UAB
Universitat Autònoma
de Barcelona

Ph.D. in Computer Science

Research line: Artificial Intelligence

A Multi-Scenario Approach to Continuously Learn and Understand the Evolution of Norm Violations

Bellaterra (Cerdanyola del Vallès), February 5, 2024

PhD Student: Thiago Freitas dos Santos

PhD Advisors: Dr. Nardine Osman and Dr. Marco Schorlemmer

Institut d'Investigació en Intel·ligència Artificial (IIIA-CSIC)

IIIA-CSIC INSTITUT D'INVESTIGACIÓ EN INTEL·LIGÈNCIA ARTIFICIAL

CSIC Consejo Superior de Investigaciones Científicas

Abstract

Online communities establish norms to regulate interactions between agents (community members) with diverse backgrounds and views. A normative system that intends to regulate community members' behavior faces the challenge of continuously learning what constitutes a norm violation as communities' views evolve (e.g., change in what characterizes a violation, emergence of new violation classes). To address this challenge, we propose a Machine Learning (ML) approach that supports normative systems to continuously learn what constitutes a norm violation from interactions and community members' feedback. Our solution uses examples of actions labeled as violations in a multi-scenario context, where actions are formalized as a set of features (tabular data) or as text sentences. This thesis focuses on low-resource or newly created online communities, domains characterized by limited available training data. To learn in this context, our proposal incorporates data from different sources (communities) to improve the performance of ML models in a new target community.

In addition to identifying norm violations, we argue that normative systems must be capable of explaining the different views manifested in the communities, providing information on which elements of an action contribute to a norm violation. Thus, our solution employs ML interpretability to provide information on how such views change within a specific domain (over time) or across different communities. We believe this constitutes a key contribution of our work, which helps describe how detrimental behavior evolves.

The main contribution of this thesis is integrating incremental learning in three frameworks: FeDAL, LaMAL, and CAL. FeDAL is an ensemble of neural networks built to handle tabular data with class distribution imbalance. LaMAL employs a Pre-Trained Language Model (PLM) to allow handling text data in binary and multi-label classification tasks. CAL combines adapters with a PLM, permitting the incorporation of text data from different sources and hence allowing us to handle the emergence of new violation classes. Regarding interpretability, FeDAL uses the LIME algorithm to explain the relevant features contributing to violation detection in the ensemble of classifiers. LaMAL and CAL use the Integrated Gradients (IG) algorithm, specifically built to understand the inner workings of transformer-based models (in our case, PLMs) by identifying the relevant words for violation detection. We evaluate these frameworks with a small set of data from the Wikipedia article editing task, where the norm that we address is the prohibition of vandalism. Results show that FeDAL, LaMAL, and CAL can detect norm violations with imbalanced data and changing community views while successfully explaining the different ML models' output. Moreover, CAL's results indicate that incorporating data from different communities enhances the model's performance in a new target community with limited labeled data.

Finally, we conduct a user study to assess whether different interpretability layouts can influence user views when evaluating sentences containing hate speech. While our statistical analysis indicates that none of the interpretability layouts significantly influence participants' views regarding the classification of hate speech, our qualitative analysis provides valuable insights into the impact of incorporating ML interpretability in our solution. Specifically, by better understanding how an ML model classifies norm violations, users can react and provide relevant feedback when they notice a disagreement with the model's outputs. As such, interpretability can be viewed as

a tool to trigger a model's update when necessary. Moreover, interpretability data can provide valuable information for engineers when evaluating the ML model's behavior beyond traditional performance metrics.

Keywords: norm violation, concept drift, ensemble learning, incremental learning, interpretability

Resum

Les comunitats en línia estableixen normes per regular les interaccions entre agents (membres de la comunitat) amb antecedents i punts de vista diversos. Un sistema normatiu que pretén regular el comportament dels membres de la comunitat s'enfronta al repte d'aprendre de manera contínua què constitueix una violació de la norma a mesura que les opinions de les comunitats evolucionen (per exemple, canvis en el que caracteritza una violació, emergència de noves classes de violacions). Per abordar aquest repte, proposem un enfocament basat en l'aprenentatge automàtic que dona suport als sistemes normatius per aprendre de manera contínua de les interaccions i les reaccions dels membres de la comunitat. La nostra solució utilitza exemples d'accions etiquetades com a violacions en un context de múltiples escenaris, permetent que les accions es formalitzin com un conjunt de característiques (dades tabulars) o com a frases de text. Aquesta tesi se centra en comunitats en línia amb pocs recursos o de nova creació, àmbits on l'aprenentatge es produeix en un context amb poques dades disponibles. Com a tal, la nostra solució pot incorporar dades de diferents fonts (comunitats) per millorar el rendiment dels models d'aprenentatge automàtic en una nova comunitat destí.

A més de la identificació de violacions de normes, argumentem que els sistemes normatius han d'explicar les diferents perspectives que es manifesten a les comunitats, incloent-hi evidències sobre els elements rellevants d'una acció que s'associen a violacions de normes, i com poden diferir a la llum del caràcter evolutiu de les interaccions en línia. Analitzar els canvis de perspectiva de les comunitats dins d'un domini específic o entre diferents comunitats és una contribució clau del nostre treball, ja que descriu com canvia el comportament perjudicial al llarg del temps i en diferents dominis.

La principal contribució d'aquesta tesi és la integració de l'aprenentatge incremental en tres marcs: FeDAL, LaMAL i CAL. FeDAL és un conjunt de xarxes neuronals dissenyat per gestionar dades tabulars amb desequilibri en la distribució de classes. LaMAL utilitza un model de llenguatge preentrenat per gestionar dades de text en tasques de classificació binària i multietiqueta. CAL combina adaptadors amb un model de llenguatge preentrenat per gestionar dades de text i incorpora informació de diferents fonts per millorar el rendiment en una comunitat destí. Per a la interpretabilitat, FeDAL utilitza l'algorisme LIME, proporcionant explicacions per a les característiques rellevants en un conjunt de classificadors. LaMAL i CAL utilitzen l'algorisme de Gradients Integrats, dissenyat específicament per comprendre el funcionament intern dels models basats en transformadors (en el nostre cas, models de llenguatge preentrenats). Avaluem aquests marcs en un petit conjunt de dades d'interacció de la tasca d'edició d'articles de la Viquipèdia, on la norma prohibeix el vandalisme. Els resultats mostren que FeDAL, LaMAL i CAL poden detectar violacions de normes amb dades desequilibrades i canvis en les perspectives de la comunitat, alhora que expliquen la sortida dels diferents models d'aprenentatge automàtic. A més, els resultats de CAL indiquen que la incorporació de dades de diferents comunitats millora el rendiment del model en una nova comunitat destí.

Finalment, realitzem un estudi d'usuaris per avaluar si diferents dissenys d'interpretabilitat poden influir en les valoracions dels usuaris en apreciar frases que contenen expressions d'odi. Tot i que el nostre anàlisi estadístic indica que cap dels dissenys d'interpretabilitat influeix significativament

en les opinions dels participants sobre la classificació de les expressions d'odi, el nostre anàlisi qualitatiu proporciona informació valuosa sobre l'impacte de la incorporació de la interpretabilitat d'aprenentatge automàtic en la nostra solució. Concretament, en comprendre de manera més precisa com un model d'aprenentatge automàtic identifica violacions de normes, els usuaris poden proporcionar informació rellevant quan noten discrepàncies amb les sortides del model. Per tant, aquesta informació es pot considerar com una eina per desencadenar una actualització del model quan sigui necessari. A més, les dades d'interpretabilitat poden oferir informació valuosa als enginyers en avaluar el comportament del model d'aprenentatge automàtic més enllà de les mètriques de rendiment tradicionals.

Paraules clau: violació de normes, deriva de concepte, mètode de conjunt, aprenentatge incremental, interpretabilitat

Resumen

Las comunidades en línea establecen normas para regular las interacciones entre agentes (los miembros de una comunidad) con orígenes y puntos de vista diferentes. Un sistema normativo que tiene como objetivo regular el compartimento de dichos miembros se enfrenta al desafío de adaptarse continuamente para identificar lo que constituye una violación de las normas establecidas, al mismo tiempo que las perspectivas de la comunidad evolucionan, como por ejemplo, los cambios en las características de una violación, o el surgimiento de nuevas clases de violación. Para abordar este desafío, la presente tesis propone una solución que emplea técnicas de aprendizaje automático para equipar a los sistemas normativos a aprender de manera continua a partir de las interacciones y las reacciones de los miembros. Nuestra propuesta utiliza ejemplos de acciones etiquetadas como violaciones en un contexto que abarca múltiples tipos de datos, lo que posibilita formalizar esas acciones como un conjunto de características (dados tabulares) o como sentencias en formato textual. Esta tesis se centra específicamente en comunidades en línea que cuentan con pocos recursos o que han sido creadas recientemente, dominios en los que el aprendizaje se desarrolla en un contexto con dados limitados. En este sentido, nuestra solución se presenta como una metodología que puede incorporar datos de diferentes fuentes (comunidades) para mejorar el rendimiento de los modelos de aprendizaje automático en una nueva comunidad.

Además de identificar violaciones de normas, argumentamos que los sistemas normativos deben explicar los diversos puntos de vista manifestados en las comunidades, incluyendo evidencia sobre los elementos relevantes de una acción que se asocian con violaciones de normas, y cómo pueden diferir a la luz de la naturaleza evolutiva de las interacciones en línea. El análisis de las transformaciones en las perspectivas de las comunidades, tanto considerando una comunidad específica o entre diferentes comunidades, constituye una contribución clave de nuestro trabajo. Este enforque permite describir cómo se modifica el comportamiento perjudicial a lo largo del tiempo y en distintos dominios.

La principal contribución de esta tesis es la integración del aprendizaje incremental en tres marcos: FeDAL, LaMAL y CAL. FeDAL consiste en un conjunto de redes neuronales creadas para abordar el desequilibrio en la distribución de clases en datos tabulares. LaMAL emplea un modelo de lenguaje preentrenado para abordar tareas de clasificación binaria y de etiquetas mútiples considerando datos de texto. CAL combina adaptadores con un modelo de lenguaje preentrenado para abordar tareas con datos de texto, incorporando datos de diferentes fuentes con el fin de mejorar el rendimiento en una comunidad destino. Con respecto a la interpretabilidad, FeDAL utiliza el algoritmo LIME, el cual proporciona explicaciones para las características relevantes en un conjunto de clasificadores. LaMAL y CAL utilizan el algoritmo de Gradientes Integrados, creado específicamente para comprender el funcionamiento interno de los modelos basados en transformadores (en nuestro caso, modelos de lenguaje preentrenados). Evaluamos estos marcos en un pequeño conjunto de datos de interacción relacionados con la tarea de edición de artículos en Wikipedia, donde la norma prohíbe el vandalismo. Los resultados indican que FeDAL, LaMAL y CAL pueden detectar violaciones de normas en presencia de desequilibrios en la distribución de clases y cambios en las perspectivas de la comunidad, al mismo tiempo que explican las decisiones de los diferentes modelos de aprendizaje automático. Adicionalmente, los resultados de CAL sug-

ieren que la inclusión de datos de diferentes comunidades mejora la performance del modelo en una nueva comunidad destino con dados limitados.

Finalmente, llevamos a cabo un estudio de usuarios para evaluar el posible impacto de diferentes diseños de interpretabilidad cuando los usuarios evalúan oraciones que contienen expresiones de odio. A pesar de que nuestro análisis estadístico no revela influencias significativas de ninguno de los diseños de interpretabilidad en las opiniones de los participantes con respecto a la clasificación de las expresiones de odio, nuestro análisis cualitativo aporta información valiosa acerca de cómo la incorporación de la interpretabilidad del aprendizaje automático afecta nuestra solución. Específicamente, al comprender de manera más precisa cómo un modelo de aprendizaje automático identifica violaciones de normas, los usuarios pueden proporcionar información relevante cuando notan un discrepancias con las salidas del modelo. Por lo tanto, esta información puede considerarse como una herramienta para desencadenar una actualización del modelo cuando sea necesario. Además, los datos de interpretabilidad pueden ofrecer información valiosa para los ingenieros al evaluar el comportamiento del modelo de aprendizaje automático más allá de las métricas de rendimiento tradicionales.

Palabras clave: violación de normas, deriva de concepto, método de conjunto, aprendizaje incremental, interpretabilidad

Acknowledgments

I would like to thank my family for their support, not only throughout this PhD project but also over the course of my entire life.

A wholehearted thank you to Zuza for all the love during my stay in Barcelona. Meeting you has truly been the best thing that ever happened to me.

I am immensely grateful to my supervisors, Dr. Nardine Osman and Dr. Marco Schorlemmer, whose guidance and support were indispensable for the completion of this research. The countless discussions, insightful ideas, corrections, patience, and teaching have been invaluable. I cannot thank them enough for the honor of having them as my supervisors over the last four years.

My special gratitude goes to my supervisors during my research visit in New Zealand, Dr. Bastin Tony Roy Savarimuthu and Dr. Stephen Cranefield. Their valuable insights and teaching contributed significantly to the improvement of my research and the production of this thesis.

I extend my thanks to my fellow Ph.D. students and colleagues at the IIIA-CSIC and the University of Otago, who were incredible people throughout these years. Athina, Dimitra, Roger, Núria, Oguz, Dave, Zara, Abira, Rosa, Jithin, Jianglin, Jairo, Bjoern, Shuxian, Nieves, Jesus, and Daniel, your friendship and contributions have made this journey memorable.

The support and resources provided by the institute (IIIA-CSIC) were fundamental to my development as a scientist and the creation of this thesis. I would like to express my gratitude to the directors of the institute, the administrative staff, and the IT team for their unwavering support throughout my project. Your contributions were essential to the successful completion of this thesis.

This research was supported by the EU funded VALAWAI (# 101070930) and WeNet (# 823783) projects, the Spanish funded VAE (# TED2021-131295BC31) and Rhymas (# PID2020-113594RB-100) projects, and the Generalitat de Catalunya funded *Ajuts a grups de recerca de Catalunya* (# 2021 SGR 00754) project.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AUC-PR	Area Under the Curve of Precision-Recall
AUC-ROC	Area Under the Curve of the Receiver Operating Characteristics
CAL	Cross-Community Adapter Learning
DL	Deep Learning
DNN	Deep Neural Network
FeDAL	Feedback-Driven Adaptive Learning
FNN	Feed-Forward Neural Network
GAM	Generalized Additive Model
IG	Integrated Gradients
LaMAL	Language Model Adaptive Learning
LIME	Local Interpretable Model-Agnostic Explanations
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
PLM	Pre-Trained Language Model
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

Online communities represent dynamic social domains where community members (agents) with diverse backgrounds and points of view interact. These interactions occur in a context where normative systems can be used to establish norms (Jones & Sergot, 1994), aiming at specifying and regulating the relevant behavior of agents (human or software) as they engage in online activities. To illustrate the importance of these norms, consider the use case of article editing in Wikipedia, an online platform where diverse people connect in an open environment and their contributions are visible to the entire world (Wikipedia, 2023). In this instance, norms dictate how community members must add or edit content, ensuring the maintenance of articles following a consistent format that respects some predefined requirements. These requirements include adopting a proper writing style, avoiding editing wars, and not expressing hate speech.[1] Adherence to these norms intends to promote inclusivity[2] and coherence within the community while violating behavior leads to exclusion and harm of agents (McLean & Griffiths, 2019; Shmargad et al., 2022).

As an online community defines its relevant norms, the first main challenge tackled in this thesis is to detect violations as interactions unfold. Notably, for many of these norms, what constitutes a violation changes over time and across communities. Consequently, we address this challenge while adapting to the evolving community views on what constitutes violating behavior, i.e., how a community understands the elements of an action that characterize detrimental interactions (Allison et al., 2019). This evolution represents the shift in communities' views leading to the prohibition of previously accepted actions, like the prohibition of certain terminologies, or the emergence of new violation classes, such as identifying a new hate speech target. For instance, what is considered hate speech may change rapidly as new members are incorporated and interactions unfold. Consider the "N-Word" (Rahman, 2012). It may be viewed differently as more African Americans join the community and begin to salute each other using this term. We argue that a normative system deployed to govern these interactions must adapt to the current view of the community. To accomplish that, normative systems may employ Machine Learning (ML) models that continuously learn what constitutes norm violations. This process becomes particularly complex considering low-resource (or newly created) online communities, where learning norm violations occur in a context with limited labeled data since we aim to learn as soon as new community views start emerging (Huang et al., 2022).

The normative systems' ability to continuously learn what constitutes a norm violation is essential because of the community-specific nature of behaviors like discrimination, hate speech, and cyberbullying. Thus, constantly identifying these behaviors is critical for the success of online

[1] Disclaimer: This thesis presents content (offensive language) that may disturb some audiences.

[2] Inclusivity refers to creating an environment where people are integrated by acknowledging and respecting their diverse characteristics (Arora & Patro, 2021). In our case, this includes promoting an online environment that actively prohibits discrimination based on gender, ethnicity, race, or other individual or group attributes.

communities. In this scenario, the manifestation of such negative behavior tends to vary across online communities and over time, and this behavior causes significant harm to individuals and negatively impacts the community experience in online platforms as a whole (Gray, 2018; Risch & Krestel, 2020). 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. For example, credit card transaction fraud may vary due to seasonal behavior patterns and different fraud strategies (Lebichot et al., 2021).

Interesting approaches to identify norm violation in online communities have already been proposed, with applications to Wikipedia (Anand & Eswari, 2019; Freitas dos Santos et al., 2022b; West & Lee, 2011), software engineering communities (Cheriyan et al., 2017, 2021), Reddit (Chandrasekharan et al., 2019) and other communities (Karim et al., 2021; Mollas et al., 2022; Xiang et al., 2021). 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. We address this limitation by proposing an approach that handles the interactions of an online community as a stream of actions with an imbalanced class distribution and the presence of concept drift (q.v. Section 2.2). We argue that we should accommodate for imbalanced class distribution because violating behavior is usually not as common as non-violating behavior, resulting in an imbalance in the training data, i.e., the stream of actions. Moreover, we should accommodate for concept drift, which represents changes in community views of what constitutes a norm violation. The novelty of this work is addressing the dynamic nature of norm violations in online communities by incorporating community members' feedback as the ground truth to the learning mechanism, which allows the learning mechanism to continuously adapt to changes in norm-violating behavior over time and across communities.

The second challenge addressed in this thesis is to learn what constitutes norm violations using different types of data input. Specifically, actions are represented by text data, such as the text of a Wikipedia edit, or formalized as a set of features, such as those representing the frequency of profane words in a Wikipedia edit.[3] Table 1.1 highlights sample classification tasks where the data type specifying online actions may vary depending on the studied domain. While some tasks require a set of features to describe an action (e.g., fraud discovery, misbehaving detection), others must handle the raw text data (e.g., hate speech detection, adversarial attacks) describing the action due to the complexity of defining action features in these cases. For instance, fraud discovery and misbehaving detection formalize an action as a set of features like user engagement and user-user interaction. Identifying deception writing styles might require mapping the input to linguistic-based features. These tasks usually benefit from tabular-related approaches. However, cases like hate speech detection, tackling the spread of fake news over social media, and adversarial attacks need solutions that handle natural language sentences directly. Given that different domains might have different types of data available (text and tabular), the main contribution of this thesis is to build a multi-scenario approach to continuously learn to detect norm violations for tabular and text-based domains since these cover a variety of classification tasks. Concretely, this thesis evaluates our multi-scenario approach within the context of learning norm violations in Wikipedia article edits. In this use case, the action of editing an article can be represented by the text of the edit or by a set of features defined by Wikipedia to describe that edit.

The third challenge of this thesis is to design interpretable learning models. We argue that normative systems can benefit from interpretability tools so that community members understand the different community views and their evolution. This implies that our proposed solution must be capable of presenting evidence on the relevant elements of an action (words in text data or features in tabular data) associated with detecting norm violation and how these elements may change in light of the evolving nature of what constitutes norm violation in online interactions. Interpretability is essential in our work because it equips our proposal with the means to address four requirements

[3]This type is also known as tabular data.

Set of Features
Fraud discover (Anowar & Sadaoui, 2021; Lebichot et al., 2021)
Misbehaving detection (Islam et al., 2022; T. C. Li et al., 2017)
Identify deception writing styles (Afroz et al., 2012; Shojaee et al., 2013)
Fake reviews detection (Barbado et al., 2019; Mohawesh et al., 2021)
Identify authors of violations (Peng et al., 2016; Skopik & Pahi, 2020)

Text
Hate speech detection (Cheriyan et al., 2021; Risch & Krestel, 2020)
Fake news detection (Kaliyar et al., 2021; Mridha et al., 2021; Szczepański et al., 2021)
Adversarial attacks (Hossam et al., 2021)
Identify AI-generated reviews (Adelani et al., 2020; Mitrović et al., 2023)
Style change detection (Strøm, 2021; Zangerle et al., 2020)

Table 1.1: Examples of classification tasks that benefit from solutions that handle tabular (set of features) and/or text datasets. Specifically, we focus on violating behavior in online interactions.

in our domain. First, it enables our system to adhere to the transparency principle of responsible artificial intelligence (Arrieta et al., 2020), enhancing community members' comprehension of what the ML model learns about what the community considers as non-acceptable behavior. Second, it enables engineers to debug the model and better comprehend the relevant elements of an action that the ML model prioritizes. The aim is to prevent drawbacks associated with black-box models, like attributing relevance to elements of an action that the community does not consider relevant to violating behavior. Third, it allows the analysis of how community views change over time or across communities. A better understanding of how detrimental behavior evolves is a key contribution of our work. Fourth, it sets the stage for a future proactive, collective, and potentially collaborative feedback elicitation process. When community members think their views are not aligned with what the model presents as the community's view, this process allows them to provide additional feedback collectively. This feedback will be on actions that community members believe the model has wrongly classified (violation or non-violation), aiming to correct the model's output. Moreover, in cases where there is a strong discrepancy between different members' views, a deliberation process enables members to argue among themselves (informed by interpretability results) about the aspects of the model's output that affect the disagreement. Then, the community members can collaboratively agree on the feedback to improve the model's performance.

The literature presents various applications of interpretability for ML models. Ribeiro et al. (2016) and Sundararajan et al. (2017) describe the two interpretability algorithms explored in this thesis (q.v. Section 2.5), which enable our solution to identify relevant elements of an action that influence the output of ML models. Moreover, studies on agents generating explanations for norm violations (Agrawal et al., 2022), and explainability in sentiment analysis, online review, and age prediction (Arora et al., 2022; Chu et al., 2020; Schuff et al., 2022) demonstrate how interpretability can enhance agents' interactions in different contexts. Notably, these approaches differ from our solution, as they do not address how detrimental behavior in online communities changes over time and across domains. Our solution accomplishes that by continuously explaining the output of models that detect norm violations.

Finally, the remainder of this chapter presents the contributions of this thesis (q.v. Section 1.1), followed by a description of its structure (q.v. Section 1.2).

1.1 Contributions

This thesis proposes a multi-scenario approach to continuously learn what constitutes a norm violation from past interactions and user feedback. The multi-scenario context involves learning norm violations from tabular and text data. In other words, we achieve our goal by using examples of behavior labeled as violations, formalized as a set of features, or collected as text sentences (such as the examples in Table 1.1). Section 1.1.1 presents our main contribution, the three ML frameworks (FeDAL, LaMAL, and CAL) that compose our multi-scenario approach for learning what constitutes norm violations. Section 1.1.2 presents this thesis' second contribution and the impact of our work in normative systems. Specifically, it highlights how normative systems can continuously learn what constitutes a norm violation from users' feedback by employing the proposed ML frameworks. Our third contribution demonstrates the effectiveness of our work through a real-life use case, which is the editing of Wikipedia articles (q.v. Section 1.1.3). Next, Section 1.1.4 shows our contribution regarding the interpretability of ML models, focusing on understanding the words in a text or the set of features usually associated with norm violation detection. Subsequently, Section 1.1.5 presents our fifth contribution, a user study assessing the impact of different interpretability layouts on users' evaluations of sentences labeled as hate speech. Lastly, we provide links to our code and data resources to facilitate transparency, reproducibility, collaboration, and promotion of good research practices, constituting our final major contribution (q.v. Section 1.1.6).

1.1.1 Machine Learning Frameworks

To create our multi-scenario approach, we employ incremental learning in three frameworks: Feedback-Driven Adaptive Learning (FeDAL) (q.v. Chapter 3), Language Model Adaptive Learning (LaMAL) (q.v. Chapter 4), and Cross-Community Adapter Learning (CAL) (q.v. Chapter 5). These frameworks are responsible for the continuous learning process of norm violations within normative systems, considering different challenges in norm violation detection that each framework addresses. The conceptual framework in Figure 1.1 illustrates the relationship between the components of our solution. Specifically, we present this conceptual view using four sections: "Data Scenario," "Learning Techniques," "ML Model," and "Interpretability."

The "Data Scenario" section presents the two types of data input handled by our multi-scenario approach. Specifically, "Text" refers to actions formalized as textual data, while "Set of Features" refers to actions formalized in a tabular format. In this context, a single data point corresponds to an individual action executed within a community, such as a single article editing in Wikipedia. In contrast, an imbalanced data block contains a collection of actions, such as all article edits in Wikipedia that occurred on a given day. The inherent imbalance in this data block is attributed to addressing the challenge of norm violation detection. In this case, violating behavior does not happen as often as regular behavior. Consequently, these blocks contain different amounts of instances for each class.

The "Learning Techniques" section presents that our frameworks (FeDAL, LaMAL, and CAL) are equipped to train ML models using imbalanced data blocks by employing the mini-batch learning approach for both text and tabular scenarios (q.v. Section 2.3.2). However, only FeDAL can train ML models using a single data point by employing the online learning approach. This distinction occurs due to the variations in size and complexity of ML models used to solve text and tabular classification tasks. Concretely, FeDAL needs only to implement an ensemble of Feed-Forward Neural Networks (FNNs) to solve tabular classification tasks ("ML Model" section), which requires a less complex ML model and handles class distribution imbalance within tabular scenarios. Thus, FeDAL can implement mini-batch and online learning, where the first offers more stability in the training process and better performance. In contrast, online learning is important when the community requires updating the ML model as soon as a single data point is made available.

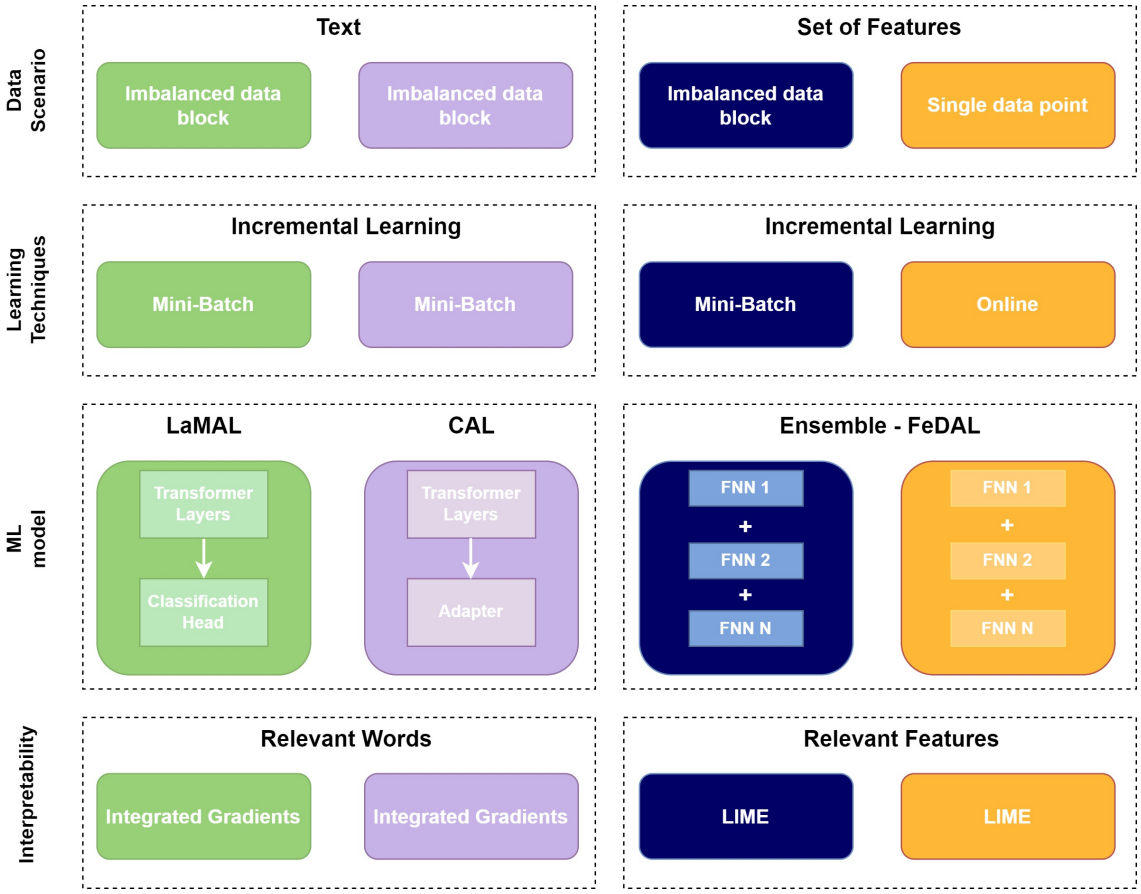


Figure 1.1: A conceptual framework of our multi-scenario approach. In the ensemble, N indicates the number of classifiers. FNN refers to a Feed-forward Neural Network model.

Regarding text classification tasks, LaMAL and CAL require employing transformer-based models ("ML Model" section), specifically the Pre-Trained Language Models (PLMs) (q.v. Section 2.4). PLMs allow LaMAL and CAL to process text sentences directly and reduce the amount of work in the featurization step (creating a set of features). Moreover, they do not require an ensemble of classifiers. Instead, they can handle imbalanced data blocks by encoding language structures due to their size and undersampling of the majority class, i.e., the class representing the instances (e.g., violation, regular) that happen more often. In this thesis, another advantage in text classification tasks is identifying specific violation classes due to sufficiently informative content, i.e., the presence of words that are usually associated with different violations, like the "N-Word" (Rahman, 2012) that can be used to manifest racism. Thus, in addition to solving binary classification tasks (only identifying violation or regular behavior), LaMAL and CAL can solve multi-label classification tasks (identifying more than one violation class that happens simultaneously).

The "ML Model" section also depicts a significant difference between the architectures of LaMAL and CAL. LaMAL integrates a classification head with the transformer layers, while CAL adopts an adapter-based approach (q.v. Section 2.4).[4] This architectural difference is important because CAL can handle the emergence of new violation classes by dynamically creating adapters as interactions unfold. In contrast, the classification head in LaMAL requires the predefinition of violation classes when it is initially created. Moreover, CAL equips our multi-scenario approach with the ability to incorporate data from different sources (communities) to improve the performance of ML models in

[4]Adapters are neural networks with a small proportion of the number of parameters present in the full PLM.

a new target community with limited labeled data. This ability is crucial for low-resource or newly created online communities, where data formalizing interactions between community members is scarce.

Lastly, the "Interpretability" section shows that FeDAL adopts the Local Interpretable Model-Agnostic Explanations (LIME) algorithm as the interpretability tool (q.v. Section 2.5.1). This choice is supported by LIME's model-agnostic nature, which allows us to obtain the explanation for the relevant *features* when using an ensemble of FNN. This approach also ensures the generation of explanations suitable for human understanding. In contrast, LaMAL and CAL adopt the Integrated Gradients (IG) algorithm to obtain the explanation for the relevant *words* when using transformer-based models (q.v. Section 2.5.2). IG is specifically built to understand the inner workings of transformer-based models, allowing our frameworks to obtain the words with the highest influence on the classification output. Like LIME, IG generates explanations that are simple enough for human understanding.

Next, we provide more details on each of the three frameworks.

FeDAL

Feedback-Driven Adaptive Learning (FeDAL) is an ensemble (q.v. Section 2.1) of Feed-forward Neural Networks (FNNs) equipped with an incremental learning approach (q.v. Chapter 3). FeDAL is specifically built to handle class distribution imbalance within tabular tasks. Additionally, FeDAL incorporates a replication by oversampling approach, which limits the ensemble size, restraining the computational power required to process incoming data points. By combining multiple ML models through a voting scheme, FeDAL can learn in a context with imbalanced datasets while obtaining superior performance compared to single components, as the different models in the ensemble compensate for the errors of a single model (we present an ablation study with this conclusion in Section 3.6.3).

To accommodate incremental learning, the FeDAL algorithm employs Stochastic Gradient Descent (SGD) to facilitate parameter updates as new interactions unfold (q.v. Section 2.3.2), effectively allowing the incorporation of new community views on norm violation. Moreover, with the SGD implementation, FeDAL avoids executing redundant computations, which makes this option more efficient when compared to batch learning (empirical results in Section 3.6).

In practice, to handle tabular datasets, FeDAL represents an action as a set of features described by the tuple (X, y), in which X is the set of features of an action and $y \in 0, 1$ is its class label, with 0 denoting regular behavior and 1 denoting norm violation. As such, this component allows FeDAL to provide community members with the features that contribute the most to the ML model's decision, which is obtained through the LIME algorithm (q.v. Section 2.5.1). In this scenario, we learn norm violations by formalizing an action (an article edit considering the Wikipedia use case) as a set of features provided by the community, e.g., number of profane words, occurrences of alphanumeric characters, etc. (complete set of features in Section 3.5).

To evaluate the performance of our framework within the Wikipedia use case, we present the experiments that describe the implementation of incremental learning techniques to train the base classifiers, namely mini-batch learning and online learning (q.v. Section 3.5). Specifically, we compare these techniques against batch learning, aiming to understand where they differ and the drawbacks present in each approach. Moreover, we describe experiments to identify the features of an action that influence norm violation detection, using LIME as the interpretability tool.

Our results indicate that FeDAL can continuously learn to detect norm violations in an online community (Wikipedia article editing task) 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) (q.v. Section 3.6). The ablation study results also indicate how the ensemble

outperforms a single model approach and the advantages of replication by oversampling. Lastly, interpretability analysis describes features that contribute to the detection of norm violations.

Resulting Publications:

- Ensemble and Incremental Learning for Norm Violation Detection. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS).*
- Is This a Violation? Learning and Understanding Norm Violations in Online Communities. *Artificial Intelligence (AIJ).*

LaMAL

Language Model Adaptive Learning (LaMAL) incorporates PLMs into an incremental learning approach (q.v. Chapter 4). The advantage of incorporating PLMs into LaMAL is that it equips our framework with a powerful language representation to tackle Natural Language Processing (NLP) tasks. Concretely, this integration enables our framework to process text sentences directly and reduce the amount of work in the featurization step (creating a set of features). In contrast to FeDAL's incorporation of FNNs, PLMs do not require an ensemble of classifiers. Instead, they can handle imbalanced datasets by encoding language structures due to their size and undersampling of the majority class. Furthermore, to learn specific classification tasks, like detecting hate speech, PLMs allow for superior performance by fine-tuning only the classification head, a Neural Network (NN) integrated on top of the pre-trained layers of the model.

Our multi-scenario approach incorporates LaMAL to solve binary (regular or violation) and multi-label (several violation classes) text classification tasks. As our real-life use case describes, a single action may comprise multiple violation classes. Thus, unlike FeDAL, LaMAL can identify specific violation classes due to sufficiently informative text content, i.e., the presence of words in a text sentence that manifests different violations. In conjunction with FeDAL, this capability enables our solution to effectively adapt to the various domains in online communities and their different data types, namely tabular and text data types (q.v. Table 1.1).

In addition to learning norm violations, LaMAL can provide information on the words that indicate violating behavior as perceived by ML models. To accomplish that, LaMAL incorporates IG to explain the behavior of PLMs. IG provides information on the relevant words related to norm violation, determining their relevance across different violation classes. Specifically, for binary classification tasks, IG presents the words that contribute the most to the ML model decision of classifying an article edit as a violation. For multi-label classification tasks, IG presents each word as relevant to 0 (neutral word), 1, or more violation classes.

Our experiments evaluate the article edits in Wikipedia by comparing two PLMs, namely DistilBERT (Sanh et al., 2019) and RoBERTa (Y. Liu et al., 2019) (q.v. Section 4.3). In the context of the multi-label scenario, the dataset contains six different violation classes. Results show how the architecture of each PLM impacts the understanding of violating behavior, with different relevant words obtained from DistilBERT and RoBERTa (q.v. Section 4.4). Furthermore, our findings indicate that LaMAL can continuously learn to detect violations considering binary and multi-label text classification.

Resulting Publications:

- A Multi-scenario Approach to Continuously Learn and Understand Norm Violations. *Autonomous Agents and Multiagent Systems (JAAMAS).*

CAL

Cross-Community Adapter Learning (CAL) incorporates PLMs and adapters into an incremental learning framework (q.v. Chapter 5). CAL equips our multi-scenario approach with the ability to incorporate data from different sources (communities) to improve the performance of ML models in a new target community with limited labeled data. This ability is crucial for low-resource or newly created online communities, where data formalizing interactions between agents (community members) is scarce. Specifically, our solution uses data from three different communities as the initial step to define violating behavior in the article editing task, as different violation classes in our context contain only a few instances (q.v. Section 5.2).

Compared to LaMAL, incorporating adapters into our framework provides the following advantages. First, each adapter is created for a specific violation class, facilitating an efficient fine-tuning process that updates the new community's views exclusively on selected violation classes (q.v. Algorithm 5.1). In contrast, LaMAL updates the PLM without selectively updating model parameters based on specific violation classes. Second, CAL can dynamically create adapters as new violation classes emerge. This is particularly important for cases where a community did not initially define all potential violation classes that may occur during interactions between community members. Consequently, CAL can effectively identify new violation classes based on feedback, such as identifying a new hate speech target. In contrast, LaMAL requires predefined violation classes when the model is initially created. Third, it allows CAL to tackle catastrophic forgetting due to interference between different violation classes,[5] which might be an issue in fine-tuning a complete PLM, such as LaMAL's fine-tuning process. Lastly, since adapters are smaller than the full PLM, CAL's training process is faster and more efficient. Specifically, the continuous update of smaller neural networks allows for greater robustness to handle over-fitting and reduced sensitivity to changes in learning rates. In this context, while we continuously update the adapter weights on our target data, the transformer layers are used only for language representation, keeping the original PLM parameters frozen. Differently, LaMAL updates the PLM.

In addition to continuously learning norm violations, CAL employs the IG algorithm to analyze the distinct views manifested in online communities, i.e., the different terms community members employ to express violating behavior. Consequently, CAL represents the first proposal of an interpretable adapter framework designed to learn and understand the differences in norm violations between communities. This ability to analyze the communities' view changes within a specific domain or across different communities is a key contribution of our solution as it describes how detrimental behavior changes over time and across domains.

Our experiments use DistilBERT with an adapter and data from three communities as the initial training step (q.v. Section 5.2). Our results indicate that CAL can continuously learn norm violations, adapt to evolving communities' views, and explain the differences in norm-violating behavior for different communities based on community members' feedback (q.v. Section 5.3).

