instance at a time). To solve this issue, modifications to the bagging procedures are necessary. Thus, Wang et al. [102] present a resampling strategy to deal with imbalanced datasets for the online case. This strategy considers two approaches, Oversampling-based Online Learning (WEOB1) and Undersampling-based Online Learning (WEOB2), with the addition of weight adjustment over time. WEOB1 and WEOB2 work to adjust the learning bias from M to P by resampling instances from these subsets. Specifically, oversampling increases the number of minority instances, while undersampling decreases the number of majority instances. Like the traditional bagging strategy, online bagging creates different classifiers and trains them a k number of times by considering only the current data point. k is defined by the $Poisson(\lambda = 1)$ distribution. As data becomes available, the λ parameter is calculated dynamically according to the imbalance ratio. In this manner, if there is a new instance in P, then k increases. However, if there is a new instance in M, then k decreases.

2.2 Incremental learning

Since we are dealing with online communities, we must consider how data is made available. Usually, systems must work with a stream of data that arrives sequentially. In this context, there are different ways to build a framework capable of solving the problem. Techniques differ in how they handle the data stream and, consequently, how the algorithms are trained. Following this idea, we can separate training techniques into two big groups: offline and incremental learning.

Offline learning deals with the complete dataset; in this case, it is impossible to update the trained model. To incorporate new knowledge, an entire training process from the beginning is necessary [40], which is the main drawback of this approach when we must handle non-stationary domains. Besides, maintaining and treating all the data for this kind of learning can be costly and complex (especially when considering data regulations specified by different entities and legislators) [50].

On the other hand, incremental learning is the technique that addresses the limitations of offline learning by continuously updating the ML model with new data as it becomes available. This approach is particularly beneficial in online communities since the models must be constantly updated as people interact and a change in understanding emerges. In this work, we are concerned with mini-batch and online learning. Mini-batch learning creates and uses small sets of data that arrive continuously to train ML models. Since we only deal with the most recent instances that compose the present data block of a fixed size, the process is neither as costly nor as complex as offline learning [50, 54]. Online learning can be seen as a special case of mini-batch learning, in which the batch size is 1. Thus, as soon as data is made available, it is possible to update the ML model, discarding the need to store this data point and consequently avoiding the complexities of data treatment. It is important to highlight one advantage of mini-batch over online learning regarding stability properties. Since in online learning, the training procedure only considers one data point at each time step, the algorithms that implement this concept usually have the poorest stability when compared to mini-batch algorithms [54] (in Sect. 5 we also demonstrate this phenomenon).

Incremental learning approaches, which involve the continuous updating of base models as new data becomes available, can be useful for investigating problems that involve concept drift, i.e., the change in the view of the community members about what is regular and violating behavior. It is possible to identify the shift in community behavior by observing the joint distribution $P_t(X, y)$ over time [56, 101], where $x \in X$ is a feature value, $y \in \{0, 1\}$ is the associated class label that denotes regular or norm-violating behavior, and t the current timestamp. Then, to compare two moments in time and detect a possible concept drift, we refer to the following: $P_t(X, y) \neq P_u(X, y)$, where u is a timestamp in the past. Gama et al. [35] define three ways to categorize concept drift: change in the prior probability of classes $p(y)$, affecting the ratio between violation and regular behavior; change in the class conditional probabilities $p(X \mid y)$, impacting how violation and regular behavior are defined; this has an impact on the posterior probabilities of classes $p(y \mid X)$, which is a change in what the community understands as a violation and regular behavior. The latter leads to real concept drift, which is the definition that interests us in this work.

2.3 The pre-trained language model (PLM)

The first part of our proposal focuses on solving tabular classification problems. However, we aim to broaden the framework's scope to a more generic solution by incorporating the ability to solve tasks in text classification scenarios. Different approaches to learning patterns from natural language sentences have been proposed in the literature, ranging from probabilistic classifiers using TF-IDF [44, 109] and Recurrent Neural Network (RNN) [89] to transformer-based models, used in this work.

Recently, transformer models have been the primary approach for addressing Natural Language Processing (NLP) tasks, surpassing previous methods and consistently achieving the highest performances across various domains [47, 61, 99]. One of the advantages of the transformer is its ability to process text data by reducing the amount of work needed in the featurization step [78]. Figure 2 presents the transformer layer architecture and the advances incorporated, such as the addition of the attention mechanisms and the use of fully connected FNN layers [105], assembled in a parallelized way to improve computational performance.

The multi-head attention mechanism enables the transformer model to learn the relationship between different words in a text sequence by calculating an attention score. Consider the sentence *"Wikipedia is important to society since it is a relevant source of*

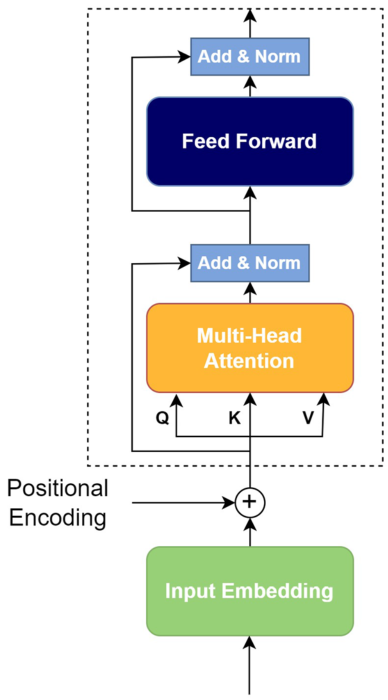


Fig. 2 The transformer layer, as proposed by Vaswani et al. [99]

information". This mechanism iteratively calculates the attention score between all words in the sentence, thus obtaining the dependency relationship between them [99]. In this specific instance, "Wikipedia" and "it" present a high attention score since they are related and represent the same concept. Meanwhile, the words "society" and "it" receive a low attention score. Besides, the attention mechanism adds context to sentences [99], enabling the model to differentiate the meaning of words, like the notions of "bank" as a financial institution and "bank" of a river. To leverage this mechanism, transformer-based models employ a multi-head strategy, with several attention heads computed in parallel. Here, words are encoded as embeddings in a vector space (input embedding) and are combined with the positional encoding, which inserts information about the word's position in a sentence and allows the model to handle long-range texts [99]. Equation 1 formalizes how attention is calculated.

$$Attention(Q, K, V) \leftarrow softmax\left(\frac{Q \odot K^T}{\sqrt{d_k}}\right) \odot V \tag{1}$$

Q, K, and V are matrices that represent every word in a sentence and $\odot$ is the dot product. These matrices receive the same input and differ only in their learned weights, acquired by training in a large-scale dataset. d_k is used as a scaling factor and encodes the dimension of interest [99] between Q and the transpose of K. Finally, the *softmax* value is combined with V to obtain the final attention score.

To improve training efficiency, the transformer normalizes the output of the intermediate sub-layers (multi-head attention and feed-forward) [12, 107]. It does that by calculating the distribution statistics (mean and standard deviation) from the addition of the output of the sub-layers, forwarding the normalized values to the next step. Equation 2 formalizes this process.

$$LayerNorm(x) \leftarrow \frac{g}{\sigma} \odot (x - \mu) + b \tag{2}$$

g and b are the gain and bias parameters, respectively. They have the same dimension as the output of the previous layers and are dynamic terms learned iteratively during the training process (large-scale datasets). σ is the standard deviation and μ the mean. x is the previous layer's output.

Since the transformer incorporates a point-wise network, each normalized node of the attention layer is forwarded through the FNN. At this step, the transformer applies two linear transformations, using the ReLU (Eq. 3) activation function [99]. To formalize the complete step, we present Eq. 4.

$$ReLU(z) \leftarrow max(0, z) \tag{3}$$

$$FFN(x) \leftarrow ReLU(x \times W_1 + b_1) \times W_2 + b_2 \tag{4}$$

ReLU (Eq. 3) executes a non-linear operation that aims to calculate the final value given by a previous NN layer (z). In Eq. 4, x represents the output of the attention layer, W_1 represents the weights of the first linear transformation, and W_2 the second. b_1 and b_2 are the bias terms added to both steps.

The architecture described above is the basic block for building a Pre-trained Language Model (PLM), a large Deep Neural Network (DNN) used to solve complex NLP tasks. To create a PLM, multiple transformer layers are stacked and initially trained on large-scale datasets [105]. Different implementations yield state-of-the-art results, e.g., BERT [47], which has around 110 million trainable parameters, RoBERTa [55], around 125 million trainable parameters, and DistilBERT [86], 66 million trainable parameters. Since we are dealing with large DNNs, it would be impossible to train these models from scratch to handle each task. Thus, PLMs take advantage of the fine-tuning paradigm to adapt to specific tasks [31].

The fine-tuning process requires using previously trained implementations and incorporating a new FNN layer on top of it, referred to as the classification head. Here, we are interested in the task of text classification in a violating-behavior setting, i.e., given a text as input, the model predicts whether the text is violating the norm of the community. In this scenario, the transformer layers are used for language representation. These layers can be applied to any domain since they were trained in large-scale datasets. On the other hand, the classification head is responsible for the output. Thus it is explicitly trained only for the task at hand, considering a given domain dataset and the community requirements, such as the number of output nodes (binary or multi-label tasks) and the number of instances used for training.

Concretely, our work explores two different PLMs. The first is RoBERTa, built on top of BERT to improve its implementation by changing the architecture design and training on a larger dataset, obtaining better performance for different NLP tasks [55]. The second is DistilBERT, which is also built on top of BERT, but it aims to create a smaller, faster, and cheaper model [86]. Section 5.2 presents the results of RoBERTa and DistilBERT applied to hate speech detection in Wikipedia article edits.

2.4 Interpretability of PLM

Unlike our tabular scenario, text-related tasks do not need a featurization process (encode text sentences into a set of attributes). Instead, it is possible to manipulate the text directly [10, 26, 72, 79]. In this context, we incorporate the Integrated Gradients (IG) algorithm [97] to understand the parts of a text sentence most relevant to the model's output. IG enables our framework to gain insights into the inner workings of transformer-based models by debugging and extracting rules from a DNN [97].

Understanding the internal mechanisms of our model is crucial for the effective interaction of people with a model's output, which is especially relevant for two main reasons. First, it is an essential part of our solution to inform community members about violated norms, allowing people to consider the elements of their actions associated with violating behavior. Second, we align with the goals of Responsible AI [10], particularly regarding the transparency of the decision-making process of an ML model.

The literature usually focuses on two interpretability techniques to explain how an ML model works. First is local interpretability, which involves identifying the words (or features) that contributed to the model's output regarding a *specific* action. Second is global interpretability, providing a broader understanding of the model's inner workings. We focus here on the local interpretability method since, at this step of our work, providing community members with information on specific text violations is the primary goal. To achieve this, IG calculates a word's contribution by a backward pass through the model, propagating its relevance from the output to the input [57]. The central assumption of this algorithm is that the tokens with the highest gradient values present the most substantial influence on the classification output.

Following the formalization in [57, 97] and considering an NLP task, let x be the sentence formed by a set of tokens $x_i, i \in 1, 2, ...n$ and $\bar{x}$ the baseline input represented by a zero embedding vector. $\frac{\partial M(x)}{\partial x_i}$ is the gradient for token i and M is our transformer-based model. Theoretically, to obtain the integrated gradients, IG considers a straight-line path from the baseline $\bar{x}$ to the input x, computing the gradients at all points of the path [97]. Thus, the integrated gradients come from the accumulation of these individual points. Equation 5 formalizes the integral calculation.

$$IntGrads(x_i) \leftarrow (x_i - \bar{x}_i) \odot \int_{\alpha=0}^{1} \frac{\partial M \times (\bar{x} + \alpha \times (x - \bar{x}))}{\partial x_i} \times d\alpha \tag{5}$$

However, to efficiently compute the integrated gradients, IG approximates $IntGrad(x_i)$ by the Riemann sum method (Eq. 6), which defines a set of finite points (m) along the straight-line path. $r(x_i)$ is the calculated relevance score and m is chosen empirically. Experiments in [97] suggest around 20-300 points along the path.

$$r(x_i) \leftarrow (x_i - \bar{x}_i) \odot \sum_{k=1}^{m} \frac{\partial M\left(\bar{x} + \frac{k}{m} \times (x - \bar{x})\right)}{\partial x_i} \times \frac{1}{m} \tag{6}$$

Finally, in our use case, after obtaining the relevance score for each token present in the original text sentence, we follow a two-step process. First, we convey to the community member (executing an action) the reasons for a model's output. Section 5.2.3 and "Appendix 2" showcase how this information is presented. Second, we prepare our framework to provide interpretability data to other community members in a future argumentation

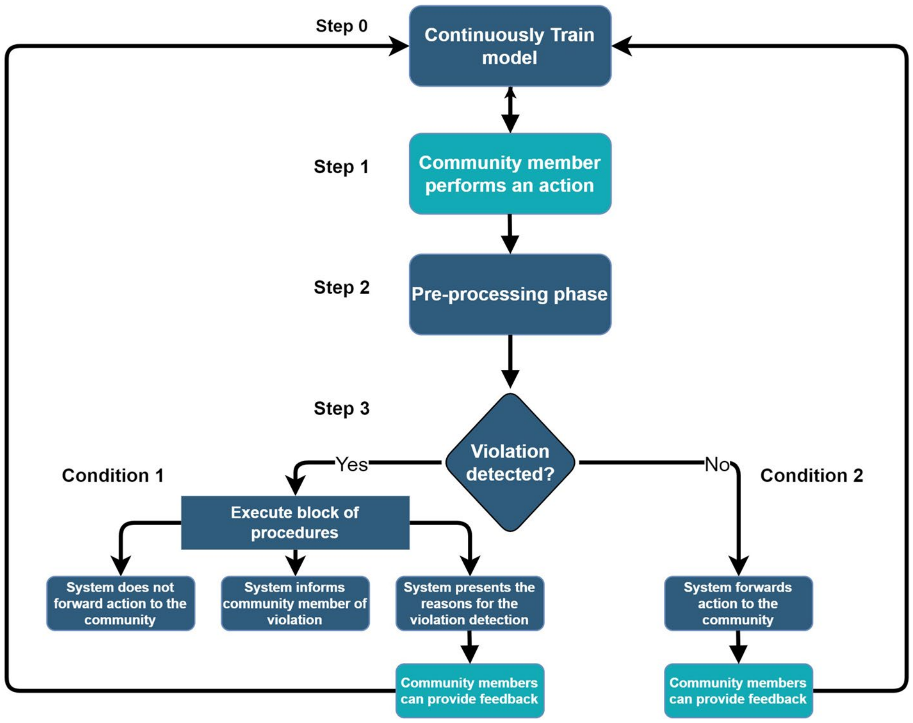

Step 0
Continuously Train model
Step 1
Community member performs an action
Step 2
Pre-processing phase
Step 3
Violation detected?
Yes
No
Condition 1
Condition 2
Execute block of procedures
System does not forward action to the community
System informs community member of violation
System presents the reasons for the violation detection
System forwards action to the community
Community members can provide feedback
Community members can provide feedback

Fig. 3 The process in which our solution would be implemented

process, focusing on discussing the reasons behind a violated norm and gathering the evolving community views. Subsequently, we use the feedback in this step to update the trained model, as we envision community members constantly defining the meaning of norm violations.

3 The multi-scenario incremental learning framework

In this section, we present the proposal of our work, a framework capable of learning the meaning of norm violations through the combination of ensemble and incremental learning for tabular tasks, and the use of PLMs for text tasks. The main idea is to deploy this framework in a normative system to support the fulfillment of norms, especially when considering prohibited behavior.

Figure 3 outlines the workflow for deploying our framework. The initial step, Step 0, involves the continuous training of machine learning models, including the ensemble of classifiers and the PLM. The framework starts by training with data blocks, and upon completion of the first block, the model is prepared to detect norm violations. Subsequently, the system starts monitoring every new action performed in the community (Step 1). In Step 2, the system can map the action to a set of features defined by the community or directly handle text input. In the latter scenario, text-processing steps such as correcting words, addressing grammatical errors, and removing non-alpha-numeric characters might be performed. Step 3 presents the two distinct paths that the system may execute. Should the action be detected as a norm violation (Condition 1), the system must execute a sequence of steps to ensure that the violation is not forwarded to the entire community. These steps include: (1) rejecting the action (i.e., the action is not executed); (2) informing the user about the violation and its blocking; (3) providing the reasons for violation detection, including the specific features or words associated with the violating behavior, allowing for community feedback and the opportunity for correction of our system. Community feedback can then be used to continuously train the base model (Step 0).

In contrast, if the executed action is not detected as a norm violation (Condition 2), the action is forwarded to the community. In this case, community members can still provide feedback due to the possibility of incorrect classification by our model. Besides, the feedback incorporates new community views to update the understanding of norm violations through continuous model training (Step 0).

The following sections delve into the different approaches for implementing our solution,[3] particularly considering some challenges in norm violation detection. For instance, norm violations often lead to working with imbalanced datasets since detrimental behavior does not happen as often as regular (or expected) behavior. Thus, when building a solution to tackle learning in this setting, it is necessary to work with an approach capable of handling imbalance in class distribution. In this research, we investigate the use of ensemble machine learning to tackle this issue in the tabular scenario while we investigate under and over-sampling for text-related tasks. We also apply two approaches to continuously update the base ML models: mini-batch and online learning. Although we use the mini-batch approach to fine-tune the PLMs, online learning is not feasible due to the size of these models. As a result, our framework considers only mini-batch learning for text classification tasks.

3.1 Mini-Batch Learning for Tabular Scenarios

As data is made available sequentially, the algorithm starts to build blocks of data with fixed size n. As soon as a data block contains n data points, the algorithm is ready to start the training procedure of the ML classifiers. The mini-batch approach explored in this work (Algorithm 1) builds on top of two incremental ensemble algorithms, the Accuracy Updated Ensemble (AUE2) [17] and the Dynamic Updated Ensemble (DUE) [54]. The differences introduced by our approach are: (1) incorporating feedback to emphasize data points that had their class labels changed by the community; (2) using a replication-based oversampling technique that randomly replicates minority class instances present in the current data block instead of using the SMOTE [21] oversampling technique that creates synthetic minority class samples. Additionally, we define a new metric (number of classifiers) to define the oversampling ratio for the minority instances (Algorithm 1, line 6).

In our case, the majority class M represents expected behavior, while the minority class P represents norm-violating behavior. Since we define an action as a set of features, we represent a data point with the tuple (X, y), in which X is the set of features of an action and $y \in \{0, 1\}$ is its class label. Thus, a data block is defined as $D_t = \{(X, y)_1, ..., (X, y)_n\}$, with n being the data block size. After the data is pre-processed, the algorithm starts by calculating the imbalance ratio (Algorithm 1, line 5) between sets P_t and M_t in the current data

[3] Source code available at https://bitbucket.org/thiago-phd/jaamas_2023/src/

Algorithm 1 The Mini-Batch Training procedure.

```
Input: Current data block ($D_t$), set of majority instances ($M_t$), set of minority instances ($P_t$), set of instances with feedback ($F_t$), max number of classifiers ($m$), max change in distribution ($d$), and number of epochs ($e$)
Output: Trained ensemble ($E$).
1: Initialize ensemble size. $s_0 \leftarrow 0$.
2: Initialize the last imbalance ratio change. $r_0 \leftarrow 0$.
3: while data block is available do
4:     Pre-process $D_t$, no past data is used.
5:     Compute the current imbalance ratio. $r_t \leftarrow |P_t| \div |M_t|$.
6:     Compute the current best ensemble size. $s_t \leftarrow |M_t| \div |P_t|$.
7:     if $s_t > m$ then
8:         Oversample minority class instances $\in P_t$.
9:         Update $s_t$ with the new value for $|P_t|$.
10:    end if
11:    if $r_{t-1} = 0$ or $r_t \div r_{t-1} < 1 - d$ then
12:        Compute the number of new classifiers. $c \leftarrow s_t - s_{t-1}$.
13:    end if
14:    Emphasize set $F_t$ by oversampling with ratio $|P_t| \div |F_t|$.
15:    Add $F_t$ to the data block $D_t$.
16:    for $i = 1$; $i \leq s_t$; $i{+}{+}$ do
17:        Get a subset $S_{t,i}$ from $M_t$, where $|S_{t,i}| = |P_t|$ and $S_{t,i} \cap S_{t,u} = \emptyset$ $(u = 1, 2, ..., i-1)$.
18:        Create a balanced dataset. $B_{t,i} = S_{t,i} \cap P_t \cap F_t$.
19:    end for
20:    Train $s_t$ classifiers with $B_t$ for $e$ epochs.
21:    Discard the current data block $D_t$ and increment $t$.
22: end while
```

The Mini-Batch Training procedure.

block D_t. Besides, set M_t and set P_t are used to calculate the number of classifiers in the ensemble s_t (Algorithm 1, line 6).