Resulting Publications:

- Cross-community Adapter Learning (CAL) to Understand the Evolving Meanings of Norm Violation. In *International Joint Conference on Artificial Intelligence (IJCAI).*

1.1.2 Normative Systems Continuously Learn From Users' Feedback

Figure 1.2 illustrates our contribution to normative systems by introducing a workflow for continuously learning what constitutes norm violations in online communities. Specifically, this workflow

[5]Catastrophic forgetting refers to the loss of knowledge acquired from previous classification tasks when learning new information (Pfeiffer et al., 2020). In our context, preventing the loss of information about a violation class when training with data from a different violation class.

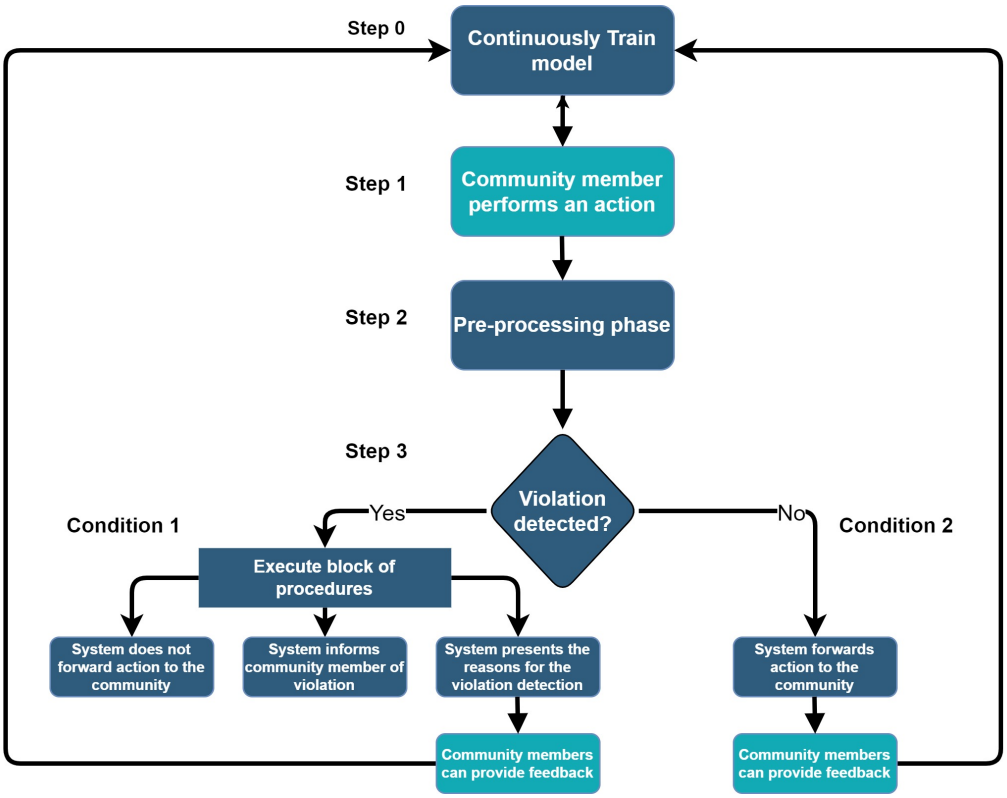


Figure 1.2: The workflow describing the continuous learning process for normative systems.

enables normative systems to adapt to the evolving community members' views using their feedback as the ground truth. This adaptive ability concretely implements our vision that a system's understanding of norm violation should reflect the understanding of its users (in our case, community members).

In online community domains, a continuous learning process becomes particularly crucial since online interactions dynamically change as people interact and new members are incorporated into the community. In this context, our workflow allows normative systems to address two key challenges related to changes in norms: 1) the evolving community view of what constitutes a norm violation, such as the shift in communities' views leading to the prohibition of previously accepted terminology; and 2) the emergence of new violation classes, such as identifying a new hate speech target. To tackle these challenges, we employ incremental learning to continuously process incoming interactions as a data stream while discarding previous data that may contain outdated information about what constitutes norm violations (q.v. Section 2.3.2). In this thesis, incorporating incremental learning involves mini-batch and online learning. While mini-batch builds small data blocks to train ML models, online learning updates a model's parameters as soon as a new interaction instance is made available (Hoi et al., 2021).

Next, to illustrate the implementation of our workflow in normative systems, we detail the steps as depicted in Figure 1.2. The initial step, Step 0, involves continuously training ML models.[6] The training process starts by using data blocks or single data points, and upon completion of the first training iteration, 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

[6]Specifically, in our approach, the ensemble (FeDAL) and the transformer-based (LaMAL and CAL) classifiers.

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 (back to 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 what constitutes norm violations through continuous model training (back to Step 0).

Resulting Publications:

- A Multi-scenario Approach to Continuously Learn and Understand Norm Violations. *Autonomous Agents and Multiagent Systems (JAAMAS).*

1.1.3 A Real-Life Use Case With Wikipedia's Article Edits

To demonstrate the impact of our multi-scenario approach in a real-life online community, we explore the article edit task in Wikipedia (Wikipedia, 2023). This use case is relevant because it allows us to evaluate our solution considering three key challenges in low-resource (or newly created) online communities: 1) learning using a limited labeled dataset;[7] 2) learning changes in community members' views on what constitutes norm violations; and 3) learning using the formalization of violation instances across different data types.[8] Experiments addressing these challenges are described in Sections 3.5, 4.3, and 5.2. The findings indicate that our multi-scenario approach can learn to identify norm violations in online communities while considering these challenges.

Furthermore, our contribution to real-life use cases extends beyond the use of Wikipedia data. Specifically, our multi-scenario approach integrates cross-community learning, which involves using data from a source community as an initial training step in a target (new) community. Including this ability in our solution not only demonstrates the adaptability of normative systems to different views within a single community but also shows its adaptability across diverse online community domains. We conduct experiments in Section 5.2 exploring three different communities and demonstrate that our multi-scenario approach can learn to identify norm violations while incorporating data from different sources.

To understand our use case, we describe the details of Wikipedia's norms. These norms include 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 thesis, the actions we focus on are those performed by the community members (the Wikipedia users), which are the attempts to edit articles. The norm we assess says: "Do not engage in vandalism behavior" (which we refer to as the "no vandalism" norm). Wikipedia provides data on what edits are marked as either regular or violating the "no vandalism" norm, where the Amazon Mechanical Turk (MTurk) is responsible for the annotation process. At least three people (Mechanical Turks) evaluate every article edit, deciding whether the edit violates the "no vandalism" norm or not.

[7] Although Wikipedia is known for its extensive content repository, the labeled dataset used in this thesis has a limited size, comprising approximately 30.000 article edits labeled either as violation or not.

[8] Text sentences and tabular (set of features), due to their relevance in various violation detection tasks (refer to Table 1.1 for an overview of tasks in which these data types are relevant).

In addition to the MTurk annotation, the author of this thesis further annotates each violation instance, categorizing these into six different hate speech classes. In this part of the labeling process, we focus only on violations of the "no hate speech" norm, as this represents a complex and particularly harmful violation within online interactions.[9] The six violation classes are: Swear, Insult and Ableism, Sexual Harassment, Racism, LGBTQIA+ Attack, and Misogyny. It is worth mentioning that a single article edit could exhibit multiple violation classes simultaneously. For instance, an article edit can contain text that shows swearing terms and expresses hatred towards the LGBTQIA+ community. Next, we compile a sample of Wikipedia article edits considered hate speech. "[INDIVIDUAL's NAME]" is used to anonymize real people's identities.

1. *...he was the mother fuckin dom...*
2. *...this is wiki not a forum for retards...*
3. *...the big lipped,hairbraned,egotistical dirty "N-Word".*
4. *[INDIVIDUAL's NAME]* also sucks dick for features.
5. *[INDIVIDUAL's NAME] was a super mega bitch and she kill the...*

Resulting Publications:

- 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.*
- A Multi-scenario Approach to Continuously Learn and Understand Norm Violations. *Journal of Autonomous Agents and Multiagent Systems (JAAMAS).*

1.1.4 Black-Box Model Interpretability

In addition to its continuous learning of what constitutes norm violations, our multi-scenario approach must provide insights into the decision-making process of ML models (e.g., an ensemble of FNNs, the PLMs). Therefore, we incorporate interpretability to obtain explanations about a model's inner workings. Here, we are interested in understanding the words in a text or the set of features usually associated with a norm violation. Concretely, incorporating interpretability allows us to:

- Analyze how detrimental behavior evolves over time within a single community (q.v. Section 3.6). Specifically, interpretability allows us to understand the differences in an ML model's assignment of relevance values to features before and after introducing changes in community views while interactions unfold;
- Compare different learning approaches (q.v. Section 3.6). Specifically, interpretability allows us to understand the impact of batch, mini-batch, and online learning when updating the parameters of the ensemble of FNNs with the FeDAL framework. By employing different learning approaches, we may obtain different relevance values for the same norm violation;
- Compare two PLMs (q.v. Section 4.4). Specifically, interpretability allows us to understand the impact of different architectures when updating the parameters of the base PLMs, DistilBERT and RoBERTa, that compose the LaMAL framework (q.v. Section 4.3);

[9] Future work shall focus on creating more annotation classes for other norms, such as those related to using a proper writing style.

- Analyze how detrimental behavior changes across communities (q.v. Section 5.3). Specifically, interpretability allows us to understand the diverse views manifested in online communities when employing cross-community learning with the CAL framework.

We use different interpretability algorithms depending on the framework. Specifically, FeDAL uses the LIME tool to explain the ML model's behavior (q.v. Section 2.5.1), and since LaMAL and CAL use PLMs to solve violation detection tasks, they employ the IG algorithm to explain their model behaviors (q.v. Section 2.5.2).

Resulting Publications:

- Is This a Violation? Learning and Understanding Norm Violations in Online Communities. *Artificial Intelligence (AIJ).*
- A Multi-scenario Approach to Continuously Learn and Understand Norm Violations. *Journal of Autonomous Agents and Multiagent Systems (JAAMAS).*
- Cross-community Adapter Learning (CAL) to Understand the Evolving Meanings of Norm Violation. In *International Joint Conference on Artificial Intelligence (IJCAI).*

1.1.5 User Study

We conduct a user study to assess the impact of interpretability and whether different interpretability layouts can influence users' views when evaluating sentences containing hate speech, i.e., sentences that violate the norm prohibiting hate speech (q.v. Chapter 6). This study aims to provide empirical evidence regarding the effective use of ML model interpretability and to investigate how presenting interpretability information using different layouts can help mitigate violating behavior online. Specifically, we consider three interpretability layouts, which are different manners to present the words that contribute to the ML model's output:

- Local interpretability (q.v. Figure 6.1a) - it presents the impact of individual words (relevance score) on identifying norm violations in a specific text sentence.
- List with the sum of relevance scores (q.v. Figure 6.1b) - it lists the sum of relevance scores for the words present in the entire dataset used during training. This layout provides an overview of general words relevant to identifying norm violations.
- A combination of both - it describes local and the sum of relevance scores together.

Our user study (an online questionnaire) asks participants how their views change when presented with interpretability results from each of the three layouts. We compare these layouts against a baseline condition where participants only interact with a text. Their task is to evaluate a sentence containing hate speech. To assess the impact of interpretability, we design a questionnaire for each layout, consisting of 20 tuples, each with two questions. The questionnaire first presents a sentence without interpretability data. Next, participants re-evaluate the same sentence, this time with interpretability information. Concretely, each sentence is evaluated twice, once without interpretability data and once with it.[10] The participant must choose whether they agree or disagree with classifying a given sentence using the 7-point Likert scale.

[10]For a comprehensive insight into the questionnaires, Annex E provides additional examples of the different layouts.

The study comprises both within-subject and between-subject analyses (q.v. Section 6.2).[11] To obtain the estimate of the participants' confidence rating while evaluating a text sentence as hate speech, we employ the Generalized Additive Model (GAM) (q.v. Section 2.6).[12] Furthermore, we include a qualitative evaluation where participants' comments complement our understanding of the statistical findings (q.v. Section 6.2.3).

In conclusion, our statistical analysis indicates that none of the interpretability layouts significantly influence participants' views regarding the classification of hate speech, specifically violations containing misogynistic and racist content. Despite this outcome, our findings contribute to a current discussion in the literature, providing empirical evidence on the limited impact of ML interpretability in the human decision-making process. Based on participants' comments, the qualitative analysis provides two key insights to comprehend why interpretability layouts do not significantly influence participants' views. First, the participants' familiarity with detecting hate speech, including the presence of explicit hate speech words, appears to diminish the significant influence of interpretability information on their decisions. Second, the alignment between the ML model's output and participants' views, where interpretability results identified relevant words similar to those already considered by the participants, further diminishes the influence of interpretability layouts. However, our qualitative analysis also highlights the positive aspects of incorporating interpretability into our solution. First, interpretability enhances people's understanding of words relevant to the ML model's output, effectively enabling them to provide corrective feedback in cases of discrepancies. This is especially important in contexts where new instances of hate speech may emerge over time within a single community or with the inclusion of data from diverse communities (cross-community learning). Second, interpretability layouts provide insights into evaluating the ML model's behavior beyond traditional performance metrics (e.g., recall, F1-score, and precision), especially in scenarios where understanding the rationale behind an ML model's decision is essential for optimal implementation.

Resulting Publications:

- Can Interpretability Layouts Influence Users' Views of Violating Sentences? *To submit (2024).*

1.1.6 Code and Data

In this link: https://bitbucket.org/thiago-phd/, we make our open-source code for the three frameworks available, including the data used for training and validation. Additionally, we have the code used for the statistical analysis and data of the user study. The goals of this initiative are: 1) to facilitate transparency and reproducibility. This allows other researchers to ensure the frameworks correctly implement the described algorithms. Additionally, data access enables replication of experiments, offering reliance on our findings; 2) to enable collaboration within the academic community. Researchers can use these resources to build on top of the methodologies present in this thesis, advancing their own investigations; and 3) to promote good research practices in the community, advocating for ethical and responsible research conduct.

[11] Within-subject refers to assessing whether the information of a given interpretability layout influences the participant's classification. This involves using the same participant group. Between-subject refers to assessing whether one of the interpretability layouts has a more significant impact on the participant's classification ratings than another, which uses different participant groups.

[12] GAM enables us to capture the relationship between variables using nonlinear functions to model the response data. Concretely, this allows us to estimate the participants' confidence in classifying a sentence as hate speech.

1.2 Thesis Structure

The remainder of this thesis opens with Chapter 2, which presents the relevant literature that our proposal builds upon, followed by introducing our three different learning mechanisms in Chapters 3–5. A user experiment assessing the impact of interpretability is presented in Chapter 6. A study of the state of the art and its comparison with our proposal is presented in Chapter 7, before concluding with Chapter 8. Each of these chapters is presented in more detail next.

Chapter 2 delves into the background literature, presenting the base concepts upon which this thesis is built. First, we introduce the ensemble strategy to deal with the imbalanced nature of the dataset (q.v. Section 2.1). Then, we describe concept drift to accommodate for changes in community views about what constitutes norm violation, including the emergence of new violation classes that forms a sub-type of concept drift (q.v. Section 2.2). Third, we describe the training approaches to continuously learning norm violations (q.v. Section 2.3). Fourth, we introduce the Pre-Trained Language Model (PLM) and the adapters to tackle text classification tasks (q.v. Section 2.4). Then, we present our two interpretability tools, the Local Interpretable Model-Agnostic Explanations (LIME) and Integrated Gradients (IG) (q.v. Section 2.5). Lastly, in Section 2.6, we present the Generalized Additive Model (GAM) that helps us analyze the user study results.

In Chapter 3, we present the Feedback-Driven Adaptive Learning (FeDAL) framework, which combines ensemble with incremental learning approaches to continuously learn norm violations from tabular data. We start by presenting three different algorithms investigated in FeDAL, followed by a computational complexity analysis. We then describe the experiments and discuss their results.

The Language Model Adaptive Learning (LaMAL) framework is introduced in Chapter 4, which is the second component of our multi-scenario approach responsible for continuously learning norm violations by handling text datasets. We present LaMAL's ability to solve binary (regular or violation) and multi-label (several violation classes) *text* classification tasks. Then, we describe the experiments and discuss their results.

Chapter 5 presents the Cross-Community Adapter Learning (CAL) framework, which combines PLMs and adapters to learn norm violations by incorporating data from different sources (communities) to improve the performance of ML models in a new target community. Then, we describe the experiments, focusing on our analysis of the differences in norm violations across communities using interpretability results.

In Chapter 6, we introduce our user study that investigates the influence of interpretability on participants' behavior. We describe the online questionnaires, which enable participants to engage with interpretability layouts. Lastly, we present the statistical and qualitative analysis results and discuss the findings of our investigation.

The literature review is provided in Chapter 7. Specifically, we cite literature focusing on detecting detrimental behavior in online communities using ML. We also analyze literature that deals with incremental and ensemble learning in an environment with concept drift and imbalanced datasets. Then, we describe the literature on cross-community learning, where works leverage the performance of a model using data from diverse sources, including solutions that employ PLMs and adapters. We also present literature addressing interpretability in tabular and text classification scenarios, demonstrating how interpretability is employed in different use cases. Lastly, we discuss literature focusing on conducting experiments exploring how users interact with interpretability information.

Finally, in Chapter 8, we provide the conclusions of our research, including directions for future work.

Chapter 2

Background

This chapter presents the base concepts upon which this thesis is built. First, we introduce the ensemble strategy to deal with the imbalanced nature of the dataset (q.v. Section 2.1). Then, we delve into concept drift, specifically targeting changes in the community members' view about norm violation (q.v. Section 2.2). Additionally, we describe the emergence of new violation classes as a sub-type of concept drift. Third, we describe the learning approaches to detecting norm violation: batch and incremental learning (q.v. Section 2.3). Batch Learning is interesting for cases where the domain does not require updating the trained model, and the complexity of keeping and managing an entire dataset does not cause disruptions. In contrast, incremental learning is most useful when we desire to continuously train machine learning models without needing to manage the complexities of large datasets. Fourth, we introduce the Pre-Trained Language Model (PLM), the transformer-based model which is responsible for handling text sentences (q.v. Section 2.4). In this section, we also present adapters, which detect specific violation classes and handle the emergence of these classes as interactions unfold. Subsequently, we detail Local Interpretable Model-Agnostic Explanations (LIME) and Integrated Gradients (IG), the interpretability tools explored in this work (q.v. Section 2.5). These tools enable our solution to promote transparency by facilitating human comprehension of a model's inner workings when it detects a norm violation. Lastly, in Section 2.6, we present the Generalized Additive Model (GAM) used to analyze the user study results.

It is worth mentioning that the different background sections refer to specific frameworks of our multi-scenario approach. For instance, only the Feedback-Driven Adaptive Learning (FeDAL) framework incorporates ensemble learning to detect norm-violating behavior (q.v. Chapter 3). Concept drift is present in all scenarios since this is the challenge inherent to the learning task in a dynamic online community. Experiments with FeDAL include the use of both batch and incremental learning techniques. Differently, for text datasets, the experiments exclusively include mini-batch learning. This is imperative in our work since training with the complete dataset (or one instance at a time) is detrimental to the computational performance of the transformer-based approaches. PLMs are employed by the Language Model Adaptive Learning (LaMAL) (q.v. Chapter 4) and the Cross-Community Adapter Learning (CAL) (q.v. Chapter 5) frameworks. Moreover, only CAL, viewed as an evolution of LaMAL, incorporates adapters to address the emergence of new violation classes.

This thesis addresses tabular and text datasets, and the work is evaluated within the Wikipedia use case (q.v. Section 1.1.3). As such, it is crucial to define what composes these datasets. Specifically, we evaluate the article editing task. In this context, an action is the editing of an article by a community member, and it can be formalized either as a set of features or as direct textual input. With our multi-scenario approach, we can continuously learn to detect norm-violating behavior with different action formalizations. FeDAL works with the tabular dataset (set of features),

while LaMAL and CAL work with text sentence input. Thus, regarding interpretability, FeDAL incorporates LIME, a model agnostic tool that can provide explanations for the relevant *features* in an ensemble of classifiers. LaMAL and CAL incorporate IG, specifically built to understand the inner workings of transformer-based models (in our case, PLMs) by obtaining the relevant *words* of a text sentence that contribute to a model's output.

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, they tend to be biased towards the class that describes regular behavior. To tackle this issue, we use ensemble learning, which can be defined as the generation and combination of different ML models (e.g., Feed-Forward Neural Network (FNN), Random Forest, and Logistic Regression) to solve a predictive task (Sagi & Rokach, 2018). 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, resulting in superior performance compared to single components (Dong et al., 2020; Galar et al., 2011).

Various classification systems have been developed employing ensemble learning techniques. Dong et al. (2020) highlight some important ones, such as Bagging (Lenka et al., 2022; Roshan & Asadi, 2020; B. Wang & Pineau, 2016; Q. Wang et al., 2017), AdaBoost (Abayomi-Alli et al., 2022; Taherkhani et al., 2020; W. Wang & Sun, 2021), Random Forest (Abayomi-Alli et al., 2022; de Freitas Barbosa et al., 2021; Kavzoglu & Teke, 2022), and Gradient Boosting (Babajide Mustapha & Saeed, 2016; Bentéjac et al., 2021; Cahyana et al., 2019; Taha & Malebary, 2020). Among these approaches, Bagging is an interesting method to address the imbalanced dataset challenge investigated in this work, especially when enhanced with data sampling techniques (Galar et al., 2011), as it presents relevant performance in similar (Alam et al., 2021; Hakak et al., 2021; S. Liu et al., 2017; Tang et al., 2019) and diverse use cases (Sagi & Rokach, 2018), including fault diagnosis (Jia et al., 2020; Wu et al., 2018; T. Zhang et al., 2022) and medical domains (Ahishakiye et al., 2020; Cahyana et al., 2019; de Freitas Barbosa et al., 2021; N. Liu et al., 2020; Suri et al., 2022).

FeDAL employs Bagging to detect norm violations by training ensemble members using different balanced subsets extracted from the imbalanced data (q.v. Figure 2.1). For instance, consider a binary classification task with an imbalanced dataset D, it is possible to divide D into two subsets, a majority class subset M and a 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 a number N of balanced datasets $B_D = \{B_1, ..., B_N\}$. Each $B_i \in B_D$ is a dataset with a similar class distribution and $N = |M|/|P|$. 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_D are created to have size $2 \times |P|$ by taking $|P|$ non-overlapping instances from M and replicating all instances of P in each subset. To limit the size of E and restrain the computational power needed to process income data points, FeDAL enhances ensemble learning by employing two data sampling techniques, *undersample* and *oversample*. The first decreases the number of majority instances, while the second increases the number of minority class instances by replication (Mohammed et al., 2020) or by generating new instances with methods like SMOTE (Chawla et al., 2002).[1] However, FeDAL only applies data sampling techniques after reaching a threshold (i.e., pre-defined ensemble size) since we aim to avoid discarding relevant information or adding more complexity to the dataset (Galar et al., 2011; S. Liu et al., 2017). Lastly, our bagging approach combines the outputs of the individual models through majority voting to decide on the final classification output.

[1] In Section 3.6.3, we compare oversampling by replication with SMOTE using the FeDAL framework.

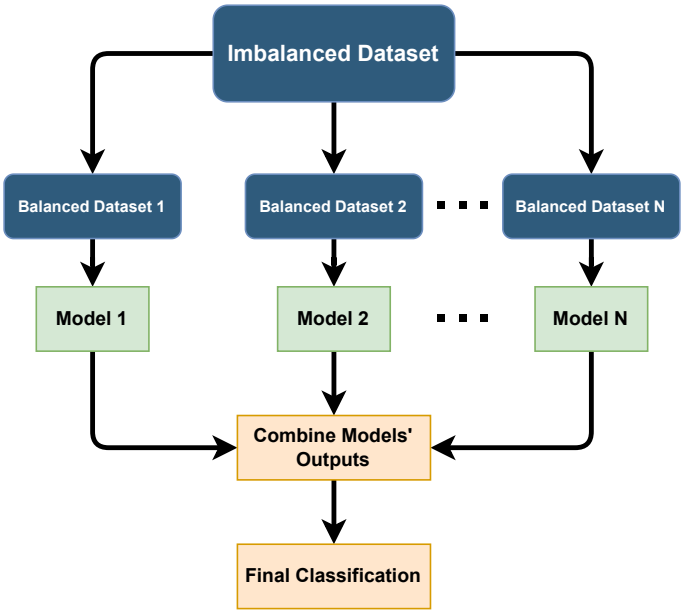


Figure 2.1: Bagging Learning to handle imbalanced datasets.

Within FeDAL's context, we explore three training approaches to build a model that adopts bagging learning: batch, mini-batch, and online learning (q.v. Section 3.5). However, we incorporate some important modifications into our online setting (where training happens one instance at a time), such as the one presented by S. Wang et al. (2015), where the resampling strategy also counts with the addition of weight adjustment over time. S. Wang et al. (2015) propose Oversampling-based Online Learning (WEOB1) and Undersampling-based Online Learning (WEOB2). WEOB1 and WEOB2 work to adjust the learning bias from M to P by using ensemble and resampling instances from these subsets. Like the bagging strategy for batch and mini-batch, online bagging creates different classifiers and then trains each classifier $C \in E$ 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. Thus, 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 Concept Drift

In addition to an imbalanced dataset, concept drift is another characteristic when dealing with norm violation detection. In this thesis, concept drift is defined as the change in the community members' view concerning what elements of an action (e.g., a set of features or words of a sentence) indicate the presence of norm-violating behavior. Essentially, it is how community members see changes in regular or violating behavior over time and across communities.

In practice, it is possible to identify the change in community behavior by observing the joint distribution $C_t(X, y)$ over time (J. Lu et al., 2018; H. Wang & Abraham, 2015), where $x \in X$ is a feature value, $y \in \{0, 1\}$ is the associated class label that denotes regular or norm-violation behavior, and t is the current timestamp. Then, to compare two moments in time and detect a possible concept drift, we refer to the following: $C_t(X, y) \neq C_u(X, y)$, where u is a timestamp in the past. To better understand concept drift, let us take as an example the use case of Wikipedia article edits and the definition present in (Gama et al., 2014) that categorizes concept drift in three

different ways: change in the prior probability of classes $c(y)$, which is a change in the ratio between norm violation and regular behavior, i.e., the number of instances of one class increases, while the number of instances of the other class decreases or stays the same; change in the class conditional probabilities $c(X|y)$, which is a change in how norm violation and regular behavior are defined, i.e., the way an acceptable action must be performed in the community changes over time, such as having the community expecting a more formal writing style lately; and change in the posterior probabilities of classes $c(y|X)$, which is a change in what is considered a norm violation. This concept drift differs from the above since it occurs without changing how actions are performed (like adopting a more formal writing style). Concretely, only the action's label (violation or not) changes. For example, a term previously prohibited in the community may now be acceptable. The latter is the kind of data that leads to real concept drift, which is the type of concept drift that interests us in this work. Furthermore, we highlight that the different types of concept drift can occur simultaneously (Gama et al., 2014).

In addition to the three types of concept drift described earlier, another important aspect of concept drift in our study is the emergence of new violation classes. As online communities evolve, novel norm violations that were previously not encountered or recognized may arise. This phenomenon introduces the challenge of not only detecting but also categorizing these new violation classes. In this context, the definition present in (Forman, 2006) assists us in defining this type of concept drift. Therefore, the emergence of new violation classes is denoted as "virtual concept drift" since it represents a shift in the definition of norm violations within the community.

In this thesis, the new violations are composed of undefined sub-classes, and their creation derives from community members' feedback. To illustrate this phenomenon, consider article editing in Wikipedia (q.v. Section 1.1.5). In addition to the binary categorization (regular and violation classes), the emergence of new violation classes serves to precisely identify the norm-violating behavior present in the community, which embraces the following classes: Swear, Insult and Ableism, Sexual Harassment, Racism, LGBTQIA+ Attack, and Misogyny.

As new violation classes emerge, the existing detection framework may not effectively identify and distinguish these classes from established definitions of violation and regular behavior. Hence, the system must be adaptive and flexible enough to incorporate and learn from these new violation classes. Specifically, with LaMAL and CAL, we explore the detection of different sub-classes of violation, in addition to the binary categorization of actions as either violation or regular. CAL is especially useful in this context since, due to the adapter architecture (q.v. Section 2.4), it can dynamically handle the emergence of new violation classes.

In summary, in addition to monitoring the traditional types of concept drift, the ability to handle the emergence of new violation classes is important to maintain the relevance and efficacy of norm violation detection systems in online communities. The adaptive nature of our multi-scenario approach (FeDAL, LaMAL, and CAL) enables our solution to address this challenge, remaining robust to the evolving definitions of what constitutes a norm violation.

2.3 Training Techniques

The presence of concept drift and the streaming characteristics of the data in online community domains prompt different ways to train an ML model. Here, we investigate two training approaches, batch and incremental learning (mini-batch and online learning). We argue that, by continuously updating the base ML model as data is made available, incremental learning approaches are the most suitable to be deployed in a system supporting an online community. However, batch learning can also be used to learn in this setting by creating new ML models when additional data is made available. For instance, a new set of actions is executed in the community, and our ML model must learn from this new information. In this case, batch learning creates new models to be trained

with the most recent data. Additionally, it sets weights for the latest data points to update the parameters of these new ML models, giving them more importance than past information.

2.3.1 Batch Learning

Batch learning deals with the entire dataset (new and past data are combined). In batch learning, it is not possible to update the trained models. To incorporate new knowledge, a complete training process from the beginning is necessary (Hastie et al., 2009), which is the main drawback of this technique when the problem requires handling non-stationary domains. Another disadvantage is the need to maintain and treat an entire dataset, increasing the system's cost and complexity (especially when different entities and organizations regulate data treatment) (Lebichot et al., 2021). Although batch learning presents these important disadvantages, it also offers interesting benefits, such as the computation of the statistical values and the stability properties during the training procedure. The algorithm calculates the statistical values (i.e., mean and standard deviation) by using the entire dataset available for training and applying it to the pre-processing phase, which could be a standardization process as FeDAL applies in Algorithm 3.4. Then, data availability avoids fluctuations due to changes in the training dataset, making the algorithm more stable (as also detailed in Section 3.6).

To further clarify how batch learning works, let us assume that the base algorithm used to find the parameters of the ML model is Gradient Descent. Variations of this algorithm exist for the incremental and batch learning cases. Considering Batch Gradient Descent, the algorithm handles the entire dataset D. Then, it proceeds to get the gradient values for each data point $X \in D$ with respect to the cost function, calculates the mean values, and then updates the algorithm parameters (Ruder, 2016). For ML models, these parameters are the weights and the coefficients. One complete iteration over the entire dataset is one epoch of the batch learning process, which can be summarized in the following equation:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.1}$$

θ represents the model's parameters. η is the learning rate, and it determines the step size the algorithm must take to reach a local minimum. $J(\theta)$ is the objective function to minimize, and ∇_θ represents the gradient with relation to the parameters (Ruder, 2016). Since this process considers D at a certain point in time, it does not embrace the update of the model. To incorporate new knowledge, it is necessary to start the training process from the beginning, leading to redundant computations (gradient values for similar instances are recalculated for each update of the parameters).

2.3.2 Incremental Learning

Incremental learning is the approach responsible for dealing with the drawbacks present in batch learning. This technique incrementally learns as new data arrives sequentially, which is particularly interesting in online communities since the ML model must be constantly updated as people interact and changes in norm understanding emerge. In this work, we are concerned with mini-batch and online learning. Mini-batch learning creates data blocks B with the data that arrives over time and uses it to continuously train ML models. Since we only deal with the most recent instances composing the current data block, the process is neither as costly nor as complex as batch learning (Lebichot et al., 2021; Z. Li et al., 2020). Online learning can be seen as a special case of mini-batch learning, in which the batch size is 1. In this scenario, it is possible to update the ML model as soon as data is made available, discarding the need to store this data point and avoiding the complexities of data treatment. The difference between these two approaches is that in mini-batch learning, we use chunks of data, while in online learning, we use one single data

point to update the base ML model. It is important to highlight that online learning algorithms have poor stability compared to mini-batch algorithms (Z. Li et al., 2020).

As in the batch learning case, Gradient Descent can be used to update the model's parameters for incremental learning, although some changes are necessary. Online learning uses Stochastic Gradient Descent (SGD), considering only one data point at each time step to update the ML model's parameters.

Since SGD updates the parameter values as new instances arrive, this approach is suitable for learning incrementally. Additionally, it does not execute redundant computations, which makes this option more efficient when compared to batch learning. The following equation describes how SGD works:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.2}$$

The difference from Equation 2.1 is that SGD relies on updating the parameter values for each new instance (i) in the dataset, which is represented by its features $(x^{(i)})$ and label $(y^{(i)})$.

Trying to combine the best of SGD and batch learning, Mini-Batch Gradient Descent updates the model's parameters after each data block B is completed (the number of instances in each data block is defined by n). The equation for Mini-Batch Gradient Decent is:

$$\theta \leftarrow \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.3}$$

The mini-batch implementation addresses the drawbacks of batch learning and SGD. First, it allows the model's parameters to be updated while reducing the variance (SGD instability), leading to more stable convergence, as described in Section 3.6. Second, it also optimizes vectorizations and matrix operations (batch learning complex computation), enhancing the computation process of the gradient (Ruder, 2016). These advantages make mini-batch learning the most suitable training technique for FeDAL (q.v. Section 3.6), LaMAL (q.v. Section 4.4) and CAL (q.v. Section 5.3). Especially for LaMAL and CAL, mini-batch learning improves computational efficiency during the training process of the PLMs investigated in this work.

2.4 Pre-Trained Language Model (PLM)

The first component of our multi-scenario approach, FeDAL (q.v. Chapter 3), focuses on solving tabular classification problems. However, we aim to broaden our approach'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 (Joachims, 1996; Yun-tao et al., 2005) and Recurrent Neural Network (Schuster & Paliwal, 1997) 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 contexts (Kenton & Toutanova, 2019; Min et al., 2023; Vaswani et al., 2017). 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 (Qiu et al., 2020). Figure 2.2 presents the transformer layer architecture and the advances incorporated, such as the addition of the attention mechanisms and the use of fully connected Feed-Forward Neural Network (FNN) layers, assembled in a parallelized way to improve computational performance (Wolf et al., 2020).

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 information." This mechanism

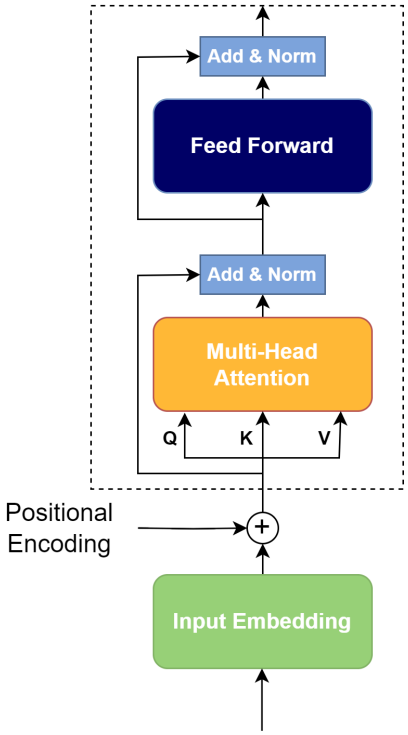


Figure 2.2: The transformer layer, as proposed by Vaswani et al. (2017).

iteratively calculates the attention score between all words in the sentence, thus obtaining the dependency relationship between them (Vaswani et al., 2017). In this 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 (Vaswani et al., 2017), 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 (Vaswani et al., 2017). Equation 2.4 formalizes how attention is calculated.

$$Attention(Q, K, V) \leftarrow softmax(\frac{Q \odot K^T}{\sqrt{d_k}}) \odot V \tag{2.4}$$

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 between Q and the transpose of K (Vaswani et al., 2017). 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) (Ba et al., 2016; J. Xu et al., 2019). 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 2.5 formalizes this process.

$$LayerNorm(x) \leftarrow \frac{g}{\sigma} \odot (x - \mu) + b \tag{2.5}$$

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 (Equation 2.6) activation function (Vaswani et al., 2017). To formalize the complete step, we present Equation 2.7.

$$ReLU(z) \leftarrow max(0, z) \tag{2.6}$$

$$FFN(x) \leftarrow ReLU(x \times W_1 + b_1) \times W_2 + b_2 \tag{2.7}$$

ReLU (q.v. Equation 2.6) executes a non-linear operation that aims to calculate the final value given by a previous NN layer (z). In Equation 2.7, 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 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 (Wolf et al., 2020). Different implementations yield state-of-the-art results, e.g., BERT (Kenton & Toutanova, 2019), which has around 110 million trainable parameters, RoBERTa (Y. Liu et al., 2019), around 125 million trainable parameters, and DistilBERT (Sanh et al., 2019), 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 (Elazar et al., 2021).

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 text classification of norm-violating behavior, i.e., given a text as input, the model predicts whether the text violates some community norm. 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 classification tasks) and the number of instances used for training.

Concretely, our experiments with LaMAL (q.v. Section 4.3) and CAL (q.v. Section 5.2) explore 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 (Y. Liu et al., 2019). The second is DistilBERT, which is also built on top of BERT, but it aims to create a smaller, faster, and cheaper model (Sanh et al., 2019). Section 4.4 presents the results of RoBERTa and DistilBERT applied to hate speech detection in Wikipedia article edits using the LaMAL framework. Furthermore, Section 5.3 presents the results of DistilBERT applied to the same domain using the CAL framework, including the adoption of the adapter architecture.

Adapter

As described above, PLMs take advantage of the fine-tuning paradigm, which uses previously trained implementations and only updates its parameters for a specific text classification task.

However, it is worth reiterating that this approach still requires updating a considerable amount of weights derived from the large number of parameters present in PLMs. To tackle this issue, we adopt a fine-tuning strategy that incorporates adapters between the transformer layers of a PLM. These adapters are neural networks with a small proportion (usually 3%) of the number of parameters present in the full model, resulting in a faster and more efficient training process (Houlsby et al., 2019).

In this context, while we continuously update the adapter weights on our target data, the transformer layers are used only for language representation, keeping the original PLM parameters frozen. Equation 2.8 (Pfeiffer et al., 2020) and Figure 2.3 describe how the adapter is combined with a PLM.

$$\Phi_n \leftarrow \underset{\Phi}{\operatorname{argmin}} L_n(D_n, \Theta, \Phi_n) \tag{2.8}$$

In Equation 2.8, $n \in \{1..., N\}$, where N is the number of text classification tasks to be solved. Φ_n represents an adapter for task n. Θ is the base PLM (shared by all adapters). D_n is the training data for the specific classification task, and L_n is the loss function we seek to minimize, which can differ depending on the task. For example, we could use categorical cross-entropy to handle multi-class classification tasks while using binary cross-entropy for the binary case.

For a comprehensive description of the adapter architecture, Figure 2.3 illustrates the adapter integration between the layers of a PLM. The left portion of the figure portrays the transformer layer proposed by Vaswani et al. (2017) and depicted in Figure 2.2, but introducing the adapter after the feed-forward component. The right portion of the figure portrays the adapter composition, specifying the *bottleneck architecture* that limits the number of parameters (Houlsby et al., 2019). In this configuration, the feed-forward down-project component projects the original d-dimensional features into a reduced dimensionality, denoted as m. Next, the adapter applies a nonlinearity function, followed by the up-project component, which restores the dimensionality to d. In total, adapters add $2md + d + m$ parameters per layer, including biases. By adopting $m << d$, the adapter limits the number of parameters added per task. In practice, this configuration employs around 0.5% - 8% of the parameters of the original model.