To illustrate in more detail how Algorithm 1 works, it is interesting to use an example. Let us say that initially $t = 1$, $s_t = 10$, and the imbalance ratio $r_t = 0.07$. Then, after some time, a concept drift is noted at time step $t = 5$, with r_t changing to 0.03 and s_t changing to 12. Next, if $s_t > m$, the algorithm oversamples set P_t by duplicating all minority instances (Algorithm 1, line 8), which prompts the update of the best ensemble size (line 9). The algorithm then checks if the imbalance ratio has changed by some pre-defined factor d in line 11 (it is worth mentioning that the community members may decide on an appropriate number for this value), computing the number of new classifiers to be included in the ensemble (line 12). After that, the algorithm incorporates community feedback (line 15) to present relevant data about the change in the community's view in the training procedure. Then, s_t balanced datasets are created from data block D_t. Each balanced dataset B comprises non-overlapping data points from M_t, all data points from P_t, and all feedback data points from F_t (line 18). Next, the algorithm executes the training procedure for each of the

Algorithm 2 The Online Training procedure.

```
Input: Current data point ($p_t$), data point feedback ($f_t$), desired class distribution ($m$), sampling rate ($g$), max change in class distribution ($d$)
Output: Trained ensemble ($E$).
1:  Initialize ensemble of classifiers. $E \leftarrow \{NeuralNetworks\}$
2:  Initialize the last imbalance ratio change. $r \leftarrow 0$
3:  while data point is available do
4:      Pre-process $p_t$, with running statistical values.
5:      Update partial class distribution $h_t$.
6:      Update number of data points $n$.
7:      if $r > 0$ and $h_t \div r < 1 - d$ then
8:          Update desired distribution.
9:          Minority class increases by the ratio $h_t \div r$.
10:     end if
11:     Compute rate to draw from distribution. $q \leftarrow g \times m \div (h_t \div n)$.
12:     for Classifier $o \in E$ do
13:         Calculate resample ratio. $v \leftarrow poisson(q)$.
14:         Train $o$ with the oversample data point.
15:     end for
16:     if data point received feedback. $f_t = True$ then
17:         Oversample data point $p_t$ by duplicating.
18:         Train all classifiers in $E$ with $p_t$.
19:     end if
20: end while
```

The Online Training procedure.

s_t ML base classifiers with the balanced datasets in B_t (line 20). Finally, the current data block D_t is discarded, and t is incremented.

3.2 Online learning for tabular scenarios

Algorithm 2 describes the procedure to train the ensemble of classifiers in an online manner, which is built on top of the concepts described by Wang et al. [102] and Montiel et al. [65]. The first step is to create the ensemble of classifiers E (Algorithm 2, line 1). The number of base classifiers in E can be defined by the community, from expert knowledge, or through initial experiments. For each data point p_t that is made available (i.e., for each action in an online community), the algorithm pre-processes p_t using the running statistical values. We are interested in the mean, and the sum of squares since these are used to normalize the incoming data point.

Unlike the mini-batch approach, the training procedure is executed in online learning as soon as a single data point is made available. However, this characteristic leads to a different way of calculating statistical values for the pre-processing phase. In this case, the algorithm must compute running statistical values, updated at each time step and less exact than the values calculated using blocks of data [65]. The algorithm uses the following equations to compute these values:

$$\mu_t \leftarrow \mu_{t-1} + ((v_t - \mu_{t-1}) \div n_t) \tag{7}$$

where μ_t is the updated running mean at time t for each feature that describes an action, μ_{t-1} is the last running mean, v_t is the new feature value, and n_t is the number of data points encountered until the current time t. With the running mean, it is possible to calculate the running sum of squares ss_t:

$$ss_t \leftarrow ss_{t-1} + (v_t - \mu_{t-1}) \times (v_t - \mu_t) \tag{8}$$

Since it is impossible to know the data distribution for the complete dataset in online training, deciding which portions of the data will be used for training as interactions happen is fundamental. To tackle this, the algorithm checks for concept drift by calculating the change in the imbalance ratio r (Algorithm 2, line 7). If the difference is bigger than a defined threshold value, then the desired distribution m is updated, which works to emphasize the minority class instances.

After updating m, the algorithm calculates the rate at which to draw a random value (Algorithm 2, line 11) for the Poisson distribution. This value determines the sampling strategy (oversampling or undersampling). For each classifier $o \in E$, the algorithm uses the Poisson distribution to determine how many times to replicate a data point for training [102] (line 13). Thus, the larger the imbalance ratio, the larger the number of times that minority data points are used for training. Although we use the work in [102] to calculate the resampling rate, future work will investigate the effect of applying alternative strategies to calculate this value [30].

Lastly, suppose the data point receives feedback from the community (represented by $f_t = True$, line 16). In that case, the algorithm oversamples p_t to emphasize the provided information and trains all classifiers in the ensemble with p_t (Algorithm 2, line 18). Empirical experiments showed that oversampling presents a higher recall performance than the weighting scheme proposed by the other approaches.

3.3 Mini-batch learning for binary text scenarios

Previously, we examined two algorithms that address binary classification tasks involving tabular data. This section extends our framework to include text classification tasks for binary and multi-label scenarios. This capability enables our framework to adapt to online communities' diverse data structure requirements. As our primary focus is on PLMs (Sect. 2.3), the online learning approach is unfeasible due to the model's size, which affects the update of the network weights and the needed time to complete the fine-tuning process.

Like Algorithm 1, mini-batch for text tasks (Algorithm 3) builds data blocks to continuously update model parameters sequentially. However, one key difference between these approaches is that Algorithm 3 can handle imbalanced datasets more efficiently just by undersampling the majority class, not requiring the creation of an ensemble of classifiers. To achieve that, Algorithm 3 takes advantage of the PLMs' architecture, which can learn representations of texts based on previous training and incorporate classification heads to solve specific tasks [47, 55, 86].

The fine-tuning process of PLMs starts by pre-processing the available text data. The classification task at hand dictates the necessary steps for this process. For instance, in the

```
Algorithm 3 The Mini-Batch Fine-Tuning procedure of PLMs.
    Input: Current data block ($D_t$), set of majority instances ($M_t$), set of
minority instances ($P_t$), min imbalance ratio ($d$), and number of epochs ($e$)
    Output: Fine-tuned PLM ($L$).
1: while data block is available do
2:     Pre-process $D_t$, no past data is used.
3:     Compute the current imbalance ratio. $r_t \leftarrow |P_t| \div |M_t|$.
4:     if $r_t < d$ then
5:         Undersample majority class by the ratio $r_t \div d$.
6:     end if
7:     Fine-tune $L$ with $D_t$ for $e$ epochs.
8:     Obtain global relevance scores for violations.
9: end while
```

The Mini-Batch Fine-Tuning procedure of PLMs.

case of detecting hate speech, it may be beneficial to remove non-alphanumeric characters, as our model considers only the terms in a sentence to determine the violation. Thus, these characters may not be relevant in this context. Another step that may be necessary is correcting words that are commonly used to bypass automatic detection tools. For example, in cases where a community member manifests racism, they may employ alternative terms to refer to African Americans, such as, “nigga”, “n1gga”, and “nigger”. On the other hand, to detect whether a sentence is violating an expected writing style, removing these characters is detrimental to the model’s performance. Thus, it is necessary to implement task-specific pre-processing to ensure the efficacy of our framework. This becomes particularly important when our community contains small datasets, or the interactions happen in a low-resource language.

Following the pre-processing phase, Algorithm 3 calculates the imbalance ratio (line 3) to determine if undersampling is required (line 4). The algorithm then applies undersampling considering the established difference between the amount of majority and minority instances (line 5). The next step (line 7) is to fine-tune the PLM with the data block for a specified number of epochs. One of the main advantages of PLM is the simplicity with which we can execute the fine-tuning process. Hence, the complete training process is more straightforward than Algorithms 1 and 2 as it requires fewer steps to deploy a PLM to a new task domain.

Lastly, in line 8, as we update the PLM, it is possible to understand the terms usually associated with violation by calculating a global relevance score based on local interpretations. The global relevance score of a word (gr_i) is the sum of all local relevance scores, calculated using integrated gradients. In Eq. 9, k is the number of occurrences of word i in the dataset, IG(i_u, 1) is the calculated relevance score for the u^{th} occurrence of i, regarding its contribution to class 1, which indicates violating behavior. The framework must only change the second parameter to 0 to get the relevance scores for the regular class.

$$gr_i \leftarrow \sum_{u=1}^{k} \mathrm{IG}(i_u, 1) \tag{9}$$

Algorithm 4 The mini-batch fine-tuning procedure for multi-label PLMs.

```
Input: current set of violation instances ($I_t$), set violation classes ($V$), min instances per class ($c$), and number of epochs ($e$)
Output: Fine-tuned multi-label PLM ($L$).
1: while violation data block is available do
2:     for violation class $v \in V$ do
3:         Get instances of $v$. $N_t^v \leftarrow I_t \in v$.
4:         if $|N_t^v| < c$ then
5:             Oversample $N_t^v$ by duplication.
6:         end if
7:     end for
8:     Fine-tune $L$ with $I_t$ for $e$ epochs.
9:     Obtain global relevance scores for each class in $V$.
10: end while
```

The mini-batch fine-tuning procedure for multi-label PLMs.

3.4 Mini-batch learning for multi-label text scenarios

In addition to identifying violations (Algorithm 3), our proposed framework can also classify the specific class of violation present. It is worth noting that a single action may comprise multiple violation classes. Thus, the framework must be equipped to handle multi-label tasks.

For each violation class $v \in V$ defined by the community (Algorithm 4, line 2), the algorithm retrieves the number of instances that belong to that class and compares it to a fixed minimum number of instances (c) that each violation class must have (line 4). Suppose the data block does not contain the minimum number of class instances v. In that case, the algorithm oversamples by duplicating all instances belonging to v and uses them for fine-tuning. This step is crucial as we attempt to maintain a balanced data distribution between the different violations. Without this step, the model would be prone to bias towards classes with a larger number of instances, potentially hindering its ability to accurately identify violations in low-represented classes. One limitation of this approach is that we do not handle the emergence of new violation classes. In this case, we have one PLM with $|V|$ output nodes, where each output node represents a violation class. Future work shall investigate the emergence of new violation classes and their incorporation into PLMs.

Line 9 obtains the global relevance score using Eq. 10. The global relevance score (gr_i^v) is calculated for each $v \in V$ and is based on local interpretations. $\mathrm{IG}(i_u, v)$ computes the local relevance score of word i in relation to class v, k is the number of occurrences of i in the dataset, and u represents a specific instance of i. Calculating gr_i^v enables community members to understand the words commonly associated with each violation class. This is particularly relevant because a word may have a relatively low relevance score for one class yet a high relevance score for another.

$$gr_i^v \leftarrow \sum_{u=1}^{k} \mathrm{IG}(i_u, v) \tag{10}$$

Table 2 Examples of sentences classified as norm violation (vandalism) in the Wikipedia community and the specific class of hate speech

Sentence	Class of hate speech
...he was the mother fuckin dom...	Swear
...this is wiki not a forum for retards...	Insult and Ableism
...the big lipped,hairbraned,egotistical dirty nigger often defecated...	Racism
[INDIVIDUAL's NAME] also sucks dick for features	Sexual Harassment
...HES GAYYYYYYYYYYY AND HES A FREAKK...	LGBTQIA+ Attack
[INDIVIDUAL's NAME] was a super mega bitch and she kill the...	Misogyny

"[INDIVIDUAL's NAME]" is used to mask real people's names

4 Experiments

This section describes how we apply the incremental learning approaches to the use case of Wikipedia article edits. Here we consider data from Wikipedia in two scenarios (tabular and text tasks) as we envision our framework deployed in different classification contexts. This use case is relevant because Wikipedia is an open and collaborative community with norms to maintain and organize its content [76], including the requirement to use proper writing style, refrain from removing content, avoid editing wars, and not engage in hate speech. Given the diverse backgrounds of individuals interacting and contributing to Wikipedia, misunderstandings about what constitutes a norm violation might emerge. In this research, we focus on violations of the hate speech norm, as this represents a complex and particularly harmful violation within online interactions.[4] In this context, a norm violation is referred to as "vandalism".

This work explores a dataset consisting of two parts. First, Wikipedia uses Amazon Mechanical Turk (MTurk) to classify an article edit either as a violation or not [2], providing no further information on the nature of the violation. Second, we further annotate each violation instance with a violation class, focusing on hate speech violations.[5] To perform this annotation, we start by considering the labels from the MTurk process (violation or regular). Then, we specify additional hate speech classes for the violation edits with messages that convey attacks to individuals or groups. Usually, these attacks focus on characteristics of people, such as ethnicity, sexual orientation, and social class [73]. Table 2 presents examples of such behavior in Wikipedia. Freitas dos Santos et al. [34] provide a detailed taxonomy for this task, including information on the relationship between features and their representation of actions in the tabular scenario. Additionally, at this step, we manually correct the misspelled insulting words (based on the identified violation classes).

In the Wikipedia dataset, we identify six different classes of hate speech. A single edit can contain elements of one or more of these classes. As such, we build our framework to address the multi-label classification task. We only solve the multi-label classification task for text sentences, as the features present in the tabular data do not encode

[4] Future work shall focus on solving other kinds of violations.

[5] The hate speech annotation is executed by one of our authors, introducing our view on the meaning of norm violation. It should be noted that the primary goal of this work is to develop a framework capable of continuously updating its parameters and adequate itself to a specific view present in an online community (which also contains diverse perspectives and may vary depending on the community in question).

relevant information for classifying a violation with a specific hate speech class. Below, we present a list detailing each of these classes:

- Swear - it describes edits that contain foul language;
- Insult and Ableism - it considers edits that insult people in general and specifically people with disabilities [16];
- Sexual Harassment - with edits that contain sexual insinuations and harassment [15];
- Racism - discrimination targeting people from different ethnicities [48];
- LGBTQIA+ Attack - insults targeting people based on their sexual orientation and/or gender identity [39];
- Misogyny - attacks targeting women [37].

To evaluate the performance of our approaches, we design specific experiments for the different task scenarios. First, considering a domain in which the community has only a tabular dataset available, we separate the experiments into two phases:

- *Learn the meaning of norm violation with no concept drift:* In this case, the goal is to evaluate if the proposed algorithms can learn the meaning of norm violation. The data set contains 32.439 edits, with 2.394 vandalism edits (around 7%) and 30.045 regular edits (around 93%). The dataset is highly imbalanced. We use 10-fold cross-validation to evaluate the performance. Classification recall is the chosen metric.
- *Learn the meaning of norm violation with concept drift:* In this case, the aim is to evaluate whether the proposed algorithms can learn the meaning of norm violation in the presence of concept drift. To do that, we start by separating the complete dataset D into two subsets, I and F. I contains data used to initially train the ensemble, with 1.197 vandalism edits and 15.022 regular edits, and F contains data that incorporates the concept drift, with 1.197 vandalism edits and 15.023 regular edits. This separation is necessary because we aim to demonstrate the algorithms' ability to adapt incrementally to new concepts. Thus, we start by training the algorithms with the subset I. Only when the algorithms process all data points in I, do we start learning from the changing dataset F. In this experiment, we are particularly interested in adding concept drift by changing what edits are labeled vandalism (swap of the class label). Since we do not have real feedback from community members, we simulate it by changing the dataset as follows: using only the vandalism subset $V_F \in F$, we apply the K-Means clustering algorithm to generate subgroups that contain data points most similar between themselves [49]. From this process, we obtain four subgroups, $G = \{0{:}\ 618, 1{:}\ 442, 2{:}\ 117, 3{:}\ 20\}$. The idea of getting these groups with similar data points is to fulfill the assumption that the feedback is consistent since we are grouping similar edits. Thus, our interpretation of the results naturally comes from this consistency. For this experiment, we swap the class label from all data points $\in G_0$. Then, the class distribution changes, resulting in 15.641 regular edits and 579 vandalism edits. Consequently, the imbalanced ratio changes as well.

We build the ensemble using the Keras library [24]. Feedforward Neural Network (FNN) is the base classifier. To compare mini-batch and online learning, the FNN architecture is the same in both cases. Stochastic Gradient Descent (SGD), with a learning rate equal to 0.01, is the optimizer and the loss function is the Cross Entropy. The experiments are executed on a 2.6GHz Intel Core i7-9750 with 16GB of RAM.

It is necessary to set specific parameters for the learning algorithms. In mini-batch learning, the batch size is 512, and the number of epochs is 200. In online learning, the initial ensemble size is set to 12, the desired distribution is 50% for each class label (regular and vandalism), the sampling rate is equal to 1, and the change in distribution is 30%. These values are found empirically (similar to a hyperparameter search) and can affect the performance of the classifiers.

Unlike the experiment above, we are not investigating concept drift for the second scenario.[6] However, we include here a multi-label classification task:

- *Learn the meaning of norm violation (hate speech) - binary task:* In this experiment, we aim to evaluate the ability of our framework to deal with norm violation in a text classification task. This step is similar to the first experiment for the tabular classification scenario. The difference is that we are interested specifically in hate speech. Thus our dataset contains 30.684 edits, with 639 hate speech edits (around 2%) and 30.045 regular edits (around 98%). The dataset is highly imbalanced. We use 2x5-fold cross-validation for the experiments, which is necessary due to the text dataset size. Classification recall is the chosen metric.
- *Learn hate speech class - multi-label task:* Here, we aim to evaluate the performance of our framework to detect the specific hate speech class. Besides the imbalanced dataset for violation/regular edits, the hate speech classes are also imbalanced. Certain violation classes occur more often than others. In total, the violation dataset is composed of 36,47% (233) Sexual Harassment edits, 33,18% (212) Insult and Ableism, 19,72% (126) Swear, 17,06% (109) LGBTQIA+ Attack, 8,76% (56) Misogyny, and 5,01% (32) Racism, in a total of 639 violation edits. To guarantee that each fold of the validation process maintains the data distribution, we apply a stratification step on the multi-label dataset using the algorithm in [90]. 2x5-fold cross-validation is also used for this experiment. Classification recall for each class is the chosen metric.

To solve text-related tasks, our framework adopts PLMs (Sect. 2.3). Specifically, we employ RoBERTa and DistilBERT following the Hugging Face implementation [105], with a batch size of 1024 for the binary classification task and 256 for the multi-layer classification task. Adam is the optimization algorithm, and focal cross entropy is the loss function. Learning rate is 10^{-4}.

To optimize the performance of the PLMs, we implement additional parameters. Specifically, we set the maximum input length to 64 words and apply padding to edits that exceed this length. We base this decision on the observation that most instances in our dataset fell within this range, allowing us to save computational resources and accelerate the fine-tuning process. It is essential to highlight that, if required in other communities, our framework uses PLMs that can accommodate text sentences up to the limit of 512 words.

In our framework, we aim not only to classify a task as norm-violating behavior but also to provide community members with the reasons for such output. Our goal is to integrate diverse community members by leveraging their mutual understanding. To achieve this, we use Integrated Gradients (IG) to obtain the relevant words contributing to the violation classification. These words are then presented to the users, as depicted in the figures

[6] The reason is that we do not possess enough data for the hate speech detection case to run such experiments. Hence, for future work, we are investigating cross-community learning. The idea is to obtain a dataset with hate speech from a different domain and improve upon that with data from our specific environment.

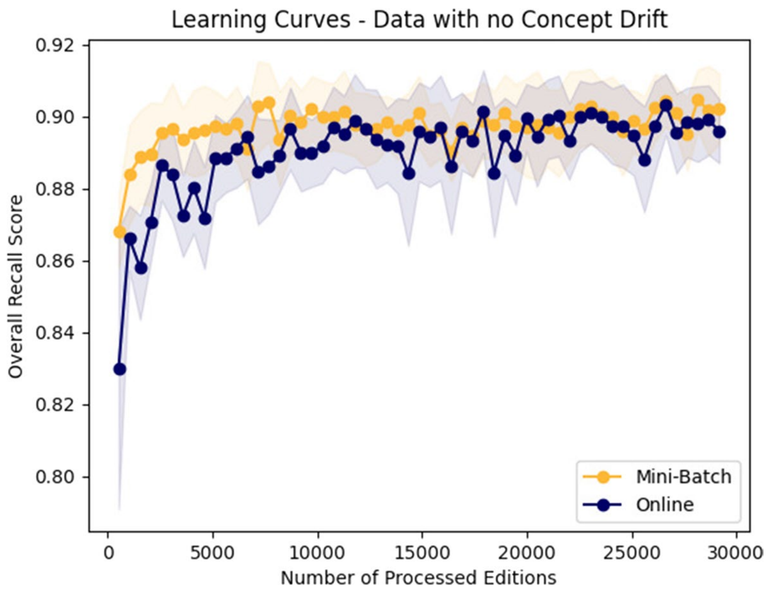


Fig. 4 Overall Recall for the Mini-Batch and Online cases with no concept drift

of Sect. 5 and "Appendix 2". Additionally, by providing access to this information, other community members can argue about the inner workings of our framework, supporting a future agreement process in which the community must collaboratively decide whether an action is indeed a violation.

The interpretation results for the binary case show which words contribute the most to the classification of text as being a violation or not (regular text). Each word in an edit can be relevant for violation classification, relevant for regular classification, or neutral. In contrast, the multi-label case allows each word to be relevant to 0, 1, or more classes. For instance, a single word may contribute to the classification of an edit as both racist and containing swear words. As we are interested in understanding the meaning of norm violations, the experiments focused only on interpretability data for these cases. Thus, we consider 639 edits (the complete violation dataset) for interpretability. Finally, we use the Transformers Interpret library for our experiments.[7]

5 Results and discussion

This section presents the results of using an ensemble of FNN to address tabular-related tasks and PLMs to address text-related tasks, considering the context of Wikipedia article edits. In this domain, the community defines norm-violating behavior as "vandalism". Additionally, we show words of an edit that contribute to the PLMs' outputs in binary and multi-label settings.

[7] pypi.org/project/transformers-interpret/

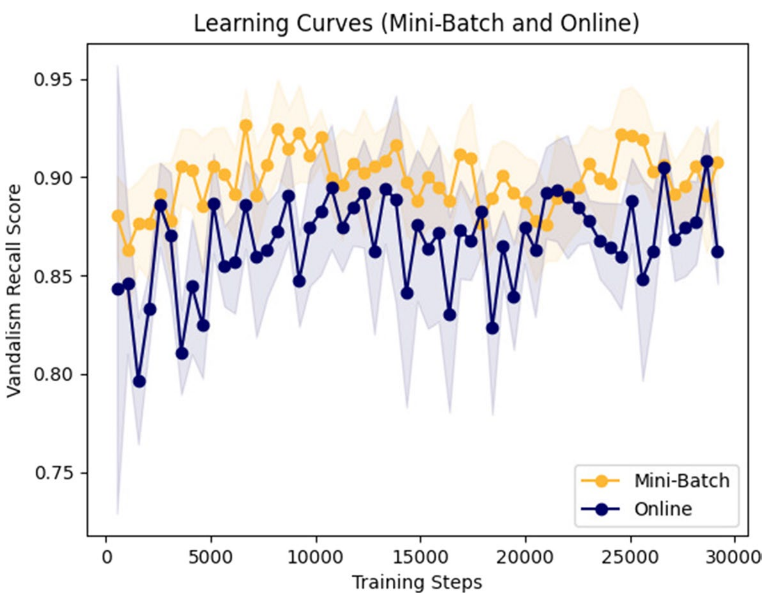


Fig. 5 Vandalism Recall for the Mini-Batch and Online cases with no concept drift

Table 3 Summary of mini-batch and online learning performance results applied to the tabular Wikipedia article edits dataset

Dataset	Method	Recall±Std	Regular recall±Std	Vandalism recall±Std
Original	Mini-Batch	**0.9023**±0.0097	0.8971±0.0091	**0.9075**±0.0219
	Online	0.8959±0.0088	**0.9297**±0.0094	0.8622±0.0164
Concept Drift	Mini-Batch	**0.8679**±0.0280	0.87085±0.0120	**0.8651**±0.0597
	Online	0.8408±0.0259	**0.9025**±0.0319	0.7792±0.0674
Re-label	Mini-Batch	0.8708±0.0120	X	X
	Online	**0.9277**±0.0284	X	X

The highest performance values are given in bold

Three settings are considered: (1) dataset with no concept drift (Original); (2) dataset with concept drift, swap of the class label; and (3) dataset with only the data that suffered the change (Re-Label)

5.1 Tabular scenario

5.1.1 Experiment 1—No concept drift

Figure 4 and Table 3 describe the overall recall score for the algorithms when applied to the first experiment (no concept drift). The learning curves for both approaches are similar, and the Wilcoxon Signed-Rank Test (Table 5) attests to this similarity. The null hypothesis is not rejected. Thus, there is no statistically significant difference between mini-batch and online learning for overall recall. Although similar in this case, the algorithms differ when explicitly dealing with vandalism instances. Table 4 presents how

Table 4 Summarized comparison between the training time of mini-batch and online learning

Measurement	Mini-batch±Std	Online±Std
Training time (s)	**4.0947**±0.7032	10.4159±0.9021

The best training time is given in bold

The number of processed edits is 512 (batch size)

Table 5 Summarized comparison between the recall performance of mini-batch and online learning

Dataset	P-values		
	Overall	Regular	Vandalism
Original	0.2754	0.0039	0.0058
Concept drift	0.1308	0.0273	0.0371
Re-label	0.0019	X	X

The paired Wilcoxon Signed-Rank Test is used to obtain the *P*-values. The learning approaches are compared after the model processes 512 edits. We use the performance results obtained from the cross-validation procedure. The null hypothesis is that the samples were drawn from the same distribution. Critical value $\alpha = 0.05$

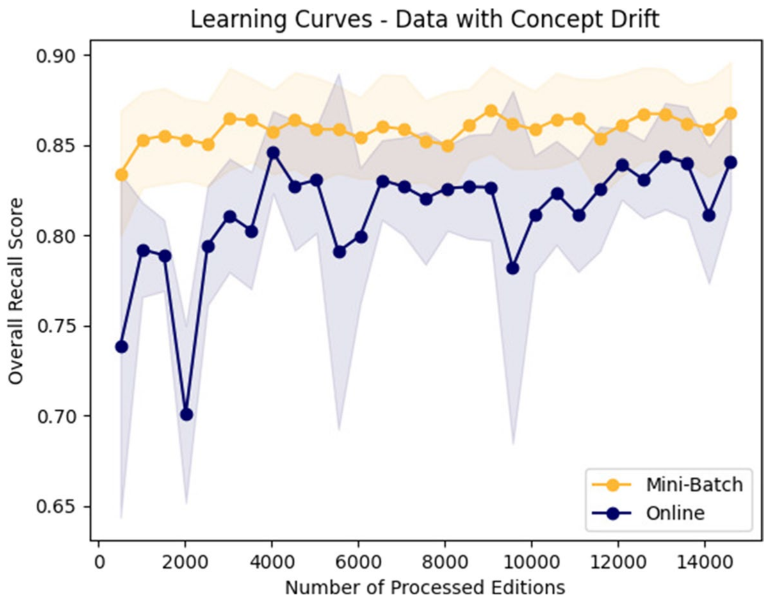


Fig. 6 Overall Recall for the Mini-Batch and Online cases in the presence of concept drift

mini-batch learning is faster, requiring less time to complete the training process since it executes the calculations on a batch of data instead of repeating this process for each data point individually.

Considering the data in Table 3 and the learning curves in Fig. 5, we can infer that the mini-batch algorithm outperforms the online algorithm in correctly classifying

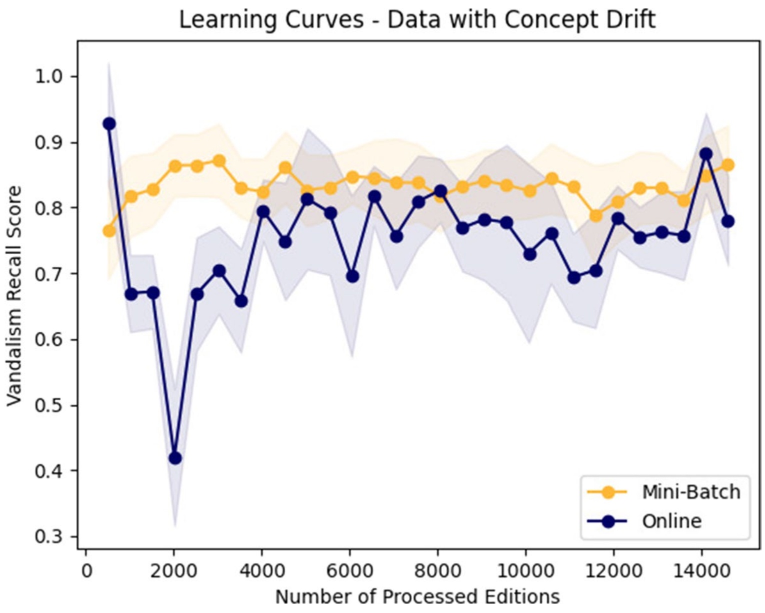


Fig. 7 Vandalism Recall for the Mini-Batch and Online cases in the presence of concept drift

vandalism edits. Additionally, the instability properties in the online case are affected by the training approach (since it considers only one point at a time) and the resampling strategy used [54, 102, 103].

5.1.2 Experiment 2—presence of concept drift

The results of the second experiment, depicted in Fig. 6 and Table 3, reveal the overall recall for the scenario involving concept drift. The mini-batch performs significantly better during most parts of training (until around 12.000 processed instances). The reason is that the introduction of concept drift causes a higher variation and instability in the online learning algorithm, leading to a slower improvement in performance and a need to process additional data points to stabilize the learning process. However, towards the end of the training procedure, both methods have similar overall performance, with no significant difference (Table 5).

Since we are working with an imbalanced dataset, comparing the results of overall and vandalism cases is essential, as failure to do so may result in misleading conclusions. In such a context, the online learning approach prioritizes the classification of the majority class, leading to an overestimation of performance through increased overall values. Figure 7 presents the learning curve specifically for vandalism classification, in which mini-batch significantly outperforms the online approach (Table 5). As in other cases, online learning is more unstable, suffering from a significant drop in performance as we introduce concept drift.

Figure 8 presents the recall specifically for the data that suffered the swap of the class label (which we will refer to as the Re-label dataset in Table 3). When we incorporate the simulated feedback, the framework's performance decreases due to introducing new information. However, as more data becomes available and the framework incrementally

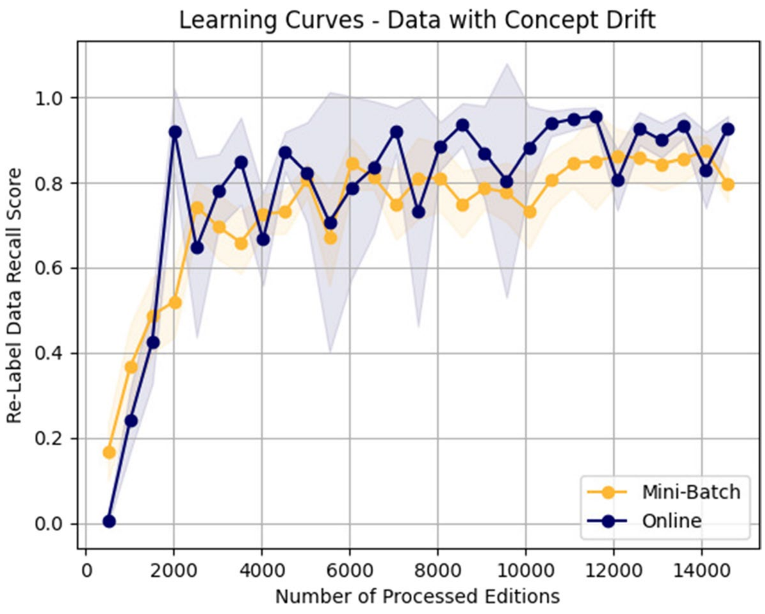


Fig. 8 Re-Label Recall for the Mini-Batch and Online cases, vandalism edits re-labeled to regular edits

trains the ML models, the ensemble adapts to the new community view by learning that specific article edits should no longer be classified as vandalism. Table 5 shows that the online learning algorithm performs significantly better in this case (however, it still presents instability properties). The bias towards the majority class impacts the performance of the online algorithm since we increase the imbalance ratio by changing the classification label from vandalism to regular behavior.

To summarize, Table 3 presents performance information of each approach in the considered datasets, and Table 4 presents the time required for the training procedure, showing that mini-batch learning is more efficient than online learning. Table 5 describes the results of the Wilcoxon Signed-Rank Test, which compares the performance of the proposed approaches. The null hypothesis is that the samples were drawn from the same distribution, and the critical value $\alpha = 0.05$. Results show that the mini-batch approach is more suitable for classifying vandalism edits, offering stable performance, and adapting quickly to concept drift. In comparison, the online approach presents a bias toward the majority class and, consequently, in our concept drift case, a bias toward the changed data. Besides, this approach significantly drops in performance when classifying vandalism edits. Here, we note the need to investigate further and explore the effects of different imbalance strategies combined with the incorporation of community feedback on the algorithm performance since the online approach can learn the new concept, but at the cost of the performance in the minority class.

Finally, it is possible to conclude that both approaches are suitable for learning the meaning of norm violation in the context of an online community for the tabular scenario. Mini-batch offers more stability, better performance at vandalism detection, and faster training since it needs to process a smaller number of instances to solve a task. On the other hand, online learning offers the flexibility of updating the model as soon as

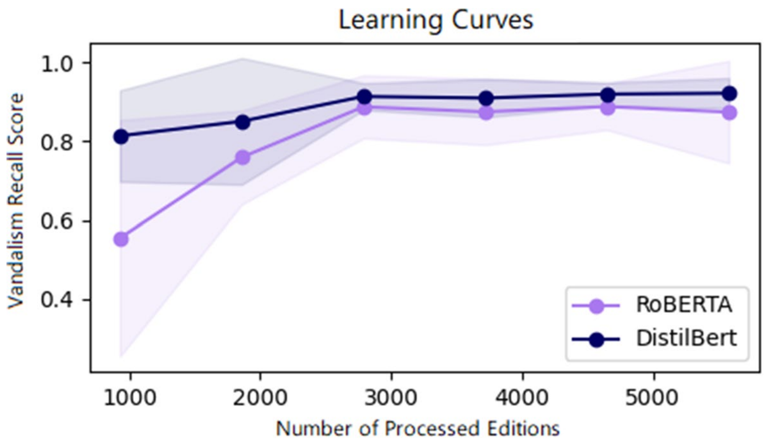


Fig. 9 Vandalism Recall for RoBERTa and DistilBert

data is made available, with no need to maintain and create data blocks while keeping an acceptable classification performance. Thus, the choice of approach must consider the community's requirements.

5.2 Text scenario

Here we present the results related to the application of RoBERTa and DistilBERT to detect vandalism in Wikipedia article edits. Additionally, we present the interpretability results for the binary and multi-label classification tasks.

5.2.1 Experiment 1—binary classification

Figure 9 shows the graph that describes the recall score (detailed in Table 6) for RoBERTa and DistilBERT when applied to the vandalism classification task. According to the Wilcoxon Signed-Rank Test in Table 8, the results show no significant difference between the two models. However, it is worth noting that RoBERTa presents a high standard deviation, which may be attributed to the small size of the dataset and a large number of trainable parameters (125 M) in the model. This behavior highlights how the presentation of data (different runs of the 2x5-folder cross-validation) affects the fine-tuning process and RoBERTa's performance. In contrast, DistilBERT (which has approximately 66 M trainable parameters) presents a more stable performance across the different executions of the experiments, dealing with a less complex language model architecture that is especially useful for our small dataset settings.

Figure 10 and Table 6 show the time required to fine-tune RoBERTa and DistilBERT. We can see a significant difference between the models, with DistilBERT requiring less time to complete the fine-tuning process. This superiority is attested by the Wilcoxon Signed-Rank Test, which yields a p-value of 0.0019, indicating a statistically significant difference at level $\alpha = 0.05$. Like the performance case analyzed above, the PLMs' size also interferes with training time. DistilBERT is smaller, with fewer parameters. Thus, it

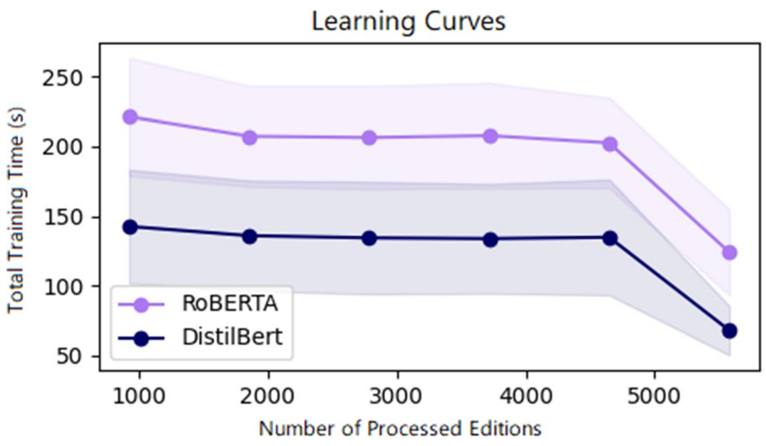


Fig. 10 Training time for RoBERTa and DistilBERT to classify vandalism. Binary task

Table 6 Summary of the performance results of RoBERTa and DistilBERT applied to the Wikipedia article edits dataset, binary task

Measurement	RoBERTa±Std.	DistilBERT±Std.
Vandalism recall	0.8741±0.1294	**0.9221**±0.0380
Regular recall	0.9906±0.0073	**0.9946**±0.0026
Training time (s)	221.22±42.300	**142.54**±40.740

The highest performance values and best training time are given in bold

We consider the task of classifying an edit as regular or vandalism behavior. We also present the total training time in seconds to process 1024 edits (batch size)

Table 7 Summary of the performance results of RoBERTa and DistilBERT applied to the Wikipedia article edits dataset in the multi-label case

Violation	RoBERTa±Std.	DistilBERT±Std.
Swear	0.7475±0.1140	**0.8180**±0.1047
Insult and ableism	**0.7802**±0.1172	0.7553±0.0886
Sexual harassment	**0.8367**±0.0662	0.8131±0.0759
Racism	0.6285±0.2054	**0.7523**±0.1594
LGBTQIA+ Attack	**0.8854**±0.0580	0.8670±0.0797
Misogyny	**0.7242**±0.1271	0.5811±0.1838
Training time (s)	294.50±30.134	**136.97**±10.922

The highest performance values and best training time are given in bold

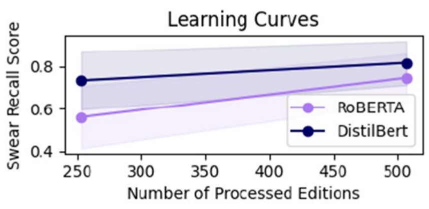
Here we consider the task of classifying a vandalism edit specifically to the class or classes of interest. We also present the total training time in seconds to process 256 edits (batch size)

takes less time to complete the whole process. The standard deviation of the results reflects the limited computational resources available for fine-tuning these models.