In conclusion, we summarize the advantages of adapter-based fine-tuning in three parts. First, it presents impressive results for domains where data is scarce, such as low-resource tasks and communities, and cross-lingual tasks (R. He et al., 2021). This is especially relevant for CAL, as we aim to learn what constitutes norm violations from a limited set of labeled data. Second, it tackles catastrophic forgetting due to interference between different text classification tasks,[2] which might be an issue in fine-tuning a complete PLM, such as LaMAL's fine-tuning process. CAL accomplishes that by dynamically creating individual adapters as the community requires solving new classification tasks (i.e., identifying new violation classes). Third, the continuous update of smaller neural networks allows for greater robustness to handle over-fitting and reduced sensitivity to changes in learning rates (R. He et al., 2021).

[2]Catastrophic forgetting refers to the loss of knowledge acquired from previous classification tasks when learning new information (Pfeiffer et al., 2020). In our context, preventing the loss of information about the detection of a violation class when training with data from a different violation class within the same community or across communities.

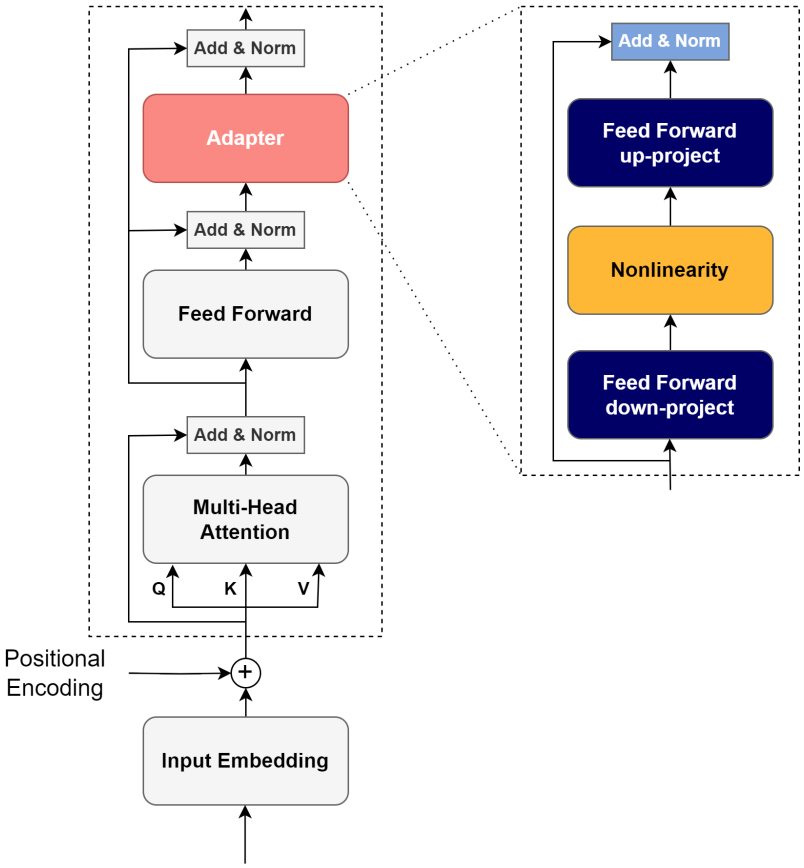


Figure 2.3: The adapter interaction with the transformer layers, as proposed by Houlsby et al. (2019). The right portion of the figure depicts the adapter's bottleneck architecture.

2.5 Interpretability

The base concepts introduced in previous sections are relevant to our challenge of detecting norm violations in online communities. However, we argue that normative systems must also provide explanations to include evidence on the relevant elements of an action associated with violations and how they may differ in light of the evolving nature of online interactions. Thus, our main reasons for implementing this ability in our multi-scenario approach are to allow for model debugging and to enable community members to understand the reasons for classifying certain actions as violations, with the ultimate goal of providing adherence to the transparency principle of responsible artificial intelligence (Arrieta et al., 2020; Barredo Arrieta et al., 2020). Moreover, 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 local interpretability methods since we primarily want to inform community members about specific norm violations.[3] Thus, to fulfill these goals, we integrate two interpretability tools into our multi-scenario approach, the Local Interpretable Model-Agnostic Explanations (LIME) and the Integrated Gradients (IG) algorithms. LIME is used by FeDAL since it is a model agnostic tool that can provide explanations for the

[3]We also investigate using a list of the words that contribute the most to detecting norm violations, such as those in Appendix C. However, we clarify that these lists do not represent global interpretability. Instead, they represent the sum of the values obtained through local interpretability.

relevant *features* in an ensemble of classifiers. IG is used by the transformer-based models, LaMAL and CAL, since it is specifically built to understand the inner workings of such models by obtaining the relevant *words*.

Since LIME and IG serve to obtain the relevant features or words associated with norm-violating behavior, our multi-scenario approach aims to follow a two-step process to provide this information to community members. First, the system conveys the reasons for a model's output to the community member executing an action. Sections 3.6, 4.4 and 5.3 and Appendix B showcase how this information is presented. Second, we prepare our approach to provide interpretability data to other community members in a future feedback elicitation process (although we do not handle this in the current work, we highlight its importance for future research), focusing on discussing the reasons behind a violation and gathering the evolving community views. This is especially important when there is a discrepancy between different members' views. A deliberation process will enable members to discuss among themselves (informed by interpretability results) the aspects of the model's output that affect the disagreement.

2.5.1 Local Interpretable Model-Agnostic Explanations (LIME)

In the context of this thesis, specifically with FeDAL, an action is represented by a set of features. Each feature describes one aspect of the action that a user executes in an online community, e.g., the number of profane words in an article edit, the percentage of alphabetic text added, and the size of the article edit. We adopt this approach to allow our framework to adapt to different application domains since modeling an action as a set of features allows the framework to deal with different classification tasks, such as identifying fake online reviews and credit card fraud. For instance, we could map the action of participating in an online meeting by features such as the amount of time present, the volume of messages exchanged, and the rate of interaction with other participants. The proposed approach can use these features in online communities to explain which aspects of an action indicate norm violation.

The features of an action can provide valuable information about the aspects that may contribute to norm-violation detection. To this end, we adopt a tool that can explain a model's prediction, the Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016). Here, LIME is responsible for helping to explain the output of three different training algorithms implemented by FeDAL, mini-batch (q.v. Section 3.1), online learning (q.v. Section 3.2), and batch (q.v. Section 3.3).

To achieve its goal, LIME learns an interpretable model around the data points of interest (the instances to explain). LIME focuses on presenting a locally faithful explanation. To do that, it perturbs the instance to be explained, creates a dataset with these perturbed data points randomly generated following the underlying data distribution, and reviews how the model's output changes due to this perturbation. Specifically, LIME works following the equation below:

$$\xi(x) \leftarrow \min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{2.9}$$

$\xi(x)$ represents the explanation found by LIME, defined as a model $g \in G$, where G is the class of possible models that humans could interpret, such as decision trees or linear regression. The interpretable model should be easily understandable to humans, either in a visual or a textual format. f is the ML model being explained. While π_x is the proximity measure, and $\pi_x(z)$ represents the distance between the data point and an instance (z) generated around x (the original instance being explained) by perturbation. Then, $\mathcal{L}(f, g, \pi_x)$ is the fidelity function that summarizes how unfaithful g is when approximating f in π_x (Ribeiro et al., 2016). Concretely, to obtain the local behavior of f, LIME minimizes the fidelity function by drawing random samples around x with the weight defined by π_x. Lastly, $\Omega(g)$ quantifies the complexity of the explanation, where the aim

is for the explanation to be simple enough for human understanding. In this context, Ribeiro et al. (2016) describe different options for $\Omega(g)$ depending on the specific model g chosen. For instance, $\Omega(g)$ can be defined as the tree depth in the case of the decision tree algorithm or as the number of non-zero weights in a linear model.

2.5.2 Integrated Gradients (IG)

Unlike our tabular scenario, text-related tasks do not need a featurization process (encode text sentences into a set of features). Instead, it is possible to manipulate the text directly (Arrieta et al., 2020; De-Arteaga et al., 2019; Niu et al., n.d.; Räuker et al., 2023). Thus, to explain models that handle text sentences, LaMAL and CAL incorporate the Integrated Gradients (IG) algorithm to understand the parts of a text most relevant to the model's output (Sundararajan et al., 2017). IG enables our frameworks to gain insights into the inner workings of transformer-based models by debugging and extracting rules from a Deep Neural Network (DNN) (Sundararajan et al., 2017).

Specifically, IG calculates a word's contribution to detecting norm violations by a backward pass through the model, propagating its relevance from the output to the input (Lyu et al., 2022). 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 (Lyu et al., 2022; Sundararajan et al., 2017) 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 (Sundararajan et al., 2017). Thus, the integrated gradients come from the accumulation of these individual points. Equation 2.10 formalizes the integral calculation.

$$IntGrads(x_i) \leftarrow (x_i - \bar{x_i}) \odot \int_{\alpha=0}^{1} \frac{\partial M \times (\bar{x} + \alpha \times (x - \bar{x}))}{\partial x_i} \times d\alpha \tag{2.10}$$

However, to efficiently compute the integrated gradients, IG approximates $IntGrad(x_i)$ by the Riemann sum method (Equation 2.11), 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 (Sundararajan et al., 2017) suggest around 20-300 points along the path.

$$r(x_i) \leftarrow (x_i - \bar{x_i}) \odot \sum_{k=1}^{m} \frac{\partial M(\bar{x} + \frac{k}{m} \times (x - \bar{x}))}{\partial x_i} \times \frac{1}{m} \tag{2.11}$$

2.6 Generalized Additive Model (GAM)

As discussed in the previous section (q.v. Section 2.5), one of our goals is to enable community members to understand why a model classifies an action as a violation by integrating interpretability into our solution. As such, we argue that it is essential to evaluate whether the interpretability tool employed in our work affects people's views on what behavior constitutes a norm violation. To address this challenge, we conduct a user study to investigate the potential influence of interpretability on an individual's understanding of norm-violating behavior, aiming at reducing detrimental behavior in online community domains.

In our user study, participants must evaluate text sentences containing hate speech (q.v. Chapter 6). The IG algorithm generates the questionnaire's interpretability data. The output from the IG

algorithm can be used to present information to community members in three different manners, which we refer to as *interpretability layouts* and depict in Figure 6.1. These layouts are:

- Local interpretability: it describes the impact of individual words (relevance score) on identifying norm violations in a specific text sentence.
- List with the sum of relevance scores: it describes the impact of words considering all text sentences in our dataset. The sum of relevance scores is not a global interpretation of our model but rather a summary of local interpretations. Specifically, this list is obtained by summing the relevance score (local interpretability) of each word's occurrence in that dataset.
- Combination of both: it describes local and the sum of relevance scores together.

Concretely, we investigate how these layouts influence an individual's view when they evaluate a text sentence as a specific class of hate speech (considering the Wikipedia domain described in Section 1.1.3). Participants answer a questionnaire that directly inquires how their views change when presented with a baseline (no interpretability information) and information generated by the different interpretability layouts.

The analysis of response data in our study requires the consideration of four factors that may influence participants' views: the interpretability layout, the demographic information, the participants, and the sentences being evaluated. Following this requirement, we adopt the Generalized Additive Model (GAM) that enables us to explore the relationship between distinct variables by using nonlinear functions to model the response data (Fahrmeir et al., 2013). Specifically, Equation 2.12 describes the additive characteristic present in GAMs that is important to our work (Baayen & Linke, 2020). This equation serves to handle ordinal responses, outlined as participants' answers on the Likert scale (y_{target}), along with random effects corresponding to individual participants (x_{part}) and text sentences (x_{sent}), as well as fixed effects associated with demographic aspects (x_{dem}) and interpretability layouts (x_{int}). Additionally, β_0 represents the intercept,[4] while β_{int} and β_{dem} refer to the weight for the interpretability layouts and demographic factors, respectively. α_{part} and α_{sent} address the weights for the random effects, considering participant-specific and sentence-specific effects, respectively.

$$y_{target} \leftarrow \beta_0 + \beta_{int} \times x_{int} + \beta_{dem} \times x_{dem} + \alpha_{part} \times x_{part} + \alpha_{sent} \times x_{sent} \tag{2.12}$$

In summary, by capturing the relationship between these variables with a GAM, we can estimate participants' confidence about norm violation, discovering whether any of these factors influence participants' views.

Theoretically, following the formalization of Hastie and Tibshirani (1987) that is described in Equation 2.13, GAMs comprise a sum of smooth functions that can embody linear effects and variables, including continuous and categorical. This composition effectively allows GAMs to capture nonlinear relationships, such as nonlinear co-variate effects, overcoming the assumption of linearity in the data.

$$E(Y|X_1, X_2, ..., X_p) \leftarrow s_0 + \sum_{j=1}^{p} s_j(X_j) \tag{2.13}$$

Y is the dependent variable (i.e., our prediction goal), and $E(Y|X_1, X_2, ..., X_p)$ represents the function that connects the expected value to the predictor variables $X_1, ..., X_p$. s_0 is a constant

[4]The intercept is a term used in the model to represent the estimated value of the dependent variable (y_{target}) when the values of the independent variables (x_{int}, x_{dem}, x_{part}, x_{sent}) are 0 (Baayen & Linke, 2020; J. D. Davis, 2007).

term, and the Backfitting algorithm estimates it.[5] The terms $s_1(X_1), ..., s_1(X_p)$ denote smooth, nonparametric functions (Larsen, 2015).[6] Moreover, several smoothers exist in the literature. In this work, GAMs use the smoothing splines (Baayen & Linke, 2020) following the implementation in the R programming language (R Core Team, 2023).[7] The individual variables are separated by the "+" operator, indicating the additive characteristic of the model.

Another critical aspect of GAM is evaluating its cost when identifying data patterns. This computational cost is calculated using Equation 2.14 that expresses the trade-off between the likelihood term and the *wiggliness* (Baayen & Linke, 2020):

$$C_f = \underbrace{\sum_{i=1}^{n}(y_i - f(x_i))^2}_{Likelihood} + \underbrace{\lambda \times \int f''(x)^2 \times dx}_{Wiggliness} \tag{2.14}$$

The likelihood term aims to keep fidelity to the observed data while minimizing the summed square errors (Baayen & Linke, 2020), whereas the *wiggliness* calculates the curve's complexity, measuring the fluctuations in its shape. This last component maintains the model as simple as possible (less *wiggly*). Consequently, a good balance between these two terms is important for a good fit. y_i denotes observed values, while $f(x_i)$ denotes predicted values. The smoothing parameter (λ) controls this balance, indicating the penalization associated with a specific degree of *wiggliness*, a value dependent upon the response data (answers from participants). In practice, this penalization operates to avoid overfitting. The integral over the square of the second derivative of function $f(x)$ is the effective measure of the *wiggliness*.

Lastly, to mitigate the potential influence of outlier participants with considerably divergent views on norm violations, we rely on GAMs' ability to manage response data rated on the Likert scale while maintaining robustness against the perturbing impact of outliers in deriving conclusions. GAMs accomplish this by assuming that outliers exhibit distinct behavior, connecting the associated uncertainty with wide confidence intervals (Baayen & Linke, 2020).

[5]The Backfiting algorithm is an iterative procedure to estimate the nonlinear effects of a GAM, enabling the model to handle the nonlinear relationship between the predictor variables and the dependent variable(Banks & Fienberg, 2003; Hastie & Tibshirani, 1987).

[6]It is worth mentioning that nonparametric functions, as opposed to their parametric counterparts, derive their shape exclusively from empirical data without reliance on a limited set of predefined parameters (Larsen, 2015).

[7]R is an Open Source programming language designed for statistical computing. It is important in our context because it provides extensive data analysis and presentation tools (R Core Team, 2023).

Chapter 3

FeDAL: The Ensemble Machine Learning Framework

This chapter introduces the Feedback-Driven Adaptive Learning (FeDAL) framework,[1] which is the first framework that composes our multi-scenario approach to continuously learn what constitutes a norm violation from past interactions and user feedback in online communities. We achieve this goal by using examples of behaviors depicted as violations, gathered either as text sentences or formalized with a set of features. Since the formalization of interactions may vary depending on the type of violation-detection task, it is essential to handle different types of datasets in online communities. For instance, while some violation-detection tasks only require a set of features to describe an action (e.g., fraud discovering, misbehaving recognition, and identifying deception writing styles), others must handle the raw text input as it takes place due to the complexity of the task (e.g., detect specific hate speech class, style change detection, and identifying AI-generated reviews). Consequently, our emphasis lies on an approach that can be employed in multi-scenario contexts, particularly in tabular and text-based domains, since these cover a variety of use cases, including the detection of violations in Wikipedia article editing (q.v. Section 1.1.3).

FeDAL learns from past interactions and user feedback, constantly adapting the definition of what constitutes a norm violation based on the evolving understanding of that norm with respect to the community members. In this context, a community member's action is represented as a set of features that FeDAL uses to define norm violations, such as the number of profane words, occurrences of alphanumeric characters, and edit size (details of the complete set of features are given in Section 3.5). Specifically, FeDAL learns to detect norm violations by integrating ensemble (q.v. Section 2.1) and two learning techniques (q.v. Section 2.3): 1) incremental learning; and 2) batch learning. We evaluate both techniques to identify their drawbacks and determine the scenarios in which one technique outperforms the other (q.v. Section 3.6).

The motivation behind ensemble learning is that detecting norm violations usually implies working with imbalanced datasets since violating behavior occurs less frequently than regular (or expected) behavior. Thus, adopting an approach capable of handling class distribution imbalance is crucial. Moreover, in FeDAL, we leverage ensemble learning by employing data sampling techniques, namely oversampling and undersampling. These techniques limit the ensemble's size and restrain the computational power needed to process incoming data points.

Regarding training procedures, we investigate the following three approaches to train the base machine learning (ML) models:

[1] Source code available at https://bitbucket.org/thiago-phd/journal_paper/.

1. The mini-batch approach (q.v. Section 3.1), where the base ML model is continuously trained using data blocks that arrive sequentially, without past information.
2. The online approach (q.v. Section 3.2), where the learning algorithm uses a single data point to train the base ML model, which allows for an immediate model update as soon as an interaction occurs.
3. The batch approach (q.v. Section 3.3), where the learning algorithm uses the entire dataset, past and new information, training the ML model from scratch every time.

In our use case, the majority class set M represents regular behavior, while the minority class set P represents violations. Since we define an action as a set of features, we represent a data point with the tuple (X, y), where X is the set of features of a given action and $y \in \{0, 1\}$ is its class label, describing if it is a violation (1) or not (0).

The remainder of this chapter is divided as follows. Sections 3.1, 3.2, and 3.3 present the three learning algorithms investigated by FeDAL, with a computational complexity analysis in Section 3.4. Subsequently, in Section 3.5, we describe the experiments conducted to evaluate the FeDAL framework, with the results outlined in Section 3.6. Finally, Section 3.7 offers a comprehensive summary of the key points discussed in this chapter.

3.1 Mini-batch Learning for Tabular Scenarios

FeDAL employs a mini-batch approach (q.v. Algorithm 3.1) that builds on top of two incremental ensemble algorithms, the Accuracy Updated Ensemble (AUE2) (Brzezinski & Stefanowski, 2014) and the Dynamic Updated Ensemble (DUE) (Z. Li et al., 2020). However, FeDAL introduces significant differences: 1) it incorporates feedback to emphasize data points that had their class labels changed by the community; 2) it uses a replication-based oversampling technique that randomly replicates minority class instances present in the current data block, as opposed to the SMOTE oversampling technique that creates synthetic minority class samples (Chawla et al., 2002) (in Section 3.6.3, we compare both data sampling techniques and analyze their impact on the proposed solution); and 3) it defines a new metric, the number of classifiers, to determine the oversampling ratio for minority instances (q.v. Algorithm 3.1, line 10).

The algorithm builds data blocks with fixed size n as data is made available sequentially. Then, when a data block contains n data points, the algorithm can start the training procedure of the ML classifiers. In this work, we define a data block as $D_t = ((X, y)_1, ..., (X, y)_n)$, where t denotes the current time step. After pre-processing the data, the algorithm starts by calculating the imbalance ratio r_t between set P_t that contains minority instances and set M_t that contains majority instances in the current data block D_t (Algorithm 3.1, line 8), which acts to track data distribution changes over time. Additionally, sets M_t and P_t are used to calculate the number of classifiers in the ensemble c_t (q.v. Algorithm 3.1, line 9).

Here, we explore an example to illustrate the workings of Algorithm 3.1. Let us say that initially $t = 1$, $c_t = 10$, 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 c_t changing to 12. Next, if $c_t > m$ (the maximum number of classifiers), the algorithm oversamples set P_t by replicating all minority instances following the ratio $|M_t| \div m$ (q.v. Algorithm 3.1, line 11), which prompts the update of the best ensemble size. The algorithm then checks if the imbalance ratio has changed by some predefined factor d (it is worth mentioning that this value is domain-dependent and defined during the development process). Following this, the algorithm incorporates oversampled community feedback to introduce relevant data about the change of community view into the training procedure (q.v. Algorithm 3.1, line 19). Subsequently, c_t balanced datasets (B_t) are created from data block D_t. Each balanced dataset in B_t is composed of non-overlapping data points from M_t, which contains

1 **Algorithm:** Mini-batch Learning
2 **Input:** Current time step (t), 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);
3 **Output:** Trained ensemble (E);
4 Initialize ensemble size. $c \leftarrow 0$
5 Initialize the last imbalance ratio change. $r \leftarrow 0$
6 **while** *data block* D_t *is available* **do**
7 Pre-process D_t, no past data is used
8 Compute the current imbalance ratio. $r_t \leftarrow |P_t| \div |M_t|$
9 Compute the current best ensemble size. $c_t \leftarrow |M_t| \div |P_t|$
10 **if** $c_t > m$ **then**
11 Oversample minority class instances $\in P_t$ by $|M_t| \div m$
12 Update c_t with the new value for $|P_t|$
13 **end**
14 **if** $c_t > c_{t-1}$ *and* $r_t \div r < 1 - d$ **then**
15 Compute the number of new classifiers. $o \leftarrow c_t - c_{t-1}$
16 Update the last imbalance ratio change $r \leftarrow r_t$
17 **end**
18 Emphasize set F_t by oversampling with ratio $|P_t| \div |F_t|$
19 Add F_t to data block D_t
20 **for** *i=1;* $i \leq c_t$*; i++* **do**
21 Get a subset $S_{t,i}$ from M_t, where $|S_{t,i}| = |P_t|$ and $S_{t,i} \cap S_{t,u} = \emptyset$ $(u = 1, 2, \ldots, i-1)$
22 Create a balanced dataset. $B_{t,i} = S_{t,i} \cup P_t$
23 **end**
24 Train c_t classifiers $\in E$ with datasets B_t for e epochs
25 Obtain the sum of relevance scores (sr_f) for violations
26 **end**

Algorithm 3.1: The mini-batch learning procedure for the ensemble of classifiers (FeDAL).

the original majority class data points and the feedback data (F_t), and all data points from P_t (q.v. Algorithm 3.1, line 22). To conclude the ensemble's parameter update, the algorithm executes the training procedure for each base classifier in E using the balanced datasets in B_t.

Lastly, in line 25 (q.v. Algorithm 3.1), as FeDAL finishes training the ensemble, it is possible to obtain the features usually associated with violation by calculating a sum of relevance scores based on local interpretations. The sum of relevance scores of a feature (sr_f) is the sum of all local relevance scores calculated using Local Interpretable Model-Agnostic Explanations (LIME) (q.v. Section 2.5.1) for the instances present in the current data block D_t. Equation 3.1 describes the sum operation, where k is the number of occurrences of feature f according to the intervals obtained by LIME in the discretization step, and $\mathsf{LIME}(f_u, 1)$ is the calculated relevance score for the u^{th} occurrence of f, regarding its contribution to violation classification. FeDAL must only change the second parameter to 0 to get the relevance scores for the regular class.

$$sr_f \leftarrow \sum_{u=1}^{k} \mathsf{LIME}(f_u, 1) \tag{3.1}$$

1 **Algorithm:** Online Learning
2 **Input:** Current data point (d), data point is of majority class (m), data point is of minority class (p), data point is feedback (f), desired class distribution (g), max oversample (o), balance window with fixed size (W), max window size (s), LIME block size (z);
3 **Output:** Trained ensemble (E);
4 Initialize the ensemble of classifiers. $E \leftarrow \{NeuralNetworks\}$
5 Set initial ensemble size c
6 Set the initial index classifier to train. $q \leftarrow 0$
7 **while** *data point is available* **do**
8 Pre-process d, with running statistical values
9 Train with the majority data point. $h \leftarrow True$
10 Add the label from d to W
11 **if** $|W| > s$ **then**
12 Remove oldest label $\in W$
13 **end**
14 Update the number of data points n
15 Get the number of the latest majority instances l from W
16 Compute minority class oversampling rate. $r \leftarrow l \div c$
17 **if** $r > o$ **then**
18 The majority class is undersampled. Thus, data point d is not used for training
19 Train with the majority data point. $h \leftarrow False$
20 **end**
21 Execute Train Classifiers (call Algorithm 3.3)
22 Obtain the sum of relevance scores (sr_f) for z violations
23 **end**

Algorithm 3.2: The online learning procedure for the ensemble of classifiers (FeDAL).

3.2 Online Learning for Tabular Scenarios

Algorithm 3.2 and Algorithm 3.3 describe how FeDAL employs the procedure to train the ensemble of classifiers online. We build this approach based on the concepts of oversampling and undersampling described by S. Wang et al. (2015). However, we further enhance the data sampling strategy in the context of online learning by proposing a novel approach to calculate the resampling ratio for the minority class. Specifically, FeDAL only considers the classes of the latest data points to accurately capture the most recent data distribution.

The first step in Algorithm 3.2 is to create the ensemble of classifiers E (line 4). In this case, the community can define the number of base classifiers $c \in E$ from expert knowledge or through initial experiments. Next, for each data point d that is made available (i.e., for each action in an online community), the algorithm pre-processes d using the running statistical values (q.v. Algorithm 3.2, line 8).[2] Unlike data block-based approaches, it is necessary to adapt the calculation of the statistical values to account for the single instance update. Therefore, online learning employs Equations 3.2 and 3.3 to perform this calculation (Montiel et al., 2021). Additionally, it is worth noting that this procedure may result in lower accuracy compared to those in mini-batch and batch learning.

$$RM_t \leftarrow RM_{t-1} + ((V_t - RM_{t-1}) \div n_t) \tag{3.2}$$

[2]We are interested in the mean and the sum of squares since these are used to normalize the incoming data point.

```
1  Algorithm: Train Classifiers Online
2  Input: Current data point (d), data point is of minority class (p), data point is feedback
     (f), minority class oversampling rate (r), ensemble of classifiers (E), train with majority
     data point (h), index classifier to train (q);
3  Output: Trained ensemble (E);
4  if p is True then
5  |   Oversampling data point d by rate r
6  |   for Classifier e ∈ E do
7  |   |   Train e with the oversample data point
8  |   end
9  else if f is True and h is True then
10 |   for i=q; i ≤ q + ((|E| ÷ 2) + 1); i++ do
11 |   |   Train classifier E_i with d
12 |   end
13 |   Update index classifier to train. q ← i
14 else
15 |   if h is True then
16 |   |   Train classifier E_q with d
17 |   |   Update index classifier to train. q ← q + 1
18 |   end
19 end
```

Algorithm 3.3: The training classifiers procedure for the online learning approach.

RM_t is the updated running mean at time t for each feature that describes the action resulting in d, RM_{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 (SQ_t):

$$SQ_t \leftarrow SQ_{t-1} + (V_t - RM_{t-1}) * (V_t - RM_t) \tag{3.3}$$

Since it is impossible to know the data distribution for the complete dataset in online learning, it is necessary to decide, as interactions happen, which portions of the data will be used for training. To tackle this challenge, FeDAL checks for concept drift by calculating the change in data distribution. It is important to note that our framework does not use all data points previously processed to perform this calculation. Instead, it only considers a fixed number of instances in a window W of fixed size (the maximum size s of this window is assumed to be provided in advance and is domain-dependent). W contains only the information about the label of the latest s data points, with no feature information present. Hence, W is a list where each element is either 0 or 1, regular or norm-violating behavior, respectively. In line 12 (q.v. Algorithm 3.2), FeDAL updates W as new instances are available.

Following the creation of W, the algorithm computes the sampling rate for the minority class, considering the number of majority instances l present in W and the number of classifiers c in the ensemble (q.v. Algorithm 3.2, line 16). This is useful in combination with the ensemble to balance the data points seen by the classifiers, dealing with the imbalanced nature of a certain domain. Next, the algorithm compares the oversampling ratio r with the maximum oversample value o. If $r > o$, the majority class is undersampled (q.v. Algorithm 3.2, line 18), and h is set to False. The parameter h controls whether the ensemble uses the current data point for training.

Upon obtaining the values for the sampling strategy, the classifiers in E can be trained following three distinct directions, depending on the current data point (q.v. Algorithm 3.2, line 21). We

design these directions to improve efficiency, as each path is dedicated to training specific base classifiers. Concretely:

- if the current data point belongs to the minority class (p is $True$), all base classifiers are trained (q.v. Algorithm 3.3, line 6). Given the imbalanced nature of the dataset with a scarcity of data instances representing the minority class, each classifier needs to incorporate these data points during the training process. Otherwise, they will not learn to identify minority instances (violating behavior).
- if the current data point contains feedback from the community (f is $True$) and the algorithm can train using the majority class (h is $True$), then half plus one (i.e., the majority) classifiers are trained (q.v. Algorithm 3.3, line 10). Training the majority of base classifiers serves to incorporate changing community views, as FeDAL's decision-making process employs majority voting.
- if the current data point is not of the minority class (p is $False$) and the algorithm can use this data point for training (h is $True$), then one base classifier is updated (q.v. Algorithm 3.3, line 16). Considering the higher frequency of instances of the majority class, we limit the amount of data points seen by the base classifiers, aiming at having a balanced number of instances (of the minority and majority classes) used during the training process.

To indicate which of the base classifiers is trained with the current data point d, we use the index q. As a result, since the algorithm considers only one training data point at a time, different base classifiers see different training instances. This feature described in Algorithm 3.3 differs completely from our initial proposal presented in (Freitas dos Santos et al., 2022a), where only the resampling strategy changed, with the algorithm training all classifiers using all data points. This led to the algorithm requiring additional time to complete the training procedure.

In Algorithm 3.2 (line 22), FeDAL calculates the sum of relevance score (sr_f). Since the online learning procedure only considers one data point at a time, LIME requires the algorithm to process z instances before executing this step. Although Algorithm 3.2 does not retain past data to learn norm violations, incorporating interpretability with LIME requires access to this data. The value of z is not predetermined and is domain-dependent. We envision each online community to provide this value. However, to ensure consistency in the number of processed instances across different experiments, we set z equal to the data block size used in the mini-batch procedure (q.v. Algorithm 3.1).[3]

3.3 Batch Learning for Tabular Scenarios

Using the batch approach to train the ML model requires the treatment of the entire dataset D. Also, it is not possible to update the ML model incrementally. Instead, a new ML model must be created to incorporate new knowledge, and the training process starts from the beginning. These characteristics make this approach the least suitable to deal with a stream of data that arrives sequentially (as in the Wikipedia use case of article edits). Algorithm 3.4 builds on top of the traditional batch learning procedure (Russell & Norvig, 2002) by incorporating two modifications: 1) emphasize the importance of feedback data by oversampling; and 2) oversample the minority class to achieve a desired number of classifiers, using this number as a metric to choose the sampling strategy.

The training procedure starts by oversampling the data points that received feedback (q.v. Algorithm 3.4, line 4). The ratio to oversample is defined by the number of minority class instances $|P|$

[3]In the experiments described in Section 3.5, we set z to 512.

1 **Algorithm:** Batch Learning
2 **Input:** Past dataset (A), new dataset (T), set of majority instances (M), set of minority instances (P), set of instances with feedback (F), max number of classifiers (m), past weight (w_A), new weight (w_T), and number of epochs (e);
3 **Output:** Trained ensemble (E), sum of relevance score (sr);
4 Emphasize F by oversampling with ratio $|P| \div |F|$
5 Assign weights for the past dataset. $A \leftarrow w_A$
6 Assign weights for the new dataset. $T \leftarrow w_T$
7 Create the complete dataset $D \leftarrow A \cup T$
8 Pre-process D
9 Compute the best ensemble size. $c \leftarrow |M| \div |P|$
10 **if** $c > m$ **then**
11 | Oversample minority class instances $\in P$
12 | Update c with the new value for $|P|$
13 **end**
14 **for** *i=1; i* $\leq c$*; i++* **do**
15 | Get a subset S_i from M, where $|S_i| = |P|$ and $S_i \cap S_u = \emptyset$ $(u = 1, 2, \ldots, i-1)$
16 | Create a balanced dataset. $B_i = S_i \cup P$
17 **end**
18 Train c classifiers in E with datasets B for e epochs
19 Obtain the sum of relevance scores (sr_f) for violations

Algorithm 3.4: The batch learning procedure for the ensemble of classifiers (FeDAL).

divided by the number of feedback instances $|F|$, which ensures that D contains a representative number of minority and feedback cases.

D is composed of two groups: past data $A(X, y)$ that contains data previously used to train the initial ML model; and new data $T(X, y)$ that contains data not used to train the initial ML model and that may contain concept drift. We separate these into two groups because, for batch learning, we set higher weights to newer data points. This step emphasizes the importance of new data to incorporate community view changes since ML classifiers use weights during the training process. In Algorithm 3.4, line 5 assigns the weights to A with value w_A and line 6 to T with value w_T. As described above, w_T is bigger than w_A.[4] In addition to these two sets, we define F as the subset of T, in which $(X, y) \in F$ received community feedback, a step exclusive to new data $T(X, y)$. Keeping past data $A(X, y)$ is important only to ensure that relevant information, not updated through community feedback, is preserved during the batch learning process when training restarts from scratch.

With the entire dataset D created, after weight assignment and concatenation, the algorithm pre-processes D (q.v. Algorithm 3.4, line 8). Specifically, this step calculates the statistical values using each data point $\in D$. In line 9, the algorithm computes the best number of classifiers for the ensemble, using the ratio between the number of majority instances $|M|$ and the number of minority instances $|P|$. This step ensures that the algorithm can create balanced datasets for each classifier in the ensemble. The next step checks if the best number of classifiers c is bigger than the maximum number m of classifiers that can be created (q.v. Algorithm 3.4, line 10). If $c > m$, then the minority class is oversampled, and c is updated with the new value for $|P|$. The main goal is to avoid creating an unnecessarily large number of classifiers to ensure computational efficiency. In Algorithm 3.4 (line 16), the algorithm proceeds to create the balanced datasets to train each of the base classifiers (q.v. Algorithm 3.4, line 18).

Lastly, like in mini-batch learning (q.v. Algorithm 3.1), the batch procedure also obtains the sum

[4] Section 3.5 presents the values used in this work.

of relevance scores for violations (q.v. Algorithm 3.4, line 19). However, batch learning calculates sr_f using the complete dataset instead of considering only a data block.

3.4 Computational Complexity

This section provides a complexity analysis of the learning algorithms employed by the FeDAL framework. We are particularly interested in four key aspects that have the potential to impact the computational requirements of our solution. As such, we examine operations in which the execution time may vary because of the following:

- Maximum number of classifiers in an ensemble (m) — this parameter constrains the computational resources that FeDAL uses, limiting the ensemble size.
- Dataset size required by each training approach ($|D|$).
- Number of majority and minority instances, $|M|$ and $|P|$, respectively.
- Number of instances that received feedback ($|F|$).

Examining the mini-batch case (q.v. Algorithm 3.1), the algorithm considers a single data block D_t instead of the complete dataset D. As a result, mini-batch learning requires $O(|D_t|)$ time in the pre-processing step and $O((|M_t| \div m) - |P_t|)$ and $O(|P_t| \div |F_t|)$ for oversampling. Furthermore, Algorithm 3.1 requires $O(m)$ time to create balanced datasets and train the classifiers. Thus, the overall computational complexity of the mini-batch approach can be expressed as $O(|D_t| + (|M_t| \div m) - |P_t| + (|P_t| \div |F_t|) + m)$.

Next, we analyze the training procedure for online learning (q.v. Algorithm 3.3). This approach handles single data points during execution (i.e., the input size does not change, and the algorithm will always process one instance at a time), resulting in a different complexity analysis compared to mini-batch and batch learning. The oversampling operation requires $O(l \div m)$ time, where l represents the number of minority class instances obtained from the fixed window W. Moreover, the worst-case scenario involves training all classifiers in the ensemble, which requires $O(m)$ time. Therefore, the online learning procedure is described with a $O((l \div m) + m)$ complexity.

Lastly, in batch learning (q.v. Algorithm 3.4), specifically in line 4, the algorithm emphasizes the feedback data, which takes $O(|P| \div |F|)$ time. The complete dataset's pre-processing step (line 8) also requires $O(|D|)$ time. As minority class oversampling considers the maximum number of classifiers, the algorithm needs $O((|M| \div m) - |P|)$ time to complete this task. The training procedure in line 12 has a $O(m)$ complexity. Consequently, the overall computational complexity of FeDAL's batch learning is $O(|D| + (|M| \div m) - |P| + (|P| \div |F|) + m)$. The complexity analysis only differs from mini-batch learning due to the dataset size.

To extend this analysis to a practical use case, we empirically evaluate the training time required by these approaches in Section 3.6. Specifically, we present a case where online learning benefits from not training all classifiers in the ensemble, which results in reduced training time. Additionally, we compare batch with mini-batch learning, demonstrating how the constant use of the complete dataset is detrimental to execution performance.

3.5 Experiments

The goals of this section are twofold: 1) to evaluate how batch and incremental learning approaches learn to detect norm violations in a context with an imbalanced dataset and concept drift, which

Group	Features
Written Edition	LANG_ALL_ALPHA; LANG_EN_PRONOUN
Comment on Edition	COMM_LEN; COMM_LEN_NO_SECT
Article After Edition	SIZE_CHANGE_RESULT; SIZE_CHANGE_CHARS
Time of Edition	TIME_TOD; TIME_DOW
User's Profile	HIST_REP_ARTICLE; USER_EDITS_DENSITY
Page's History	PAGE_AGE; WT_NO_DELAY
Reversions	HASH_REVERTED; HASH_REC_DIVERSITY

Table 3.1: Example of features present in the taxonomy groups.

denotes the shift in community views that reclassify actions previously considered norm violation as regular behavior; and 2) to understand the features of an action that lead to norm-violation detection. As described in Section 1.1.3, we investigate the use case of Wikipedia article edits, focusing on the "no vandalism" norm. Thus, in this context, norm-violating behavior is referred to as *vandalism*. Since FeDAL handles tabular datasets, an important step in our solution is to map the instances of actions (Wikipedia article edits) to a set of features and categorize them.

We manually created a taxonomy that separates the features that compose an article edit into categories, illustrating their relationship (q.v. Figure 3.1). We consider the features described in (West & Lee, 2011) plus three that are present in the dataset: LANG_MARKUP_IMPACT, the measure of the addition/removal of Wiki syntax/markup; LANG_EN_PROFANE_BIG and LANG_EN_PROFANE_BIG_IMPACT, the measure of addition/removal of English profane words. In this case, features ending with _IMPACT are normalized by the difference in article size after the edition.

In Figure 3.1, we allocate the features into four main groups. The first group, "User's Direct Actions," represents aspects related to the user's action, i.e., the text edit in a Wikipedia article. This group is further divided into four sub-groups: a) "Written Edition," which contains features about the content of the edited text; b) "Comment on Edition," incorporating features related to the comments provided by the user on the article edit; c) "Article After Edition," which includes features linked to the article edit after the user completed the action; and d) "Time of Edition," which contains features about the time of the edit.

The second group, "User's Profile," represents general information related to the user. The third group, "Page's History," describes how articles have changed with past edits. Lastly, the fourth group, "Reversions," focuses on information related to past reversions.[5]

In total, these groups have 57 features and the label (specifying whether the edit is vandalism or not). However, due to simplification purposes, Table 3.1 only presents a subset of those features.[6] The definitions of these features are presented in (West & Lee, 2011). Specifically, LANG_ALL_ALPHA and LANG_EN_PRONOUN denote the percentage of text added that is alphabetic (excluding symbols and numbers) and the measure of pronoun addition/removal, respectively. COMM_LEN describes the size of the edit comment, and COMM_LEN_NO_SECT indicates the size of the comment revision. Regarding the article after the edit, SIZE_CHANGE_RESULT depicts the difference in the article size after post-edit, while SIZE_CHANGE_CHARS quantities the number of characters removed or added by the edit. TIME_TOD represents the edit time, and TIME_DOW refers to the day of the week. HIST_REP_COUNTRY measures the user's behavior historically, while USER_EDITS_DENSITY quantifies the total user edits. PAGE_AGE indicates the time in seconds since page creation, and WT_NO_DELAY represents the WikiTrust score (West & Lee, 2011). Lastly, HASH_REVERTED indicates whether the article content receives reversion, and HASH_REC_DIVERSITY measures the percentage of recent page edits made by the user.

[5] A reversion is when an article reverts to a version before the vandalism.

[6] For the complete taxonomy, the reader can refer to bit.ly/complete-taxonomy.

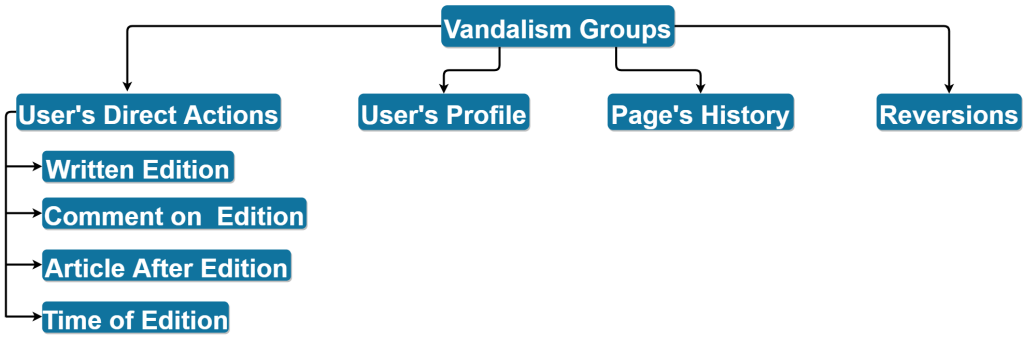


Figure 3.1: The groups of features present in our use case.

3.5.1 Learning to Detect Vandalism

The dataset contains 32,439 edits, with 2,394 vandalism edits (around 7%) and 30,045 regular edits (around 93%). Since the dataset is highly imbalanced, we use stratified 10-fold cross-validation to evaluate the performance. Then, we repeat the process three times for each fold to enhance the estimate of FeDAL's performance. In total, 30 experiments are executed for each of the three learning approaches. By stratifying the dataset, we guarantee that each fold of the validation process maintains the data distribution between majority and minority classes (regular and vandalism edits). Classification recall, Area Under the Curve of the Receiver Operating Characteristics (AUC-ROC), and Area Under the Curve of Precision-Recall (AUC-PR) are the chosen metrics for evaluation.

We conduct a two-step experiment to evaluate FeDAL's performance. Here, concept drift refers to changes in community members' views about the components of an action (e.g., the set of features) that indicate the presence of vandalism behavior. Specifically, we consider changes in the label of article edits as interactions unfold.

- ***Learn to detect vandalism before introducing concept drift:*** In this case, the goal is to evaluate if the proposed algorithms can simply learn vandalism without considering changes in community view. We separated the dataset into two halves (D_1 and D_2). D_1 contains 14.597 article edits, while D_2 contains 14.598 article edits. The data in D_1 does not present concept drift and only has the imbalanced dataset with regular and vandalism edits, specifically with 13,520 regular and 1,077 vandalism edits. Since this part of the experiment aims to learn to detect vandalism before introducing concept drift, only D_1 was used. The testing dataset T_1 contains 3,005 regular edits and 239 vandalism edits, which maintains the data distribution of the original dataset D_1.

- ***Learn to detect vandalism after introducing concept drift:*** In the second part of the experiment, we introduce changes in the community view within the D_2 data, resulting in a concept drift. The aim is to evaluate if FeDAL can learn to detect vandalism in the presence of concept drift. For the incremental learning cases (mini-batch and online), the algorithm does not process the original data D_1: it only uses the new information in D_2 and updates the ML model previously trained with D_1. However, for the batch learning case, the algorithm must process the entire dataset $D_1 \cup D_2$, giving different weights for D_1 and D_2 to emphasize the new data points. Additionally, the ML models created in the first part of the experiment are not updated. Instead, new ML models are created based upon the training set $D_1 \cup D_2$. Since we do not have real feedback from community members, we simulate it by introducing concept drift to the D_2 dataset. We add concept drift by changing which edits $\in D_2$ are labeled as vandalism (swap of class labels). We do so as follows: using only the vandalism subset $V_{D_2} \in D_2$, we apply the K-Means clustering algorithm to generate

subgroups that contain data points most similar between themselves (Krishna & Murty, 1999). We obtain four subgroups from this process, $G = \{0: 555, 1: 435, 2: 87, 3: 1\}$. The idea of having groups with similar data points is based on our assumption that community feedback would be consistent and that this consistency would be present in these datasets. Thus, our interpretation of the results naturally comes from this consistency. We swap the class label from all data points $\in G_0$ for this experiment.[7] Then, class distribution in D_2 changes, resulting in 14,075 regular and 523 vandalism edits. Consequently, the imbalanced ratio changes as well to $IR = 0.037$. The testing dataset T_2 also suffers change, with 117 vandalism and 3,127 regular edits.

We use the Keras library (Chollet et al., 2015) to build the ensemble, adopting the Feed-Forward Neural Network (FNN) as the base classifier. FeDAL employs identical FNN architectures in all cases to compare batch, mini-batch, and online learning. The base classifiers comprise an input layer with 57 units (each representing one feature of an article edit) and two hidden layers, the first with six nodes and the second with three nodes. Mini-Batch Gradient Descent, Stochastic Gradient Descent (SGD), and Batch Gradient Descent, with a learning rate equal to 0.01, are used as optimizers. The loss function is the Cross-Entropy (De Boer et al., 2005).

It is imperative to set specific parameters for the learning algorithms. For all three approaches, the maximum ensemble size is 12. In mini-batch learning, the batch size is 512, and the number of epochs is 128. In online learning, the desired class distribution is 50% for both regular and vandalism classes, which the algorithm achieves by employing undersampling and oversampling. In batch learning, the incoming data is assigned a weight (w_T) 100 times greater than the weight assigned to past instances (w_A). We evaluate each ensemble's performance after training with 512 edits to ensure comparability across different learning approaches. Hence, for each block of 512 edits that the algorithm uses to train the ML model, we evaluate the performance, generating graphs that describe the performance progress over time. The parameters described above are found empirically and may affect the classifiers' performances, indicating that different values should be explored for specific domains.

3.5.2 Understanding the Features that Contribute to Vandalism Detection

Besides learning to detect vandalism, our work also focuses on understanding the features of an edit that contribute to the ML model's output. To fulfill this goal, we use the Local Interpretable Model-Agnostic Explanations (LIME) algorithm as the interpretability tool (q.v. Section 2.5.1). The first step is to train the ML models. Then, LIME provides a score for each feature of an edit being explained. This score describes the influence of the feature on the classification result. Since the investigated use case is a binary classification problem, the feature score influences the classification in two directions, either contributing to classifying it as regular or as a vandalism edit.

The experiments with LIME also consider datasets D_1 and D_2, before and after concept drift, respectively. Thus, for each learning approach—batch, mini-batch, and online—there are two sets of relevant features (q.v. Section 3.6). Since we aim to investigate the relevant features for vandalism detection, the experiments only consider the testing data in which the class label is vandalism. Therefore, for the first part of the experiment, 239 edits from the vandalism subset $V_{T_1} \in T_1$ are explained by LIME, and for the second part, 117 edits $V_{T_2} \in T_2$ are explained.

Upon defining the testing dataset, the next step is configuring LIME parameters. In this study, LIME generates 1024 data points around each evaluated edit and calculates the relevance score of

[7] For future work, we plan to conduct experiments where not all data points have a swap of class labels. This experiment design aims to evaluate how FeDAL addresses potential inconsistencies in the labeling process that may arise when people interact online.

all 57 features, ranking them based on their score values. Additionally, LIME discretizes continuous variables, transforming each continuous feature into a set of ranges. This assists community members in understanding the features that contributed the most to vandalism detection. For instance, the feature "LANG_ALL_ALPHA" $\in [0, 1]$ is a continuous variable representing the percentage of the text that is only alphanumeric. In our case, LIME discretizes this feature in four different ranges:

1. "LANG_ALL_ALPHA" ≤ 0.68.
2. $0.68 <$ "LANG_ALL_ALPHA" ≤ 0.72.
3. $0.72 <$ "LANG_ALL_ALPHA" ≤ 0.8.
4. "LANG_ALL_ALPHA" > 0.8.

To obtain these values, LIME uses two sets of data points. For the incremental learning cases, it only considers the latest data points seen in the training procedure, 512 edits. On the other hand, for batch learning, LIME considers the entire dataset. Thus, changes in these ranges are expected due to the different training approaches.

3.6 Results and Discussion

This section presents the results of applying FeDAL to the task of learning vandalism detection in the context of Wikipedia article edits. Additionally, it presents information about the relevant features of an edit that contribute to classifying an action as vandalism. Lastly, it describes ablation studies providing two key comparisons: 1) between an ensemble and a single model approach; and 2) between oversampling by replication and SMOTE.

3.6.1 Learning to Detect Vandalism

As presented in Section 3.5, we separate the experiments into two parts, before and after introducing concept drift. Thus, the dataset is also separated into two parts (D_1 and D_2).

Figure 3.2 presents the overall recall graph. Concept drift is introduced after 14,500 edits are processed. For D_1, the learning curves for the three approaches are similar. However, mini-batch presents significantly better performance than batch learning, as attested by the Friedman-Nemenyi Test (q.v. Figure 3.3a).[8] We use this test to statistically compare the different approaches in this work (ensemble of classifiers). To demonstrate that the approaches have a statistical difference in performance, the corresponding average rank must have a difference bigger than the critical difference (CD) value, which is calculated based on the Studentized range statistics (here, we adopt the statistical significance $\alpha = 0.05$) (Demšar, 2006). After introducing concept drift, the algorithms behave differently. Even though the batch learning approach has the worst performance as soon as changes are introduced, by the end of the training procedure, it is significantly better than the other two approaches (q.v. Figure 3.3b). This shows that batch learning can build new models that learn even with concept drift by setting weights for newly added data points and providing enough data.

However, since we are dealing with an imbalanced dataset, evaluating only the overall recall is not representative enough. Thus, we aim to further detail the performance metrics for each class present in our user case (regular and vandalism).

[8]The complete statistical results are presented in Appendix A.

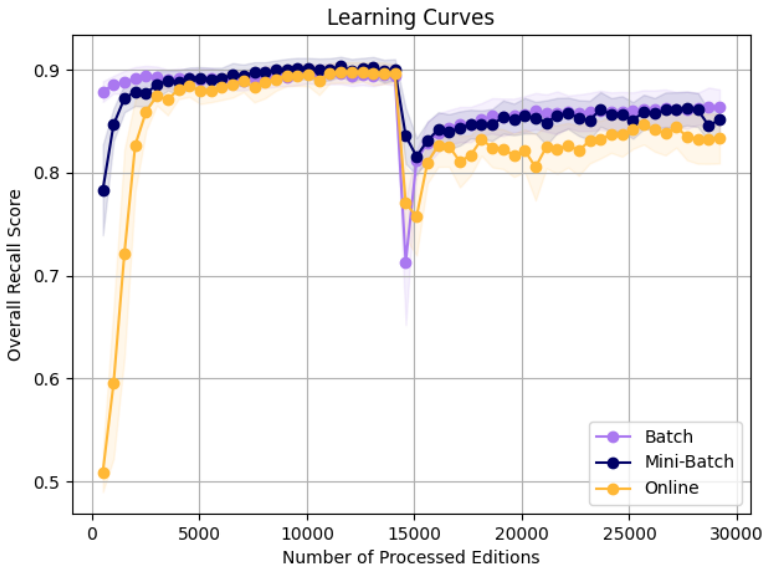
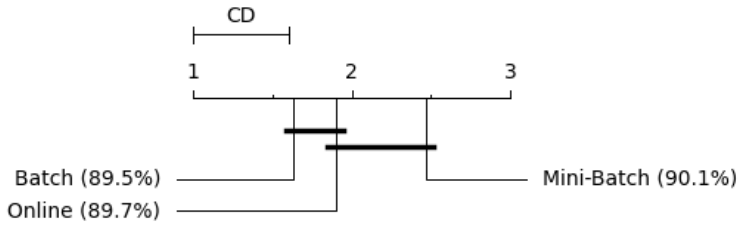
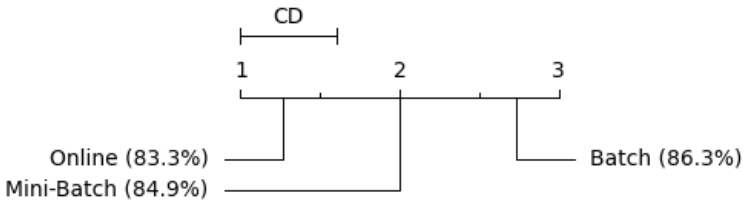


Figure 3.2: Overall recall for the batch, mini-batch, and online cases. After around 14,500 processed edits, concept drift is introduced.



(a) Before introducing concept drift.



(b) After introducing concept drift.

Figure 3.3: Friedman-Nemenyi test comparing overall recall for the batch, mini-batch, and online learning approaches. Tables A.1 and A.2 present the complete statistical results.

Regarding the classification performance for the regular class (q.v. Figure 3.4), mini-batch learning outperforms the other two approaches in the first part of the experiment. The difference is statistically significant, as attested by the Friedman-Nemenyi test (q.v. Figure 3.5a). As the training procedure advances, the three approaches present less performance instability in classifying regular edits compared to the vandalism case. We argue that this behavior is an effect of the high number of regular data points (majority class) and the re-label strategy, which increases the imbalance ratio. Additionally, since batch learning handles concept drift by adding a bias towards vandalism (minority class), this approach presents higher variation and instability after introducing concept

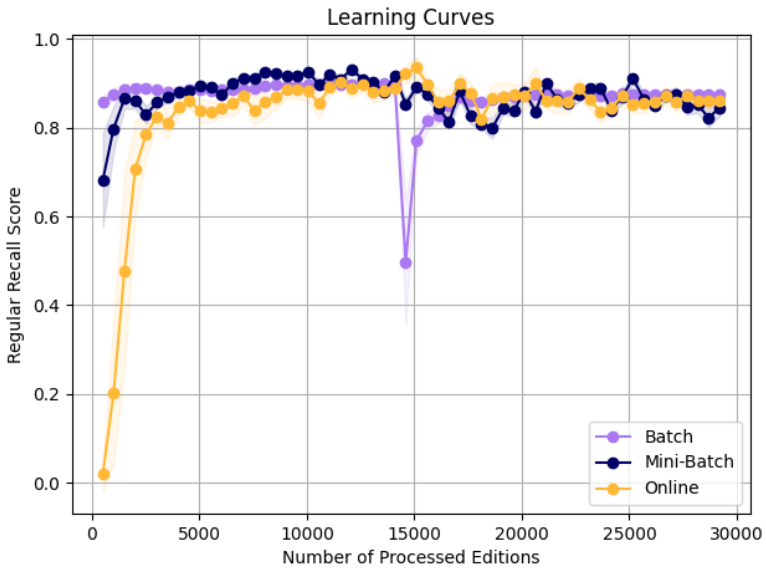
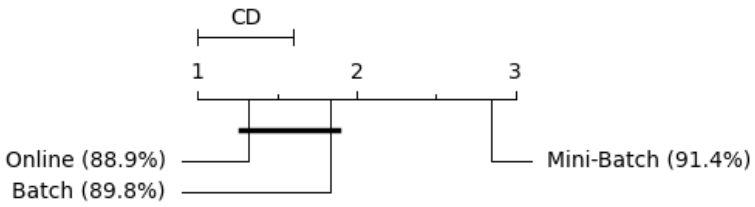
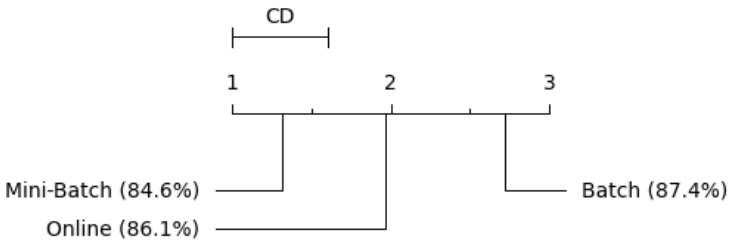


Figure 3.4: Regular recall for the batch, mini-batch, and online cases. After around 14,500 processed editions, concept drift is introduced.



(a) Before introducing concept drift.



(b) After introducing concept drift.

Figure 3.5: Friedman-Nemenyi test comparing regular recall for the batch, mini-batch, and online learning approaches. Tables A.3 and A.4 present the complete statistical results.

drift, consequently leading to a significant drop in performance. In other words, it improves the performance of vandalism detection at the cost of detecting regular edits (q.v. Figure 3.6). This behavior is corrected as the algorithm processes more edits. At the end of the training procedure, the batch learning performance is significantly better than the other two approaches (q.v. Figure 3.5b). To conclude, although we can see that the difference is statistically significant, in practice, all three approaches present impressive results (specifically, regular recall values greater than 0.8) in classifying regular edits.

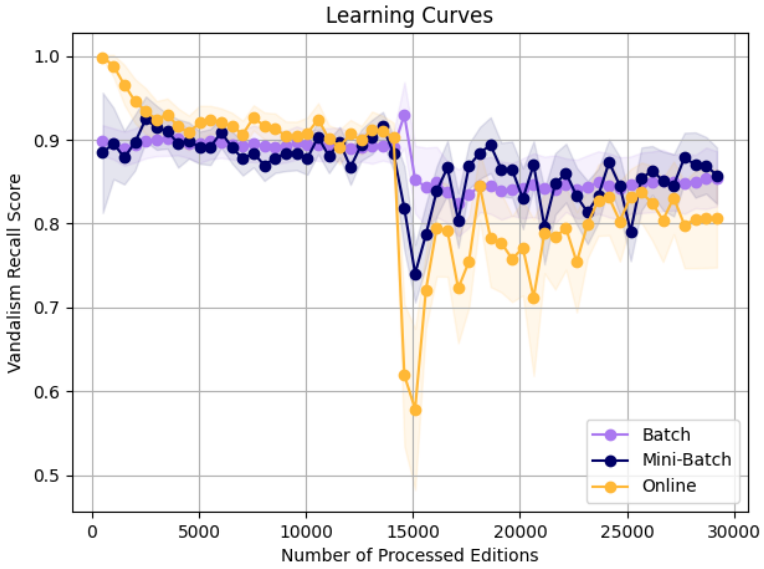
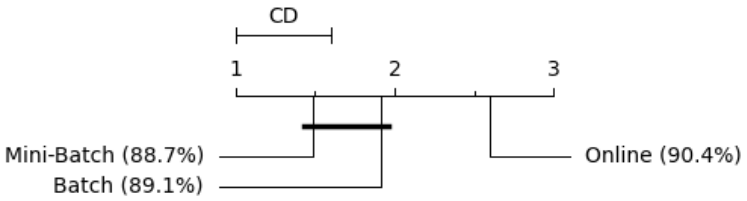


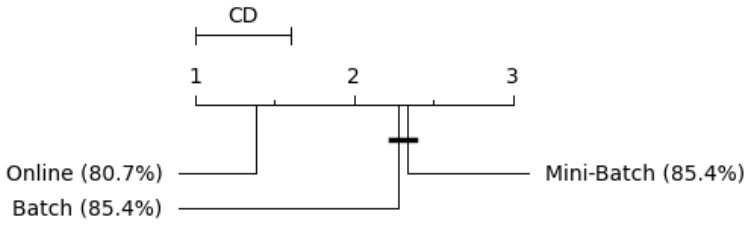
Figure 3.6: Vandalism recall for the batch, mini-batch, and online cases. After around 14.500 processed editions, concept drift is introduced.

To analyze the results for vandalism edits, we use Table 3.2, the learning curve of Figure 3.6, and the Friedman-Nemenyi tests (q.v. Figures 3.7a and 3.7b). It is possible to conclude that before introducing concept drift, the online learning approach outperforms the other two in correctly classifying vandalism. At the beginning of the training procedure, we see that online learning presents a positive bias towards the vandalism case at the cost of regular classification. However, as the ensemble processes new data, the performances for both classes reach similar levels. Additionally, after introducing concept drift, the three approaches present a performance drop, and the instability properties for the incremental learning cases are evident (q.v. Figure 3.6). The instability can be explained by how the training procedure incorporates incoming data. Since incremental learning approaches use only parts of the dataset (a data block or a single instance) to train the ensemble, these approaches are not guaranteed to immediately train the models with representative vandalism data. Specifically, this drawback is accentuated by the data sampling strategy used in the online learning case (Z. Li et al., 2020; S. Wang et al., 2015, 2018), leading to a 10% performance drop. Although, in the beginning, mini-batch learning also presents this instability property, towards the end of the training procedure, it reaches a performance comparable to the batch learning approach (q.v. Figure 3.7b). Finally, despite describing the introduction of concept drift only once (in an abrupt manner), we note that the same approach could be used in the case of multiple concept drifts since our results show that the proposed framework can continuously learn as interactions happen.

Figure 3.8 presents the recall for data that suffered the swap of class labels after introducing concept drift (the re-labeled dataset). As feedback data is incorporated, the framework's overall performance drops almost to zero (illustrated by the beginning of Figure 3.8). This behavior is expected since we are giving a different label to old definitions and introducing it in the dataset. However, as more data becomes available, the ensemble can learn that certain article edits should no longer be classified as vandalism, thus adapting to the new community view. Figure 3.9 demonstrates that online learning significantly outperforms the other approaches in this case, which can be attributed to the bias towards the majority class and the re-label strategy (increasing the imbalance ratio by changing vandalism to regular edits). Additionally, we observe that batch learning presents less instability and no significant difference in performance compared to mini-batch learning.



(a) Before introducing concept drift.



(b) After introducing concept drift.

Figure 3.7: Friedman-Nemenyi test comparing vandalism recall for the batch, mini-batch, and online learning approaches. Tables A.5 and A.6 present the complete statistical results.

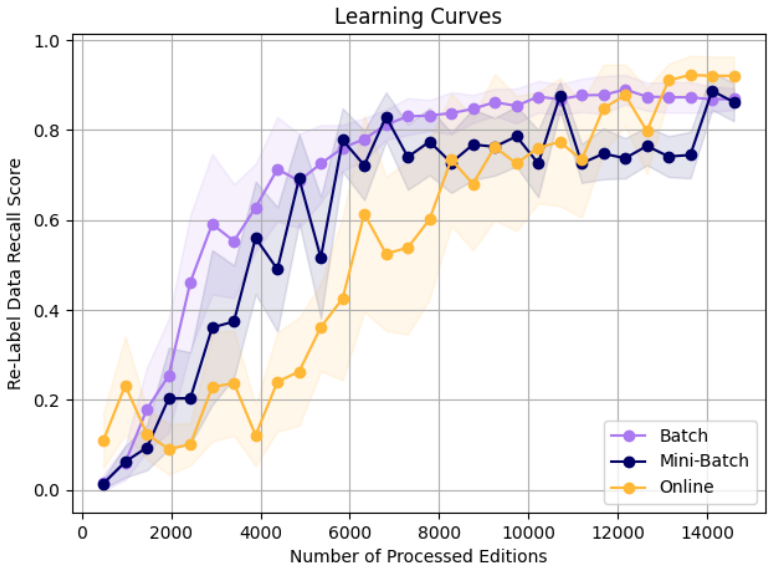


Figure 3.8: Re-label recall for the batch, mini-batch, and online cases after concept drift.

In addition to the recall, other interesting metrics are relevant to evaluate the performance of ML classifiers. Two such metrics are the Area Under the Curve of the Receiver Operating Characteristics (AUC-ROC) and the Area Under the Curve of Precision-Recall (AUC-PR). Here, we are interested in the relationship between the majority and minority instances, specifically the ratio between these two values, which is the basis for calculating the AUC-ROC and AUC-PR scores. Therefore, we aim to investigate how useful these metrics can be for a domain with imbalanced datasets and concept drift.

Both metrics (AUC-ROC and AUC-PR) rely on the confusion matrix from the test procedure. For AUC-ROC, we note that, at first sight, this is not a suitable metric to evaluate ML models trained

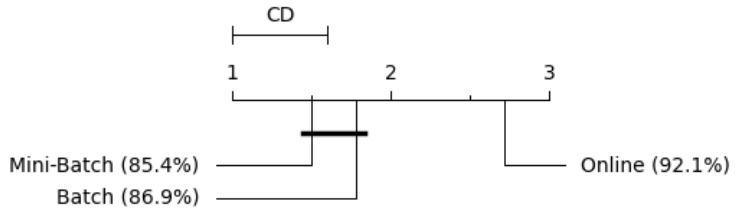


Figure 3.9: Friedman-Nemenyi test comparing re-label recall for the batch, mini-batch, and online learning approaches after concept drift is introduced. Table A.7 presents the complete statistical results.

with an imbalanced dataset since it overestimates the model's performance with a bias towards the true positive rate. In this case, a large change in the number of false positives (the regular edits wrongly classified as vandalism edits) leads only to a small change in the ROC score (J. Davis & Goadrich, 2006; H. He & Garcia, 2009). However, suppose the domain requires giving the most importance to identifying true positives even at the cost of false positives. In such a scenario, this metric may be useful for understanding performance. We argue that this is the context for norm violation in online community domains since it may be more critical to block offensive actions than to make mistakes classifying regular behavior.

Results show that all three approaches present results above 90% for AUC-ROC (q.v. Figure 3.10 and Figure 3.11a). This outstanding result indicates that the ML model is very useful in identifying true vandalism cases, at the cost of giving little importance to the regular edits that were wrongly classified. As concept drift is introduced, all three approaches present a significant drop in performance, with batch learning receiving the highest impact on its AUC-ROC score. However, as the training procedure continues, the batch approach performs better than the two incremental learning approaches (with a statistically significant difference in performance as depicted in Figure 3.11b). This behavior is in accordance with the instability properties present in these cases and explained in Figure 3.4 and Figure 3.6.

The AUC-PR plot shows the relationship between precision and recall by comparing the false positive and the true positive rates (Saito & Rehmsmeier, 2015). It also uses precision and captures the effect of the large number of regular instances on the algorithm's performance (differently from AUC-ROC) (Saito & Rehmsmeier, 2015). To obtain a more realistic evaluation of this metric, Siblini et al. (2020) propose a calibrated version of AUC-PR, in which the goal is to make the metric independent of the class prior (i.e., it monitors the performance based on the changing data distribution after introducing concept drift). Figure 3.12 shows that mini-batch and online learning perform significantly better than batch learning before introducing concept drift (q.v. Figure 3.13a). However, after concept drift, batch learning presents less instability, thus leading to a smaller drop in performance and the highest AUC-PR scores (q.v. Figure 3.13b).

In Figure 3.14, we present the training time (in seconds) necessary to learn to detect norm violation. The ML models process 512 article edits at each time step. Incremental learning takes a constant amount of time to train as new data is made available. Since mini-batch and online learning do not consider past data, they only train the ML model on the latest article edits. Differently, batch learning uses the entire dataset at each time step, which increases the number of instances to process by 512 anytime new knowledge is incorporated. Hence, incremental learning approaches present better training time and less data management complexity for an online community domain with a constant stream of actions. By the end of the training procedure, online learning trains faster than mini-batch learning since it updates only part of the ensemble of classifiers and processes fewer data points. This second effect happens as a result of the online learning undersampling strategy, which decreases the number of data points to compute by decreasing the number of majority class instances. It is worth noting that this behavior is domain-dependent. If the data distribution is

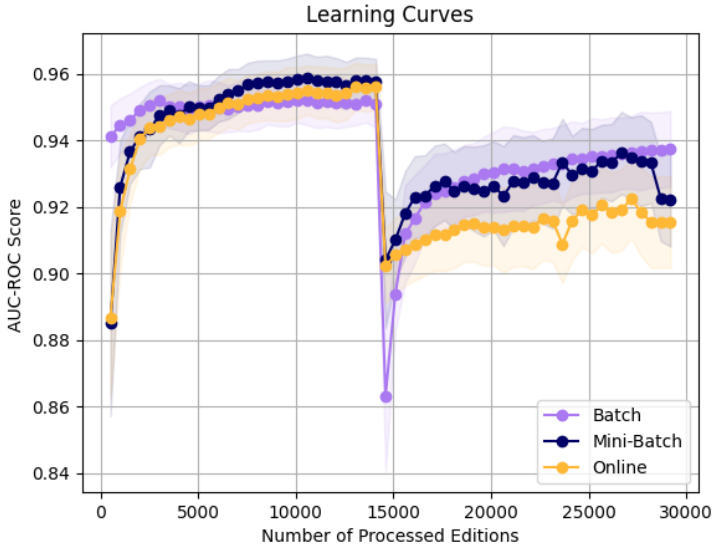
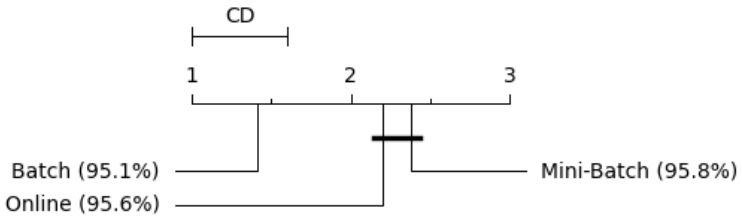
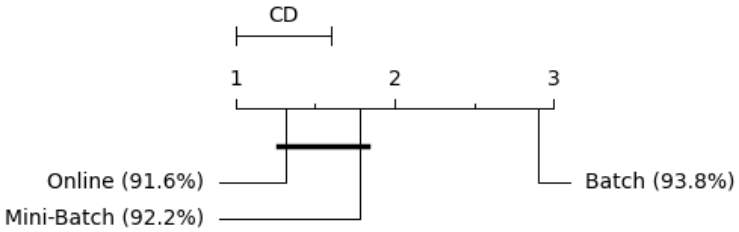


Figure 3.10: AUC-ROC scores for the batch, mini-batch, and online learning approaches. After around 14.500 processed editions, concept drift is introduced.



(a) Before introducing concept drift.



(b) After introducing concept drift.

Figure 3.11: Friedman-Nemenyi test comparing AUC-ROC scores for the batch, mini-batch, and online learning approaches. Tables A.8 and A.9 present the complete statistical results.

different in another setting, then the decrease in training time might not be noticed.

Table 3.2 and Table 3.3 summarize the results by presenting the performance information for each approach concerning the different datasets (D_1 and D_2). For the Friedman-Nemenyi Test, the null hypothesis is that the samples are drawn from the same distribution with the statistical significance value set to $\alpha = 0.05$. This test suits our experimental setup as it allows for the comparison of multiple (three in our case) learning approaches without relying on assumptions of normality (the performance estimates follow a normal distribution) or homogeneity of variance (the

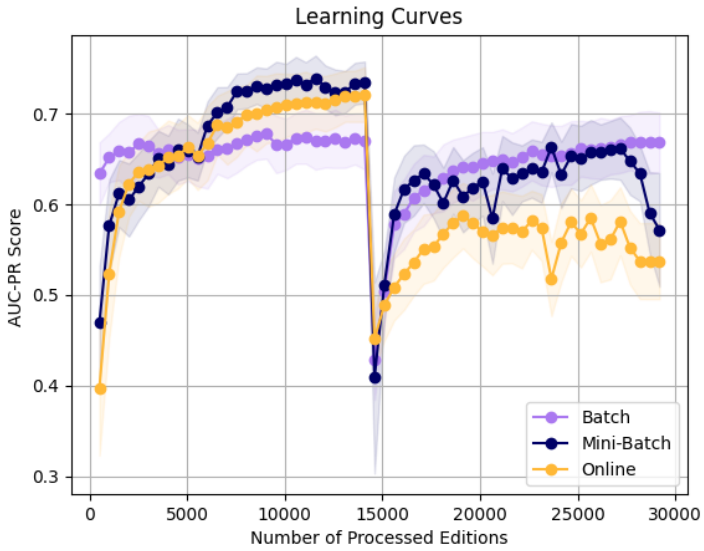
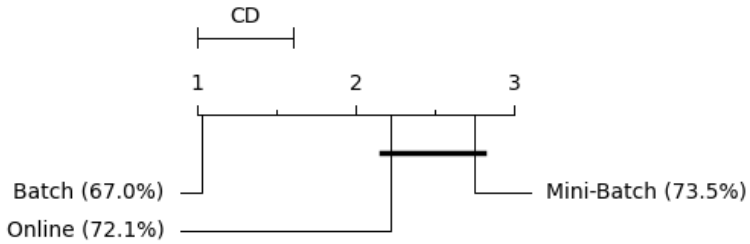
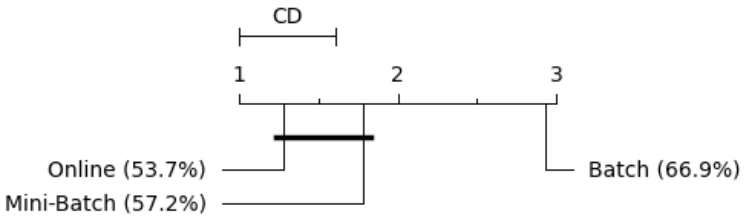


Figure 3.12: AUC-PR scores for the batch, mini-batch, and online cases. After around 14.500 processed editions, concept drift is introduced.



(a) Before introducing concept drift.



(b) After introducing concept drift.

Figure 3.13: Friedman-Nemenyi test comparing AUC-PR scores for the batch, mini-batch, and online learning approaches. Tables A.10 and A.11 present the complete statistical results.

dispersion of the performance estimates is similar across different learning techniques), which our repeated cross-validation procedure does not guarantee. Additionally, this test is robust to outliers, ensuring that extreme performance values (regardless of being too good or too bad) do not greatly impact overall estimates. Results show that the batch and mini-batch approaches present higher performance scores to classify vandalism edits, with a significant statistical difference compared to online learning. Specifically, the batch approach offers a more stable performance. It adapts more quickly to concept drift, while the online approach presents a bias towards the majority class and,

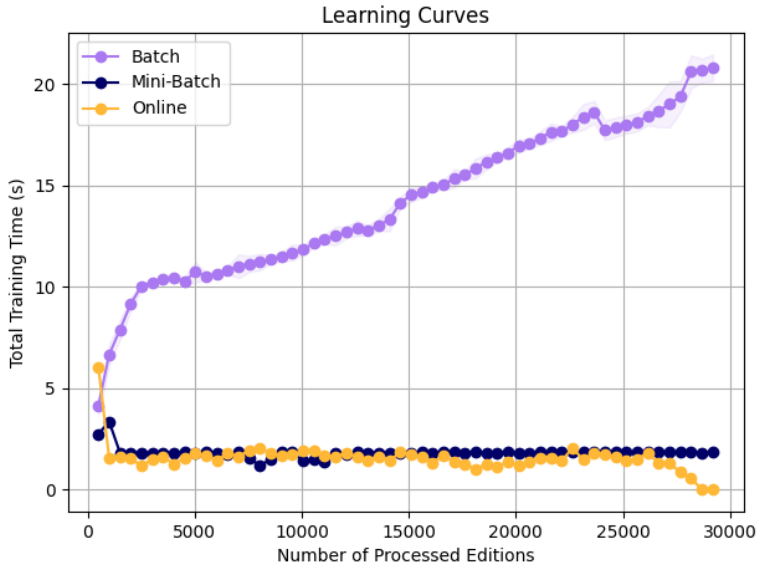


Figure 3.14: Training time for the batch, mini-batch, and online cases.

Dataset	Method	Recall Mean±Std	Regular Recall Mean±Std	Vandalism Recall Mean±Std
Original	Batch	0.895±0.01	0.898±0.07	0.891±0.0019
	Mini-Batch	**0.901**±0.01	**0.914**±0.008	0.887±0.021
	Online	0.897±0.008	0.889±0.0015	**0.904**±0.017
Concept Drift	Batch	**0.863**±0.017	**0.874**±0.007	**0.854**±0.033
	Mini-Batch	0.849±0.019	0.846±0.016	**0.854**±0.039
	Online	0.833±0.024	0.861±0.021	0.807±0.058
Re-label	Batch	0.869±0.024	X	X
	Mini-Batch	0.854±0.055	X	X
	Online	**0.921**±0.043	X	X

Table 3.2: Summary of Batch, Mini-Batch, and Online Learning performance results applied to the Wikipedia article edits dataset. Three settings are considered: 1) dataset before introducing concept drift (original); 2) dataset after introducing concept drift, swap of class labels; and 3) dataset with only the data that suffered the change (re-label).

consequently, in our concept drift case, a bias towards the changed data (with the highest recall to the re-label dataset). Additionally, the significant drop in performance in online learning occurs even when using different sampling and imbalance strategies, such as that in (Montiel et al., 2021; S. Wang et al., 2015).

In conclusion, our findings indicate that all three learning approaches can effectively deal with the challenge of learning to detect norm violations (vandalism) in the context of an online community (Wikipedia). However, each approach has its complexities and requirements. On the one hand, batch learning offers more stability and faster recovery from the drop in performance after introducing concept drift, but at the cost of adding complexity to assign weights and keep the entire dataset consistent (handling data storage and management). On the other hand, incremental learning approaches offer faster training time and are designed to constantly update the ML model. Thus, we argue that it is imperative to choose the most appropriate approach considering the specific requirements of a certain online community.