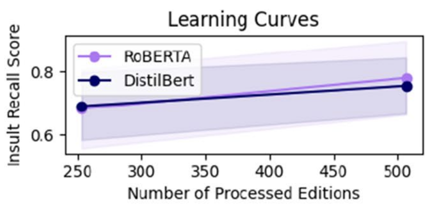
Table 8 Summarized comparison between the recall performance of RoBERTa and DistilBERT

Dataset	*P*-values	
	Regular	Violation
Complete	0.1309	0.6250
Swear	X	0.1134
Insult and ableism	X	0.4316
Sexual harassment	X	0.3571
Racism	X	0.0632
LGBTQIA+ attack	X	0.4055
Misogyny	X	0.0407

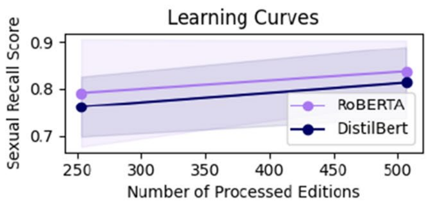
The Wilcoxon Signed-Rank Test is used to obtain the *P*-values. The null hypothesis is that the samples were drawn from the same distribution. Critical value $\alpha = 0.05$



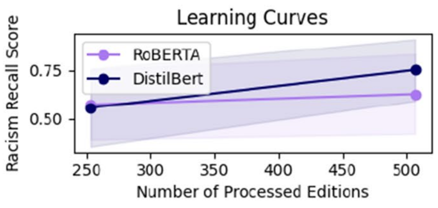
(a) Swear Recall score for RoBERTa and DistilBERT.



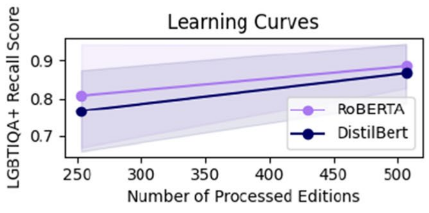
(b) Insult and Ableism Recall score for RoBERTa and DistilBERT.



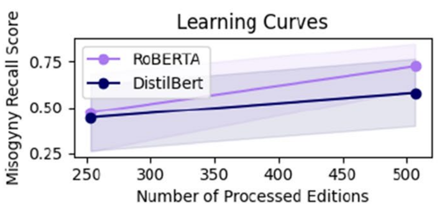
(c) Sexual Harassment Recall score for RoBERTa and DistilBERT.



(d) Racism Recall score for RoBERTa and DistilBERT.



(e) LGBTQIA+ Attack Recall score for RoBERTa and DistilBERT.



(f) Misogyny Recall score for RoBERTa and DistilBERT.

Fig. 11 Recall scores for the violation classes: Swear, Insult and Ableism, Sexual Harassment, Racism, LGBTQIA+ Attack, and Misogyny

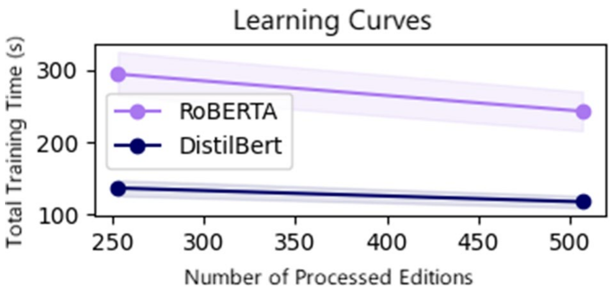
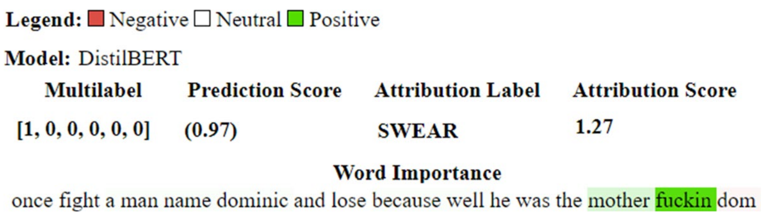


Fig. 12 Training time for RoBERTa and DistilBERT to classify the violation classes. Multi-label task


Legend: 🟥 Negative ☐ Neutral 🟩 Positive

Model: DistilBERT

Multilabel	Prediction Score	Attribution Label	Attribution Score
[1, 0, 0, 0, 0, 0]	(0.97)	SWEAR	1.27

Word Importance

once fight a man name dominic and lose because well he was the mother fuckin dom

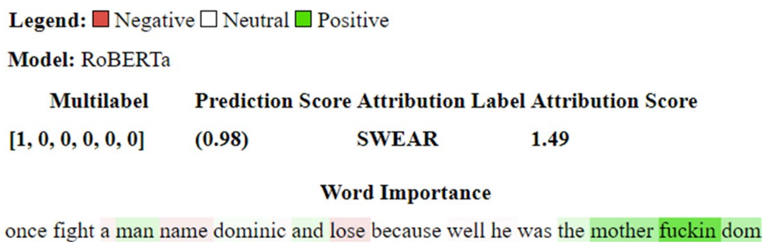
Fig. 13 The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is SWEAR. The relevance score is calculated using Integrated Gradient (Sect. 2.4)

5.2.2 Experiment 2—multi-label classification of vandalism

This section presents the evaluation of RoBERTa and DistilBERT applied to the multi-label classification task. We aim to categorize vandalism considering the six classes mapped for hate speech in Wikipedia, e.g., Swear, Insult and Ableism, Sexual Harassment, Racism, LGBTQIA+ Attack, and Misogyny. In the investigated use case, hateful content can attack different individuals and groups at the same time. For instance, in a single sentence, a community member can utter insults based on a person's ethnicity and sexual orientation. Thus, our proposed framework must be able to identify when these violations occur simultaneously.

Figure 11 presents the recall scores (detailed in Table 7) for each class in the context of our use case, which involves handling only vandalism data. As each class consists of a small number of edits, our approaches exhibit a higher variation in the recall scores for the 2x5-fold cross-validation experiments. Concerning performance values, the learning curves for both PLMs are similar, attested by the Wilcoxon Signed-Rank Test (Table 8). The only significant difference is the Misogyny class, in which RoBERTa outperforms DistilBERT. This class occurs in only 8,76% of the violation instances and presents the lowest performance score for both models, especially for DistilBERT. To address this issue, future work shall focus on cross-community learning to enhance the model's performance by leveraging the fine-tuning process with data from different communities.

Finally, Fig. 12 shows the training time needed for the multi-label task. Similar to the binary case, the DistilBERT model has a significantly faster fine-tuning process, as attested by the Wilcoxon Signed-Rank Test with a p-value of 0.0019 (below the critical value $\alpha = 0.05$). We use a batch size of 256 vandalism edits for the multi-label, trained over three epochs. On a smaller scale, the same behavior regarding the spread in training time, as seen for the binary case, can also be observed here.



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

Model: RoBERTa

Multilabel	Prediction Score	Attribution Label	Attribution Score
[1, 0, 0, 0, 0, 0]	(0.98)	SWEAR	1.49

Word Importance

once fight a man name dominic and lose because well he was the mother fuckin dom

Fig. 14 The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is SWEAR. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

5.2.3 Interpretability—binary and multi-label cases

This experiment investigates which words of an edit affect the output of the PLMs. For that, we present three different pieces of information. One describes the relevant words for a specific vandalism edit, as depicted in Figs. 13 and 14. Our framework calculates the relevant values using the Integrated Gradients (IG) algorithm (Sect. 2.4). The second describes a summarization of words usually associated with a specific (Swear) violation class and their frequency in our complete training dataset, as depicted in Figs. 15 and 16.[8] Lastly, Figs. 17 and 18 present the summarization of words usually associated with vandalism behavior in general. The sum of scores considers the local relevance calculated using IG. With this, we aim to give an overall view of the meaning of hate speech in our domain.[9]

To describe local interpretation, we analyze Figs. 13 and 14 for DistilBERT and RoBERTa, respectively. The vandalism class considered here is Swear.[10] For local interpretations, the stronger the green shade, the higher the highlighted word's relevance score. On the other hand, the stronger the shade of red, the more significant the influence of the highlighted word on decreasing the vandalism confidence (classification as non-Swear).

One crucial aspect is that the relevance of certain words may vary depending on the model. Let us consider the word "man" in Figs. 13 and 14. For RoBERTa, it is relevant to the model's classification. However, for DistilBERT, this word has no importance since it contains a neutral score. There are two main reasons for this variation. First, while RoBERTa prioritizes performance accuracy, DistilBERT was built to be smaller, faster, and cheaper. Hence, their architectures differ, and individual words affect the classification results differently. Second, they employ different tokenization processes and vocabularies, influencing how each PLM encodes words in the input layer. For instance, in DistilBERT's tokenization process, the word "nerd" is split into two "ne" and "rd". In contrast, RoBERTa's tokenization handles the complete word with no modification. This difference is especially critical for our hate speech use case since these PLMs do not initially map most terms associated with this behavior.

Besides local interpretations, it is also interesting to describe the summary of words related to specific hate speech classes. As discussed earlier, each edit may contain more than one vandalism class (people may express hatred towards various groups or individuals). Therefore, it is essential to understand the words associated with each hate speech

[8] "Appendix 1" presents the relevance score for all the other vandalism classes.

[9] To clarify, the sum of scores is not a global interpretation of our model but rather a summary of local interpretations.

[10] "Appendix 2" presents local interpretability examples for all other vandalism cases.

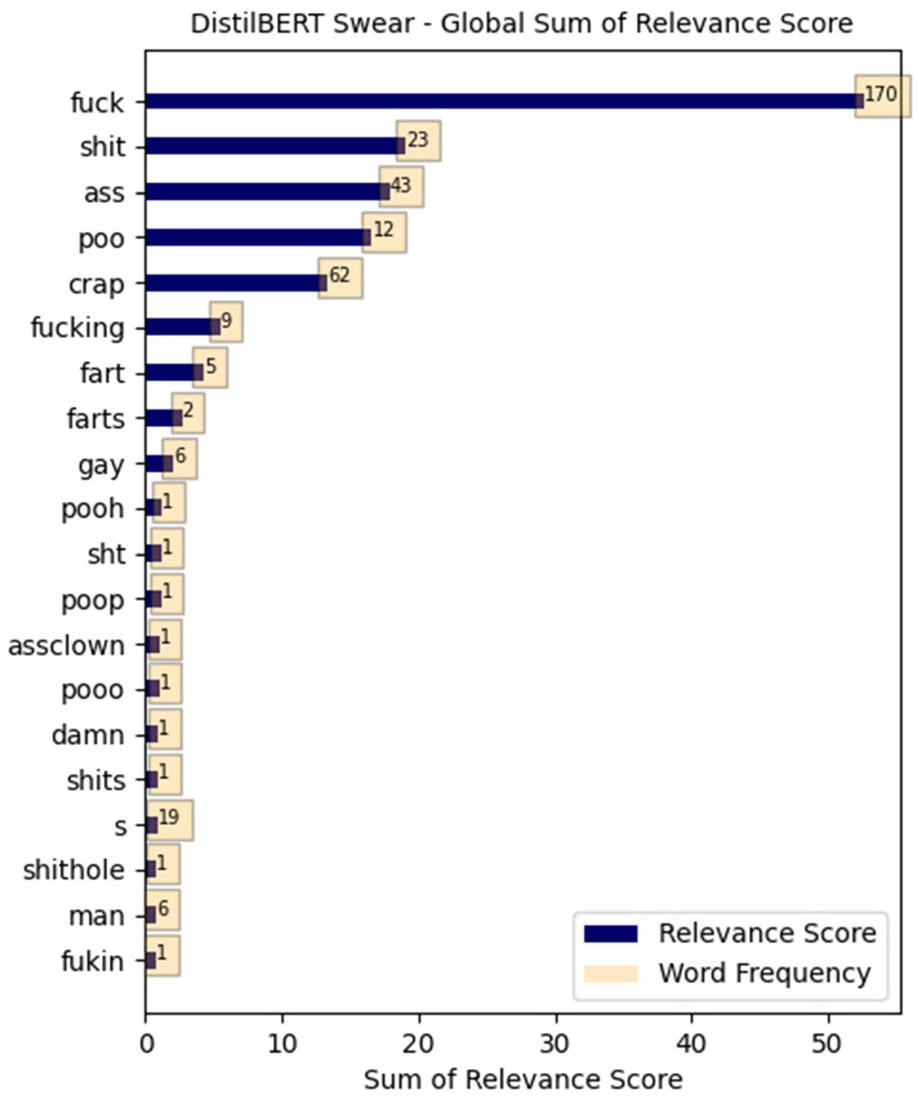


Fig. 15 The global sum of relevance score for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is Swear. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (Sect. 2.4)

class. Figures 15 and 16 present the terms with the highest sum of relevance score for the Swear class. From the top 20 relevant words, DistilBERT and RoBERTa disagree on six. Additionally, some relevant words do not align with our understanding of the Swear class, such as “307” and “s”. Identifying these words demonstrates another advantage of incorporating interpretability. With this information, community members have a visualization tool to identify when a model follows faulty logic since it considers influential words that are not coherent with their understanding.

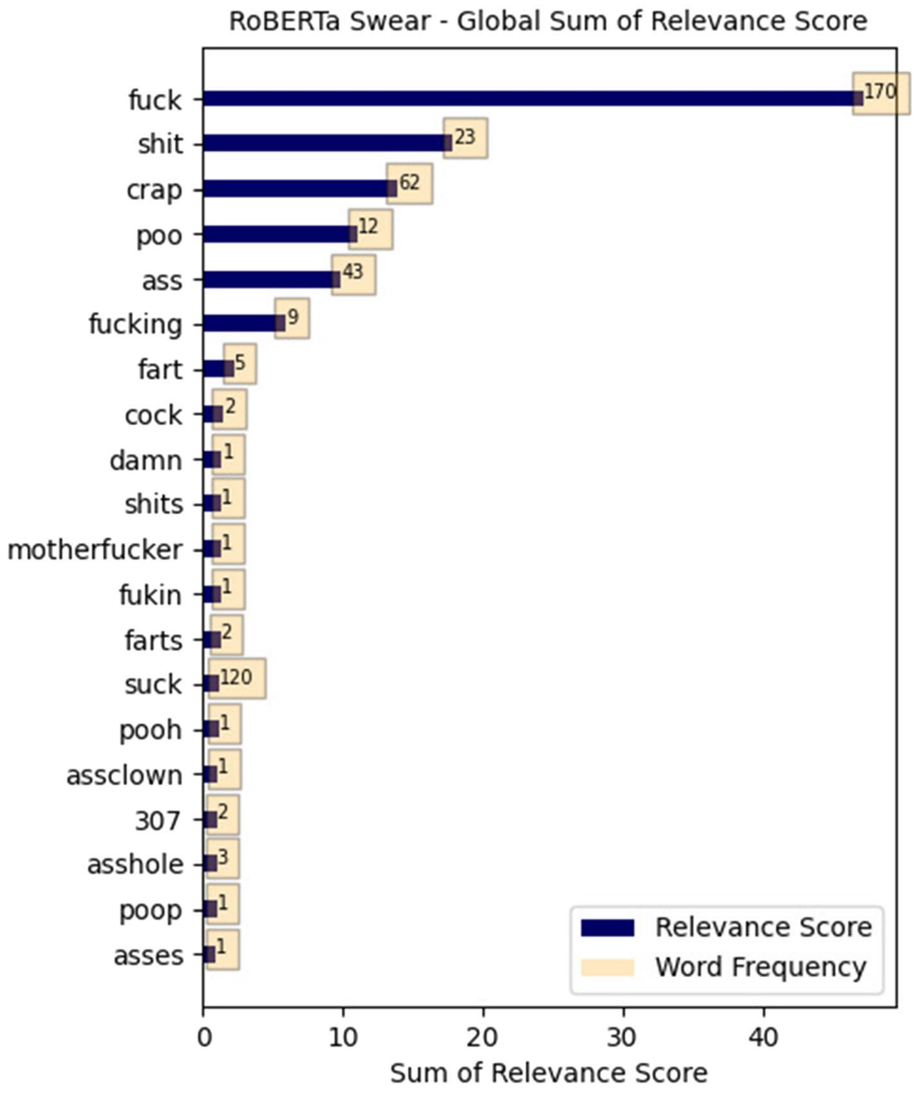


Fig. 16 The global sum of relevance score for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is Swear. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (Sect. 2.4)

The last part of our interpretation is in Figs. 17 and 18. These graphs summarize the words usually associated with vandalism behavior for the binary classification task. As expected for hate speech in general, the words in the community dataset are insulting and related to cyberbullying. Both models consider similar words relevant for detecting vandalism. However, they disagree on the assessment of six of them. This discrepancy

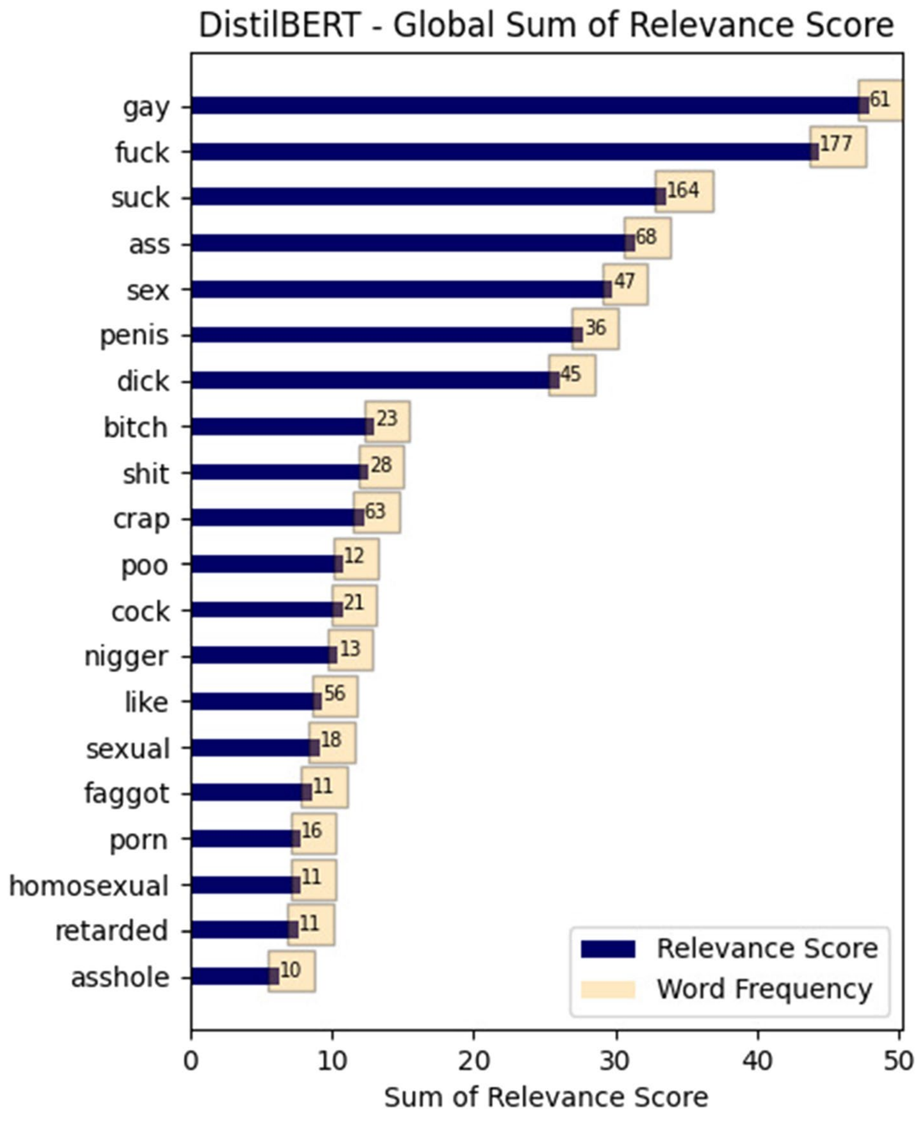


Fig. 17 The global sum of relevance score for the top 20 words considering the DistilBERT model. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

in scores (higher or lower) does not necessarily indicate a lack of relevance. Instead, it reflects the differences in the internal mechanisms (different numbers of transformer layers and embeddings) of the PLMs. For instance, in DistilBERT, the word with the highest global sum of relevance scores is “gay”, while RoBERTa presents “fuck” with the highest scores. These findings highlight how the two PLMs solve this task.

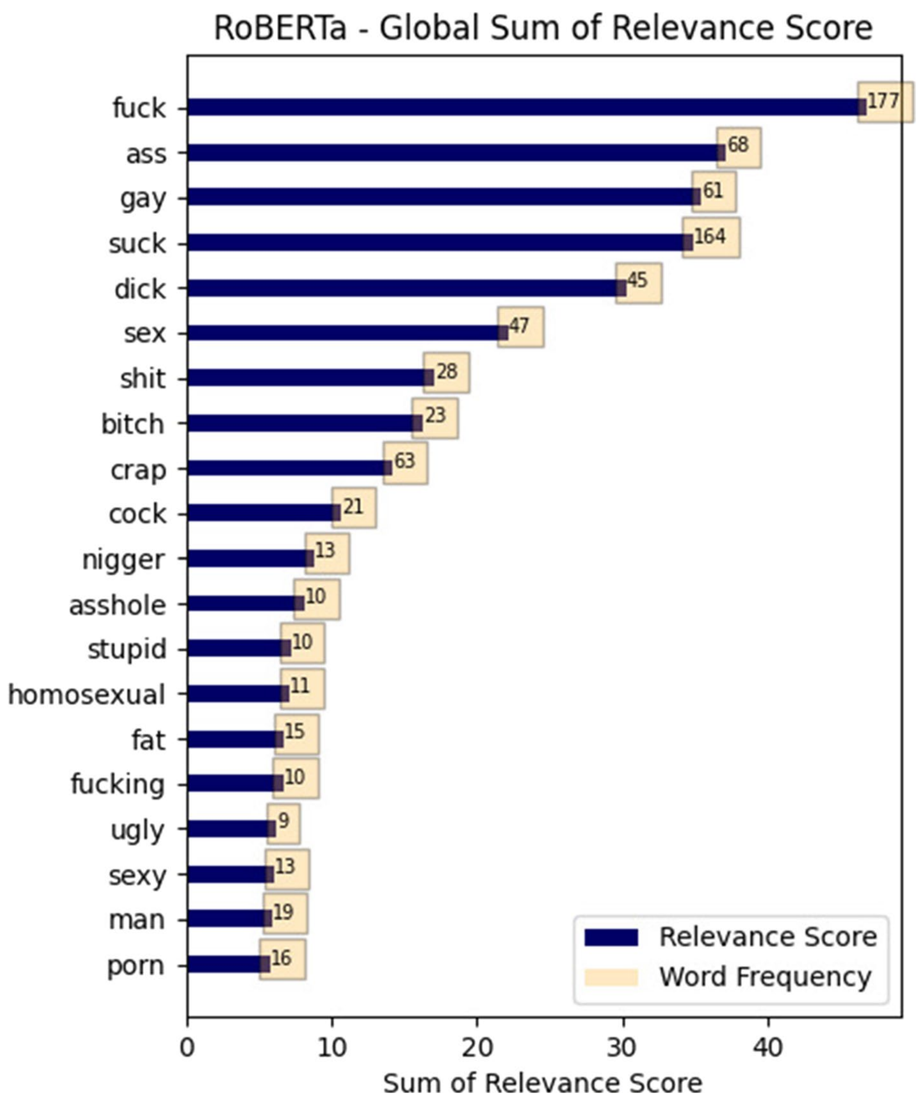


Fig. 18 The global sum of relevance score for the top 20 words considering the RoBERTa model. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

6 Literature review

This section presents related work to the research reported in this paper. Specifically, we cite literature focusing on detecting detrimental behavior in online communities using Machine Learning (ML). The idea is to present different approaches that deal with this issue in other communities, highlighting the importance of research in the field. For example, Risch and Krestel [81] describe several Deep Learning (DL) approaches to deal with toxic comments. In [6], the authors use Natural Language Processing (NLP), ML, and feature representation techniques as the basis to build a solution that handles hate speech. Chandrika et al. [20] report and compare the application of several ML algorithms to detecting abusive comments online, with Neural Network (NN) presenting better results than other approaches. We also focus on works that deal with incremental learning in an environment with concept drift and imbalanced dataset [35, 56, 80]. Gama et al. [35], and Lu et al. [56] present surveys on concept drift, with different applications to solve this challenge in several domains, while Ren et al. [6] build an ensemble to deal with imbalanced dataset and concept drift using a sampling strategy that considers previously seen data to enhance the current minority set. Lastly, we present works that handle interpretability in a text classification context, demonstrating how this technique has been used to improve user interaction in such domains [14, 77, 100].

6.1 Norm violation detection

Previous works have also focused on the Wikipedia online community to detect norm violations. Anand and Eswari [8] apply Deep Learning to classify a comment as abusive or not based on a dataset from the talk page edit. Freitas dos Santos et al. [34] use the Logistic Model Tree to learn the meaning of norm violation. Additionally, they provide a taxonomy that describes the relationship between the features of an action. However, these differ from our research because they do not cope with concept drift and do not incorporate community feedback to update their models.

Cheriyan et al. [22] present a work that explores ML to detect norm violations in the Stack Overflow (SO) community. As in our work, Cheriyan et al. [22] use specific data about the context of the SO community to train the ML models. In this case, instead of article edits, Cheriyan et al. [22] analyze comments posted on the site. The presence of hate speech and abusive language defines the violation. The main difference is our focus on applying an incremental learning approach to continuously update the ML models, while Cheriyan et al. [22] focus on using a recommendation system to detect and recommend alternatives to the community members' posts. Continuing their work, in [23], the authors expand their solution to incorporate the detection of offensive language in four different Software Engineering (SE) communities. They consider three violation classes, personal, racial, and swearing. Other ML techniques were also evaluated, ranging from Random Forest and Support Vector Machines to BERT-based language models, which present the best classification performance. Unlike our work, Cheriyan et al. [22, 23] use TF-IDF Vectorizer to obtain features, while the community provides our attributes [34].

Using an approach that applies ensemble learning to help in the task of comment moderation in Reddit, Chandrasekaran et al. [19] created a system that uses the concept of cross-community learning to train different ML models on additional data (provided by several communities), namely the Crossmod approach. The goal is to detect a violation in a specific community by understanding how other communities would classify a particular comment. Unlike our proposal, which uses ensemble learning to create ML models with balanced portions of the dataset, Crossmod collects information from different communities to train the ensemble of classifiers. Future work shall investigate incorporating data from different communities to leverage norm violation detection in our use case.

Different researchers use BERT-based language models for violation detection in text classification tasks. The work by [59] creates an ensemble using transformers-based models, SVM, and feature information. Since their application considers the Dutch language, BERTje was used, which is explicitly trained on top of Dutch text sentences. The authors used data from comments on Facebook and Twitter to evaluate their approach. While Markov et al. [59] mix text and features to solve violation classification tasks, our framework tackles them separately, aiming to avoid the need to have a featurization process when a community provides text data. For instance, we can tackle hate speech detection without defining a set of attributes that encodes a text sentence. Thus our system does not require the community members to create these attributes. Intending to detect aggression and misogyny, Samghabadi et al. [85] use BERT in a multi-task setting. Their solution first classifies an action as not aggressive, covertly aggressive, or overtly aggressive. Then, they discover the target of the violation, focusing on the gender of a person or group. Their work achieves state-of-the-art performance.

Muslim and Purwarianti [70] investigate the use of offensive language in social media. They employ a combination of ensemble and cost-sensitive learning to enhance the performance of BERT for this task, divided into three subgroups: a) offensive language identification; b) automatic categorization of offense types; and c) offense target identification. The authors focus on building an ensemble of BERT models to address the issue of high variance in small datasets. Similar to other studies on detrimental behavior, the dataset is also imbalanced. In contrast to Muslim and Purwarianti [70], which uses cost-sensitive learning to evaluate the costs of mistakes made by the model, we employ binary focal loss to assess errors in the calculation of the loss function during the training process.

In addition to the comparisons above, we note the potential of our work to contribute to another line of research, which focuses on continuously revising norms as agents interact [18, 27, 66]. For instance, Dell'Anna et al. [27] propose the Data-Driven Norm Revision (DDNR) approach that relies on previously obtained knowledge of whether individual actions violate a norm or not. DDNR requires access to labeled data acquired from automated (or manual) classification processes, which is precisely the type of information our approach provides. In this context, our framework complements DDNR's strategy by enabling the identification of multiple classes of norm violations occurring simultaneously and handling text-based scenarios.

6.2 Incremental learning

Regarding the use of incremental learning in a setting with a class distribution that is highly imbalanced, the work by Lebichot et al. [50] builds a solution capable of detecting credit/debit card frauds. Like our use case, these transactions have a sequential nature, are highly imbalanced, and present concept drift. The proposed approach in [50] reports better results than the traditional offline learning approaches. To enhance incremental learning, Lebichot et al. [50] use ensemble learning to reduce variance and improve stability. At the same time, transfer learning deals with information learned in a different task. One difference in our approach is that we use an active process to detect concept drift, better suited to deal with major changes in time. Lebichot et al. [50], on the other hand, apply a passive strategy to concept drift since, in their domain, several concept drifts happen daily. Another major difference is the way to deal with an imbalanced dataset. While we use an ensemble, their work uses parameter tuning of a dense neural network model. In this case, the models that compose the ensemble are independently trained. The final output is the average of the probability scores.

In their work, Zeng et al. [54] present an incremental learning approach that emphasizes misclassified instances in the update procedure of the models that compose the ensemble (DUE). Another interesting characteristic of DUE is that it keeps a limited number of classifiers in the ensemble to ensure efficiency. Like our work, DUE uses an ensemble to handle data imbalance without needing to access past data. The oversampling technique used in [54] is the SMOTE, while we oversample by duplication. A key distinction between our approaches is the inclusion of a feedback component that uses data provided by the community to emphasize instances with a swap of class labels, i.e., we duplicate edits that receive feedback from community members, which works to update our framework on a new community view.

Zhang et al. [111] introduce an ensemble framework to handle concept drift in an imbalanced dataset context, the Resample-based Ensemble Framework for Drifting Imbalance Stream (RE-DI). This approach employs a resampling buffer to keep instances of the minority class, enabling the framework to handle the class distribution over time. Additionally, ensemble members that perform poorly in the minority class receive less weight. RE-DI incorporates a long-term static classifier to handle gradual changes and a set of dynamic classifiers to address sudden concept drift, which only considers recently received data. The framework dynamically creates these classifiers using a block-based method, chosen due to their ability to be updated incrementally and their strong initialization power. The goal is for these dynamic classifiers to learn the most recent concepts by the end of the training process. While RE-DI employs a buffer (using past information). We oversample to emphasize the minority class and undersample to decrease the influence of the majority class.

6.3 Interpretability

Different approaches to handling interpretability exist, Atanasova et al. [11] present a comparison of interpretability methods applied to several ML models. Besides, competitions are also common in the field, providing interesting solutions to the problem [28, 51, 84].

In their work, Xiang et al. [106] propose an approach to enhance the interpretability of PLMs. Unlike our work, the authors compute the relevance of each word's contribution to the output of a text and use max pooling to aggregate these values to determine the overall relevance of an entire sentence. To evaluate the effectiveness of this approach, Xiang et al. [106] conducted a user experiment, discovering that the explanations generated by their method outperform those produced by inherently interpretable models (e.g., Logistic Regression). Future work shall evaluate the differences between IG and the proposal in [106], focusing on analyzing how the understanding of norm violation differs depending on the interpretability algorithm.

Interpretability is also relevant in the health domain. Novikova and Shkaruta [74] use BERT to detect depression marks in text. While Wawer et al. [87] present an approach to detect objective markers of schizophrenia, showing parts of the text that are usually associated with this disorder. They used a perturbation method (LIME) to explain the output of a PLM, namely ElMo. In their investigation, the goal is to identify patients and healthy individuals. The use of interpretability provides additional information about the words usually associated with patient behavior. For instance, spiritual words are sometimes connected to non-healthy behavior, while work and professional words indicate healthy behavior.

The relationship between interpretability and PLMs can also be beneficial for low-resource languages, characterized by a scarcity of labeled data and language models [42, 92]. In [46], the authors present a study that applies an interpretability approach to the investigation of hate speech in Bengali, focusing on political, personal, geopolitical, and religious hate targets. In contrast to our work, their approach uses the Layer-wise Relevance Propagation (LRP) technique to obtain interpretations when hate speech is detected.

Some researchers also use interpretability to simulate and evaluate how ML models behave in an adversarial attack situation [7, 53, 108], in which small perturbations to the input can significantly degrade model performance. In addition, other works focus on leveraging cross-domain interactions. Hossam et al. [41] present a model that learns using data from a similar domain, extracting relevant features. Their assumption for creating this substitute model is that text structures are similar across different domains (such as reviews of movies and restaurants). A possible future direction is to investigate cross-domain interactions, focusing on getting relevant information about violating behavior from different communities and understanding their evolving meanings.

7 Conclusion and future work

In this work, we propose mechanisms that support normative systems to learn from the interactions and feedback of agents (human or artificial) to determine what is considered a norm violation. Our framework handles norms whose meanings may change, such as hate speech or acceptable response times. To demonstrate the effectiveness of our approach, we conduct experiments in tabular and text scenarios. We employ an ensemble of classifiers in tabular tasks, while in text classification tasks, we use the Pre-trained Language Model (PLM). Both approaches incorporate incremental learning techniques to continuously adapt to the evolving community view.

We evaluate our approach in the Wikipedia article editing task. Specifically, we focus on two challenges that emerge in such domains, the imbalanced nature of the dataset and the adaptation to the changing community view on the meaning of norm violation. Thus, our main contributions are: (1) incorporating feedback data (to be collected from a real online community in the future) to update the machine learning model as interactions unfold; and (2) using interpretability to enhance a community understanding of norm-violating behavior.

In the context of tabular tasks, we start evaluating the algorithms in the case with no concept drift, which focuses on learning the meaning of norm violation. Following this step, we evaluate our approach in a context with concept drift, specifically the drift due to the swap of class labels. For this experiment, we highlight the need for feedback from community members to enhance our framework's performance. However, as we do not have access to real feedback from community members, we use a simulation strategy to create different subgroups of the violation dataset and change their labels. This simulation assumes that the feedback is consistent since we are grouping similar edits. Thus, our interpretation of the results naturally comes from this consistency.

For the text classification tasks, we evaluate RoBERTa and DistilBERT in two different settings. First, we run experiments for the binary case, in which both PLMs should learn whether an article edit is a violation. Second, we evaluate the PLMs to solve a multi-label task, aiming at learning the specific hate speech classes in an article edit. Since we handle text data directly in these cases, we incorporate an interpretability component into our framework.

Results show that our proposal can learn the meaning of norm violations in an online community considering different scenario requirements. While ensemble and incremental learning can efficiently handle imbalanced datasets and continuously adapt to concept drift, DistilBERT and RoBERTa incorporate prior language knowledge to leverage the learning process of hate speech in our specific domain. Through interpretability analysis, we could examine the community's description of non-acceptable behavior and identify the most relevant words in the dataset usually associated with norm violation, providing a summary view of this concept.

Finally, as we argue that feedback from community members can provide information on how a community understands norm violations, future work shall focus on getting real feedback. This is not only interesting because of feedback collection but also from the point of view of how the community members will agree on the definition of norm violation. Additionally, for the ensemble of classifiers to decide if an action is a norm violation, we will investigate the adoption of different strategies, from a simple voting scheme (used in this paper) to something more complex as deliberation. For future interpretation, our work shall investigate the global interpretability of ML models. We plan to use this information as a resource to explain concept drift and how violation-related words change as community members interact. To validate our approach, user experiments shall be conducted.

Appendix 1: Global sum of relevance scores for all violation classes

See Figs. 19, 20, 21, 22, 23, 24, 25, 26, 27, 28.

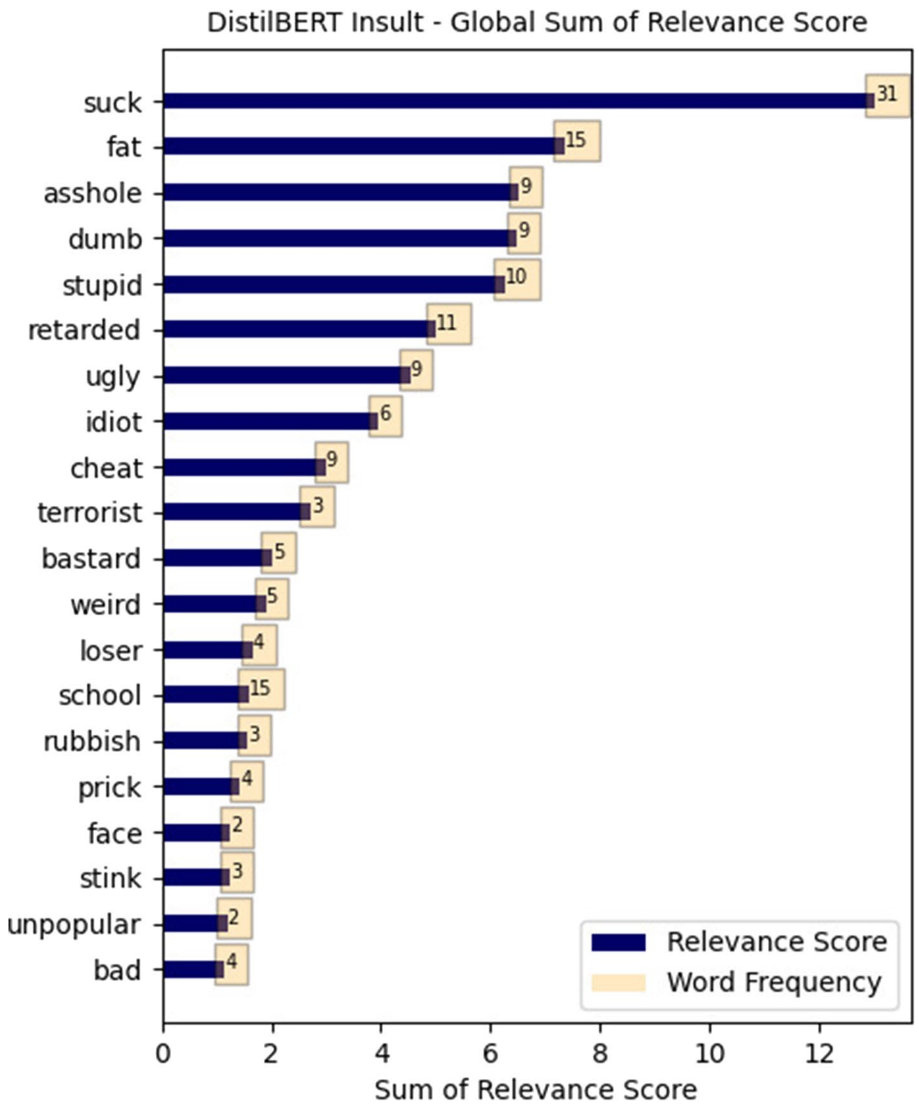


Fig. 19 The global sum of relevance score for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is INSULT AND ABLEISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