Dataset	Method	AUC-ROC Mean±Std	AUC-PR Mean±Std
Original	Batch	0.951±0.007	0.670±0.0031
	Mini-Batch	**0.958**±0.007	**0.735**±0.023
	Online	0.956±0.007	0.721±0.003
Concept Drift	Batch	**0.938**±0.011	**0.669**±0.033
	Mini-Batch	0.922±0.014	0.572±0.063
	Online	0.916±0.014	0.537±0.042

Table 3.3: Considering the composed metrics, the summary of Batch, Mini-Batch, and Online Learning performance results applied to the Wikipedia article edits dataset. Two settings are considered: 1) dataset before introducing concept drift (original); 2) dataset after introducing concept drift, swap of class labels.

3.6.2 Understanding the Features that Contribute to Vandalism Detection

The second part of the experiment focuses on the interpretability of the ML models. In our work, we are interested in investigating which features of an article edit contribute the most to detecting vandalism. This is relevant not only to promote transparency to end-users but also as a valuable tool during development (M. Du et al., 2019), enabling engineers to debug and obtain a deeper comprehension of the elements of an action the model focuses on (Hong et al., 2020; Iadarola et al., 2021). This understanding helps prevent drawbacks associated with black-box models, like attributing relevance to features of an article edit that the community does not consider while defining vandalism. For example, let us assume that a community member edited an article by adding text with profane words. In this case, these words would make the feature PROFANE_IMPACT have a bigger value, contributing to vandalism detection (since this feature is usually present when such classification occurs). Thus, with information on the features that triggered the ML model's output, the community member can understand the type of edit the entire community defines as violating behavior and specifically the parts of the action that contain the issue.

Similar to the previous performance metrics, we also conduct a two-part experiment for interpretability: 1) before concept drift; and 2) after introducing concept drift. In total, the results of these experiments are presented in five bar charts (q.v. Figures 3.15–3.19). Specifically, each training approach has one chart that describes the relevant features for vandalism detection under both conditions (q.v. Figures 3.15–3.17). In addition, we directly compare the three learning approaches also under both conditions (q.v. Figures 3.18 and 3.19). It is worth noting that although we do not include this information in the present work, the LIME tool also provides data about the features that contribute the most to the classification of regular edits.

To build the interpretability charts, we calculate the sum (sr_f) of all local relevance scores associated with vandalism detection. The relevance scores are obtained by applying LIME to each instance that the model processes. Equation 3.1 describes the sum operation.

We then present the top ten features with the highest sum of relevance scores for each learning approach. These features can provide enough information for community members to understand how the most relevant parts of their actions affect the classification output. However, different relevant features might exist when comparing learning techniques or under diverse conditions (before and after concept drift). Thus, in some cases, more than ten features might be present. While some features are consistently relevant in different cases, indicating their importance to vandalism detection in general, we also note changes in the set of relevant features, particularly in the ranking order and impact of these features on the classification output. This finding suggests that the base ML models have distinct internal parameters affected by the training approaches, prompting LIME to find a different set of relevant features.

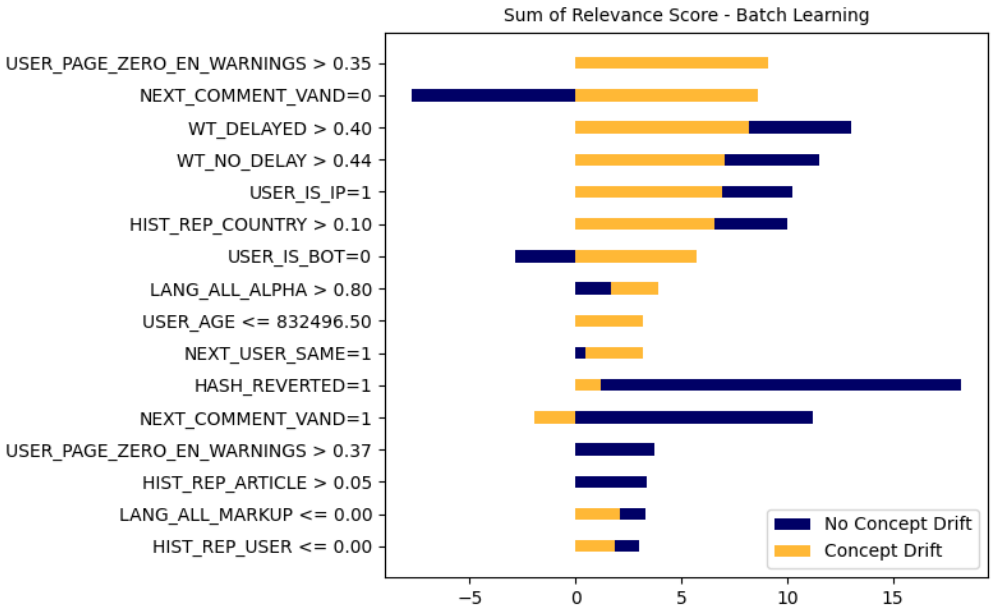


Figure 3.15: The sum of relevance scores for vandalism classification. The graph presents the top ten features of the batch learning case before and after introducing concept drift.

Figure 3.15 describes interpretability for the ensemble of classifiers trained with batch learning. The figure reveals that, before the introduction of concept drift, the presence of the feature NEXT_COMMENT_VAND (a piece of metadata indicating if the following comment is related to vandalism) with a value of 1 increases the probability of vandalism classification. However, after introducing concept drift, this feature loses relevance and starts to negatively impact vandalism detection, while the opposite, NEXT_COMMENT_VAND equals 0, gains relevance. This result is consistent with the modification introduced in the dataset, where 97.5% of the data points that suffered the swap of class labels from vandalism to regular had this feature altered. Before concept drift, NEXT_COMMENT_VAND equals 1 was present in 59% of vandalism edits. After concept drift, only about 18.8% of the vandalism edits possessed this feature with a value of 1. Additionally, the feature HASH_REVERTED also shows a significant loss in relevance score, as 66.5% of the data points initially contained this feature, but after introducing concept drift, the percentage dropped to around 35% of the data. In contrast, 96.7% of re-labeled instances had this feature equal to 1.

Similar to batch learning, it is possible to note the changes in relevance scores for the mini-batch approach (q.v. Figure 3.16). Specifically, the feature NEXT_COMMENT_VAND shows a similar behavior, with a loss in relevance score after the introduction of concept drift. Additionally, Figure 3.16 reveals slight differences in the discretization process employed by LIME. Features like HIST_REP_COUNTRY and USER_PAGE_ZERO_EN_WARNINGS have different ranges before and after concept drift, which is a result of the sequential nature of incremental learning approaches. In these cases, LIME processes only the last data block instead of using the complete dataset. Despite these differences, the above features remain relevant for violation classification in both cases, with high relevance scores.

The interpretability analysis in the online learning scenario presents an interesting finding (q.v. Figure 3.17). The sum of relevance scores for most features is low, only around 5 points, suggesting that most of the 57 features have similar relevance in vandalism detection. Consequently, this may have negative implications for user experience on the platform, as community members would need to make significant changes to their actions to comply with the community's view. The features

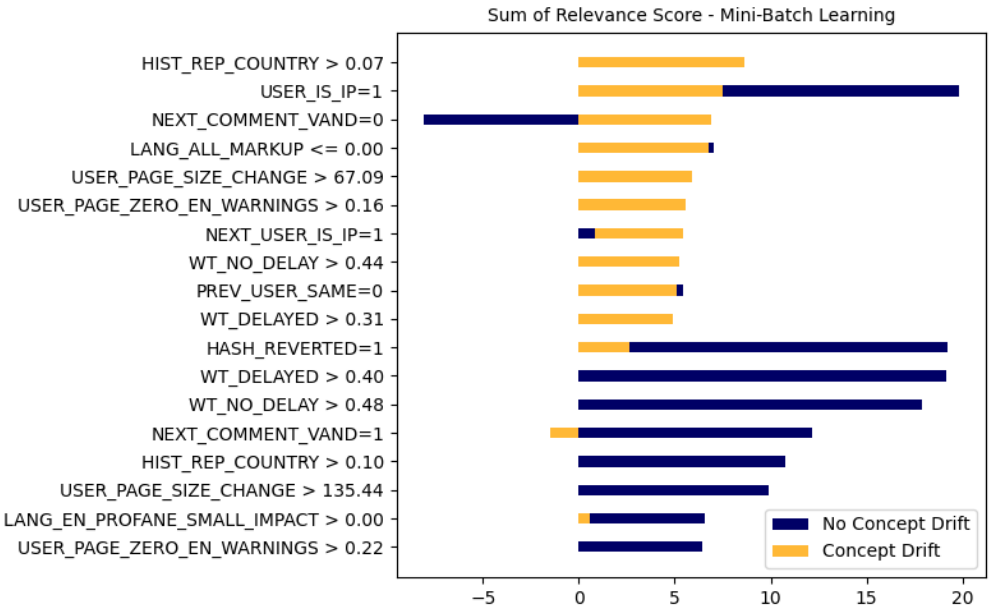


Figure 3.16: Sum of relevance scores for vandalism classification. The graph presents the top ten features for the mini-batch learning case before and after introducing concept drift.

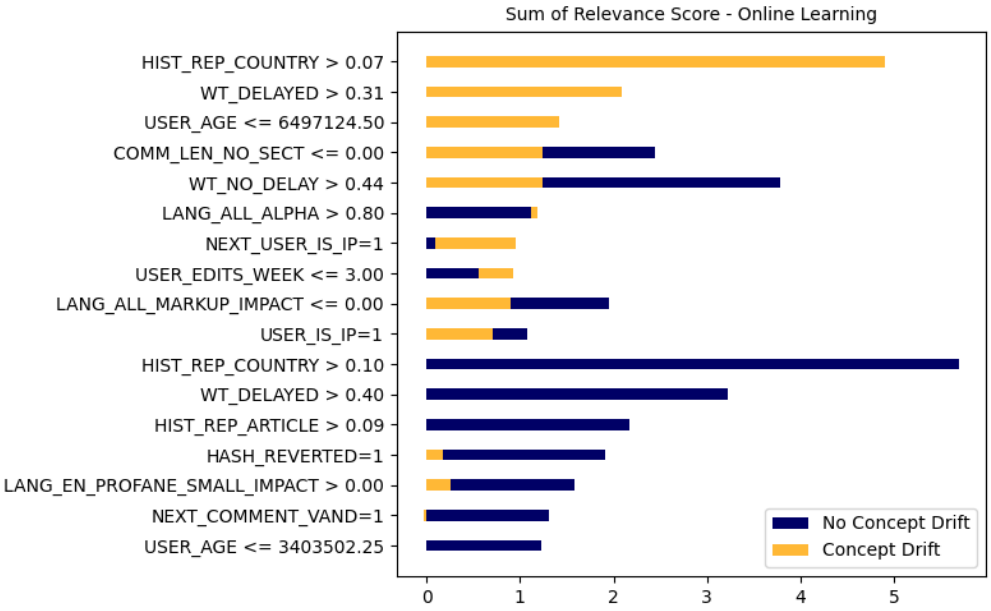


Figure 3.17: Sum of relevance scores for vandalism classification. The graph presents the top ten features for the online learning case before and after introducing concept drift.

that have the most significant impact are HIST_REP_COUNTRY and WT_DELAYED. Similar to mini-batch learning, these features also show a difference in range before and after concept drift. This difference is also an effect of the sequential nature of the data in online learning scenarios.

Following our analysis of the relevant features for the same learning approach, we investigate the

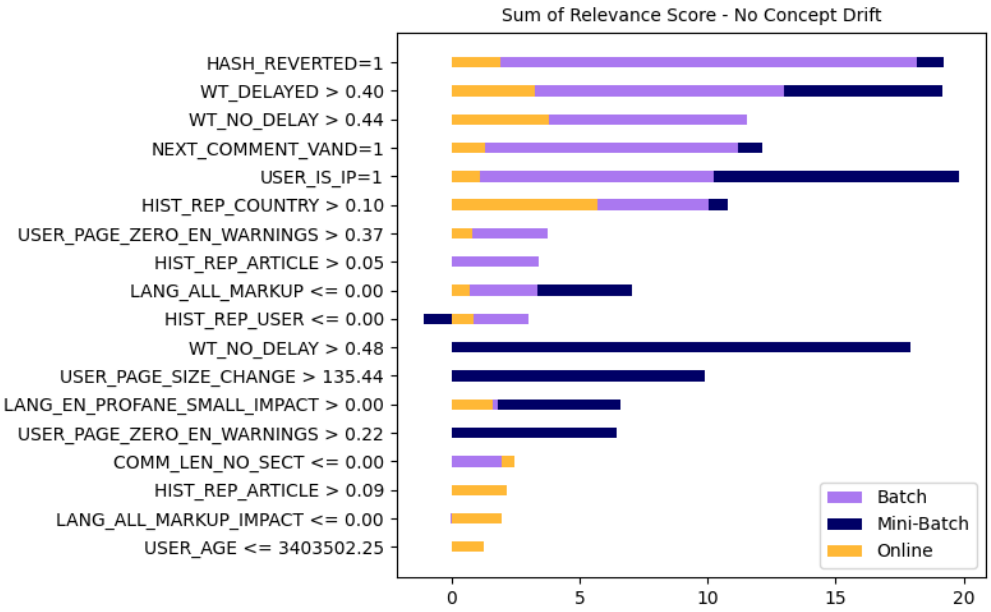


Figure 3.18: Sum of relevance scores for vandalism classification. The graph presents the top ten features for batch, mini-batch, and online learning cases before concept drift.

changes in relevance scores across different techniques, as depicted in Figures 3.18 and 3.19. We aim to demonstrate that not only do the data and base classifiers influence the relevant features for vandalism classification, but the training approach adopted also plays a crucial role in defining these features and their scores. Our results reveal that online learning presents the lowest relevance scores for most features, with values of approximately 5 points, while batch and mini-batch present higher values of approximately 15 and 20 points. The relevance scores directly impact a model's predicted probabilities. Consequently, on average, online learning yields the lowest probability values for vandalism classification. Before concept drift (q.v. Figure 3.18), mini-batch produces the highest sum of relevance scores, while after concept drift (q.v. Figure 3.19), batch learning presents the highest scores. This difference can be attributed to the distinct weight update procedures adopted by these techniques during the training process, which leads to variation in their output and feature relevance, despite processing the same dataset when training is completed. Specifically, batch learning uses the entire dataset in a single epoch to compute the model's parameters, calculating the gradients for all data points. In contrast, mini-batch learning uses only a subset of the data in an epoch, and online learning uses the gradients of each edit individually. Additionally, we note differences in the ranges of feature values, such as those observed for WT_NO_DELAY, HIST_REP_ARTICLE, and USER_PAGE_ZERO_EN_WARNINGS. The procedure employed by LIME is responsible for these divergences since it creates the explanations using either the complete dataset (batch learning) or only a subset of the training data (incremental learning).

In summary, interpretation is an interesting manner of understanding why the ML model classified an article edit as vandalism. This enhances FeDAL with the ability to better serve the community, providing useful information about a user's action. However, we also note the importance of investigating different explainability techniques to obtain a more robust and consistent set of relevant features considering the different learning approaches.

We aim to use this interpretability information for future work, providing data that supports community members to review edits that help correct the output of ML models. In other words, when a user performs an action, and the ML classifier detects it as violating behavior, FeDAL

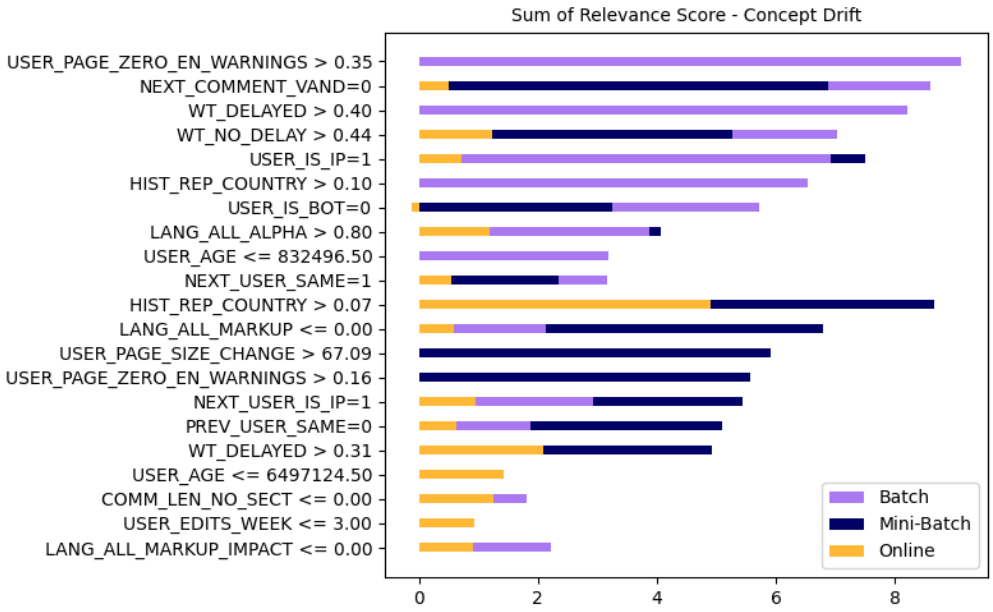


Figure 3.19: Sum of relevance scores for vandalism classification. The graph presents the top ten features for batch, mini-batch, and online learning cases after introducing concept drift.

would provide the user with the reasons for such detection. If the user disagrees with the output, they can engage with other community members to evaluate the action with information about the relevant features, assessing the correctness of the ML classifier accordingly. The user's original edit will not be relabeled until the community reaches a consensus, and only then will the system allow the relabeling of that edit.

Dataset	Mini-Batch		
	Regular	Vandalism	Re-label
Original	**0.0004**	0.5291	X
Concept Drift	**0.0000**	**0.0000**	**0.0000**

Table 3.4: P-values of comparing the recall performance of ensemble and a single classifier training using mini-batch learning. Performance values are presented in Figure 3.20. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Dataset	Online		
	Regular	Vandalism	Re-label
Original	**0.0000**	**0.0000**	X
Concept Drift	**0.0000**	**0.0000**	**0.0017**

Table 3.5: P-values of comparing the recall performance of ensemble and a single classifier training using online learning. Performance values are presented in Figure 3.21. Bold p-values indicate results below the critical value $\alpha = 0.05$

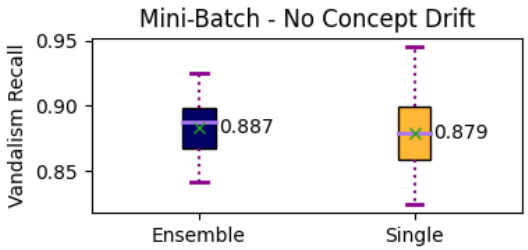
3.6.3 Ablation Studies

In this section of our work, we investigate the impact of two modifications on FeDAL. First, we compare the ensemble approach against the use of a single model. Since a single model approach is simpler and computationally less expensive, we argue that it is important to demonstrate the advantages of employing ensemble learning in FeDAL. Second, we compare data sampling techniques, presenting results for replication and SMOTE (Chawla et al., 2002). This comparison is relevant because SMOTE is the original approach employed by the DUE algorithm (Z. Li et al., 2020) that serves as the basis for FeDAL.

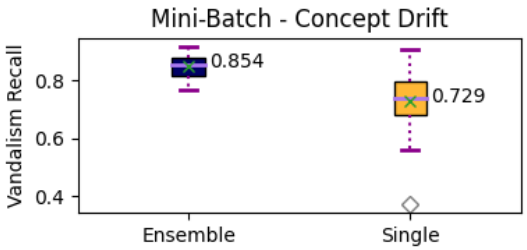
Ensemble and single model. Figure 3.20 presents the empirical results obtained by comparing an ensemble against a single classifier trained with mini-batch learning. Our findings show that, before concept drift, the identification of vandalism and regular behavior remains largely unaffected regardless of the adopted model, as demonstrated by Figures 3.20a and 3.20c. However, the ensemble significantly outperforms the single model approach after the introduction of concept drift (q.v. Figures 3.20b and 3.20d). In Table 3.4, the statistical significance is attested by the Wilcoxon Signed-Rank Test (Woolson, 2007),[9] which compares the performance of both approaches. The null hypothesis is that the samples were drawn from the same distribution, and the critical value $\alpha = 0.05$. Furthermore, the single model exhibits a strong bias towards re-labeled instances (q.v. Figure 3.20e), indicating that the model learned the new patterns at the cost of previously acquired knowledge.

Next, we analyze the other incremental learning approach, namely online learning, as depicted in Figure 3.21. For both conditions, before and after introducing concept drift, the single model presents the worst performance in vandalism classification (q.v. Figures 3.21a and 3.21b), with a strong bias towards the majority class (q.v. Figures 3.21c and 3.21d). The differences are statistically significant, as attested in Table 3.5. Additionally, the single model has poor performance and high variance in identifying re-labeled instances (q.v. Figure 3.21e). These findings demonstrate that online learning significantly benefits from adopting an ensemble of classifiers, as this approach presents the most substantial gains in performance.

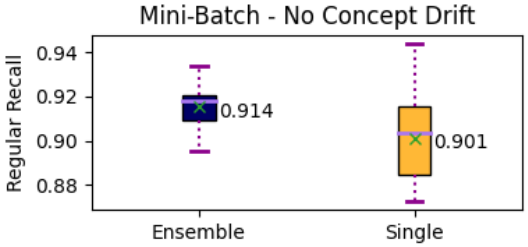
[9]For the ablation studies, we use the Wilcoxon Signed-Rank Test instead of the Friedman-Nemenyi Test. This is the case because we are comparing two distributions at each evaluation (ensemble and single classifier, and replication against SMOTE).



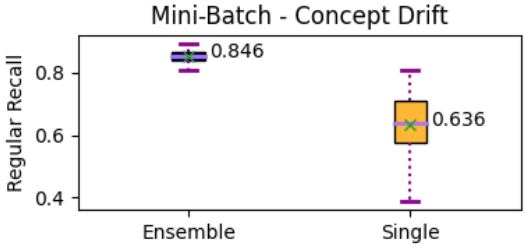
(a) Mini-Batch - no concept drift; the metric is vandalism recall. Ensemble and single classifiers.



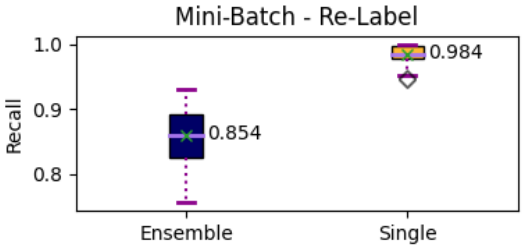
(b) Mini-Batch - with concept drift; the metric is vandalism recall. Ensemble and single classifiers.



(c) Mini-Batch - no concept drift; the metric is regular recall. Ensemble and single classifiers.



(d) Mini-Batch - with concept drift; the metric is regular recall. Ensemble and single classifiers.

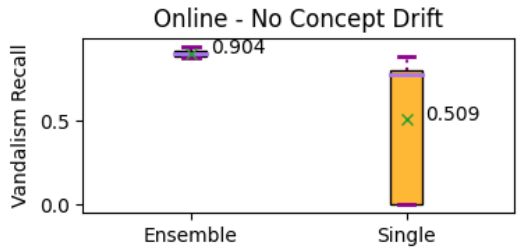


(e) Mini-Batch - with concept drift; the metric is re-label recall. Ensemble and single classifiers.

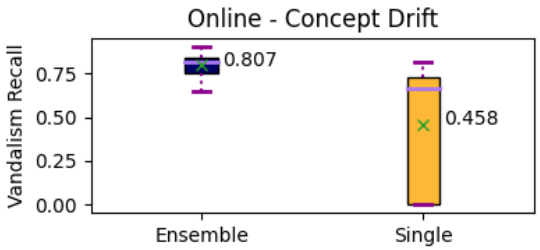
Figure 3.20: Box-plots for ablation studies considering the mini-batch learning approach to compare the performance of ensemble against single classifiers.

Models trained with the traditional batch learning approach (q.v. Figure 3.22) also present performance differences as attested by the statistical results in Table 3.6. It is worth noting that this difference is especially relevant for the regular class. In this case, the performance of a single model presents several outliers across different executions of the cross-validation procedure (q.v. Figures 3.22c and 3.22d). Concretely, the single model failed to provide an appropriate solution to the problem, resulting in poor performance values. Thus, although the single model can learn to identify minority class instances (including those that suffered the swap of class labels), it does so at the cost of the majority (regular) class. This finding highlights the advantage of adopting an ensemble approach, which offers the best performance by integrating algorithmic and data sampling techniques to tackle the imbalanced nature of the data.

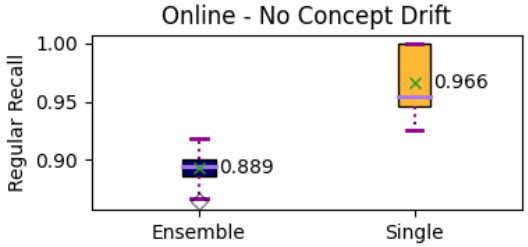
In summary, the results in Figures 3.20 and 3.21 demonstrate that using an ensemble approach improves the performance of our incremental learning techniques under different conditions. Additionally, for the traditional batch learning case, Figure 3.22 shows that the single model approach is subject to variance and that we can address this issue by adopting an ensemble of classifiers, which is an expected benefit of using ensemble methods (Galar et al., 2011).



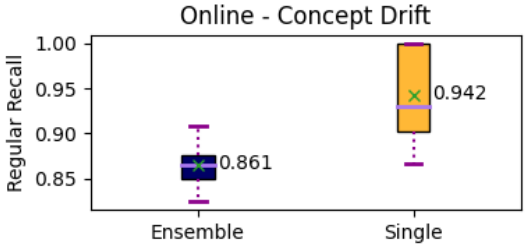
(a) Online - no concept drift; the metric is vandalism recall. Ensemble and single classifiers.



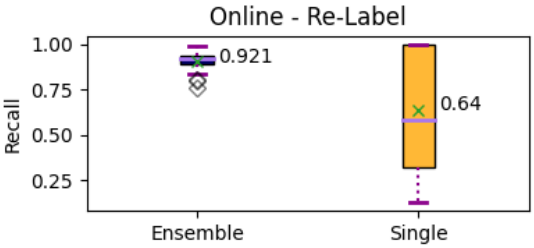
(b) Online - with concept drift; the metric is vandalism recall. Ensemble and single classifiers.



(c) Online - no concept drift; the metric is regular recall. Ensemble and single classifiers.



(d) Online - with concept drift; the metric is regular recall. Ensemble and single classifiers.



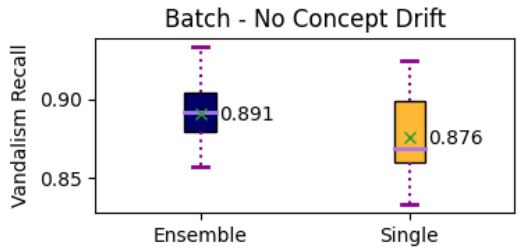
(e) Online - with concept drift; the metric is re-label recall. Ensemble and single classifiers.

Figure 3.21: Box-plots for ablation studies considering the online learning approach to compare the performance of ensemble against single classifiers.

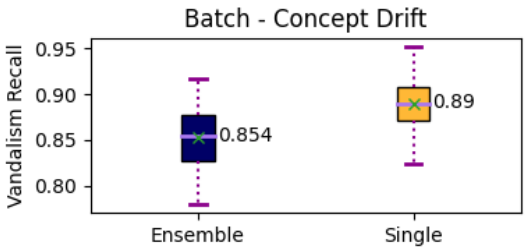
Dataset	Batch		
	Regular	Vandalism	Re-label
Original	**0.0001**	**0.0021**	X
Concept Drift	**0.0000**	**0.0000**	**0.0000**

Table 3.6: P-values of comparing the recall performance of ensemble and a single classifier training using batch learning. Performance values are presented in Figure 3.22. Bold p-values indicate results below the critical value $\alpha = 0.05$

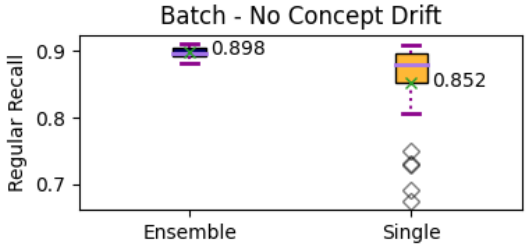
Replication and SMOTE. The mini-batch method adopted in our solution is based on the proposal presented by Z. Li et al. (2020), where they use SMOTE as the data sampling strategy. Additionally, other works have also shown the benefits of combining SMOTE and ensemble techniques in different fields, positively impacting the performance of the minority class classification (Fernández et al., 2018; Galar et al., 2011). However, in this research, we incorporate an oversample by replication strategy. Therefore, it is important to present a comparison between



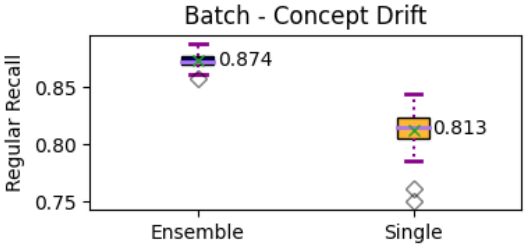
(a) Batch - no concept drift; the metric is vandalism recall. Ensemble and single classifiers.



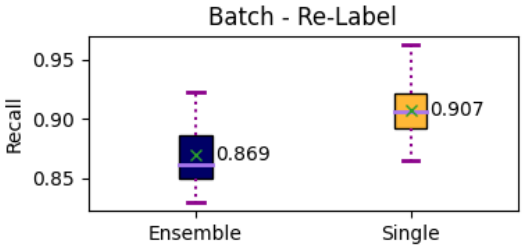
(b) Batch - with concept drift; the metric is vandalism recall. Ensemble and single classifiers.



(c) Batch - no concept drift; the metric is regular recall. Ensemble and single classifiers.



(d) Batch - with concept drift; the metric is regular recall. Ensemble and single classifiers.

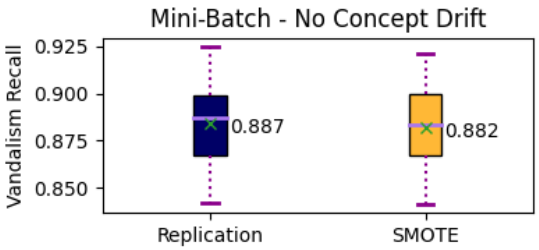


(e) Batch - with concept drift; the metric is re-label recall. Ensemble and single classifiers.

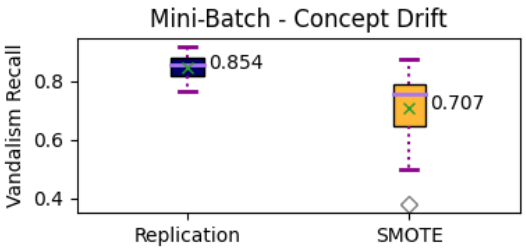
Figure 3.22: Box-plots for ablation studies considering the batch learning approach to compare the performance of ensemble against single classifiers.

replication and SMOTE to justify the use of a different data sampling strategy. We note that our use case poses a unique challenge to data sampling strategies since it embodies the complexities associated with learning from scarce data and overlaps between class representations. In SMOTE, this challenge is further accentuated by the incremental introduction of concept drifts through the presentation of data in blocks, which directly affect the quality of the data generated (Fernández et al., 2018). To summarize, the data sampling strategy in our context must consider class overlap, feedback noise (re-label data), and limited datasets.

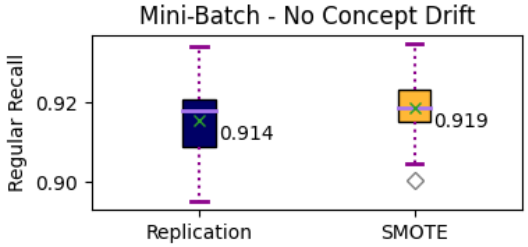
Results show that before concept drift, the performances of both data sampling strategies are similar (q.v. Figures 3.23a and 3.23c) as supported by the statistical results in Table 3.7. However, after introducing concept drift (q.v. Table 3.7), mini-batch learning benefits from using the replication-based technique to detect vandalism (q.v. Figure 3.23b). Another finding is that although the incorporation of SMOTE leads to higher performance values for regular and re-labeled instances after concept drift (q.v. Figures 3.23d and 3.23e), it results in reduced accuracy in vandalism classification, thus hindering the implementation of this strategy in our case.



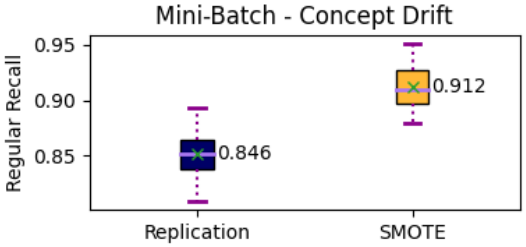
(a) Mini-Batch - no concept drift; the metric is vandalism recall. Replication and SMOTE.



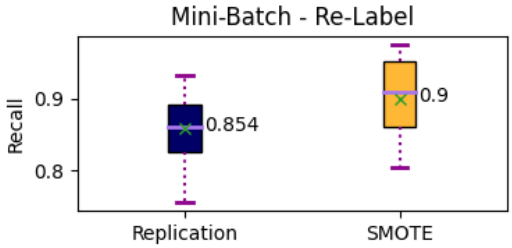
(b) Mini-Batch - with concept drift; the metric is vandalism recall. Replication and SMOTE.



(c) Mini-Batch - no concept drift; the metric is regular recall. Replication and SMOTE.



(d) Mini-Batch - with concept drift; the metric is regular recall. Replication and SMOTE.



(e) Mini-Batch - with concept drift; the metric is re-label recall. Replication and SMOTE.

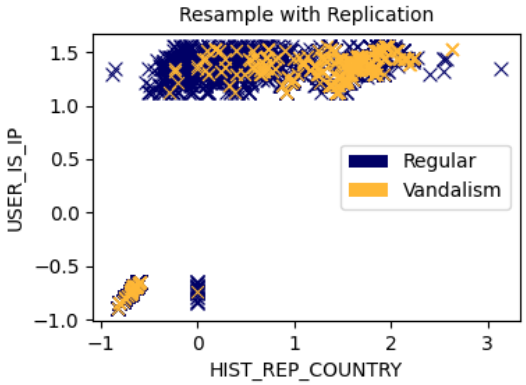
Figure 3.23: Box-plots for ablation studies considering mini-batch learning approach to compare the performance of oversampling by replication against SMOTE.

Dataset	Mini-Batch Replication		
	Regular	Vandalism	Re-label
Original	0.0740	0.4990	X
Concept Drift	**0.0000**	**0.0000**	**0.0000**

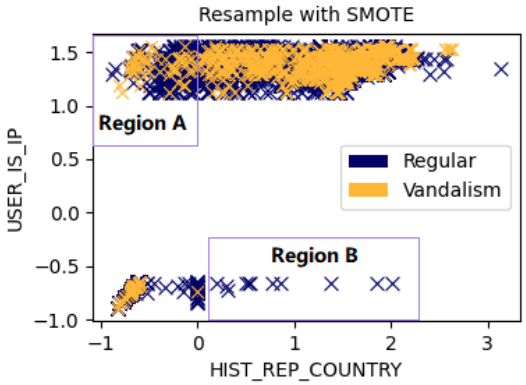
Table 3.7: P-values of comparing the recall performance of oversampling by replication and SMOTE training using mini-batch learning. Performance values are presented in Figure 3.23. Bold p-values indicate results below the critical value $\alpha = 0.05$.

To comprehend the reasons that account for the differences between oversampling by replication and SMOTE, we examine two sources of information. First, Figure 3.24 illustrates the class distribution after applying each sampling strategy. Second, Figure 3.25 presents the interpretability data, which assists in selecting the features to build the data space outlined in step one and in understanding the contrast of feature scores.

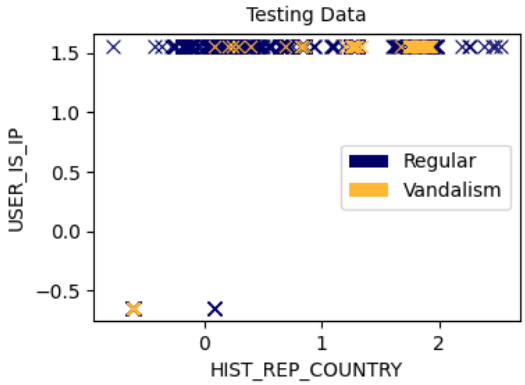
As described by Luengo et al. (2011), one of the characteristics of the SMOTE method is its ability



(a) Class distribution after replication.



(b) Class distribution after SMOTE.



(c) Class distribution in the testing dataset.

Figure 3.24: Class distribution considering oversampling by replication, SMOTE, and testing data. The features are USER_IS_IP and HIST_REP_COUNTRY.

to generate more general data points, covering a larger portion of the data space. In our particular context, SMOTE generates instances in certain regions (Region A and B in Figure 3.24b) that leads to the creation of a more complex data space. As a result, this causes the newly generated instances to disrupt the contrary class space (Region A in Figure 3.24b), negatively impacting the model's performance. This effect has also been reported in different cases and identified as a source of errors (Luengo et al., 2011; Weiss, 2009). In our case, we note that 4.6% of the violation data is in Region A for SMOTE, while only 0.7% of the data is in the same region for the replication strategy, and no data point exists in this region for the testing data. Furthermore, SMOTE generates instances in regions of the data space that do not conform to the featurized edits provided by the online community (Region B in Figure 3.24b).[10] In contrast, the oversampling by replication strategy does not add data points in this region, nor is it present in the testing dataset.

We select the features USER_IS_IP and HIST_REP_COUNTRY to describe the data space above based on interpretability information provided in Figure 3.25. Our results reveal the importance of these features in the vandalism classification task and how they have the biggest difference in relevance score between the data sampling techniques. Specifically, USER_IS_IP gains importance when we apply SMOTE, while HIST_REP_COUNTRY reaches higher relevance when the oversampling by replication strategy is used. The divergences in scores are important because a

[10]It should be noted that Region B represents oversampling for the feedback data that suffered the swap of class labels.