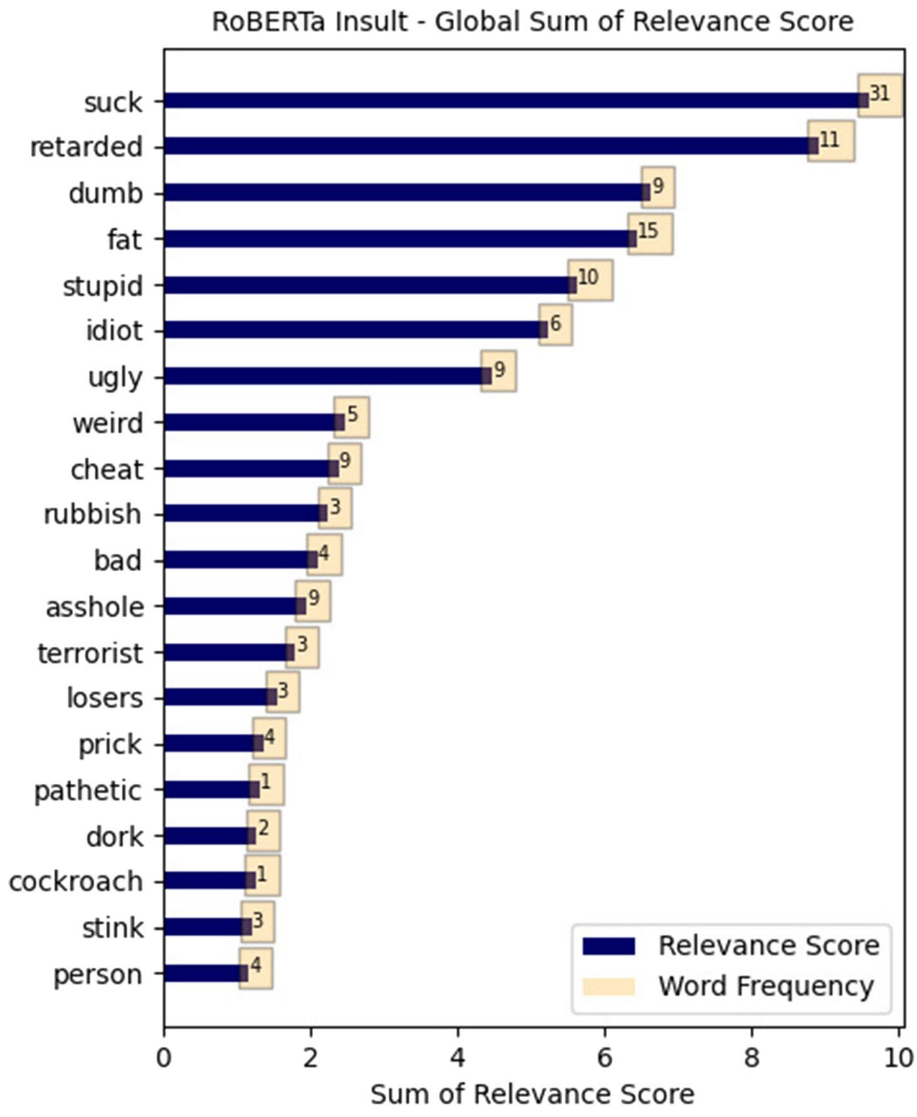


Fig. 20 The global sum of relevance score for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is INSULT AND ABLEISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

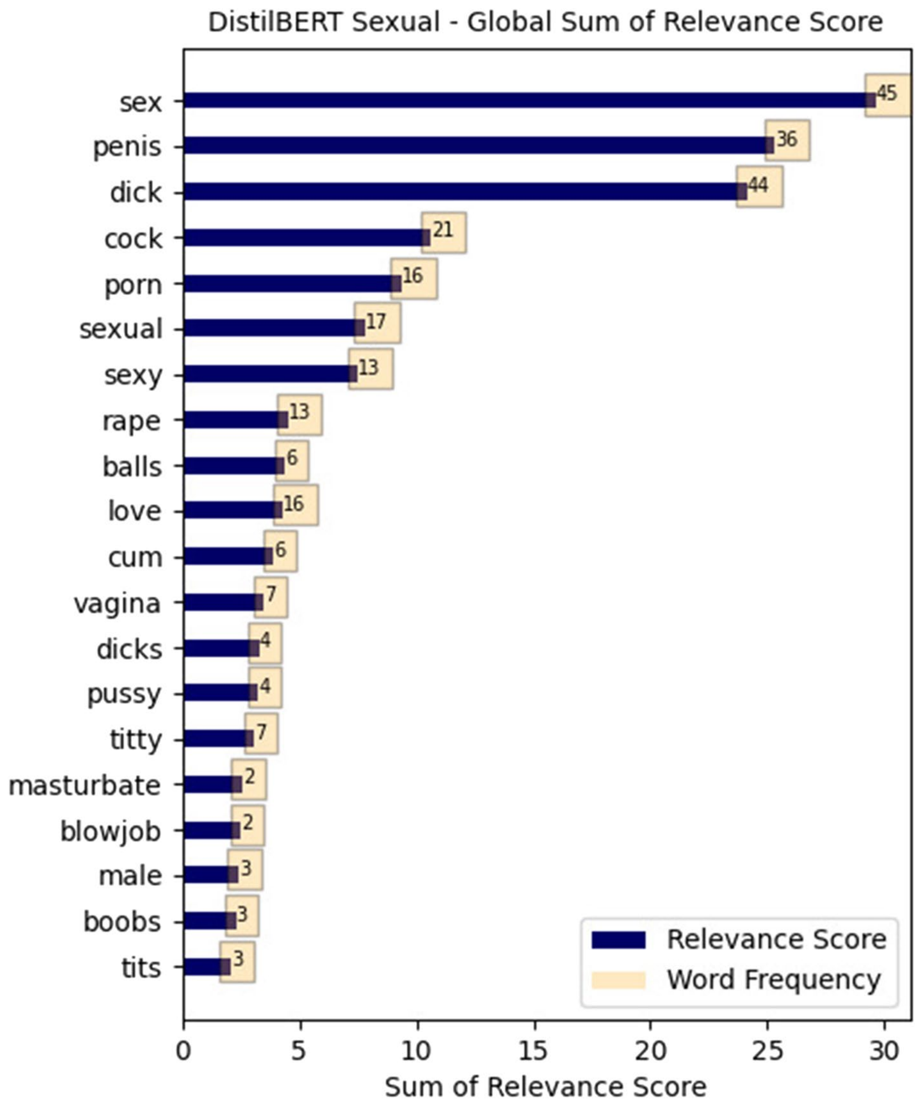


Fig. 21 The global sum of relevance score for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is SEXUAL HARASSMENT. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

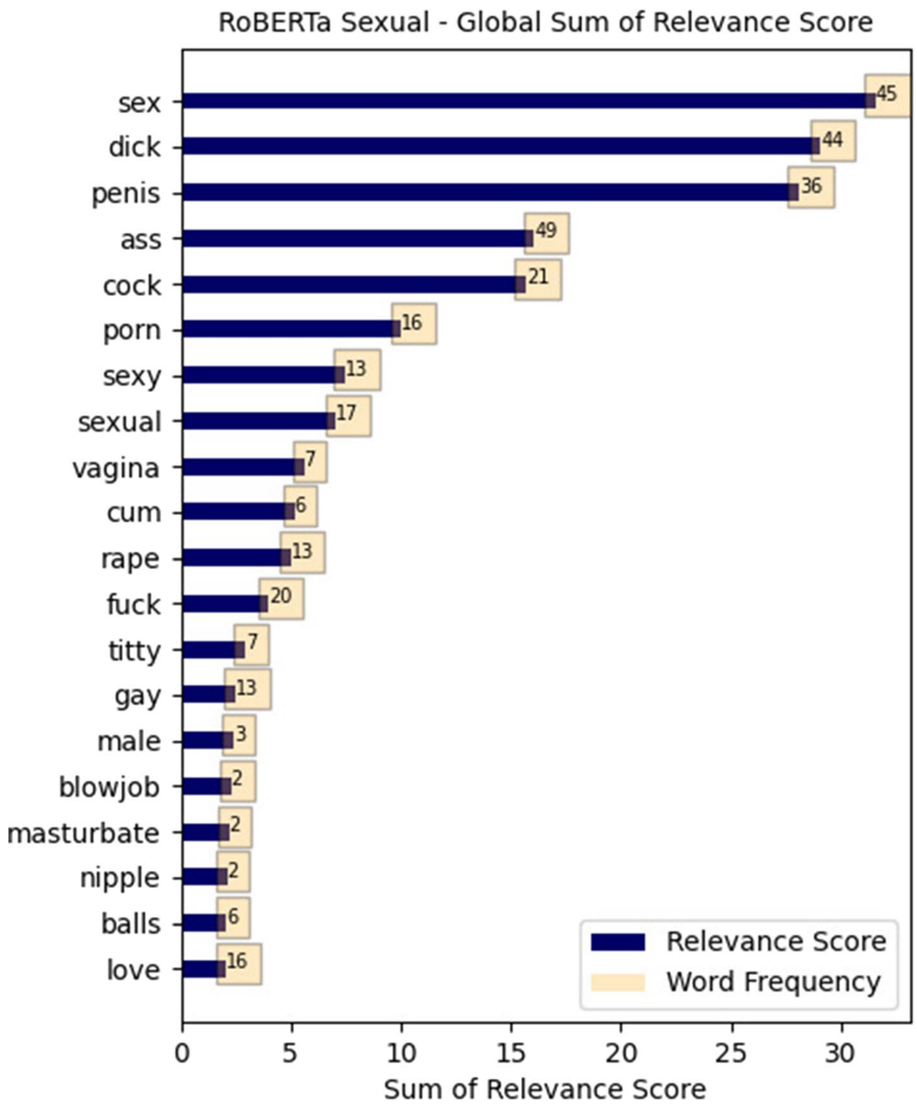


Fig. 22 The global sum of relevance score for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is SEXUAL HARASSMENT. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

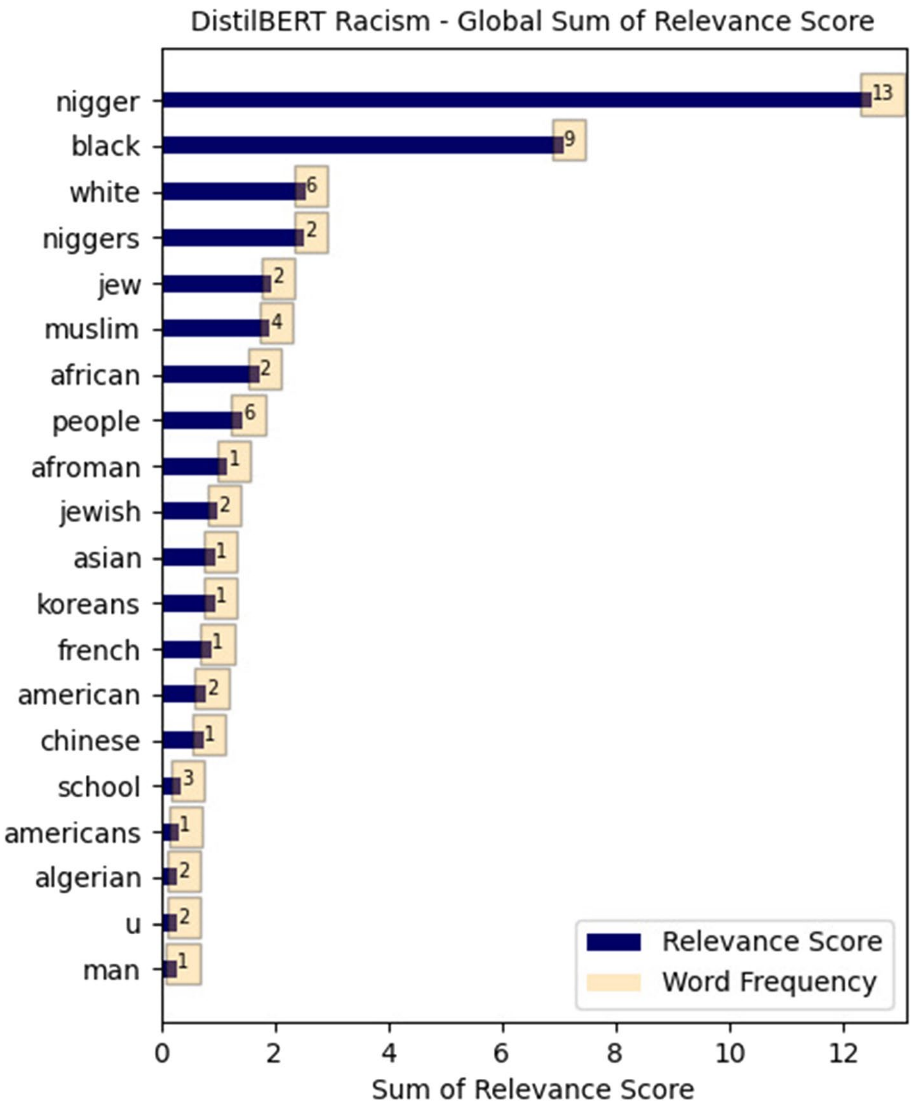


Fig. 23 The global sum of relevance score for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is RACISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

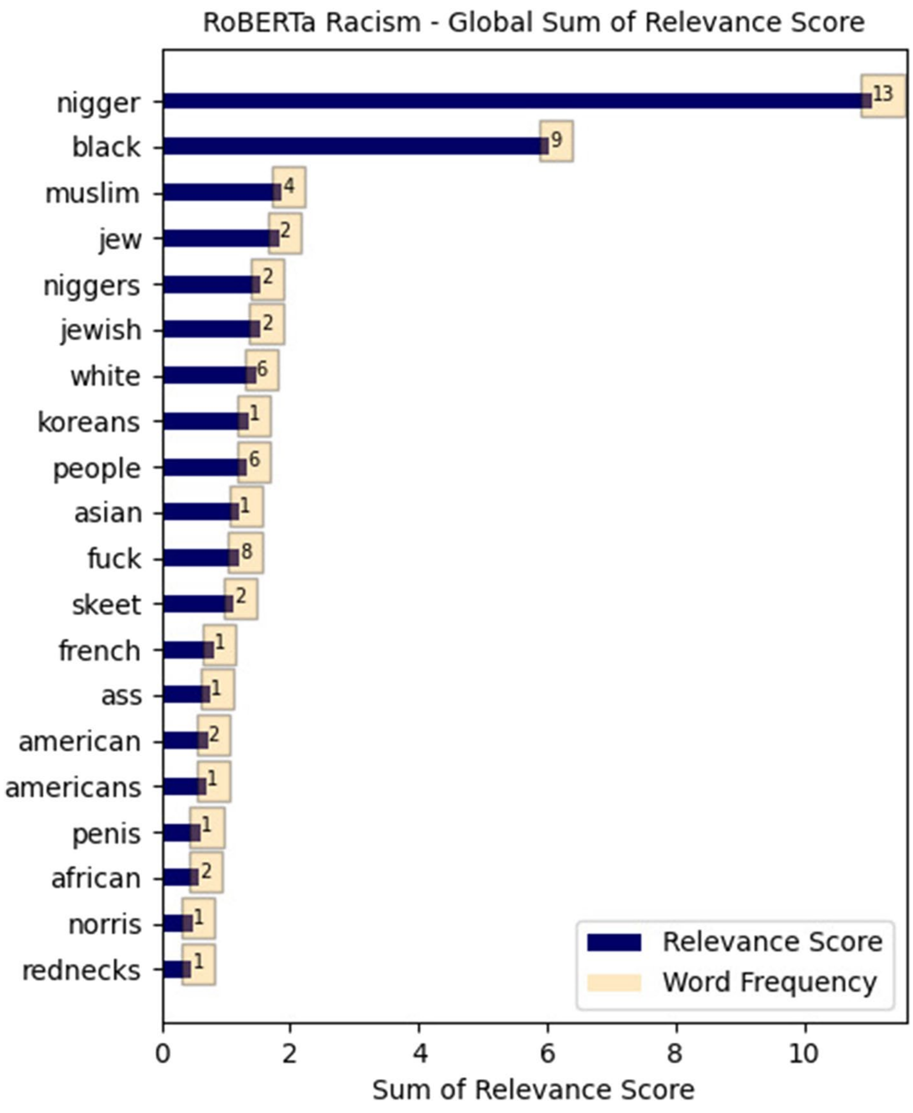


Fig. 24 The global sum of relevance score for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is RACISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

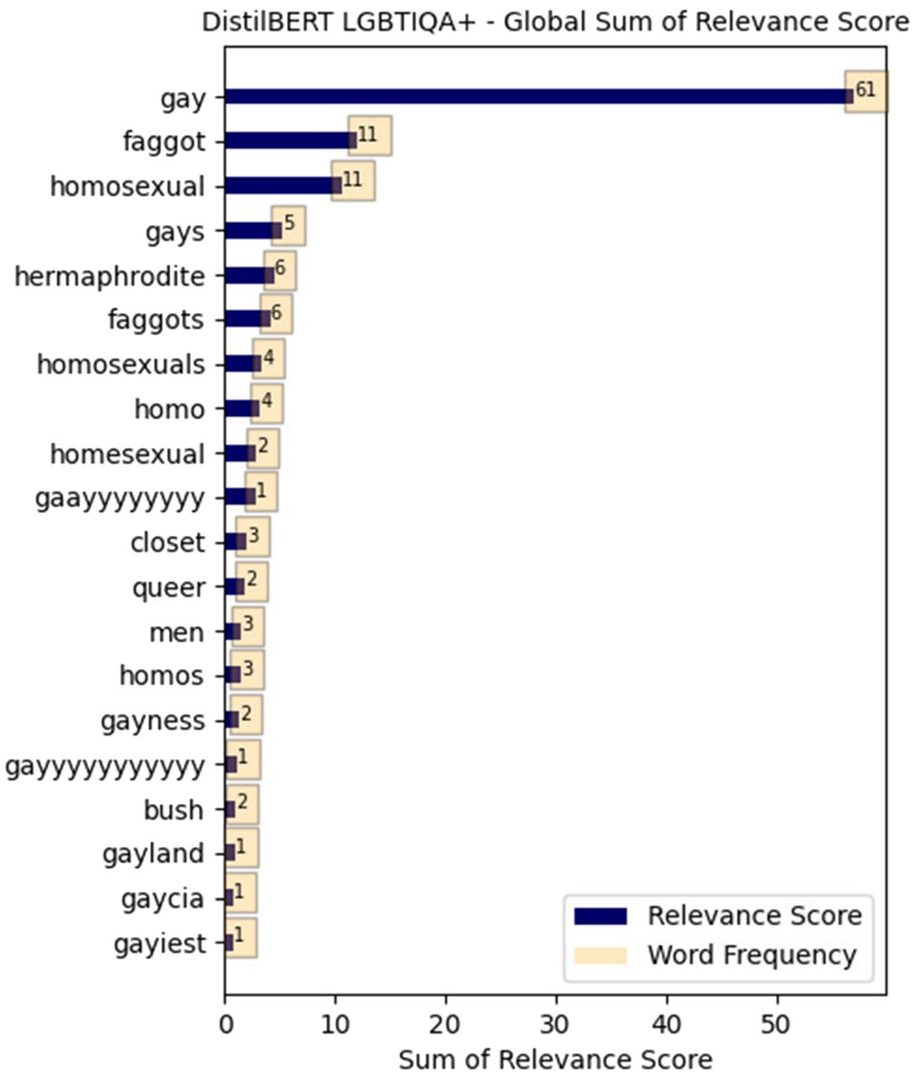


Fig. 25 The global sum of relevance score for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is LGBTQIA+ Attack. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