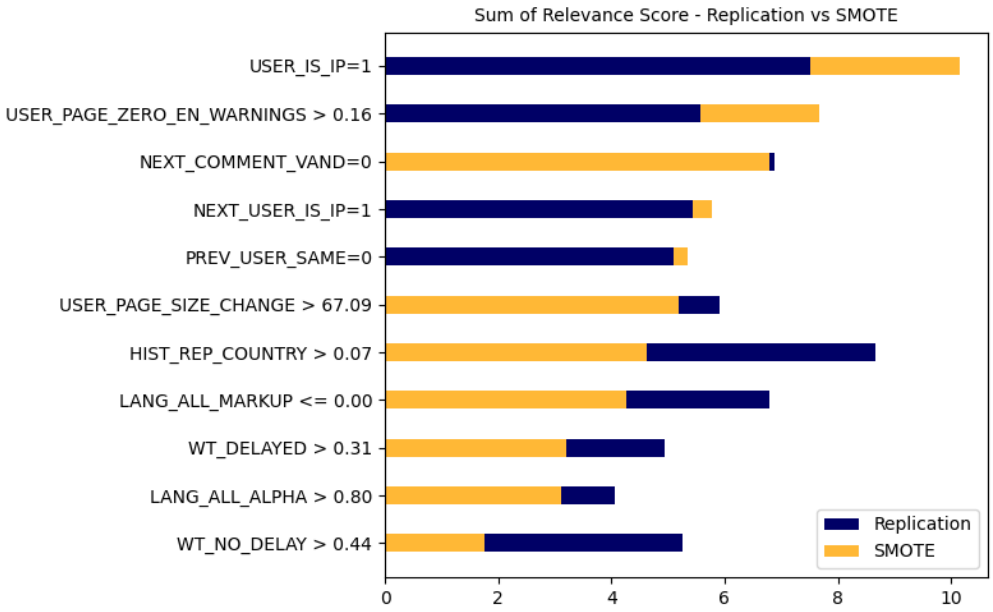


Figure 3.25: Sum of relevance scores for vandalism classification. The graph presents the top ten features when resampling with replication and SMOTE.

considerable percentage (i.e., 73.33%) of the errors affecting the performance of the SMOTE-based solution is related to a decrease in the relevance score of the HIST_REP_COUNTRY feature, thus, explaining the lower performance of this technique in our use case.

3.7 Summary

In this chapter, we presented the Feedback-Driven Adaptive Learning (FeDAL) framework, which aims to support normative systems by learning to detect norm violations from interaction data and continuous feedback (human or artificial)[11] in the context of online communities. Specifically, FeDAL can work with norms whose meaning can change over time, like what is the set of features that characterize hate speech.

FeDAL integrates ensemble with machine learning approaches to address two challenges that emerge in online community domains: 1) the imbalanced nature of the dataset; and 2) the need to adapt to the evolving community view on what characterizes norm violations. We used a dataset from Wikipedia article edits to evaluate the efficacy of our framework for detecting violating behavior in a binary classification task. In this dataset, each action (article edit) is annotated as either regular or violating behavior (referred to as *vandalism* in our context). Furthermore, we note a highly imbalanced class distribution, with only 7% of the edits representing the vandalism class.

In addition to detecting violating behavior, we argue that it is important to incorporate interpretability into FeDAL to promote transparency in the decision-making process. Thus, FeDAL employed LIME to assist in not only explaining to users how a particular community defines non-acceptable behavior but also to serve as a tool for engineers to debug and comprehend the inner

[11] In this thesis, an artificial agent refers to the ML model that classifies an action as violation or regular behavior. The output of the ML model is the feedback.

workings of different ML models to mitigate their limitations, like considering features of an article edit that humans do not consider when evaluating a violation.

We conducted experiments on two different dataset configurations (q.v. Section 3.5). First, we evaluated the algorithms before introducing concept drift, specifically focusing on learning to detect vandalism in the context of Wikipedia article edits. We examined the recall metric for the overall, regular, and vandalism cases to assess FeDAL's performance (q.v. Section 3.6). Moreover, we analyzed AUC-ROC and AUC-PR to further comprehend the behavior of the three learning techniques. Second, we designed a scenario to evaluate the learning techniques after introducing concept drift, particularly when class labels are swapped. Our aim was to assess how FeDAL adapts after receiving feedback. Our current approach adopted a simulation strategy since we did not have access to real feedback from community members (which we plan to address in future work). This strategy involved creating different subgroups within the vandalism class and changing their labels to simulate the effect of concept drift. Lastly, we conducted ablation studies to evaluate changes in FeDAL's architecture and data sampling strategy (q.v. Section 3.6.3). Specifically, we evaluated the benefits of using an ensemble of classifiers compared to a single model. Additionally, we analyzed how the replication strategy outperforms the SMOTE technique in our use case.

Results indicated that we can use the three learning approaches to detect norm violations in an online community. However, important differences must be addressed. First, compared to incremental approaches, batch learning presented a drawback regarding training time. This resulted from retraining the model using the entire dataset when new data became available, which was time-consuming and added complexity to data treatment and management. Second, after introducing concept drift, batch and mini-batch learning exhibited more stability in the training process and significantly better performance in classifying norm violations. In contrast, online learning presented a significant drop in performance for the same case due to the bias towards the majority class (regular behavior).

Finally, the next chapter presents the Language Model Adaptive Learning (LaMAL) framework, which tackles norm violations by handling text datasets in binary and multi-label classification tasks.

Chapter 4

LaMAL: The Large Language Model Framework

This chapter delves into the second framework of our multi-scenario approach, the Language Model Adaptive Learning (LaMAL) framework, which learns to detect norm violations by handling text datasets.[1] While FeDAL handles tabular datasets (q.v. Chapter 3), LaMAL extends our approach to solve binary and multi-label *text* classification tasks. This provides our multi-scenario approach with a range of solutions to addressing communities with diverse data structure requirements.

The multi-label classification task refers to assigning distinct labels to a violation instance. This is particularly relevant within the scope of this thesis, as we aim to handle domains in which a single action may comprise multiple violation classes. For instance, our experiments with LaMAL outline the application of this framework to the Wikipedia use case to identify hate speech in article editing tasks (q.v. Section 4.3). In this context, each label corresponds to a particular hate speech class, such as swearing, racism, or misogyny.

In contrast to FeDAL, LaMAL can identify these specific violation classes due to sufficiently informative text content, i.e., words in a text that are usually associated with different violations, like the "N-Word" (Rahman, 2012) that can be used to manifest racism. In our Wikipedia use case, this characteristic differs substantially from the tabular data structure since the set of features lacks the encoding of pertinent information necessary to identify the specific violation classes.

To handle text datasets, LaMAL adopts Pre-Trained Language Models (PLMs) (q.v. Section 2.4) fine-tuned using mini-batch learning (q.v. Section 2.3). Unlike the FeDAL framework, LaMAL does not employ online and batch learning techniques due to the limitations in PLMs' settings. Online learning is unfeasible due to the model's size, which impacts the update of the network weights and the time needed to complete the fine-tuning process. Similarly, batch learning is unfeasible because of the computational power required to repeatedly fine-tune a PLM as the dataset grows in addition to the complexities regarding data management.

LaMAL also faces challenges associated with imbalanced datasets. Like FeDAL, which uses data-sampling solutions to improve the ensemble learning process in the tabular scenario (q.v. Section 3.1), LaMAL investigates data-level solutions, specifically oversampling and undersampling techniques for text-related classification tasks (q.v. Sections 4.1 and 4.2).

The remainder of this chapter is divided as follows. Section 4.1 presents mini-batch learning for binary classification tasks, while 4.2 presents mini-batch learning for multi-label classification tasks. Subsequently, in Section 4.3, we describe the experiments to evaluate the LaMAL framework,

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

1 **Algorithm:** Binary Mini-Batch Fine-Tuning
2 **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);
3 **Output:** Fine-tuned PLM (Θ).
4 **while** *data block is available* **do**
5 Pre-process D_t, no past data is used.
6 Compute the current imbalance ratio. $r_t \leftarrow |P_t| \div |M_t|$.
7 **if** $r_t < d$ **then**
8 Undersample majority class by the ratio $r_t \div d$.
9 **end**
10 Fine-tune Θ with D_t for e epochs.
11 Obtain the sum of relevance scores for violations.
12 **end**

Algorithm 4.1: The binary mini-batch fine-tuning procedure for PLMs (LaMAL).

with the results outlined in Section 4.4. Finally, Section 4.5 offers a comprehensive summary of the key points discussed in this chapter.

4.1 Mini-Batch Learning for Binary Text Scenarios

Similar to Algorithm 3.1 for tabular data, mini-batch for text tasks (q.v. Algorithm 4.1) builds data blocks to continuously update the model's parameters over time (in a sequential manner). However, one key difference between these approaches is that Algorithm 4.1 can handle imbalanced datasets more efficiently just by undersampling the majority class, without the need to create an ensemble of classifiers. To achieve that, Algorithm 4.1 takes advantage of the PLMs' architecture, which can learn representations of texts based on previous training and incorporate classification heads to solve specific classification tasks (Kenton & Toutanova, 2019; Y. Liu et al., 2019; Sanh et al., 2019).

The fine-tuning process of PLMs starts by pre-processing the available text data (q.v. Algorithm 4.1, line 5). In LaMAL's context, the type of detection task dictates the necessary pre-processing steps. For instance, in the case of hate speech detection, 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 be irrelevant in this context. On the other hand, to detect whether a sentence violates an expected writing style, removing these characters is detrimental to the model's performance since they indicate that the text is not following the required format. Another case that might require a pre-processing step is correcting words commonly used to bypass automatic detection tools. For example, in cases where a community member uses swear words, they may employ alternative terms such as "f1ck," "fooock," and "fuk." These two cases demonstrate that it is necessary to implement task-specific pre-processing to ensure the efficacy of LaMAL. This becomes particularly relevant when the community contains limited labeled data or the interactions happen in a low-resource language.

Following the pre-processing step, Algorithm 4.1 calculates the imbalance ratio to determine whether under-sampling is required (lines 6 and 7). The algorithm then undersamples the majority class by considering the established difference between the number of majority and minority instances (q.v. Algorithm 4.1, line 8). The next step is fine-tuning the PLM with the data block for e epochs (q.v. Algorithm 4.1, line 10). One of the main advantages of PLMs is their inherent simplicity in executing the fine-tuning process. Hence, the complete training procedure is more straightforward than FeDAL's Algorithms 3.1, 3.2 and 3.4, as it requires fewer steps to deploy a

```
1  Algorithm: Multi-Label Mini-Batch Fine-Tuning
2  Input: current set of violation instances (I_t), set violation classes (V), min instances per
     class (c), and number of epochs (e);
3  Output: Fine-tuned multi-label PLM (Θ).
4  while violation data block is available do
5      for violation class v ∈ V do
6          Get instances of v. N_t^v ← I_t ∈ v.
7          if |N_t^v| < c then
8              Oversample N_t^v by duplication.
9          end
10     end
11     Fine-tune Θ with I_t for e epochs.
12     Obtain the sum of relevance scores for each class in V.
13 end
```

Algorithm 4.2: The multi-label mini-batch fine-tuning procedure for PLMs (LaMAL).

PLM to a new classification task.

Lastly, in line 11 (q.v. Algorithm 4.1), as we update the PLM, it is possible to understand the terms usually associated with violation by calculating a sum of relevance scores obtained from local interpretations. The sum of the relevance scores of a word (s_w) is the sum of all local relevance scores, calculated using the Integrated Gradients (IG) algorithm (q.v. Section 2.5.2). This differs from the Local Interpretable Model-Agnostic Explanations (LIME) method employed by FeDAL (q.v. Section 2.5.1). In equation 4.1, k is the number of occurrences of word w in the dataset, $\mathsf{IG}(w_u, 1)$ is the calculated relevance score for the u^{th} occurrence of w, regarding its contribution to class 1, where class 1 represents violating behavior. The framework must only change the second parameter to 0 to get the relevance scores for the regular class.

$$s_w \leftarrow \sum_{u=1}^{k} \mathsf{IG}(w_u, 1) \tag{4.1}$$

4.2 Mini-Batch Learning for Multi-Label Text Scenarios

In addition to identifying violations (q.v. Algorithm 4.1), LaMAL can also classify the specific class of violation present (q.v. Algorithm 4.2). It is worth noting that a single action may comprise multiple violation classes.[2] Thus, the framework must be equipped to handle multi-label classification tasks.

For each violation class $v \in V$ defined by the community (q.v. Algorithm 4.2, line 5), LaMAL 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 (q.v. Algorithm 4.2, line 7). Suppose the data block does not contain the minimum number of class instances v. In that case, the algorithm over-samples 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.

[2] In Section 4.3, we describe the different violation classes considered in our use case.

Line 12 (q.v. Algorithm 4.2) obtains the sum of relevance scores using Equation 4.2. The sum of relevance scores (s_w^v) is calculated for each $v \in V$ and is based on local interpretations. $\mathsf{IG}(w_u, v)$ computes the local relevance score of word w in relation to class v, k is the number of occurrences of w in the dataset, and u represents a specific instance of w. Calculating s_w^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 (in Section 4.4, we analyze these differences).

$$s_w^v \leftarrow \sum_{u=1}^{k} \mathsf{IG}(w_u, v) \tag{4.2}$$

Finally, one limitation of the LaMAL framework is that it does not handle the emergence of new violation classes. LaMAL has one PLM Θ with $|V|$ output nodes, with each output node corresponding to a distinct violation class. Consequently, to extend our multi-scenario approach, the next chapter presents the CAL framework (q.v. Section 5), which employs adapters (q.v. Section 2.4) as the component to tackle the emergence of new violation classes.

4.3 Experiments

This section describes how we apply mini-batch learning to fine-tune PLMs. Similar to the experiments with FeDAL (q.v. Section 3.5), we also investigate the use case of Wikipedia article edits in this context. However, we directly process text data instead of handling a set of features. This thesis focuses on violations of the norm that prohibits hate speech, as this represents a complex and particularly harmful violation within online interactions.[3] We remind the reader that a norm violation is referred to as *vandalism* in our use case, encompassing the norm prohibiting hate speech.

Since LaMAL can solve multi-label classification tasks, the dataset labeling process consists of two parts. First, Wikipedia uses Amazon Mechanical Turk (MTurk) to classify an article edit either as vandalism or not, providing no further information on the nature of the violation (Adler et al., 2011). Second, the author of this thesis[4] further annotates each vandalism instance with a violation class, focusing on the norm prohibiting hate speech. To perform this annotation, the author starts by considering the labels from the MTurk process (vandalism or regular). Then, the author specifies additional hate speech classes for the vandalism edits with texts that convey attacks to individuals or groups. Usually, these attacks focus on characteristics of people, such as ethnicity, sexual orientation, and social class (Nockleby, 2000). Table 4.1 presents examples of such behavior in Wikipedia and their associated hate speech class. At this step, we manually correct the misspelled insulting words (based on the identified violation classes).

The author of this thesis assigns the multi-label annotation, introducing the author's view on what is considered a norm violation. We assert that this annotation reflects one of many possible communities' views. Given that we aim to develop a framework capable of continuously updating its parameters and that can adapt to different views, regardless of whether a single person (e.g., a moderator) or multiple individuals assign the violation class, it must incorporate the feedback provided in the annotation process to continuously learn new definitions of norm violation (which also contains diverse perspectives and may vary depending on the community in question). For instance, the use of the "N-Word" (Rahman, 2012) may not represent detrimental behavior in an African American community, but it violates the hate speech norm on Wikipedia. Therefore, our system must learn that this term could indicate a violation in this new community.

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

[4] Here, we highlight the "author of this thesis" to clarify that a *single* person related to this research provides the annotation.

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 "N-Word" often defecated...	Racism
[INDIVIDUAL's NAME] also sucks dick for features.	Sexual Harassment
...HES GAYYYYYYYYYY AND HES A FREAKK...	LGBTQIA+ Attack
[INDIVIDUAL's NAME] was a super mega bitch and she kill the...	Misogyny

Table 4.1: Examples of sentences classified as norm violation (vandalism) in the Wikipedia community and the specific class of hate speech. "[INDIVIDUAL's NAME]" is used to mask real people's names.

In the Wikipedia dataset, the author of this thesis identifies six different classes of hate speech by analyzing each instance of norm-violation behavior (article edit). 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. In the current use case, We only solve the multi-label classification task for text data, as the features present in the tabular data do not encode relevant information for classifying vandalism with a specific hate speech class. Below, we present a list detailing each of the hate speech classes:

- Swear - it describes edits that contain foul language;
- Insult and Ableism - it describes edits that insult people in general and specifically people with disabilities (Bogart & Dunn, 2019);
- Sexual Harassment - it describes edits that contain sexual insinuations and harassment (Biber et al., 2002);
- Racism - it describes edits with attacks targeting people from different ethnicities (Keum & Miller, 2018);
- LGBTQIA+ Attack - it describes edits with insults targeting people based on their sexual orientation and/or gender identity (Harper & Schneider, 2003);
- Misogyny - it describes edits with attacks targeting women (Ging & Siapera, 2018).

4.3.1 Learning to Detect Hate Speech

In contrast to the experiments with FeDAL, we are not investigating concept drift with LaMAL. We cannot conduct such experiments with LaMAL due to the limited amount of data for the hate speech detection case. However, considering that concept drift is an important aspect of interactions in our use case, we later introduce the CAL framework to address this challenge (q.v. Chapter 5).[5] Thus, this section describes experiments to detect hate speech in a binary classification task and identify specific vandalism classes in a multi-label classification task.

- ***Learn to detect hate speech — the binary classification task:*** This experiment aims to evaluate LaMAL's ability to deal with norm violation in Wikipedia edits, where the text

[5]CAL aims to use a dataset with hate speech from a different community and improve upon that with data from our specific environment (Wikipedia article edits) while tackling the type of concept drift that introduces new violation classes.

of the edit is classified as violation or not. 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%). We use data from our annotation process to specifically obtain hate speech edits. The dataset is highly imbalanced, so we use 2x5-fold cross-validation for the experiments, which is necessary due to the text dataset size. Classification Recall, Area Under the Curve of the Receiver Operating Characteristics (AUC-ROC), and Area Under the Curve of Precision-Recall (AUC-PR) are the chosen metrics for evaluation.

- ***Learn to identify specific violation classes — the multi-label classification task:*** Here, we aim to evaluate LaMAL's performance to detect specific hate speech classes. Besides the imbalanced dataset for violation/regular edits, the hate speech classes are also imbalanced. Certain hate speech classes occur more often than others. In total, the hate speech 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 hate speech 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 (Sechidis et al., 2011). A 2x5-fold cross-validation is also used for this experiment. Classification recall for each class is the chosen metric for evaluation.

To classify a given text, our framework adopts PLMs (q.v. Section 2.4). Specifically, we employ RoBERTa (Y. Liu et al., 2019) and DistilBERT (Sanh et al., 2019) following the Hugging Face implementation (Wolf et al., 2020), with a batch size of 1024 for the binary classification task and 256 for the multi-label classification task. Adam is the optimization algorithm (Kingma & Ba, 2014), and the focal cross entropy is the loss function (T.-Y. Lin et al., 2017). The learning rate is 10^{-4}.

To optimize the performance of the PLMs, we implement additional parameters. Concretely, 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 fall 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 texts up to the limit of 512 words.

4.3.2 Understanding the Words that Contribute to Hate Speech Detection

The LaMAL framework aims not only to classify a text as violating the "no-hate-speech" norm but also to provide community members with information on which parts of that text result in such a classification. We aim to accommodate diverse community members and leverage their mutual understanding of what constitutes a norm violation. To achieve this, LaMAL employs 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 of Section 4.4 and Appendix B. Additionally, by providing such information, other community members can argue about the inner workings of our framework, supporting community members in deliberating about those words and whether they should trigger violation classification. The collaborative decision of the community can provide feedback for the model to adapt.

The interpretation results for the binary classification task show which words contribute most to the classification of a text as being hate speech or not (regular text). Each word in the text (Wikipedia edit) can be relevant for hate speech classification, relevant for regular classification, or neutral. In contrast, the multi-label classification case allows each word to be neutral or relevant to

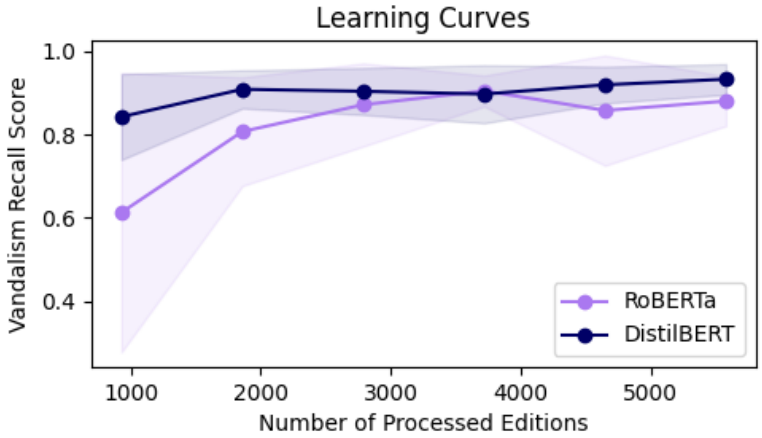


Figure 4.1: Vandalism Recall for RoBERTa and DistilBert.

one or more classes. For instance, a single word may contribute to the classification of a Wikipedia edit as both racist and containing swear words. As we are interested in understanding the terms that compose hate speech, the experiments focus only on interpretability data for the six hate speech classes. Thus, we consider 639 edits (the complete hate speech dataset) for interpretability. Finally, we use the Transformers Interpret library (Pierse, 2023) to implement the IG algorithm in our experiments.

4.4 Results and Discussion

This chapter presents the results of using PLMs to address text classification tasks when evaluated within the Wikipedia article edits use case. Specifically, we describe the results of implementing RoBERTa and DistilBERT. In this case, the community defines norm-violating behavior as *vandalism*. We show words of a Wikipedia article edit contributing to the PLMs' output in binary and multi-label classification tasks.

4.4.1 Learning to Detect Hate Speech

Binary Classification Task

Figure 4.1 shows the graph that describes the recall score for RoBERTa and DistilBERT when applied to the hate speech classification task, and Table 4.2 presents the complete summarized performance values. According to the Wilcoxon Signed-Rank Test in Table 4.3, the results show that DistilBERT outperforms the RoBERTa model. Additionally, it is worth noting that since the beginning of the training process, RoBERTa presents a higher standard deviation (q.v. Figure 4.1), which may be attributed to the small size of the dataset and the large number of trainable parameters (125M) 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 66M 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.

In addition to the recall metric, we also evaluate LaMAL's performance using the following metrics: the Area Under the Curve of the Receiver Operating Characteristics (AUC-ROC) and the Area

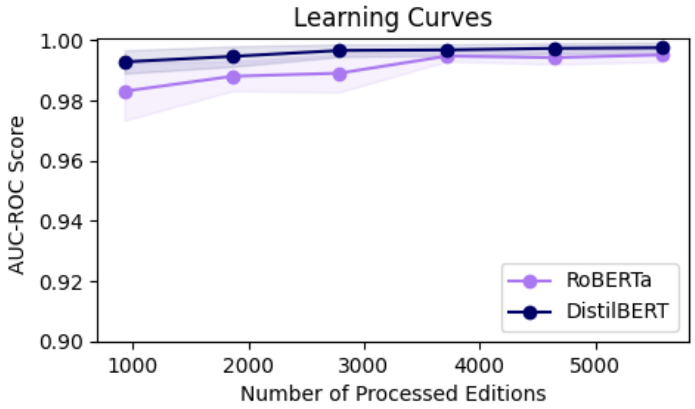


Figure 4.2: AUC-ROC score for RoBERTa and DistilBERT.

Under the Curve of Precision-Recall (AUC-PR). It is worth mentioning that both metrics rely on the ratio between the majority and minority instances to calculate their scores. Next, we detail the usefulness of these metrics for text-based domains with an imbalanced dataset.

As in FeDAL's case (q.v. Section 3.6), initially, AUC-ROC does not appear to be a suitable metric to evaluate ML models trained with an imbalanced dataset, given that it overestimates the model's performance with a bias towards the true positive rate. Consequently, a large change in the number of false positives (the regular edits wrongly classified as vandalism edits) yields only a small change to the ROC score (J. Davis & Goadrich, 2006; H. He & Garcia, 2009). However, suppose the domain requires giving the most importance to identifying true positives even at the cost of false positives. In such contexts, the AUC-ROC metric may be useful for understanding performance. We argue that this is the case for norm violation in online community domains, where blocking offensive actions is more important than making mistakes when classifying regular behavior.

Figure 4.2 illustrates the results for the AUC-ROC metric. Both models exhibit equal AUC-ROC scores, with no discernible differences. The lack of variability between the two models suggests similar performance, even though, as described in Table 4.3, the Wilcoxon Signed-Rank Test fails to statistically demonstrate their similarity with a valid p-value. Additionally, it is possible to see that the AUC-ROC score presents higher values than the vandalism recall (q.v. Table 4.2). Specifically, the AUC-ROC scores for RoBERTa and DistilBERT have values around 99%, while vandalism recall values are around 88% and 93%, respectively. This difference demonstrates the bias towards the true positive rate that characterizes the AUC-ROC metric.

The AUC-PR plot shows the relationship between precision and recall by comparing the false positive and the true positive rates (Saito & Rehmsmeier, 2015). It also uses precision and captures the effect of the large number of regular instances on the algorithm's performance (differently from AUC-ROC) (Saito & Rehmsmeier, 2015). To obtain a more realistic evaluation of this metric, Siblini et al. (2020) propose a calibrated version of AUC-PR, in which the goal is to make the metric independent of the class prior (i.e., it monitors the performance based on the data distribution, regardless if it changes or not). In Figure 4.3, the beginning of the training process shows that DistilBERT outperforms RoBERTa, yielding higher performance values. As training progresses, the performance gap between the two models gradually diminishes. However, it is worth noting that by the conclusion of the training procedure, a statistically significant difference persists between the two models, as indicated by the p-values detailed in Table 4.3.

Figure 4.4 and Table 4.2 illustrates 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 (q.v.

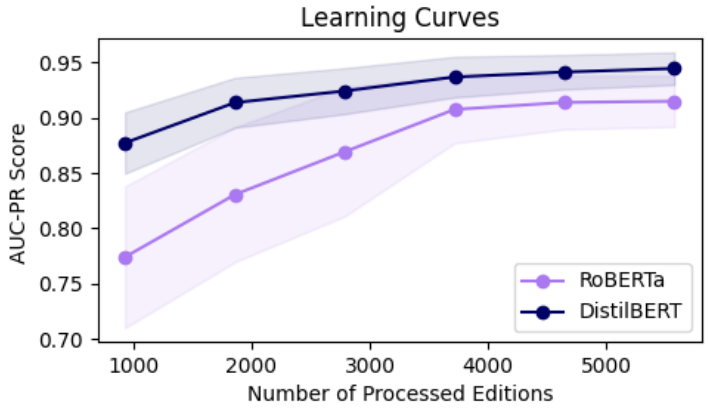


Figure 4.3: AUC-PR score for RoBERTa and DistilBERT.

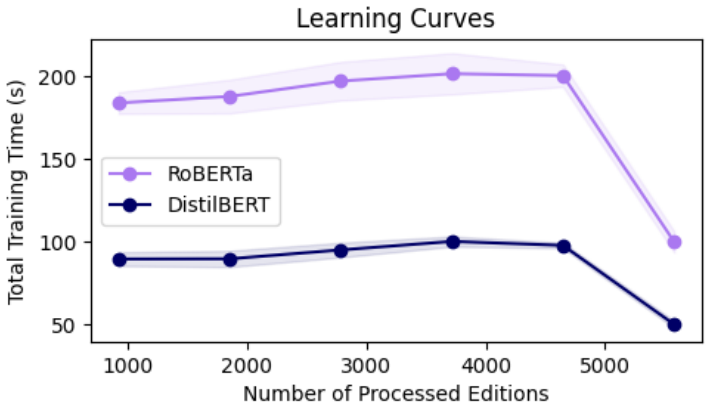


Figure 4.4: Training time for RoBERTa and DistilBERT to classify vandalism — binary task.

Table 4.3), 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 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.

Measurement	RoBERTa±Std	DistilBERT±Std
Vandalism Recall	0.8805±0.0594	**0.9331**±0.0366
Regular Recall	**0.9944**±0.0046	0.9934±0.0043
AUC-ROC	0.9951±0.0023	**0.9975**±0.0018
AUC-PR	0.9149±0.0231	**0.9445**±0.0147
Training Time (s)	183.98±6.5272	**89.56**±4.1958

Table 4.2: Summary of the performance results of RoBERTa and DistilBERT applied to the Wikipedia article edits dataset, binary task. 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). Bold values refer to the best performance, while for training time, bold values refer to the fastest approach.

Measurement	P-values
Vandalism Recall	0.0412
Regular Recall	0.5566
AUC-ROC	X*
AUC-PR	0.0020
Training Time	0.0019

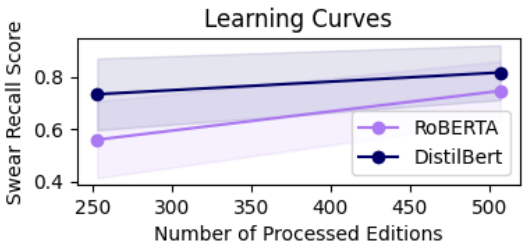
Table 4.3: Summarized comparison between the recall, AUC-ROC, and AUC-PR performances of RoBERTa and DistilBERT. Training time is also presented. The Wilcoxon Signed-Rank Test is used to obtain the p-values. The null hypothesis is that the samples were drawn from the same distribution. Critical value $\alpha = 0.05$. *Examining the AUC-ROC metric and given the similarities between the score values for the two models, we note no discernible difference in this case. It is worth mentioning that the Wilcoxon Signed-Rank Test fails to provide a valid p-value due to the occurrence of ties between the scores. Specifically, seven out of the 2x5-fold experiments yielded identical values for both models, thus negatively affecting the statistical test's ability to generate a conclusive p-value.

Multi-Label Classification Task

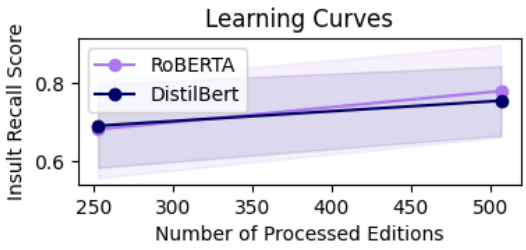
This section presents the evaluation of RoBERTa and DistilBERT applied to the multi-label classification task. We aim to categorize vandalism considering the six classes mapped for hate speech in Wikipedia, e.g., "Swear," "Insult and Ableism," "Sexual Harassment," "Racism," "LGBTQIA+ Attack," and "Misogyny." In the Wikipedia 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, LaMAL must be able to identify when these violations occur simultaneously.

Figure 4.5 presents the recall scores (detailed in Table 4.4) 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 Wikipedia 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 (q.v. Table 4.5). 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. We later address this issue (i.e., a small number of violation instances) with the CALmodel, which incorporates cross-community learning to enhance the model's performance by leveraging the fine-tuning process with data from different communities (q.v. Section 5.3).

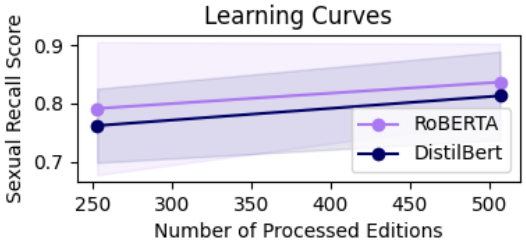
Finally, Figure 4.6 shows the training time needed for the multi-label task. Similar to the binary case, the DistilBERT model has a significantly faster fine-tuning process, as the Wilcoxon Signed-Rank Test (q.v. Table 4.5) attests with a p-value of 0.0019 (below the critical value $\alpha = 0.05$). We use a batch size of 256 vandalism edits for the multi-label classification task, 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.



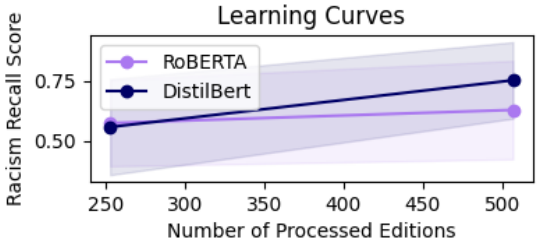
(a) "Swear" Recall score for RoBERTa and DistilBERT.



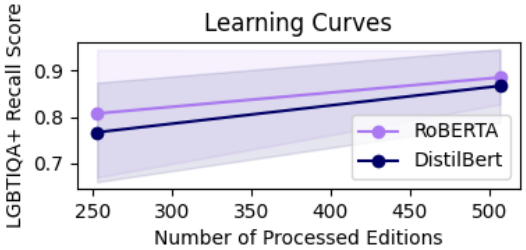
(b) "Insult and Ableism" Recall score for RoBERTa and DistilBERT.



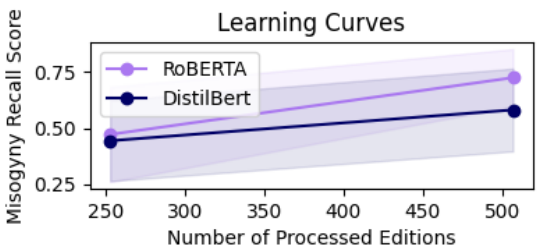
(c) "Sexual Harassment" Recall score for RoBERTa and DistilBERT.



(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.

Figure 4.5: Recall scores for the violation classes: "Swear," "Insult and Ableism," "Sexual Harassment," "Racism," "LGBTQIA+ Attack," and "Misogyny."

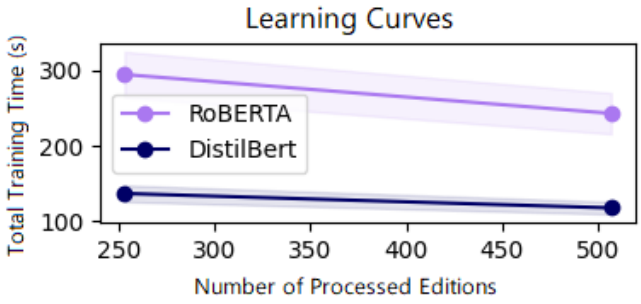


Figure 4.6: Training time for RoBERTa and DistilBERT to classify the violation classes — multi-label task.

Measurement	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

Table 4.4: Summary of the performance results of RoBERTa and DistilBERT applied to the Wikipedia article edits dataset in the multi-label case. 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). Bold values refer to the best performance, while for training time, bold values refer to the fastest approach.

Measurement	P-values
Swear	0.1134
Insult and Ableism	0.4316
Sexual Harassment	0.3571
Racism	0.0632
LGBTQIA+ Attack	0.4055
Misogyny	0.0407
Training Time	0.0019

Table 4.5: Summarized comparison between the recall performance of RoBERTa and DistilBERT. Additionally, training time is also presented. 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 $\alpha = 0.05$

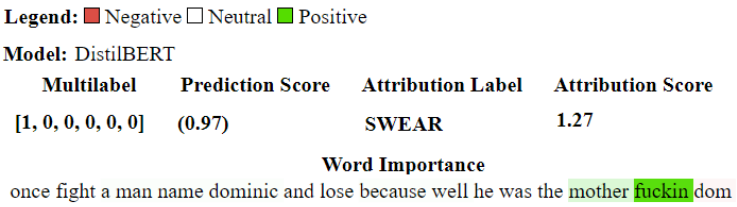


Figure 4.7: The local interpretation of a specific edit considering the DistilBERT model in the multi-label case. The label considered is "Swear." The relevance score is calculated using IG (q.v. Section 2.5.2).

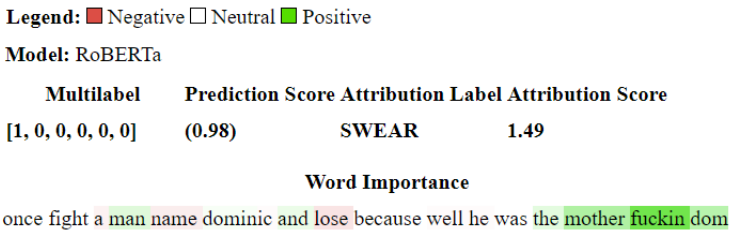


Figure 4.8: The local interpretation of a specific edit considering the RoBERTa model in the multi-label case. The label considered is "Swear." The relevance score is calculated using IG (q.v. Section 2.5.2).

4.4.2 Understanding the Words that Contribute to Hate Speech Detection

This experiment investigates which words of an article edit affect the classification output of the PLMs. For that, we present three different pieces of information. One describes the relevant words for a specific hate speech instance (i.e., a single article edit), as depicted in Figures 4.7 and 4.8. LaMAL calculates the relevance scores using the IG algorithm (q.v. Section 2.5.2). The second presents a summary of the words usually associated with a specific hate speech class (such as "Swear") and their frequency in our complete training dataset, as depicted in Figures 4.9 and 4.10.[6] Lastly, Figures 4.11 and 4.12 present a summary of words usually associated with hate speech in general. The sum of scores considers the local relevance calculated using IG. With this, we aim to give an overall view of the terms in our domain that result in classifying text as hate speech.[7]

To describe local interpretability, provided by the IG algorithm, we analyze Figures 4.7 and 4.8 for DistilBERT and RoBERTa, respectively. The violation class considered here is "Swear."[8] For local interpretations, the stronger the green shade, the higher the highlighted word's relevance score concerning classifying the text with the "Swear" class. On the other hand, the stronger the red shade, the more significant the influence of the highlighted word on decreasing the hate speech confidence (classifying the text as not belonging to the "Swear" class).

One crucial aspect is that the relevance of certain words may vary depending on the model. Let us consider the word "man" in Figures 4.7 and 4.8. 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,

[6] Appendix C presents the relevance score for all the other hate speech classes.

[7] To clarify, the sum of scores is not a global interpretation of our model (q.v. Section 2.5) but rather a summary of the sum of local interpretations (the output of the IG algorithm).

[8] Appendix B presents local interpretability examples for all other violation classes.

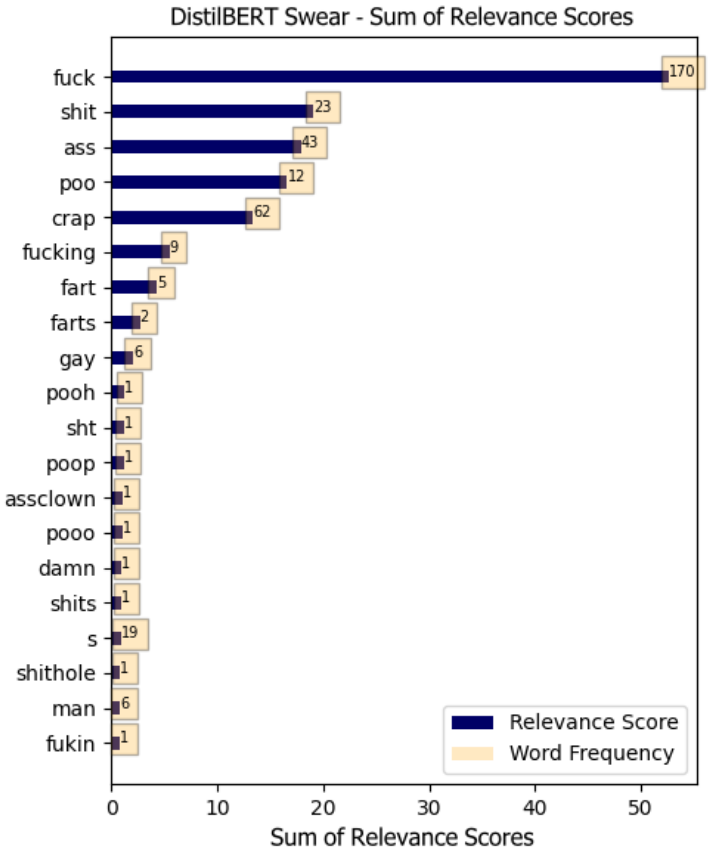


Figure 4.9: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model in the multi-label case. The label considered is "Swear." Besides, we present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

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. In other words, when the PLMs are initially trained, the datasets do not contain instances of texts with these terms as part of the violation classes we identify in our use case.