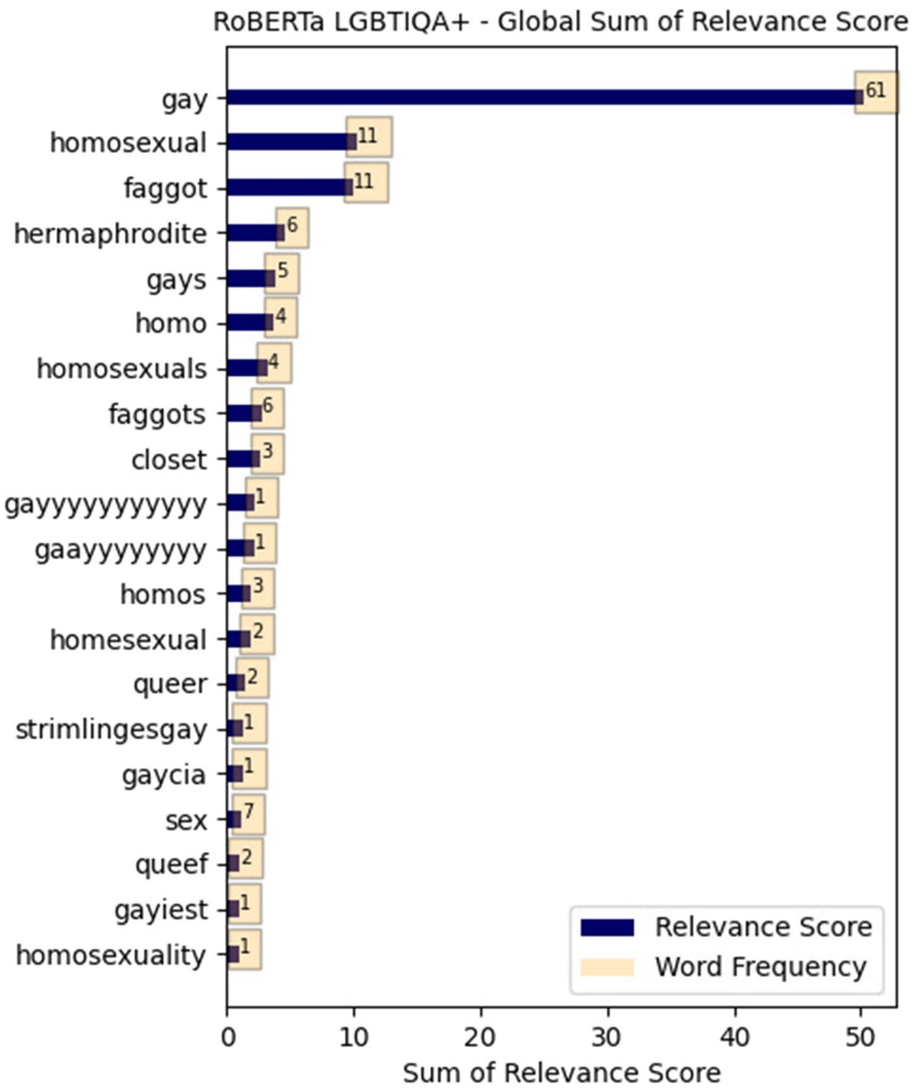


Fig. 26 The global sum of relevance score for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is LGBTQIA+ Attack. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

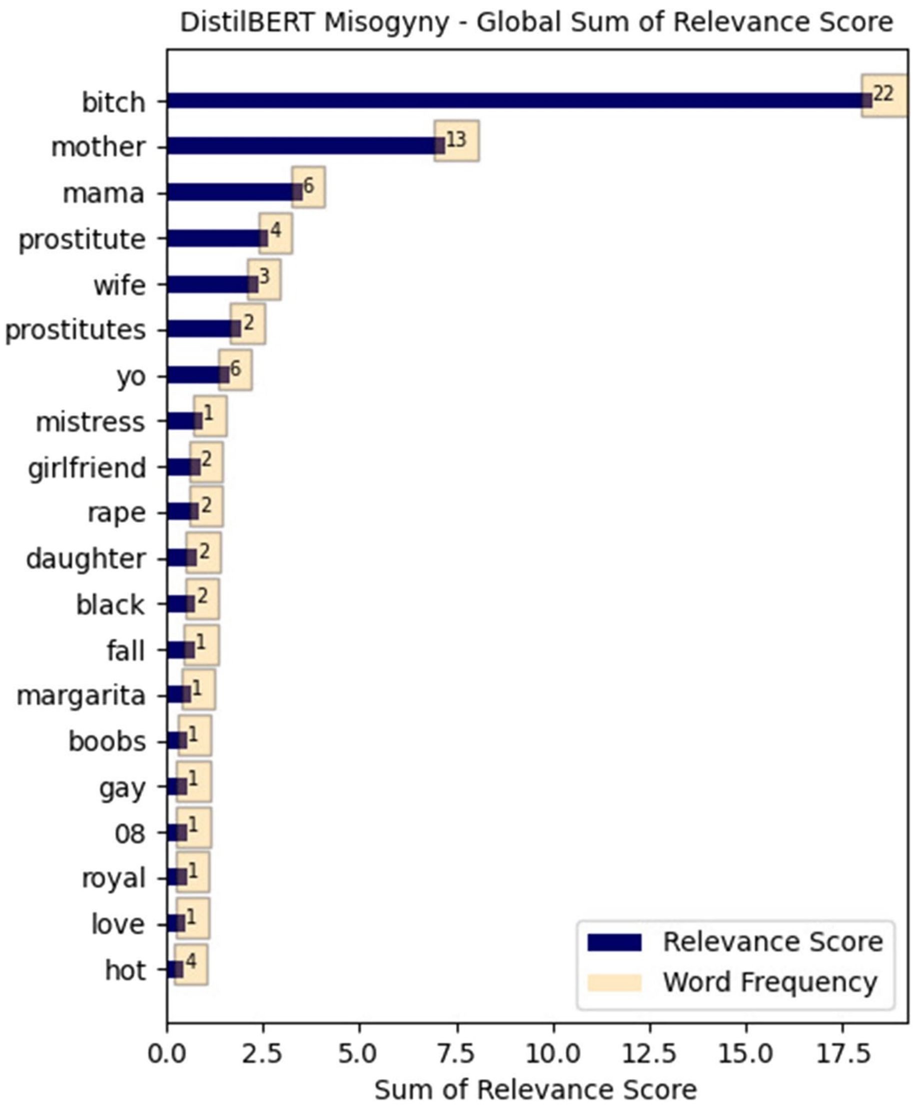


Fig. 27 The global sum of relevance score for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is MISOGYNY. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

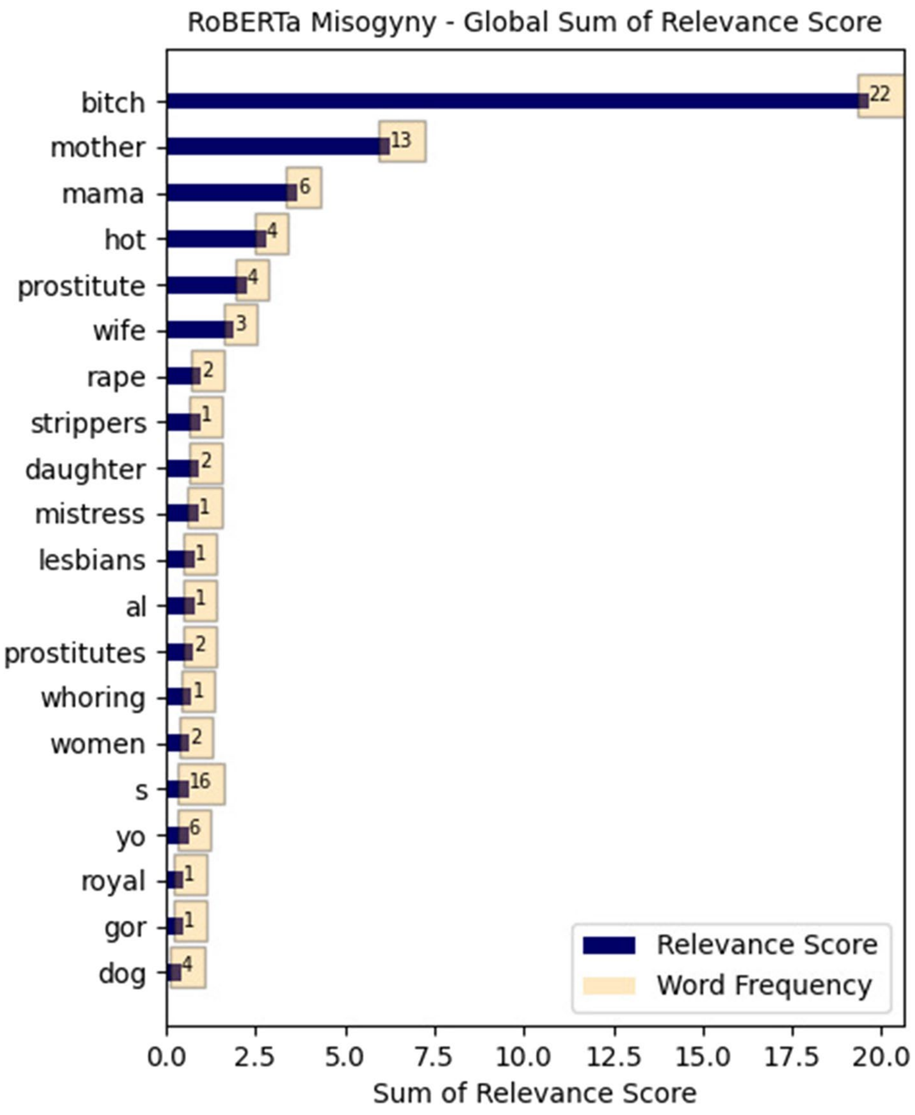
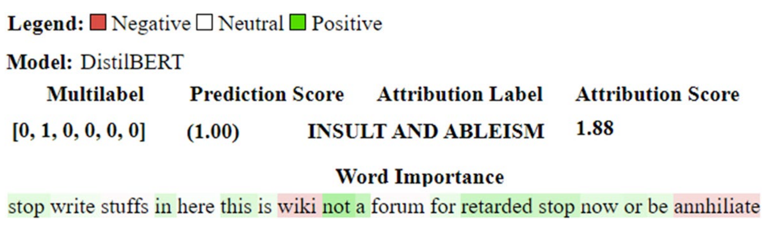


Fig. 28 The global sum of relevance score for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is MISOGYNY. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

Appendix 2: Local interpretation examples for all violation classes

See Figs. 29, 30, 31, 32, 33, 34, 35, 36, 37, 38.



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

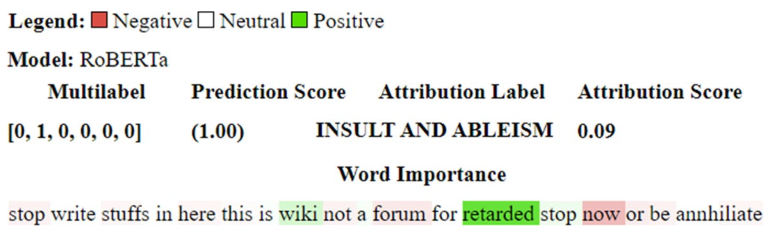
Model: DistilBERT

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 1, 0, 0, 0, 0]	(1.00)	INSULT AND ABLEISM	1.88

Word Importance

stop write stuffs in here this is wiki not a forum for retarded stop now or be annhiliate

Fig. 29 The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is INSULT AND ABLEISM. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

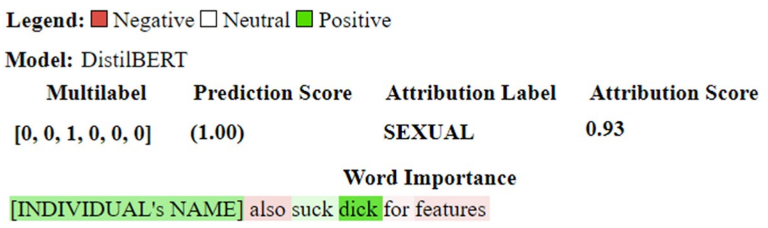
Model: RoBERTa

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 1, 0, 0, 0, 0]	(1.00)	INSULT AND ABLEISM	0.09

Word Importance

stop write stuffs in here this is wiki not a forum for retarded stop now or be annhiliate

Fig. 30 The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is INSULT AND ABLEISM. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

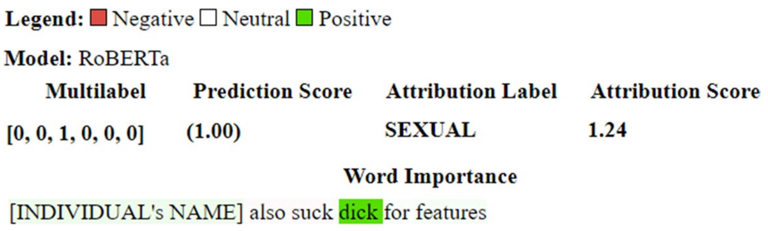
Model: DistilBERT

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 0, 1, 0, 0, 0]	(1.00)	SEXUAL	0.93

Word Importance

[INDIVIDUAL's NAME] also suck dick for features

Fig. 31 The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is SEXUAL HARASSMENT. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

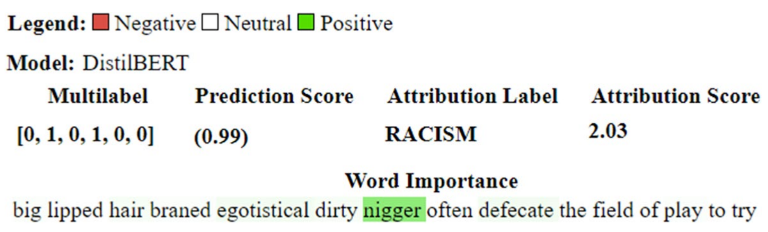
Model: RoBERTa

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 0, 1, 0, 0, 0]	(1.00)	SEXUAL	1.24

Word Importance

[INDIVIDUAL's NAME] also suck dick for features

Fig. 32 The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is SEXUAL HARASSMENT. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

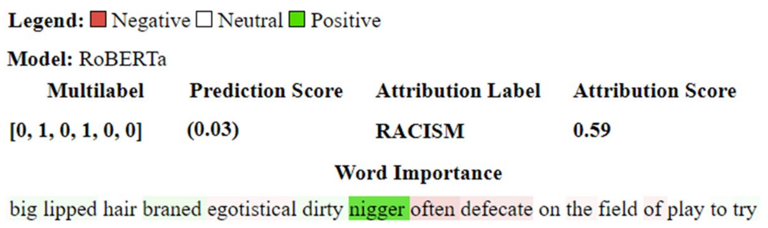
Model: DistilBERT

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 1, 0, 1, 0, 0]	(0.99)	RACISM	2.03

Word Importance

big lipped hair braned egotistical dirty nigger often defecate the field of play to try

Fig. 33 The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is RACISM. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

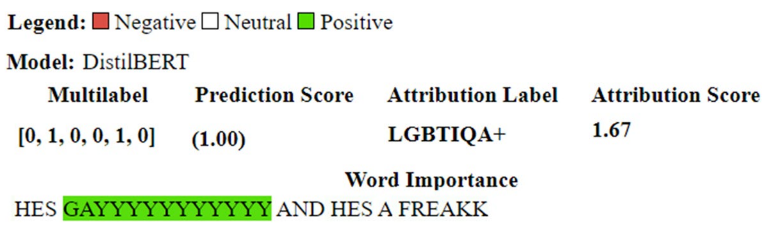
Model: RoBERTa

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 1, 0, 1, 0, 0]	(0.03)	RACISM	0.59

Word Importance

big lipped hair braned egotistical dirty nigger often defecate on the field of play to try

Fig. 34 The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is RACISM. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

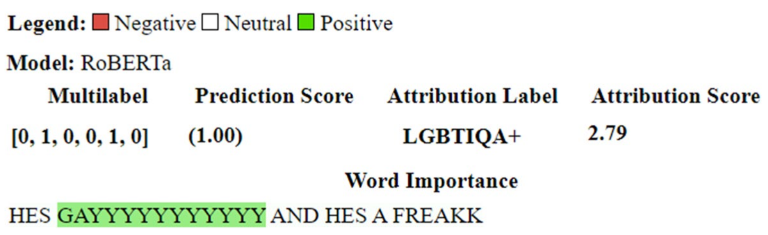
Model: DistilBERT

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 1, 0, 0, 1, 0]	(1.00)	LGBTIQA+	1.67

Word Importance

HES GAYYYYYYYYYYY AND HES A FREAKK

Fig. 35 The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is LGBTQIA+ Attack. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

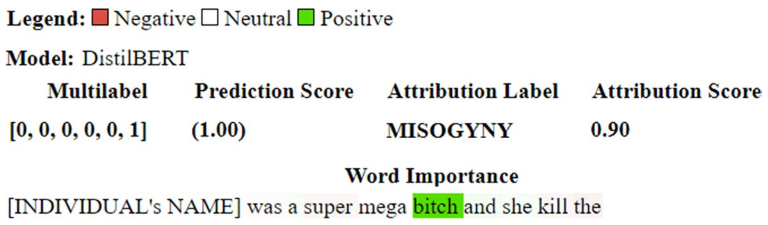
Model: RoBERTa

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 1, 0, 0, 1, 0]	(1.00)	LGBTIQA+	2.79

Word Importance

HES GAYYYYYYYYYYY AND HES A FREAKK

Fig. 36 The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is LGBTQIA+ Attack. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

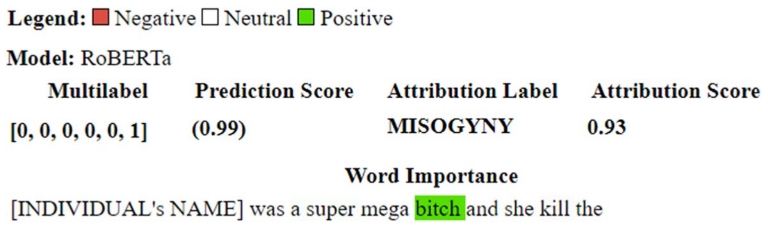
Model: DistilBERT

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 0, 0, 0, 0, 1]	(1.00)	MISOGYNY	0.90

Word Importance

[INDIVIDUAL's NAME] was a super mega bitch and she kill the

Fig. 37 The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is MISOGYNY. The relevance score is calculated using Integrated Gradients (Sect. 2.4)



Legend: 🟥 Negative ☐ Neutral 🟩 Positive

Model: RoBERTa

Multilabel	Prediction Score	Attribution Label	Attribution Score
[0, 0, 0, 0, 0, 1]	(0.99)	MISOGYNY	0.93

Word Importance

[INDIVIDUAL's NAME] was a super mega bitch and she kill the

Fig. 38 The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is MISOGYNY. The relevance score is calculated using Integrated Gradients (Sect. 2.4)

Acknowledgements This research is supported by the EU funded VALAWAI (# 101070930) and WeNet (# 823783) projects, the Spanish funded VAE (# TED2021-131295B-C31) and Rhymas (# PID2020-113594RB-100) projects, and the Generalitat de Catalunya funded *Ajuts a grups de recerca de Catalunya* (# 2021 SGR 00754) project.

Author contributions TFS created the source code, collected data, executed the experiments and the statistical analysis, organized the content, and wrote the main manuscript text. NO and MS conceived the initial idea, provided intellectual guidance, coordinated the study, and reviewed and corrected the manuscript. All authors participated in the study's conceptualization and the discussion of the results.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Adelani, D. I., Mai, H., Fang, F., Nguyen, H. H, Yamagishi, J., & Echizen, I. (2020). Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection. In: *Advanced information networking and applications: Proceedings of the 34th international conference on advanced information networking and applications (AINA-2020)*, (pp. 1341–1354), Springer.
2. Thomas Adler, B., de Alfaro, L., Mola-Velasco, S. M., Rosso, P., & West, A. G. (2011). Wikipedia vandalism detection: Combining natural language, metadata, and reputation features. *Computational linguistics and intelligent text processing* (pp. 277–288). Berlin Heidelberg: Springer.
3. Afroz, S., Brennan, M., & Greenstadt, R. (2012). Detecting hoaxes, frauds, and deception in writing style online. *2012 IEEE symposium on security and privacy* (pp. 461–475). IEEE, San Francisco, CA, USA: IEEE.
4. Aires, J. P., & Meneguzzi, F. (2021). Norm conflict identification using a convolutional neural network. In A. A. Tubella, S. Cranefield, C. Frantz, F. Meneguzzi, & W. Vasconcelos (Eds.), *Coordination, organizations, institutions, norms, and ethics for governance of multi-agent systems XIII* (pp. 3–19). Cham: Springer International Publishing.
5. Ajmeri, N., Guo, H., Murukannaiah, P. K., & Singh, M. P. (2020). Elessar: Ethics in norm-aware agents. In: *Proceedings of the 19th international conference on autonomous agents and multiagent systems*. (pp. 16–24), International foundation for autonomous agents and multiagent systems, Richland, SC.
6. Al-Hassan, A. & Al-Dossari, H. (2019). Detection of hate speech in social networks: A survey on multilingual corpus. In *6th international conference on computer science and information technology*.
7. Alsmadi, I., Ahmad, K., Nazzal, M., Alam, F., Al-Fuqaha, A., Khreishah, A., & Algosaibi, A. (2021). Adversarial attacks and defenses for social network text processing applications: Techniques, challenges and future research directions. arXiv preprint arXiv:2110.13980
8. Anand, M., & Eswari, R. (2019). Classification of abusive comments in social media using deep learning. In *2019 3rd international conference on computing methodologies and communication (ICCMC)*, (pp. 974–977).
9. Anowar, F., & Sadaoui, S. (2021). Incremental learning framework for real-world fraud detection environment. *Computational Intelligence, 37*(1), 635–656.
10. Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion, 58*(2020), 82–115.
11. Atanasova, P., Simonsen, J. G., Lioma, C., & Augenstein, I. (2020). A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*. (pp. 3256–3274), Association for computational linguistics, Online. https://doi.org/10.18653/v1/2020.emnlp-main.263
12. Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450
13. Barbado, R., Araque, O., & Iglesias, C. A. (2019). A framework for fake review detection in online consumer electronics retailers. *Information Processing and Management, 56*(4), 1234–1244.
14. Belle, V., & Papantonis, I. (2021). Principles and practice of explainable machine learning. *Frontiers in Big Data, 2021*, 39.
15. Biber, J. K., Doverspike, D., Baznik, D., Cober, A., & Ritter, B. A. (2002). Sexual harassment in online communications: Effects of gender and discourse medium. *Cyber Psychology and Behavior, 5*(1), 33–42.
16. Bogart, K. R., & Dunn, D. S. (2019). Ableism special issue introduction. *Journal of Social Issues, 75*(3), 650–664.
17. Brzezinski, D., & Stefanowski, J. (2014). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems, 25*(1), 81–94.
18. Campos, J., Lopez-Sanchez, M., Salamó, M., Avila, P., & Rodríguez-Aguilar, J. A. (2013). Robust regulation adaptation in multi-agent systems. *ACM Transactions on Autonomous and Adaptive Systems (TAAS), 8*(3), 1–27.
19. Chandrasekharan, E., Gandhi, C., Mustelier, M. W., & Gilbert, E. (2019). Crossmod: A cross-community learning-based system to assist reddit moderators. *Proceedings of the ACM on Human-Computer Interaction, 3*, 30.
20. Chandrika, C. P., & Kallimani, J. S. (2020). Classification of abusive comments using various machine learning algorithms. In P. K. Mallick, V. E. Balas, A. K. Bhoi, & G.-S. Chae (Eds.), *Cognitive Informatics and soft computing* (pp. 255–262). Springer Singapore.

21. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of artificial intelligence research, 16*(2002), 321–357.
22. Cheriyan, J., Savarimuthu, B. T. R., & Cranefield, S. (2017). Norm violation in online communities–A study of stack overflow comments. In: *Coordination, organizations, institutions, norms, and ethics for governance of multi-agent systems XIII*, (pp. 20–34), Springer.
23. Cheriyan, J., Savarimuthu, B. T. R., & Cranefield, S. (2021). Towards offensive language detection and reduction in four software engineering communities. In *Evaluation and assessment in software engineering*, (pp. 254–259).
24. Chollet, F. et al. (2015). Keras. https://keras.io.
25. Criado, N., Ferrer, X., & Such, J. M. (2020). A normative approach to attest digital discrimination. arXiv preprint arXiv:2007.07092
26. De-Arteaga, M., Romanov, A., Wallach, H., Chayes, J., Borgs, C., Chouldechova, A., Geyik, S., Kenthapadi, K., & Kalai, A. T. (2019). Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the conference on fairness, accountability, and transparency(FAT* '19)*. (pp. 120–128), Association for computing machinery, New York, NY, USA. https://doi.org/10.1145/3287560.3287572
27. Dell'Anna, D., Alechina, N., Dalpiaz, F., Dastani, M., & Logan, B. (2022). Data-driven revision of conditional norms in multi-agent systems. *Journal of Artificial Intelligence Research, 75*(2022), 1549–1593.
28. Ding, H., & Jurgens, D. (2021). HamiltonDinggg at SemEval-2021 Task 5: Investigating toxic span detection using RoBERTa pre-training. In *Proceedings of the 15th international workshop on semantic evaluation (Semeval-2021)*. (pp. 263–269), Association for computational linguistics, Online. https://doi.org/10.18653/v1/2021.semeval-1.31
29. Dong, X., Zhiwen, Yu., Cao, W., Shi, Y., & Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science, 14*(2), 241–258.
30. Hongle, D., Zhang, Y., Gang, K., Zhang, L., & Chen, Y.-C. (2021). Online ensemble learning algorithm for imbalanced data stream. *Applied Soft Computing, 107*(2021), 107378. https://doi.org/10.1016/j.asoc.2021.107378
31. Elazar, Y., Kassner, N., Ravfogel, S., Ravichander, A., Hovy, E., Schütze, H., & Goldberg, Y. (2021). Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics, 9*(2021), 1012–1031.
32. Fenech, Stephen, Pace, Gordon J., & Schneider, Gerardo. (2009). Automatic conflict detection on contracts. *International colloquium on theoretical aspects of computing* (pp. 200–214). Springer.
33. Freitas dos Santos, T. , Osman, N., & Schorlemmer, M. (2022a). Ensemble and incremental learning for norm violation detection. In *Proceedings of the 21st international conference on autonomous agents and multiagent systems* (pp. 427–435).
34. Freitas dos Santos, T., Osman, N., & Schorlemmer, M. (2022b). Learning for detecting norm violation in online communities. In: *Coordination, organizations, institutions, norms, and ethics for governance of multi-agent systems XIV: International workshop, COINE 2021, London, UK, May 3, 2021, Revised Selected Papers* (pp. 127–142), Springer.
35. Gama, J., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*, 4.
36. Gao, X., & Singh, M. P. (2014). Extracting normative relationships from business contracts. In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. (pp. 101–108).
37. Ging, D., & Siapera, E. (2018). Special issue on online misogyny.
38. Gray, K. L. (2018). Gaming out online: Black lesbian identity development and community building in Xbox Live. *Journal of Lesbian Studies, 22*(3), 282–296.
39. Harper, G. W., & Schneider, M. (2003). Oppression and discrimination among lesbian, gay, bisexual, and transgendered people and communities: A challenge for community psychology. *American Journal of Community Psychology, 31*(3), 243–252.
40. Hoi, S. C. H., Sahoo, D., Jing, L., & Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing, 459*(2021), 249–289. https://doi.org/10.1016/j.neucom.2021.04.112
41. Hossam, M., Le, T., Zhao, H., & Phung, D. (2021). Explain2Attack: Text adversarial attacks via cross-domain interpretability. In: *2020 25th international conference on pattern recognition (ICPR)*. (pp. 8922–8928), IEEE.
42. Ishmam, A. M., & Sharmin, S. (2019). Hateful speech detection in public facebook pages for the bengali language. In *2019 18th IEEE international conference on machine learning and applications (ICMLA)*, (pp. 555–560). https://doi.org/10.1109/ICMLA.2019.00104

43. Islam, R., Ben Treves, Md., Rokon, O. F., & Faloutsos, M. (2022). HyperMan: Detecting misbehavior in online forums based on hyperlink posting behavior. *Social Network Analysis and Mining, 12*(1), 1–14.
44. Joachims, T. (1996). *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization.* Technical Report. Carnegie-mellon univ pittsburgh pa dept of computer science.
45. Kaliyar, R. K., Goswami, A., & Narang, P. (2021). FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. *Multimedia Tools and Applications, 80*(8), 11765–11788.
46. Karim, M. R., Dey, S. K., Islam, T., Sarker, S., Menon, M. H., Hossain, K., Hossain, M. A., & Decker, S. (2021). Deephateexplainer: Explainable hate speech detection in under-resourced bengali language. In *2021 IEEE 8th international conference on data science and advanced analytics (DSAA)*. (pp. 1–10), IEEE.
47. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of naacL-HLT*, (pp. 4171–4186).
48. Keum, B. T. H., & Miller, M. J. (2018). Racism on the internet: Conceptualization and recommendations for research. *Psychology of Violence, 8*(6), 782.
49. Krishna, K., & Murty, M. N. (1999). Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 29*(3), 433–439.
50. Lebichot, B., Paldino, G. M., Siblini, W., He-Guelton, L., Oblé, F., & Bontempi, G. (2021). Incremental learning strategies for credit cards fraud detection. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)* (pp. 785-786). IEEE.
51. LekshmiAmmal, H. R. I., Ravikiran, M., & Madasamy, A. K. (2022). NITK-IT_NLP@ TamilNLP-ACL2022: Transformer based model for offensive span identification in Tamil. *DravidianLangTech, 2022*(2022), 75.
52. Li, T. C., Gharibshah, J., Papalexakis, E. E., & Faloutsos, M. (2017). TrollSpot: Detecting misbehavior in commenting platforms. In *Proceedings of the 2017 IEEE/ACM international conference on advances in social networks analysis and mining 2017*. (pp. 171–175).
53. Li, Y., Cheng, M., Hsieh, C.-J., & Lee, T. C. M. (2022). A review of adversarial attack and defense for classification methods. *The American Statistician, 2022*, 1–17.
54. Li, Z., Huang, W., Xiong, Y., Ren, S., & Zhu, T. (2020). Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm. *Knowledge-Based Systems, 195*(2020), 105694.
55. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M, Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692
56. Jie, L., Liu, A., Dong, F., Feng, G., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering, 31*(12), 2346–2363.
57. Lyu, Q., Apidianaki, M., & Callison-Burch, C. (2022). Towards faithful model explanation in NLP: A survey. arXiv preprint arXiv:2209.11326
58. Mahmoud, S., Griffiths, N., Keppens, J., & Luck, M. (2012). Efficient norm emergence through experiential dynamic punishment. In *ECAI 2012*. (pp. 576–581), IOS Press.
59. Markov, I., Gevers, I., & Daelemans, W. (2022). An ensemble approach for Dutch cross-domain hate speech detection. In *International conference on applications of natural language to information systems*. (pp. 3–15), Springer.
60. McLean, L., & Griffiths, M. D. (2019). Female gamers' experience of online harassment and social support in online gaming: A qualitative study. *International Journal of Mental Health and Addiction, 17*(4), 970–994.
61. Min, B., Ross, H., Sulem, E., Pouran B. V., Amir, N., Thien H., Sainz, O., Agirre, E., Heinz, I., & Roth, D. (2021). Recent advances in natural language processing via large pre-trained language models: A survey. arXiv preprint arXiv:2111.01243
62. Mitrović, S., Andreoletti, D., & Ayoub, O. (2023). Chatgpt or human? Detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text. arXiv preprint arXiv: 2301.13852
63. Mohawesh, R., Tran, S., Ollington, R., & Shuxiang, X. (2021). Analysis of concept drift in fake reviews detection. *Expert Systems with Applications, 169*(2021), 114318.
64. Mollas, I., Chrysopoulou, Z., Karlos, S., & Tsoumakas, G. (2022). ETHOS: A multi-label hate speech detection dataset. *Complex and Intelligent Systems, 2022*, 1–16.
65. Montiel, J., Halford, M., Mastelini, S. M., Bolmier, G., Sourty, R., Vaysse, R., Zouitine, A., Gomes, H. M., Read, J., Abdessalem, T., & Bifet, A. (2021). River: Machine learning for streaming data in Python. *Journal of Machine Learning Research, 22*(110), 1–8.
66. Morales, J., López-Sánchez, M., Rodríguez-Aguilar, J. A., Wooldridge, M., & Vasconcelos, W. (2015). Synthesising liberal normative systems. In *Proceedings of the 2015 international conference*

on autonomous agents and multiagent systems(AAMAS '15). (pp. 433–441), International foundation for autonomous agents and multiagent systems, Richland, SC.

67. Morales, J., Wooldridge, M., Rodríguez-Aguilar, J. A., & López-Sánchez, M. (2018). Off-line synthesis of evolutionarily stable normative systems. *Autonomous Agents and Multi-Agent Systems, 32*(5), 635–671.
68. Morris-Martin, A., De Vos, M., & Padget, J. (2019). Norm emergence in multiagent systems: A viewpoint paper. *Autonomous Agents and Multi-Agent Systems, 33*(6), 706–749.
69. Mridha, M. F., Keya, A. J., Hamid, M. A., Monowar, M. M., & Rahman, M. S. (2021). A comprehensive review on fake news detection with deep learning. *IEEE Access*.
70. Muslim, F., Purwarianti, A., & Ruskanda, F. Z. (2021). Cost-sensitive learning and ensemble bert for identifying and categorizing offensive language in social media. In *2021 8th international conference on advanced informatics: Concepts, theory and applications (ICAICTA)*. (pp. 1–6), IEEE.
71. Nir, R., Shleyfman, A., & Karpas, E. (2020). Automated synthesis of social laws in strips. *In Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 9941–9948.
72. Niu, R., Wei, Z., Wang, Y., & Wang, Q. (2022). Attexplainer: Explain Transformer via Attention by Reinforcement Learning.
73. Nockleby, J. (2000). Hate speech. In L. Levy, K. Kenneth, A. Winkler (Eds.). *Encyclopedia of the American Constitution*, Vol 6. (pp. 1277–1279).
74. Novikova, J., & Shkaruta, K. (2022). DECK: Behavioral tests to improve interpretability and generalizability of BERT models detecting depression from text. arXiv preprint arXiv:2209.05286
75. Peng, J., Choo, K.-K.R., & Ashman, H. (2016). Bit-level n-gram based forensic authorship analysis on social media: Identifying individuals from linguistic profiles. *Journal of Network and Computer Applications, 70*(2016), 171–182.
76. Potthast, M., & Holfeld, T. (2010). Overview of the 1st International Competition on Wikipedia Vandalism Detection. In *CLEF*.
77. Qiang, Y., Pan, D., Li, C., Li, X., Jang, R., & Zhu, D. (2022). AttCAT: Explaining transformers via attentive class activation tokens. In *Advances in neural information processing systems*.
78. Qiu, X., Sun, T., Yige, X., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences, 63*(10), 1872–1897.
79. Räukur, T., Ho, A., Casper, S., & Hadfield-Menell, D. (2022). Toward transparent AI: A survey on interpreting the inner structures of deep neural networks. arXiv preprint arXiv:2207.13243
80. Ren, S., Liao, B., Zhu, W., Li, Z., Liu, W., & Li, K. (2018). The gradual resampling ensemble for mining imbalanced data streams with concept drift. *Neurocomputing, 286*(2018), 150–166.
81. Risch, J., & Krestel, R. (2020). Toxic comment detection in online discussions. In *Deep learning-based approaches for sentiment analysis*. (pp. 85–109), Springer.
82. Rosso, P., Correa, S., & Buscaldi, D. (2011). Passage retrieval in legal texts. *The Journal of Logic and Algebraic Programming, 80*(3–5), 139–153.
83. Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery, 8*(4), e1249.
84. Salemi, A., Sabri, N., Kebriaei, E., Bahrak, B., & Shakery, A. (2021). UTNLP at SemEval-2021 Task 5: A comparative analysis of toxic span detection using attention-based, named entity recognition, and ensemble models. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*. (pp. 995–1002) Association for Computational Linguistics, Online. https://doi.org/10.18653/v1/2021.semeval-1.136
85. Samghabadi, N. S., Patwa, P., Pykl, S., Mukherjee, P., Das, A., & Solorio, T. (2020). Aggression and misogyny detection using BERT: A multi-task approach. In *Proceedings of the second workshop on trolling, aggression and cyberbullying*, (pp. 126–131).
86. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108
87. Sarzynska-Wawer, J., Wawer, A., Pawlak, A., Szymanowska, J., Stefaniak, I., Jarkiewicz, M., & Okruszek, L. (2021). Detecting formal thought disorder by deep contextualized word representations. *Psychiatry Research, 304*(2021), 114135.
88. Savarimuthu, B. T. R., , Purvis, M., Purvis, M., & Cranefield, S. (2008). Social norm emergence in virtual agent societies. In *International workshop on declarative agent languages and technologies*. (pp. 18–28), Springer.
89. Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing, 45*(11), 2673–2681.
90. Sechidis, K., Tsoumakas, G., & Vlahavas, I. (2011). On the stratification of multi-label data. In: *Joint European conference on machine learning and knowledge discovery in databases*. (pp. 145–158), Springer.

91. Serramia, M., Lopez-Sanchez, M., & Rodriguez-Aguilar, J. A. (2020). A qualitative approach to composing value-aligned norm systems. In *Proceedings of the 19th international conference on autonomous agents and multiagent systems*. (pp. 1233–1241).
92. Sharma, A., Kabra, A., & Jain, M. (2022). Ceasing hate with MoH: Hate speech detection in Hindi–English code-switched language. *Information Processing and Management, 59*(1), 102760. https://doi.org/10.1016/j.ipm.2021.102760
93. Shojaee, S, Murad, M. A. A., Azman, A. B., Sharef, N. M., & Nadali, S. (2013). Detecting deceptive reviews using lexical and syntactic features. In *2013 13th international conference on intellient systems design and applications*. (pp. 53–58), IEEE.
94. Skopik, F., & Pahi, T. (2020). Under false flag: Using technical artifacts for cyber attack attribution. *Cybersecurity, 3*(2020), 1–20.
95. Somasundaram, A., & Reddy, S. (2019). Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance. *Neural Computing and Applications, 31*(1), 3–14.
96. Strøm, E. (2021). Multi-label style change detection by solving a binary classification problem. In *CLEF (working notes)*. (pp. 2146–2157).
97. Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *International conference on machine learning*. PMLR, pp. 3319–3328.
98. Szczepański, M., Pawlicki, M., Kozik, R., & Choraś, M. (2021). New explainability method for BERT-based model in fake news detection. *Scientific Reports, 11*(1), 1–13.
99. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems* 30.
100. Ventura, F., Greco, S., Apiletti, D., & Cerquitelli, T. (2022). Trusting deep learning natural-language models via local and global explanations. *Knowledge and Information Systems, 64*(7), 1863–1907.
101. Wang, H., & Abraham, Z. (2015). Concept drift detection for streaming data. In *2015 International joint conference on neural networks (IJCNN)*. (pp. 1–9), IEEE.
102. Wang, S., Minku, L. L., & Yao, X. (2015). Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering, 27*(5), 1356–1368.
103. Wang, S., Minku, L. L., & Yao, X. (2018). A systematic study of online class imbalance learning with concept drift. *IEEE Transactions on Neural Networks and Learning Systems, 29*(10), 4802–4821.
104. West, A. G., & Lee, I. (2011). Multilingual vandalism detection using language-independent and ex post facto evidence. In *CLEF Notebooks*.
105. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*. (pp. 38–45), Association for Computational Linguistics, Online. https://www.aclweb.org/anthology/2020.emnlp-demos.6
106. Xiang, T., MacAvaney, S., Yang, E., & Goharian, N. (2021). ToxCCIn: Toxic content classification with interpretability. In *Proceedings of the eleventh workshop on computational approaches to subjectivity, sentiment and social media analysis*. (pp. 1–12), Association for Computational Linguistics, Online. https://aclanthology.org/2021.wassa-1.1
107. Xu, J., Sun, X., Zhang, Z., Zhao, G., & Lin, J. (2019). Understanding and improving layer normalization. *Advances in Neural Information Processing Systems* 32.
108. Yang, P., Chen, J., Hsieh, C.-J., Wang, J.-L., & Jordan, M. I. (2020). Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *Journal of Machine Learning Research, 21*(43), 1–36.
109. Yun-tao, Z., Ling, G., & Yong-cheng, W. (2005). An improved TF-IDF approach for text classification. *Journal of Zhejiang University-Science A, 6*(1), 49–55.
110. Zangerle, E., Mayerl, M., Specht, G., Potthast, M., & Stein, B. (2020). Overview of the style change detection task at PAN 2020. In *CLEF (Working Notes)* 93.
111. Zhang, H., Liu, W., Wang, S., Shan, J., & Liu, Q. (2019). Resample-based ensemble framework for drifting imbalanced data streams. *IEEE Access, 7*(2019), 65103–65115. https://doi.org/10.1109/ACCESS.2019.2914725