Presenting the summary of words related to specific hate speech classes in general, as opposed to only those related to a specific text (local interpretations), is also interesting. Figures 4.9 and 4.10 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 a faulty logic since it considers influential words that are not coherent with their understanding. As such, they can initiate a new training process, using additional instances of norm-violating behavior to improve the model's classification output.

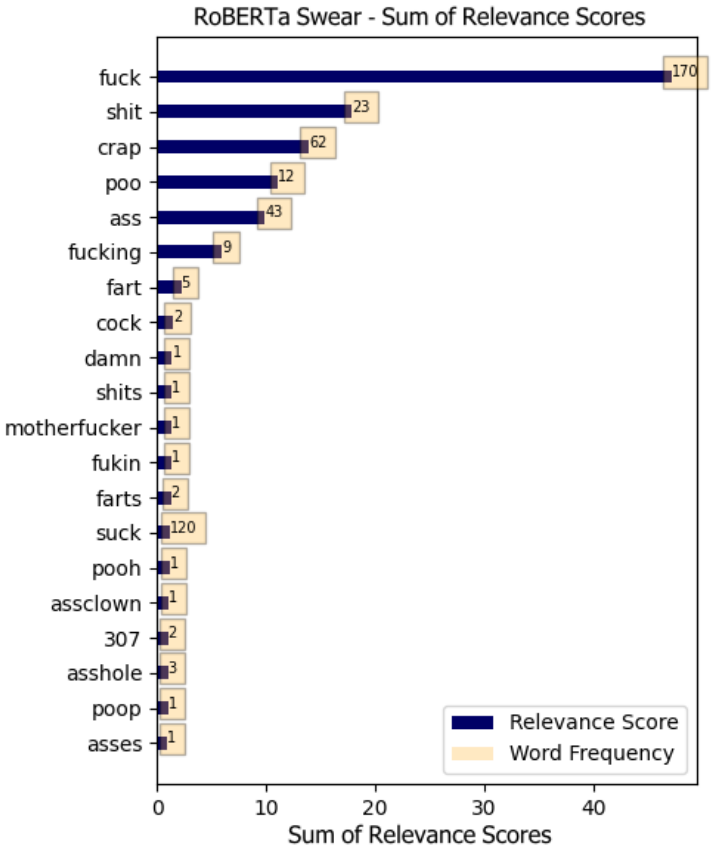


Figure 4.10: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is "Swear." Besides, we present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

The last part of our interpretation is in Figures 4.11 and 4.12. These graphs summarize the words usually associated with hate speech, which is the binary classification task. The words in the community dataset are insulting and related to cyberbullying. Both models consider similar words relevant for detecting hate speech. However, they disagree on the assessment of six of them. This discrepancy 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 differently.

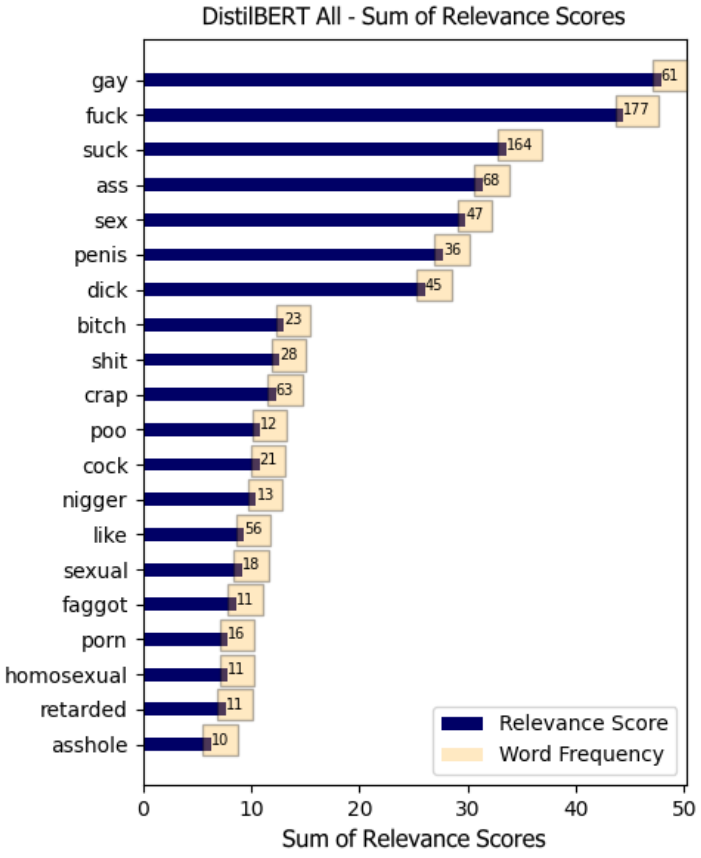


Figure 4.11: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model for all violations. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

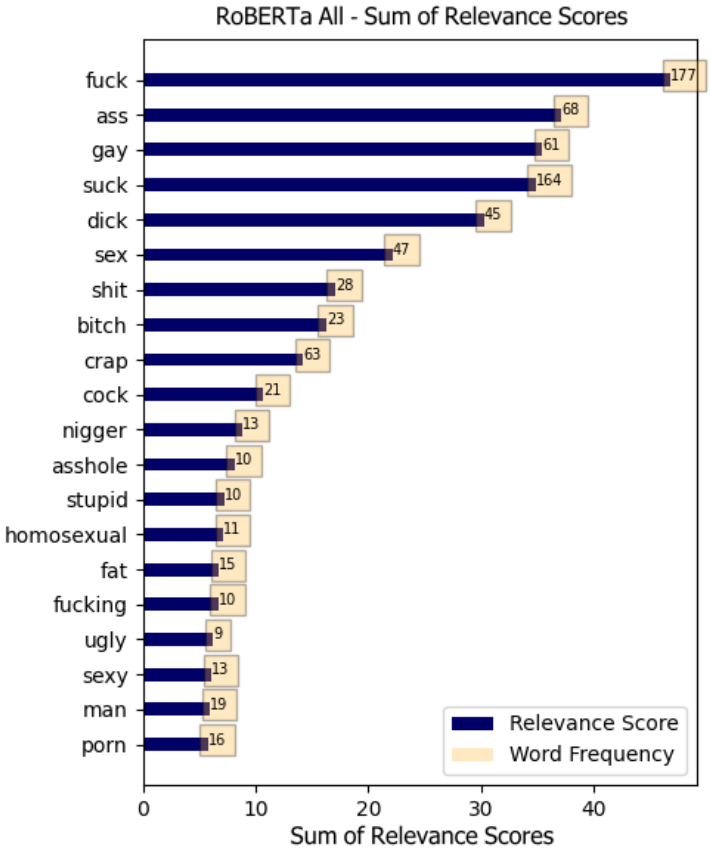


Figure 4.12: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model for all violations. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

4.5 Summary

This chapter proposed the Language Model Adaptive Learning (LaMAL) framework, which aims to support normative systems in continuously learning to detect norm violations in text classification tasks. To accomplish this goal, LaMAL used data from community members' views on what constitutes a norm violation to fine-tune Pre-Trained Language Models (PLMs) (q.v. Section 2.4) through an incremental learning approach (mini-batch).[9]

We evaluated LaMAL in the Wikipedia article editing use case (q.v. Section 1.1.3). In this community, we were interested in detecting violations of the "no-vandalism" norm, specifically targeting article edits that contained hate speech. Two challenges in such communities are crucial to our investigation: the imbalanced nature of the dataset; and the incremental update of PLMs to incorporate the most recent definition of what constitutes norm violations. Thus, LaMAL's main contributions were 1) creating a framework that handled imbalance between violation classes (different types of hate speech) using a replication approach; 2) enabling our framework to incrementally incorporate feedback data to update the machine learning model as interactions unfold; and 3) using interpretability to enhance a community's understanding of what constitutes norm-violating behavior (which words were associated with hate speech).

The experiments with LaMAL explored two PLMs, RoBERTa and DistilBERT, in distinct settings. First, we conducted experiments for the binary classification task, in which both PLMs should learn whether an article edit is a violation (hate speech). Second, we evaluated the PLMs to solve a multi-label classification task, aiming at identifying the specific hate speech classes associated with a given article edit. This enabled LaMAL to detect multiple violations that occur simultaneously. Since we could handle text data directly in these cases (as opposed to working with tabular data like FeDAL (q.v. Chapter 3)), we incorporated into our framework an interpretability component that identified relevant words for hate speech detection. Specifically, LaMAL employed the Integrated Gradients (IG) algorithm (q.v. Section 2.5.2).

Results showed that LaMAL could learn to detect hate speech in the case of Wikipedia article edits while performing both binary and multi-label text classification tasks. With the incorporation of DistilBERT and RoBERTa, LaMAL fine-tuned prior language knowledge to improve the learning process for hate speech detection in our use case. Additionally, through interpretability, LaMAL offered insights into the community's view of what the members consider non-acceptable behavior, specifically identifying the most relevant words usually associated with hate speech and providing a summary view of this concept.

Finally, in the next section, we present the Cross-Community Adapter Learning (CAL) framework, which tackles the limitations of LaMAL regarding the emergence of new violation classes and learning from datasets with a limited size.

[9]For LaMAL, an action was defined as a Wikipedia article edit, and the words were the elements that helped classify the text as a violation or not.

Chapter 5

CAL: The Cross-Community Adapter Learning Framework

This chapter introduces the third framework that composes our multi-scenario approach, the Cross-community Adapter Learning (CAL) framework.[1] CAL can learn to detect norm violations using data from different sources, such as various online communities. Like FeDAL (q.v. Chapter 3) and LaMAL (q.v. Chapter 4), the main objective of CAL is to be deployed in a normative system to support the adherence to norms, especially when analyzing actions specified through text. CAL analyzes each action that community members perform to minimize norm violations. Furthermore, it addresses the limitations present in LaMAL, specifically by handling the emergence of new violation classes and learning from datasets with a limited size.

The major difference between CAL and LaMAL is their approach to classifying violation classes. While LaMAL employs a classification head comprising several nodes, each responsible for identifying a distinct violation class, CAL adopts a different strategy combined with Pre-Trained Language Models (PLMs). It employs adapters (q.v. Section 2.4) to represent specific classes of violations, which can be dynamically created as interactions unfold. For instance, in contexts where community feedback indicates a shift in community members' views that leads to identifying classes that did not exist, CAL can learn from this feedback by incorporating new adapters.

In this thesis, we aim to work with low-resource (or newly created) online communities (q.v. Section 1.1.3), where low-resource implies working with a limited labeled set of norm-violating actions (Huang et al., 2022). To improve a model's performance in a particular community, CAL adopts cross-community learning to incorporate data from different sources. This initially helps train a Machine Learning (ML) model in a new target community with limited labeled data, where we deploy the framework (Chandrasekharan et al., 2019; Zhuang et al., 2020).

Figure 5.1 presents an overview of the CAL framework (which is further explored in Section 5.1 with details of the algorithm). Our work uses data from various source communities as the initial step to define norm-violating behavior in a new community with limited labeled data (target community). First, the new community defines the specific violation class that CAL should detect.[2] CAL then follows two different paths depending on whether an adapter that identifies the specified violation class exists or not:

1. Suppose the adapter does not exist, i.e., a new violation class emerges. In that case, the framework uses data from another community in the initial training process, creating a new

[1] Source code available at https://bitbucket.org/thiago-phd/ijcai_2023/.

[2] Although we do not tackle this challenge here, we envision this process to use feedback data from a deliberation process where community members define the new violation class (provide a new label) that CAL should identify.

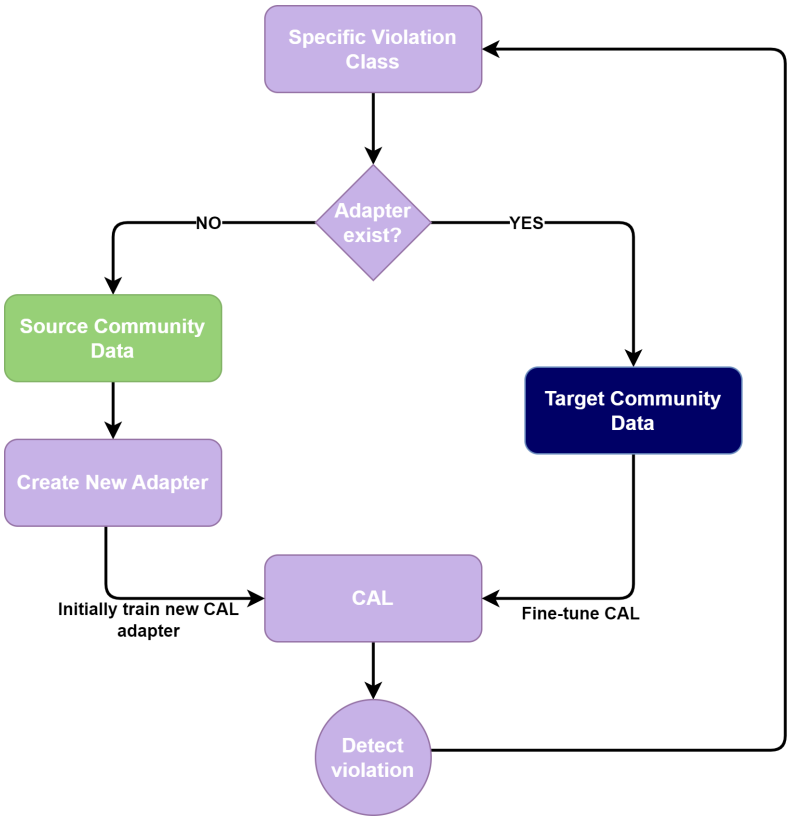


Figure 5.1: The Cross-community Adapter Learning (CAL) framework. The "Source Community Data" and the "Target Community Data" blocks have different colors because they represent different datasets, source and target, respectively.

adapter for the specified violation class. Upon completing this step, CAL is ready to detect norm-violating behavior in the new (target) community with limited labeled data. However, it is worth noting that if the source and target communities exhibit divergent views of what constitutes norm-violating behavior, the model may present low-performance values at this step (the results in Section 5.3 describe this case). Thus, fine-tuning with target data is important.

2. If the specific violation class adapter already exists, we simply fine-tune CAL with target community data. This step aims to improve the framework's performance with the newest feedback data.

To accomplish its goal, CAL incorporates adapters between the layers of a PLM, learning to detect norm violations from the feedback of interacting agents (referred to as community members). Additionally, CAL employs the Integrated Gradients (IG) algorithm (q.v. Section 2.5.2) to help understand the distinct views manifested in the communities. Concretely, IG allows CAL to obtain the relevant words contributing to the violation classification in source and target communities. The ability to analyze how communities' views change within a specific domain or across different communities is a key contribution of this component, as it describes how detrimental behavior changes over time and across domains.

To summarize, each component in CAL offers a unique benefit. First, transformer-based models, specifically the PLM, provide powerful language representation for tackling NLP tasks (T. Lin et al., 2022; Vaswani et al., 2017). Second, incorporating adapters in our framework allows for

1 **Algorithm:** The Cross-Community Adapter Learning (CAL) Algorithm
2 **Input:** Current time step (t), violation classes (V_t), set of all violation instances (I_t), set of augmented instances (A_t), data block size (s), set of source task adapters ($\Phi^{V_{t-1}}$), base PLM (Θ), and number of epochs (e);
3 **Output:** Trained adapters (Φ^{V_t})
4 **while** *violation instances available* **do**
5 **for** *each violation class* $v \in V_t$ **do**
6 Get current violation class instances $VI \in I_t$.
7 Get other violation instances $BI \in I_t$ to balance D_t, where $BI \cap VI = \emptyset$ and $|BI| = |VI|$.
8 Create a balanced data block. $D_t \leftarrow VI \cup BI$.
9 **if** $|D_t| < s$ **then**
10 Incorporate augmented instances. $D_t \leftarrow A_t$.
11 **end**
12 **if** $\Phi^v \in \Phi^{V_{t-1}} = NULL$ **then**
13 Create a new adapter Φ^v.
14 **else**
15 Load previously trained adapter. $\Phi^v \leftarrow \Phi^{v_{t-1}}$.
16 **end**
17 Fine-tune Φ^v on top of Θ with D_t for e epochs.
18 Add trained adapter to the list Φ^{V_t}.
19 **end**
20 Obtain the sum of relevance scores for adapters $\in \Phi^{V_t}$.
21 **return** fine-tuned adapters Φ^{V_t}.
22 **end**

Algorithm 5.1: The Cross-community Adapter Learning (CAL) Algorithm

an efficient fine-tuning process that reflects the new community's view on what constitutes norm violations while also allowing for extensibility through the dynamic creation of adapters as new violation classes emerge. Lastly, the IG algorithm supports our analysis of the model to understand the elements of a text relevant to norm violation detection.

The remainder of this chapter is divided as follows. Section 5.1 presents CAL's learning algorithm. Next, Section 5.2 describes the experiments conducted to evaluate the CAL framework. We divide this section into two parts. First, we describe experiments regarding learning to detect violations using data from different communities. Second, we employ interpretability to understand the words usually associated with norm-violating behavior. Then, Section 5.3 outlines the results. Finally, Section 5.4 offers a comprehensive summary of the key points discussed in this chapter.

5.1 Mini-Batch Learning for Cross-Community Text Scenarios

This section delves into Algorithm 5.1 that describes the Cross-Community Adapter Learning (CAL) framework. CAL is trained using a mini-batch learning approach to solve text classification tasks. As norm-violating actions are made available (for instance, community members start labeling actions as norm-violating behavior), CAL builds balanced data blocks D_t of fixed size s for each violation class $v \in V_t$ defined by the community (q.v. Algorithm 5.1, line 8). To create D_t, CAL uses $s/2$ instances of class v, while the other $s/2$ are randomly drawn from the remaining instances (q.v. Algorithm 5.1, line 7).

If the size of the current balanced dataset $|D_t|$ is smaller than the fixed data block size (q.v. Algorithm 5.1, line 9), we generate and augment extra violation instances (q.v. Algorithm 5.1, line 10). The augmented instances are generated by modifying the original text previously identified as norm-violating behavior. This process includes substituting synonyms and randomly removing words. If CAL is processing a newly emerged violation class, we perturb the ground-truth data at the current time step (t), asking for feedback from the community members. This feedback is essential because, by modifying a text, we may remove the violation. However, suppose CAL is processing an already identified violation class. In that case, we use the classification output of our model in the previous time step ($t-1$) and generate the perturbed instances only from the texts detected as violations, asking for augmented data relabeled by the community. Step $t-1$ represents either the training step in a source community or in the same community but at a previous moment (past actions).

In line 12, Algorithm 5.1 checks for an existing adapter corresponding to violation class v. If v is a newly emerged violation for which no adapter has been previously created, CAL initiates a new adapter Φ^v (q.v. Algorithm 5.1, line 13). The ability to dynamically generate adapters enables CAL to incorporate new classes for violations as interactions unfold and communities' views evolve. However, if a previously trained adapter $\Phi^{v_{t-1}}$ is related to v, then CAL loads it (q.v. Algorithm 5.1, line 15). Each violation class v has a single associated adapter Φ^v.

Algorithm 5.1 executes the incremental learning procedure in line 17, updating Φ^v using D_t for e epochs. As we update the adapters, it is possible to observe the evolution of their behavior over time by calculating the relevance scores based on local interpretations obtained from the IG algorithm (q.v. Algorithm 5.1, line 20). The differences between the adapters may occur due to the application of distinct fine-tuning procedures. In this case, adapters Φ^v and $\Phi^{v_{t-1}}$ are different because they were trained within the same community but at separated moments, or Φ^v and $\Phi^{v_{t-1}}$ were trained using cross-community learning (using source data from a different community). The ability to analyze how the communities' views change within a specific domain or across different communities is a key contribution of our work. Finally, in line 21, Algorithm 5.1 returns the continuously trained adapters for norm-violating detection.

5.2 Experiments

This section outlines the application of our incremental learning approach to detecting and understanding norm-violating behavior in a cross-community setting, specifically within the context of Wikipedia article editing (q.v. Section 1.1.3). Wikipedia has a set of norms to regulate interactions during article edits, including the requirement to use proper writing style, to refrain from removing content, to avoid editing wars, and not to express hate speech. This chapter focuses on violations of the hate speech norm, as this represents a complex and particularly harmful norm-violating behavior within online interactions.[3]

We remind the reader that we employ a two-step process to collect interaction data for this study. First, Wikipedia uses Amazon Mechanical Turk (MTurk) to classify an article edit either as a violation or not, providing no further information on the nature of the violation (Adler et al., 2011). Second, the author of this thesis[4] further annotates each violation instance with a violation class, introducing the author's view on what is considered a norm violation. We assert that this annotation reflects one of many possible community views. Thus, instead of arguing about the introduction of bias, we aim to demonstrate that our framework can continuously update its parameters to new views, regardless of whether a single person (e.g., a moderator) or multiple

[3]Future work shall focus on solving other types of violations.

[4]Here, we highlight the "author of this thesis" to clarify that a *single* person related to this research provides the annotation.

individuals assign the violation class. In other use cases, this annotation process may be different, with labels containing multiple perspectives and variations depending on the community in question. For instance, the use of the "N-Word" (Rahman, 2012) may not represent detrimental behavior in an African American community, but it violates the hate speech norm on Wikipedia. Therefore, our system must learn, from the feedback of a new community, that this term could indicate a violation specifically in this new community. Table 4.1 shows examples of hate speech classes considered in this work.

In this thesis, we source data from three publicly available community datasets, namely a software engineering community on Slack (Chatterjee et al., 2020), the abusive language towards conversational systems (ConvAbuse) (Curry et al., 2021), and a dataset built using humans and machine learning models to generate hate speech (DynamicGenerated) (Vidgen et al., 2021). Each dataset represents a unique community and includes text sentences for specific classes of hate speech. Specifically, the Slack community provides "Swear" instances. The ConvAbuse dataset includes "Insult and Ableism" and "Sexual Harassment" instances. The DynamicGenerated dataset includes "Racism," "LGBTQIA+ Attack," and "Misogyny" instances. To evaluate our approach, we design the following two experiments. First, we evaluate CAL's ability to identify hate speech using data from different communities. Second, we use interpretability to analyze how the definition of hate speech varies across communities.

5.2.1 Learning to Detect Hate Speech

We aim to evaluate whether CAL can learn to detect hate speech by initially using a source community to train the adapters. The number of instances for each violation class in the source data is 349 for "Swear," 273 for "Insult and Ableism," 456 for "Sexual Harassment," 512 for "Racism," 512 for "LGBTQIA+ Attack," and 512 for "Misogyny." Our focus is on learning from a limited number of examples of disruptive behavior. The target task consists of 639 edits, with 233 for "Sexual Harassment," 212 for "Insult and Ableism," 126 for "Swear," 109 for "LGBTQIA+ Attack," 56 for "Misogyny," and 32 for "Racism." In this case, a single edit may contain multiple violation classes simultaneously. Consequently, the sum of the amount of edits belonging to these violation classes surpasses the total of 639 edits. To ensure that each fold of the validation process maintains a similar data distribution, we employ stratification on the multi-label dataset with the algorithm from (Sechidis et al., 2011). 2x5-fold cross-validation is used for this experiment.

To implement our solution, we use DistilBERT, which is smaller and faster than other state-of-the-art PLM alternatives (Sanh et al., 2019), with the adapter implementation by HuggingFace (Wolf et al., 2020). The data block size is 256, AdamW is the optimization algorithm (Loshchilov & Hutter, 2019), and the number of epochs is 12. Adapters have a reduction factor of 16 and ReLu as the non-linearity function (S. Sharma et al., 2017). The IG algorithm was implemented following the Transformers Interpret library (Pierse, 2023). We use TextAttack to create the augmented instances (Morris et al., 2020). The experiments are executed on an NVIDIA GeForce GTX 1650 with 4GB memory.

5.2.2 Understanding Different Communities' Views on Detecting Hate Speech

We use the local interpretability of PLMs to analyze the impact of words on detecting hate speech. To do so, we first examine the words that receive high relevance scores when the model is trained on data from the source community. Next, we gather information on the relevant words when the model is incrementally fine-tuned on data from the target community. Finally, we compare the difference in the relevance score between these two steps. The change in the relevance score reveals how interactions differ between these communities.

5.3 Results and Discussion

5.3.1 Learning to Detect Hate Speech

Setting	Violation	Precision±Std	Recall±Std	F1±Std
Source – Target	Swear	0.5090±0.0438	0.5066±0.0355	0.3466±0.0281
	Insult and Ableism	0.5925±0.0361	0.6680±0.0725	0.6005±0.0434
	Sexual Harassment	0.7661±0.0381	0.7564±0.0397	0.7413±0.0415
	Racism	0.6022±0.0120	0.8534±0.0357	0.6038±0.0224
	LGBTQIA+ Attack	0.5963±0.0286	0.5973±0.0303	0.3978±0.0284
	Misogyny	0.5848±0.0243	0.7083±0.0588	0.5553±0.0360
Target – Target	Swear	**0.8757**±0.0261	**0.8945**±0.0197	**0.8831**±0.0203
	Insult and Ableism	**0.6937**±0.0380	**0.8478**±0.0557	**0.7236**±0.0437
	Sexual Harassment	**0.9147**±0.0281	**0.9151**±0.0279	**0.9144**±0.0280
	Racism	0.8290±0.0306	**0.9760**±0.0214	**0.8850**±0.0252
	LGBTQIA+ Attack	**0.8843**±0.0271	**0.9340**±0.0363	**0.9047**±0.0294
	Misogyny	**0.8635**±0.0657	**0.7535**±0.0618	**0.7915**±0.0573
Only Target	Swear	0.8407±0.0427	0.8701±0.0315	0.8515±0.0385
	Insult and Ableism	0.6351±0.0294	0.7642±0.0574	0.6484±0.0359
	Sexual Harassment	0.9012±0.0329	0.9005±0.0311	0.8998±0.0320
	Racism	**0.8662**±0.0886	0.7682±0.0961	0.8002±0.0893
	LGBTQIA+ Attack	0.8046±0.0441	0.8535±0.0434	0.8234±0.0446
	Misogyny	0.5470±0.2025	0.5085±0.0185	0.4884±0.0349

Table 5.1: Summary of the performance results (2x5-fold cross-validation) of incremental DistilBERT using adapter-based fine-tuning to evaluate hate speech detection on the Wikipedia article editing task. Three settings are considered: 1) cross-community training and testing on our target (*Source – Target*); 2) fine-tuning on target community and testing on target (*Target – Target*); and 3) training only on target data and testing on target data (*Only Target*).

Hate Speech	P-values		
	Precision	Recall	F1-Score
Swear	0.0273	0.0645	0.0273
Insult	0.0020	0.0059	0.0020
Sexual	0.0506	0.0525	0.0827
Racism	0.1934	0.0020	0.0840
LGBTQIA+	0.0020	0.0020	0.0020
Misogyny	0.0098	0.0020	0.0020

Table 5.2: Comparison between *Target – Target* and *Only Target*. 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, and the critical value $\alpha = 0.05$.

Table 5.1 presents the results for each experiment. We describe performance values per hate speech class using precision, recall, and F1-score metrics. *Source – Target* refers to training on the source community and testing on the target Wikipedia interactions, with no fine-tuning. *Target – Target* is the experiment after fine-tuning the source model on our target community. Finally, *Only Target* refers to the results obtained when the model is trained and evaluated solely on the target community data. Results indicate that fine-tuning a cross-community model presents the best performance in most cases. Although directly using a model trained on a source task with no fine-tuning (*Source – Target*) yields the lowest performance, it can serve as an initial point for our task, leveraging the performance of our approach after fine-tuning is applied. The Wilcoxon Signed-Rank

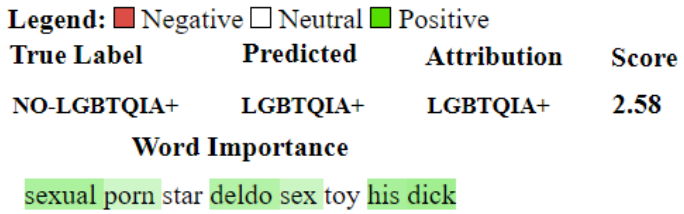


Figure 5.2: Local interpretation when trained on source community data. The model wrongly classifies.

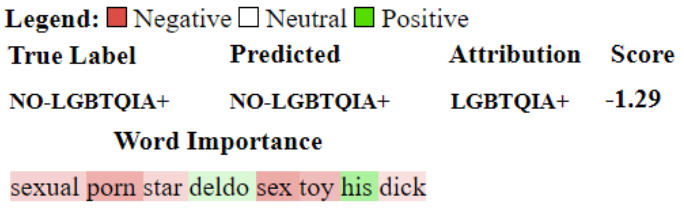


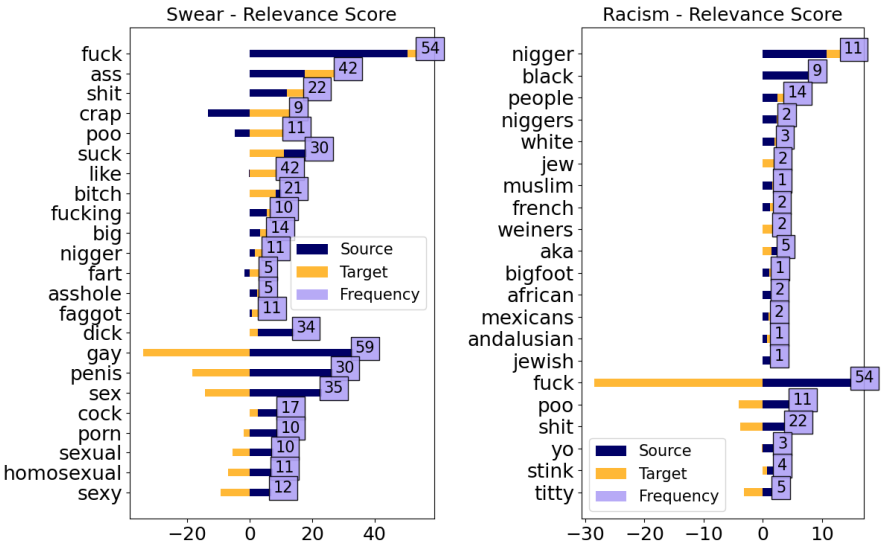
Figure 5.3: Local interpretation after continuously fine-tuning on target data. The model changes its behavior as expected by our new community view.

Test in Table 5.2 attests that, except for the "Sexual Harassment" class and precision for "Racism," our cross-community learning approach significantly outperforms *Only Target*, suggesting that our framework benefits from incorporating data from multiple communities. These exceptions are explained as follows. First, for "Sexual Harassment," *Only Target* performs very well (above 90%), with no need to add data to improve its performance since this class is already well represented in our target community. Second, the discrepancy in "Racism" precision occurs due to the divergence in the terms used to express racist remarks in both the source and target communities. For instance, the term "fuck" presents a high sum of relevance scores for "Racism" detection in the source community (q.v. Figure 5.4). However, this relevance diminishes when considering the target community, demonstrating the shift in the term's relevance for detecting this violation class in a new use case. As such, because we have a small number of instances of the "Racism" class in our target dataset, the model's fine-tuning process is sub-optimal (i.e., incomplete parameter updating), and an outdated definition of violation is still detected, affecting the precision performance.

5.3.2 Understanding Different Communities' Views on Detecting Hate Speech

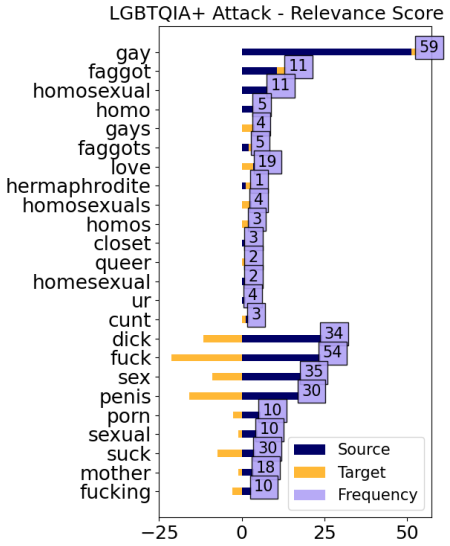
Figures 5.2 and 5.3 present local interpretability as calculated by the IG algorithm. The intensity of the green shade indicates the relevance of the highlighted word to violation detection, demonstrating what words trigger norm violation. In contrast, the intensity of the red shade is related to the decrease in the violation confidence (not detecting as an instance of "LGBTQIA+ Attack," but rather labeling the edit as "non-LGBTQIA+ Attack"). Figure 5.2 presents how the model trained on source community data classifies an article edit from Wikipedia. Words usually associated with "Sexual Harassment" content are deemed relevant to the model. However, as fine-tuning is executed, Figure 5.3 shows how the model drifts from the previous view about what constitutes an "LGBTQIA+ Attack," resulting in negative relevance scores for the same words as the model adapts to the new community view.

Figure 5.4 illustrates the differences in the sum of relevance scores determined by CAL between



(a) Swear

(b) Racism



(c) LGBTQIA+ Attack

Figure 5.4: Sum of relevance scores for three violation classes. "Insult and Ableism," "Sexual Harassment," and "Misogyny" are provided in Appendix D. The source task refers to the model trained using the source community data, while the target task is the result after fine-tuning the model with our data. "Frequency" refers to the number of occurrences of a word in the training dataset. We present the increase (or decrease) of the relevant words for the violation classification, presenting the difference between the two communities and how norm violations are defined.

communities or at different moments in time.[5] To obtain a word's relevance measure, CAL sums the relevance scores of each occurrence of the word (local interpretability), as those depicted in Figures 5.2 and 5.3. The term *Source* refers to the model trained exclusively on the source-community data, while *Target* represents the relevance score after fine-tuning the model. The graph is organized as follows: First, we identify the 15 most relevant words for *Source* and *Target*.

[5]Appendix D presents the data for the other three violation classes.

Then, the relevant words for *Target* are displayed at the top. To demonstrate the changes in relevance scores (either increase or decrease), we include the *Source* value for the same word. Finally, we completed the rest of the rank with the words from the *Source* that had high relevance scores but saw a decrease in value after fine-tuning. For example, considering the "LGBTQIA+ Attack" class (q.v. Figure 5.4c), we can see that the word "gay" becomes more relevant as we fine-tune CAL with our target data (update what constitutes a norm violation based on a new community view). However, simultaneously, the words "dick," "fuck," and "sex" lose relevance. The NEGATIVE value informs us that these words are relevant for identifying an edit as a "non-LGBTQIA+ Attack."

In addition to understanding different communities' views, we can identify the factors contributing to the under-performance of the *Source – Target* task (q.v. Table 5.1) through the analysis of relevance scores. As an example, for the "Swear" class (q.v. Figure 5.4a), the source community data includes instances that associate "Sexual Harassment" and "LGBTQIA+ Attack" content with "Swear," e.g., "gay," "penis," "sex," which are words with high relevance scores. Since the communities use parts of the violating-behavior vocabulary differently when the *Source – Target* model attempts to solve a new task, instances containing these words are wrongly classified as "Swear." However, the adaptable character of our proposal allows for updating relevance scores and improving the model's performance as interactions unfold. We can also observe this phenomenon in the "Racism" case, where words like "N-Word," "Jew," and "Muslim" have higher relevance scores, while "fuck," "poo," and "shit" present a significant drop.

5.4 Summary

This chapter presented the Cross-Community Adapter Learning (CAL) framework that aimed to continuously learn to detect norm violations using interaction data from diverse communities. Specifically, CAL incorporated data from source communities to improve the performance of ML models in a new community with a limited labeled dataset, referred to as the target community. Our goal was to provide the basis to work with norm violations where views on what constitutes a norm violation can change based on community members' feedback. Furthermore, we equipped CAL to adapt to different communities' views of identical violation classes.

For instance, consider the case of hate speech, particularly racism. The interpretation of sentences deemed racist may vary depending on the community. To illustrate this point, imagine a community composed of African Americans where the use of the "N-Word" is regarded as a friendly salute. In contrast, this term is regarded as a severe racial offense within a different domain, such as the Wikipedia article editing task. Thus, we aimed to learn the differences between communities, providing insights into how violation evolved and is defined in different domains. To accomplish this goal, we employed interpretability tools.

CAL adopted a bottleneck adapter architecture on top of a PLM (q.v. Section 2.4), fine-tuned using a mini-batch approach (q.v. Section 2.3). Additionally, we presented an interpretability analysis of the cross-community adapters to understand that what constitutes norm violations varied between communities. The Integrated Gradients (IG) algorithm calculated the local relevance scores of words in text sentences (q.v. Section 2.5.2), which were combined to determine their sum of relevance scores in a particular community.

We conducted experiments within the context of Wikipedia article editing (q.v. Section 1.1.3). The norm in question regulated the prohibition of hate speech. This was the same domain in which we evaluated FeDAL (q.v. Section 3.5) and LaMAL (q.v. Section 4.3). Nevertheless, some challenges these approaches faced were efficiently addressed here due to CAL's architecture. Concretely, since we used adapters to handle specific violation classes, the imbalance between violation classes did not hinder the performance of our model. Furthermore, CAL demonstrated efficiency in parameter updates. Unlike FeDAL and LaMAL, which updated all parameters of a neural network or the

classification heads simultaneously, CAL optimized this process. It only updated the adapter associated with each violation class after obtaining the required data to build a data block. Lastly, CAL could handle the emergence of new violation classes, creating new adapters in response to community members' feedback indicating the need to detect a new class.

To evaluate cross-community learning, we used data from three distinct sources. The results showed that by initially training an adapter with source community data, we could leverage the performance of our model (q.v. Section 5.3), demonstrating how CAL learned to detect violations and incorporated new knowledge based on a novel community view. Since interactions have evolving characteristics, we have argued throughout this thesis that a current community view is the most critical input for defining what constitutes a norm violation.

Finally, in the next chapter, we present a user experiment to assess whether our model interpretation affects the user's perception of violating behavior. As such, we use data derived from the IG algorithm, both local and as the sum of relevance scores, to compare three different interpretability layouts.

Chapter 6

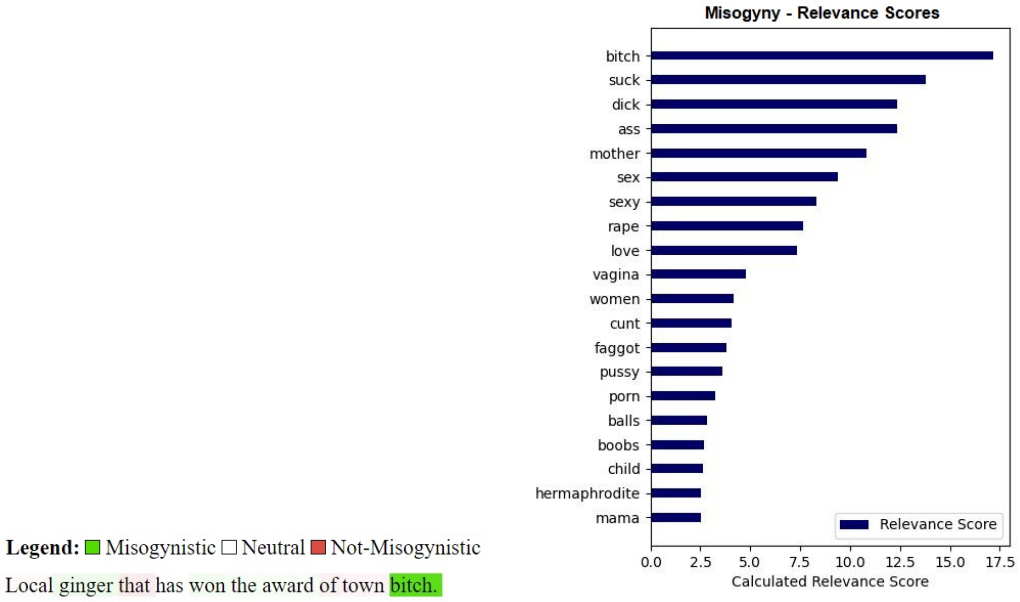
User Study: Assessing the Impact of Interpretability

This thesis describes our multi-scenario approach to continuously learn what constitutes norm violations and detect when such violations occur. This approach continuously updates the parameters of Machine Learning (ML) models to incorporate the evolving views about norm violations in online communities as interactions unfold. Chapters 3, 4, and 5 introduce the frameworks that compose this approach, which we believe are particularly important for the normative systems literature. Our argument is that any system that intends to regulate the behavior of its interacting agents (community members) in an open and dynamic environment should be able to identify evolving violating behavior. This ability is especially critical when considering issues like discrimination, hate speech, and cyberbullying that represent real damage to people's lives and interactions, besides affecting the community experience and engagement in online platforms.

Moreover, in addition to learning norm violations, our multi-scenario approach incorporates interpretability tools to present the elements of an action that contribute to its identification as a norm violation, where these elements can be a set of features that represent that action or words in a text that the action is introducing. In this context, we employ two algorithms, namely Local Interpretable Model-Agnostic Explanations (LIME) (q.v. Section 2.5.1) and Integrated Gradients (IG) (q.v. Section 2.5.2). These algorithms enable the presentation of interpretability information to community members in three layouts: a) local interpretability, which presents relevant elements of a specific action; b) the sum of relevance scores, which presents the relevant elements considering all actions in the dataset so far; c) and a combination of both.

Figure 6.1 illustrates these different layouts. First, Figure 6.1a depicts an example of the local interpretability layout obtained with the IG algorithm. It describes the impact of each word on identifying a given text as a norm violation, enabling users to understand specific problematic words in their text that may contribute to norm violations. Second, Figure 6.1b presents the sum of the relevance scores, which is obtained by summing the relevance value of each word based on its occurrence in the entire dataset used to train the ML model (q.v. Algorithm 5.1, line 20). This layout provides users with a comprehensive general overview of problematic words that may result in norm violations, which can also represent the community's current view. The third layout combines Figures 6.1a and 6.1b. In this case, the layout first presents the local interpretability, followed by the list of the most relevant words.

As the layouts in Figure 6.1 present different information to community members, it is worth conducting a user study to asses their potential influence on community members' views regarding norm violations. Thus, our user study aims to provide empirical evidence regarding the effective



(a) Local interpretability.

(b) sum of relevance scores.

Figure 6.1: The local interpretability and sum of relevance scores layouts. Specifically, Figure 6.1a represents the interpretability of classifying the specific text as belonging to the "Misogyny" class, while Figure 6.1b illustrates the sum of relevance scores for the "Misogyny" class considering words across all instances in the training dataset. Regarding local interpretability (q.v. Figure 6.1a), the intensity of the green shade indicates the relevance of the highlighted word to violation detection, demonstrating the contribution of those words to identifying the text as misogynistic. In contrast, the intensity of the red shade is related to the decrease in the violation confidence, demonstrating the contribution of those words to classifying the text as not misogynistic).

use of ML-model interpretability and to investigate how these layouts can help mitigate violating behavior online.

Specifically, we design this user study to assess three interpretability layouts based on participants' responses. The IG algorithm provides the interpretability data, while the CAL framework trains the ML model (q.v. Chapter 5).[1] We directly ask participants how their views change when presented with interpretability results from each of the three layouts. To achieve this, we create a questionnaire for each layout, consisting of 20 tuples, each comprising two questions. The questionnaire begins by presenting a sentence without interpretability data, such as Figure 6.2. Next, after answering the question with no interpretability data, participants reevaluate the same sentence. However, this time with information from the interpretability tool, as illustrated in Figure 6.3. Concretely, the participants assess each sentence twice, one with no interpretability data and the other with interpretability data.[2] These questions allow us to investigate the impact of interpretability information on participants: do they change people's views?

Finally, the remainder of this chapter is divided as follows. Section 6.1 presents the design of our user study to assess the potential influence of three interpretability layouts when participants evaluate text sentences containing hate speech (namely, misogyny and racism). Then, Section 6.2 outlines the results with statistical and qualitative analysis of questionnaire responses. Lastly, Section 6.3 provides a comprehensive summary of the key points discussed in this chapter.

[1] We use IG because it provides information on relevant words as considered by the ML model, and CAL because it learns with text data and can handle datasets with a limited size.

[2] Annex E provides the sentences and the examples for the other two interpretability layouts.

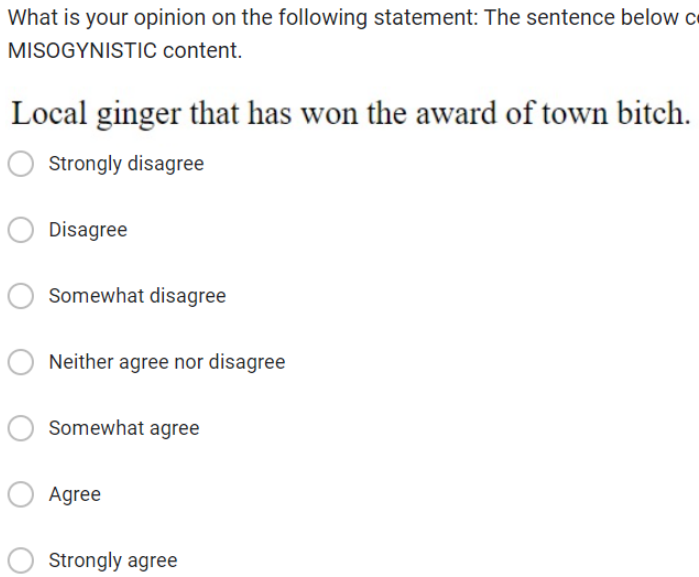


Figure 6.2: A question that inquires participants about their views on misogyny with no interpretability information. The participant must choose whether they agree or disagree with classifying this text as misogynous using the 7-point Likert scale.

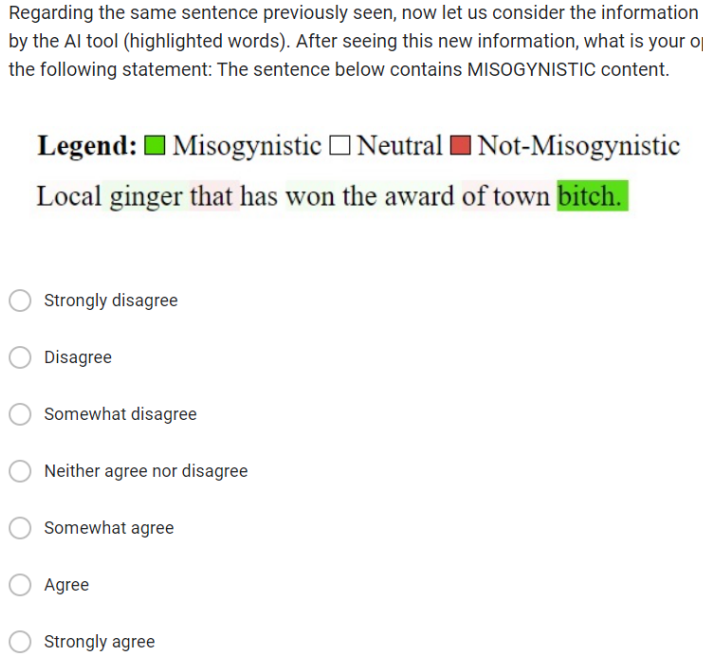


Figure 6.3: A question that inquires participants about their views on misogyny while considering local interpretability information provided by the IG algorithm. Participants must answer this question right after responding to the question about the same sentence with no interpretability data, as depicted in Figure 6.2. Again, the participant must choose whether they agree or disagree with classifying this text as misogynous using the 7-point Likert scale.

6.1 Study Design

Our user study evaluates three distinct interpretability layouts, comparing them against a baseline condition where participants only interact with a text (q.v. Figures 6.2 and 6.3). The study design comprises both within-subject and between-subject settings. Within-subject refers to assessing whether the information of a given interpretability layout influences the participant's classification of a sentence as hate speech or not. This experiment setting considers the same group of participants. In contrast, the between-subject component of the study assesses whether one of the interpretability layouts has a more significant impact on the participant's classification than another, which involves using different groups of participants. We randomly assign participants to one of three groups. Each group evaluates a different interpretability layout[3] and consists of 38 participants, totaling 114, and all groups share the same demographic distribution (gender and ethnicity). In adherence to ethical considerations associated with human studies, the experiment protocol was subject to review by the Universitat Autònoma de Barcelona ethics committee.

We employ a power analysis to determine the appropriate group size, which requires defining the following factors:

- Statistical power - the probability that the statistical test will reject the null hypothesis and obtain a statistically significant result (Cohen, 1992).
- Effect size - the quantitative indicator that measures the magnitude of the difference between distributions (i.e., the impact of a layout) when conducting a statistical analysis (Brysbaert & Stevens, 2018).
- Significance level (α) - the threshold to define the presence of statistically significant difference (Cohen, 1992).
- Cohen's d value - the measure of effect size (Brysbaert & Stevens, 2018).

This power analysis considers the effect size measured with Cohen's d value (Cumming, 2014) obtained from the investigation by F. Yang et al. (2020). This work is particularly relevant to our user study because it addresses a similar experiment setting, i.e., the investigation of interpretability in ML models. We aim to achieve a statistical power of 0.95 while maintaining a significant level of $\alpha = 0.05$, for which we set Cohen's d value to 0.84 (F. Yang et al., 2020). Moreover, we use the two-tailed T-test method. This statistical hypothesis test examines the existence of a significant difference between two distributions in both directions, whether the difference indicates an increase or decrease in the influence of an interpretability layout (Park, 2010). Consequently, following this calculation, we get to 38 participants per group. Lastly, it is worth noting that our required number of participants is consistent with the sizes employed in other studies in the literature (Arora et al., 2022; Chu et al., 2020; Schuff et al., 2022; F. Yang et al., 2020), aligning with established protocols in the field and further supporting our decision.

Each participant must evaluate 20 sentences, ten for each of the two hate speech classes considered in this study, "Misogyny" and "Racism." Each sentence will receive two questions, as shown in Figures 6.2 and 6.3. The first obtains the participant's view on identifying the sentence as misogynistic or racist without providing interpretability information. This serves as a baseline. The second obtains the participant's view after introducing interpretability information from one of the interpretability layouts, which could be any of the layouts in Figure 6.1. It is worth noting that the participants across the three distinct groups see the same 20 sentences. The difference between the groups lies in their usage of different interpretability layouts.

[3]We do this to ensure that participants do not use information from one interpretability layout when answering the questionnaire related to another interpretability layout.

Additionally, to determine the number of sentences in the study, we consider three main points: 1) ensure that the task completion time is not excessively long to avoid potential fatigue effects (Y. Zhang et al., 2018); 2) limit the number of hate-speech sentences evaluated by the participants to avoid any negative impact (e.g., anxiety, stress, and sadness) on their mental health (Karunakaran & Ramakrishan, 2019; Steiger et al., 2021); and 3) align the number of sentences with previous studies that achieved significant results (Chu et al., 2020; P. De Vries et al., 2003; F. Yang et al., 2020).

The selected sentences are balanced to ensure that both violation classes are represented, choosing ten distinct sentences for each of the two respective classes. To avoid any effect grammatical mistakes might have on participants' evaluations, we apply grammatical corrections when needed,[4] without changing the sentences' meaning. We obtained 40 answers from each participant during the user study. This includes two answers for each sentence. First, the participant evaluates the sentence without information from the interpretability layout (baseline). Next, the participant evaluates the same sentence for the second time but now with information from the interpretability layout. In both cases, participants rate the classification of each sentence on a 7-point Likert scale. Across all three interpretability layouts, we collected 1,520 answers per layout,[5] resulting in 4,560 answers for the entire study. We note that the questionnaire presents the text sentences randomly to address potential carry-over effects.

It is worth noting that although our use case comprises six violation classes (q.v. Section 1.1.3), our user study evaluates only two ("Misogyny" and "Racism"). The reason behind this choice is to maintain the study concise and allow us to control the variables in the experiment, which we accomplish by accounting for demographic factors (gender and ethnicity) in our statistical analysis. Specifically, this involves limiting gender identification to *male* and *female*, and ethnicity identification to *black* and *white*.[6] This approach takes into consideration that gender may influence perceptions of misogynistic behavior, with women reportedly identifying instances of this behavior more frequently than men (Kirkman & Oswald, 2020). Furthermore, black and white individuals perceive racism differently, with distinct perceptions about attention to racial issues, inequality, and violence targeting black people (Center, 2020). Including these demographic variables in our statistical analysis enables an evaluation of whether participant characteristics influence hate speech classification. In other words, our statistical analysis aims to isolate the variable of interest (interpretability layout) while simultaneously accounting for potential confounding variables (gender and ethnicity). By adopting this approach, we preserve the conclusions of our statistical analysis, ensuring that confounding variables do not incorrectly impact our results about the true relationship between the variables of interest.

After collecting participants' responses,[7] it is necessary to estimate their classification ratings of a text sentence as a violation. To accomplish this, we employ the Generalized Additive Model (GAM) (q.v. Section 2.6).[8] Our GAM model comprises fixed effects, including interpretability layouts (also referred to as treatment) and demographic aspects (gender and ethnicity), and random effects, such as the participants and text sentences. Equation 6.1 describes the specific model for this study, while Section 6.2 presents the results of fitting this model to our response data.

$$y_{target} \leftarrow \beta_0 + \beta_{treat} \times x_{treat} + \beta_{dem} \times x_{dem} + \alpha_{user} \times x_{user} + \alpha_{sent} \times x_{sent} \tag{6.1}$$

y_{target} refers to participants' answers on the Likert scale, while x_{user} and x_{sent} refer to random effects corresponding to individual participants and text sentences, respectively. Regarding fixed

[4] For instance, the sentence "fuck y all N-Word" was corrected to "Fuck you all N-Word."

[5] 38 participants providing 40 answers each.

[6] We note the existence of other gender and ethnicities identities (Center, 2020; Pega & Veale, 2015). However, aiming to keep the study concise and statistically sound, we limit the options to only two for each class.

[7] Response data available at https://bitbucket.org/thiago-phd/user_study.

[8] GAM enables us to capture the relationship between variables using nonlinear functions to model the response data. Concretely, this allows us to estimate the participants' confidence in classifying a sentence as hate speech.

effects, x_{dem} represents demographic aspects, and x_{int} represents the different interpretability layouts. Furthermore, β_0 describes the intercept.[9] β_{int} and β_{dem} refer to the intercept for interpretability layouts and demographic aspects, respectively. Lastly, α_{user} addresses user-specific effects, while α_{sent} addresses sentence-specific effects.

To conduct this user study, we recruit participants from the Prolific online crowdsourcing platform (Prolific, 2023a), which complies with the European GDPR data protection and treatment framework (Voigt & Von dem Bussche, 2017). Specifically, to complete the study, participants must read the research goal, give their consent, and answer the questionnaire. We select this platform because it covers the following requirements: 1) easy integration with external forms (e.g., Alchemer) and easy management of the study lifecycle (i.e., the response collection stages); 2) our previous experience with the platform; and 3) as presented in the Fairwork Cloudwork report (Fairwork, 2022), it implements policies aiming to improve work conditions by mitigating precarity and overwork while allowing researchers to directly set the amount paid to each worker.

We set the payment to £13,00 an hour.[10] Since the maximum time of the experiment is 30 minutes, each participant is paid £6,50, regardless of whether they use all this time or not (within this 30-minute window). Additionally, our consent form informs participants that if they decide to leave the study without completing it, they will receive compensation according to the time spent on the questionnaire (based on the £13,00 per hour rate). Following ethical guidelines defined in the platform (Prolific, 2023b), we set payment to all participants equally, avoiding negative psychological effects on crowdsourcing workers due to payment based on performance conditions (e.g., getting the "right answers") (Sayre, 2023). For questionnaires, the platform recommends payments of £9,00 per hour. However, due to the nature of our sentences (hate speech), we add £4 per hour (44% increase).

To ensure the quality of answers, we use attention checks to measure the participant's attention while answering our questionnaire. Six of the 114 initially recruited participants failed these attention checks. Thus, we recruited six additional people to achieve our desired number of participants.

6.2 Results

6.2.1 Within-Subject

The within-subject part of our user experiments presents three sets of results, one for each interpretability layout. Our analysis compares the interpretability layouts to the baseline to obtain these results. In practice, this comparison allows us to understand whether the interpretability layouts influence the participants' classification of a sentence as a particular class of violating behavior, namely "Misogyny" and "Racism." For the GAM model implementation described in Equation 6.1, we use the R programming language (R Core Team, 2023).[11]

This work fits the GAM model employing linear terms, a useful approach when a solution requires handling categorical variables as predictors. In our implementation, categorical variables such as interpretability layout (treatment), gender, and ethnicity are transformed into factors and referred to as fixed terms. This transformation allows the GAM function to fit the model with a fixed effect associated with each level (i.e., a value that can be assigned to the variable) of the category. Thus, this feature design allows us to depict the results specifically for each category level and provide details on these levels' influence.

[9] The intercept is a term in the model to represent the estimated value of the dependent variable when the values of the independent variables are 0.

[10] £ refers to Pound Sterling.

[11] R is an Open Source programming language designed for statistical computing. It is important in our context because it provides extensive data analysis and presentation tools. The source code for this implementation is available at https://bitbucket.org/thiago-phd/user_study.

In addition to the linear terms above, the smoothing parameter[12] is optimized in the model implementation when we fit a GAM to the data. This parameter is particularly important in this context, as it assists in fitting the underlying data trends rather than the noise, thus effectively preventing issues related to over-fitting and under-fitting. In practice, to select the smooth parameter, our implementation employs the Restricted Maximum Likelihood (REML) method since this is the most likely to provide reliable and stable results (Wood, 2011).

In summary, with these concepts, we can define the GAM formula implemented in our within-subject part of the user study as follows:

$$Answer \leftarrow Treatment + Gender + Ethnicity + Smooth(User) + Smooth(Sentence) \quad (6.2)$$

Equation 6.2 represents the implementation of the model described in Equation 6.1. $Smooth()$ applies the smooth function in the selected variables ($User$ and $Sentence$). $Treatment$ (interpretability layouts), $Gender$, and $Ethnicity$ are the fixed terms with the following possible values:

- Treatment - a) local interpretability; b) sum of relevance scores ("list"); and c) combined approach.
- Gender - a) male; and b) female.
- Ethnicity - a) black; and b) white.

$User$ and $Sentence$ are random factors. Thus, the possible values in these variables are the individual participants of the user study (38 for each interpretability layout) and the 20 sentences we present to these participants, respectively.

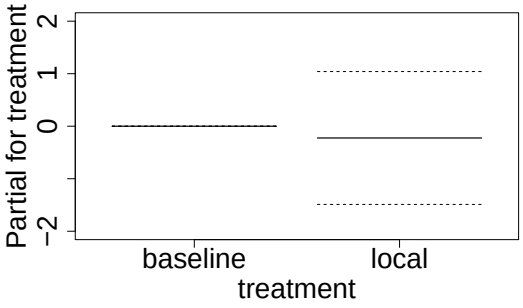
Figure 6.4 provides an overview of the results for the local interpretability layout compared to the baseline, including a sub-figure for each fixed term. Specifically, results indicate that none of the factors significantly influence participants' classifications, as corroborated by the $\Pr(>|z|)$ values in Table 6.1. In other words, the participants' views about violating behavior are not affected by the interpretability layout, nor their gender or ethnicity.

Still comparing the local interpretability setting to the baseline, Figure 6.5 depicts the results for the smooth factors. Specifically, Figures 6.5a and 6.5b illustrate a QQ-plot[13] that describes the observed distribution of the random effects' estimated values against the theoretical (Gaussian) expectation. Our findings indicate that the individual sentences and users influence the classification ratings, as corroborated in Table 6.2. The reasons for this conclusion are twofold: 1) some sentences might exhibit more explicit violations than others. Thus, a sentence containing explicit racist terms may prompt participants to assign higher ratings on the Likert scale than sentences containing subtle expressions of racism; and 2) some participants may have divergent perspectives regarding the severity of these violations. Thus, these participants may assign higher ratings on the Likert scale than others, reflecting a greater inclination towards classifying hate speech with severity.

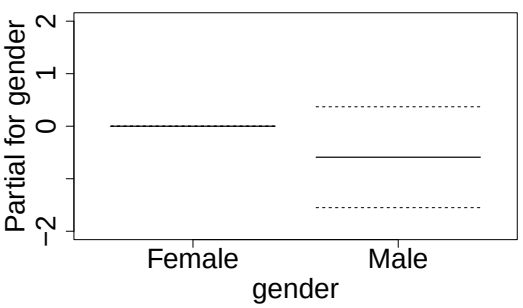
Regarding the interpretability layout that uses the sum of relevance scores (referred to as "list" in the figures), results are depicted in Figures 6.6 and 6.7, with corresponding statistical details provided in Tables 6.3 and 6.4. Like the local interpretability case, the results indicate that no fixed terms (treatment, gender, and ethnicity) influence participants' classification of violating behavior. The individual users and sentences are the only effects influencing the violating classification ratings.

[12] A smoothing parameter controls the balance (i.e., find the trade-off) between capturing data patterns and avoiding over-fitting, keeping the model as simple as possible.

[13] A Quantile-Quantile plot is a graphical method to compare probability distributions (Wilk & Gnanadesikan, 1968). In our case, it compares the distribution of values for random factors ($User$ and $Sentence$) with the normal population, assessing the similarity of their distributions.



(a) Treatment



(b) Gender

(c) Ethnicity

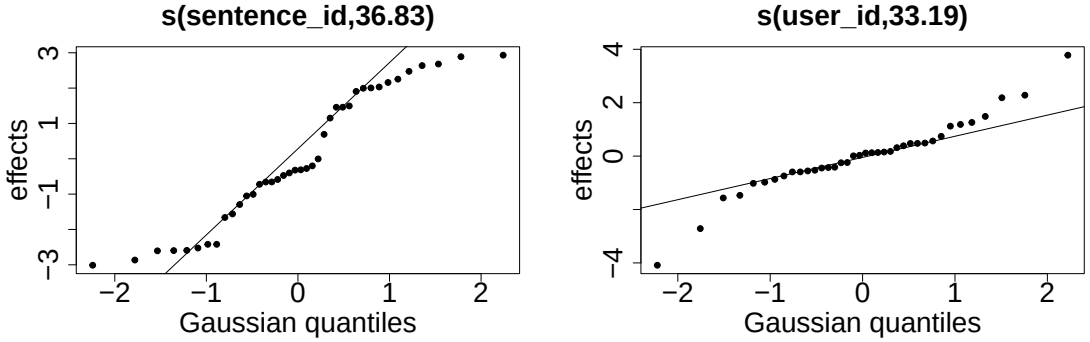
Figure 6.4: Fixed terms of the GAM model for the experiment considering the local interpretability layout. The baseline, female, and black values for the partial effects are zero because they represent the reference level. In contrast, the values for the specific coefficients, e.g., local, male, and white, vary since they represent the estimation of their deviation from the intercept (reference values). Lastly, the dotted lines present the interval of the standard (Std.) error.

Factors	Estimate	Std. Error	z-value	$\Pr(>\|z\|)$
Intercept	4.0429	0.6049	6.683	$2.33e^{-11}$
Treatment - Local	-0.2243	0.6324	-0.355	0.723
Gender - Male	-0.5905	0.4803	-1.230	0.219
Ethnicity - White	-0.3223	0.4802	-0.671	0.502

Table 6.1: The fixed terms for the experiment considering the local interpretability layout. The parametric coefficients have values for Estimate, Standard Error, z-value, and $\Pr(>|z|)$. The z-value relates to the estimation's mean, representing the number of standard deviations from this mean. Lastly, the $\Pr(>|z|)$ column depicts the p-value for the coefficients. Specifically, it indicates the probability of obtaining a value of z bigger than our calculated absolute z-value.

The set of results for the combined interpretability layout is the last part of the within-subject experiments. Figure 6.8 describes the results for fixed terms, while Figure 6.9 depicts results for the smooth terms. Moreover, Tables 6.5 and 6.6 detail the corresponding statistical values. As in the previous interpretability layout cases, no fixed term influences participants' classification of violating behavior, with only the specific user and sentence being relevant.

In summary, our findings demonstrate that no interpretability layout (local, sum of relevance scores, or combined) influences participants' views on misogyny and racism. Furthermore, the



(a) Sentence Id

(b) User Id

Figure 6.5: Smooth terms of the GAM model for the experiment considering the local interpretability layout. The Y-axis (effects) refers to the estimated slope, which is a consequence of the effect being modeled as a random slope. The Gaussian quantiles reflect the values of a standard normally distributed variable, while the Y-axis shows the predicted values of the random intercept. Consequently, the deviations from the line indicate the deviation of the data points from the expected normal distribution.

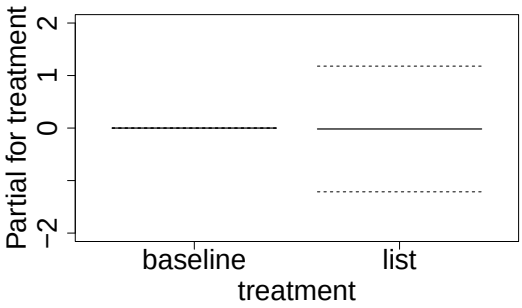
Smooth Terms	EDF	Ref.df	Chi.sq	P-value
User Id	33.19	35	1374	$< 2e^{-16}$
Sentence Id	36.83	38	2367	$< 2e^{-16}$

Table 6.2: Approximate significance of smooth terms for the experiment considering the local interpretability layout. With Effective Degrees of Freedom (EDF), Reference Degrees of Freedom (Ref.df), Chi.sq, and p-value. EDF represents the complexity of the smooth. For instance, an EDF of 1 indicates a straight line. Higher EDFs represent more wiggly curves. The Ref.df column contains the maximum degrees of freedom for each term used in calculating the p-value. The Chi.sq represents the test statistic to determine the overall significance of the smooth. Lastly, the p-value is the result of the test.

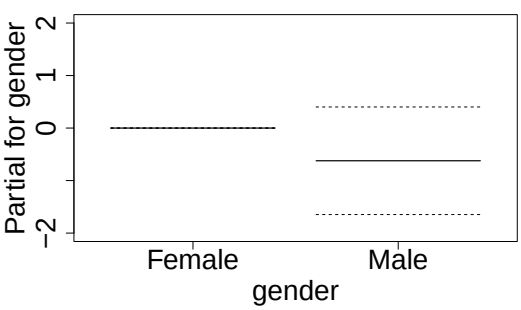
Factors	Estimate	Std. Error	z Value	$\Pr(>\|z\|)$
Intercept	4.8605	0.6170	7.88	$3.3e^{-15}$
Treatment - List	-0.0174	0.5984	-0.03	0.98
Gender - Male	-0.6228	0.5121	-1.22	0.22
Ethnicity - White	-0.1098	0.5121	-0.21	0.83

Table 6.3: The fixed terms for the experiment considering the sum of relevance scores interpretability layout ("list"). The parametric coefficients have values for Estimate, Standard Error, z-value, and $\Pr(>|z|)$. The z-value relates to the estimation's mean, representing the number of standard deviations from this mean. Lastly, the $\Pr(>|z|)$ column depicts the p-value for the coefficients. Specifically, it indicates the probability of obtaining a value of z bigger than our calculated absolute z-value.

demographic factors under investigation, specifically gender and ethnicity, do not yield any significant influence on participants' assessments. We argue that participants start with a given view of what constitutes hate speech when they answer the questionnaire, or our model's output already aligns with their classification of hate speech. Thus, any information provided by the different interpretability layouts fails to change the participant's rating while evaluating hate speech. This conclusion is supported by our qualitative analysis (q.v. Section 6.2.3), with comments highlighting participants' reliance on their own view of hate speech when assessing violating behavior.



(a) Treatment



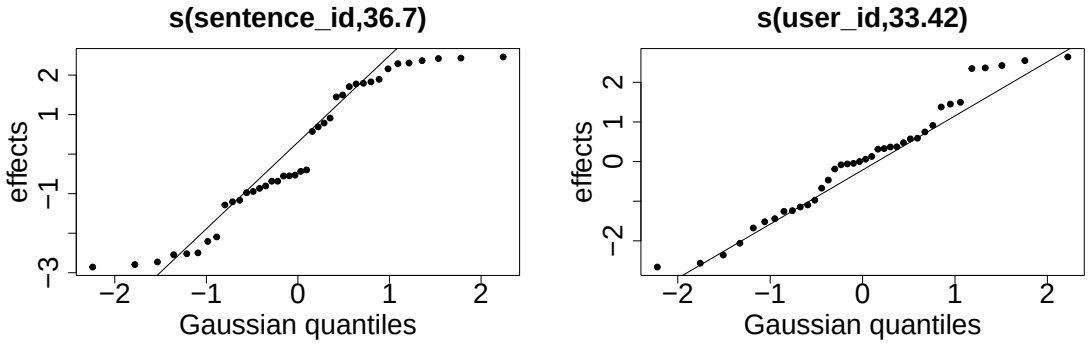
(b) Gender

(c) Ethnicity

Figure 6.6: Fixed terms of the GAM model for the experiment considering the sum of relevance scores layout (referred to as "list"). The baseline, female, and black values for the partial effects are zero because they represent the reference level. In contrast, the values for the specific coefficients, e.g., local, male, and white, vary since they represent the estimation of their deviation from the intercept (reference values). Lastly, the dotted lines present the interval of the Std. error.

Smooth Terms	EDF	Ref.df	Chi.sq	P-value
User Id	33.4	35	1623	$< 2e^{-16}$
Sentence Id	36.7	38	2426	$< 2e^{-16}$

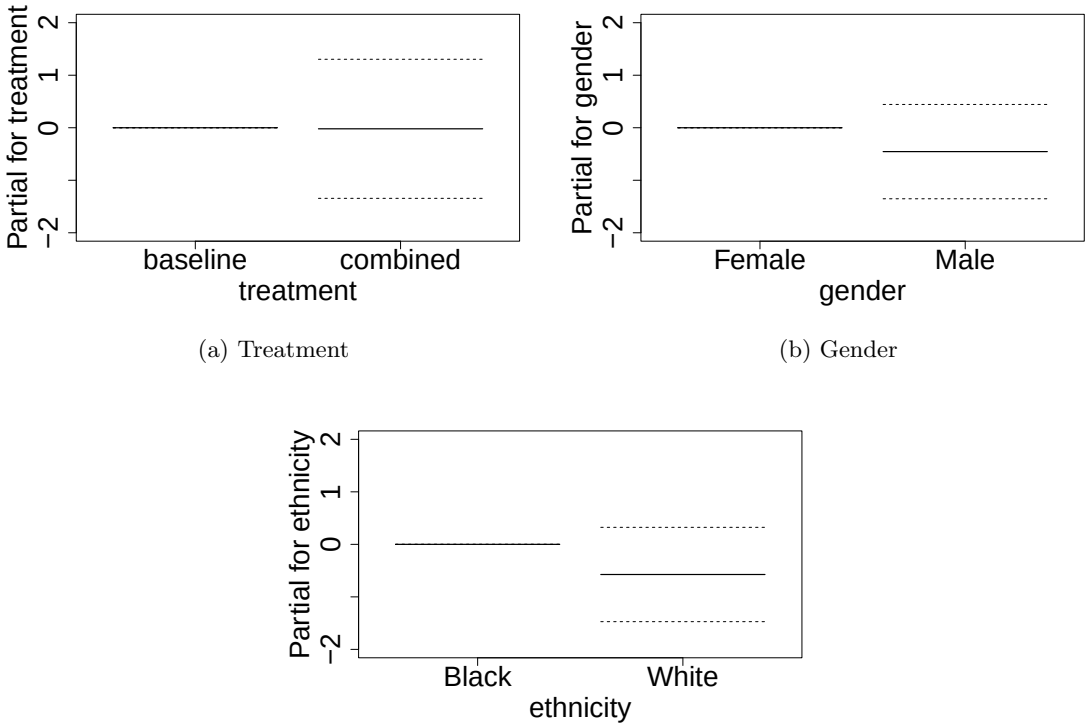
Table 6.4: Approximate significance of smooth terms for the experiment considering the sum of relevance scores interpretability layout ("list"). With Effective Degrees of Freedom (EDF), Reference Degrees of Freedom (Ref.df), Chi.sq, and p-value. EDF represents the complexity of the smooth. For instance, an EDF of 1 indicates a straight line. Higher EDFs represent more wiggly curves. The Ref.df column contains the maximum degrees of freedom for each term used in calculating the p-value. The Chi.sq represents the test statistic to determine the overall significance of the smooth. Lastly, the p-value is the result of the test.



(a) Sentence Id

(b) User Id

Figure 6.7: Smooth terms of the GAM model for the experiment considering the sum of relevance scores interpretability layout. The Y-axis (effects) refers to the estimated slope, which is a consequence of the effect being modeled as a random slope. The Gaussian quantiles reflect the values of a standard normally distributed variable, while the Y-axis shows the predicted values of the random intercept. Consequently, the deviations from the line indicate the deviation of the data points from the expected normal distribution.

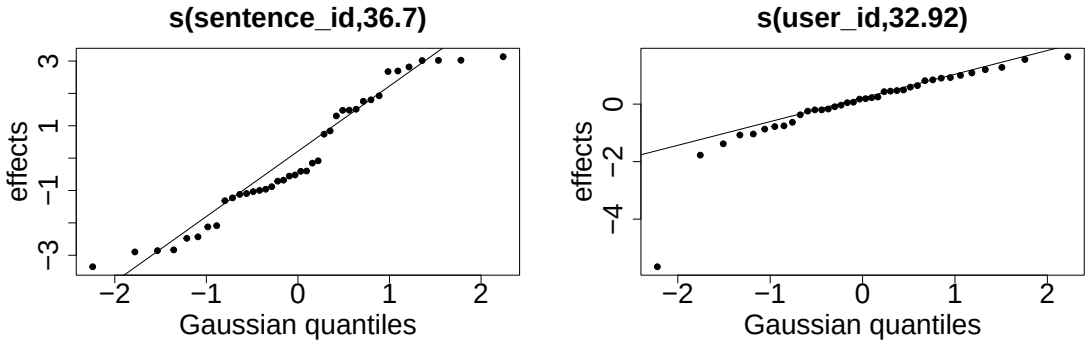


(a) Treatment

(b) Gender

(c) Ethnicity

Figure 6.8: Fixed terms of the GAM model for the experiment considering the combined interpretability layout. The baseline, female, and black values for the partial effects are zero because they represent the reference level. In contrast, the values for the specific coefficients, e.g., local, male, and white, vary since they represent the estimation of their deviation from the intercept (reference values). Lastly, the dotted lines present the interval of the Std. error.



(a) Sentence Id

(b) User Id

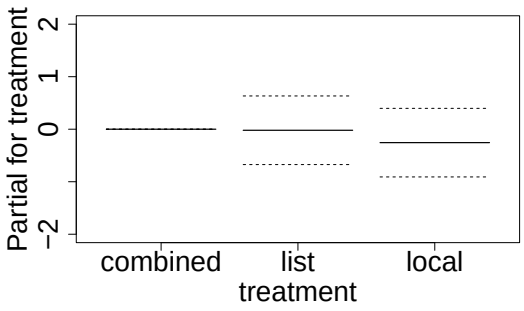
Figure 6.9: Smooth terms of the GAM model for the experiment considering the combined interpretability layout. The Y-axis (effects) refers to the estimated slope, which is a consequence of the effect being modeled as a random slope. The Gaussian quantiles reflect the values of a standard normally distributed variable, while the Y-axis shows the predicted values of the random intercept. Consequently, the deviations from the line indicate the deviation of the data points from the expected normal distribution.

Factors	Estimate	Std. Error	z Value	$\Pr(>\|z\|)$
Intercept	4.1039	0.6110	6.72	$1.9e^{-11}$
Treatment - Combined	-0.0202	0.6620	-0.03	0.98
Gender - Male	-0.4544	0.4490	-1.01	0.31
Ethnicity - White	-0.5741	0.4490	-1.28	0.20

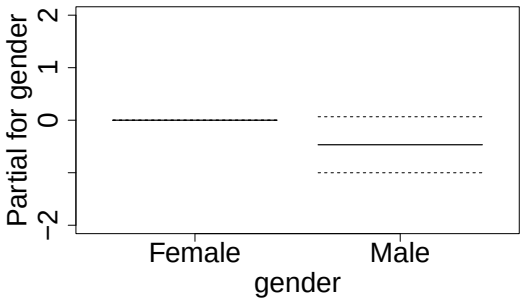
Table 6.5: The fixed terms for the experiment considering the combined interpretability layout. The parametric coefficients have values for Estimate, Standard Error, z-value, and $\Pr(>|z|)$. The z-value relates to the estimation's mean, representing the number of standard deviations from this mean. Lastly, the $\Pr(>|z|)$ column depicts the p-value for the coefficients. Specifically, it indicates the probability of obtaining a value of z bigger than our calculated absolute z-value.

Smooth Terms	EDF	Ref.df	Chi.sq	P-value
User Id	32.9	35	983	$< 2e^{-16}$
Sentence Id	36.7	38	1718	$< 2e^{-16}$

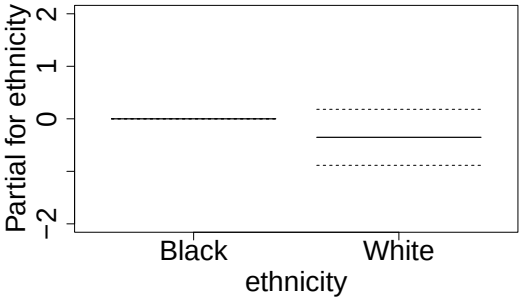
Table 6.6: Approximate significance of smooth terms for the experiment considering the combined interpretability layout. With Effective Degrees of Freedom (EDF), Reference Degrees of Freedom (Ref.df), Chi.sq, and p-value. EDF represents the complexity of the smooth. For instance, an EDF of 1 indicates a straight line. Higher EDFs represent more wiggly curves. The Ref.df column contains the maximum degrees of freedom for each term used in calculating the p-value. The Chi.sq represents the test statistic to determine the overall significance of the smooth. Lastly, the p-value is the result of the test.



(a) Treatment



(b) Gender



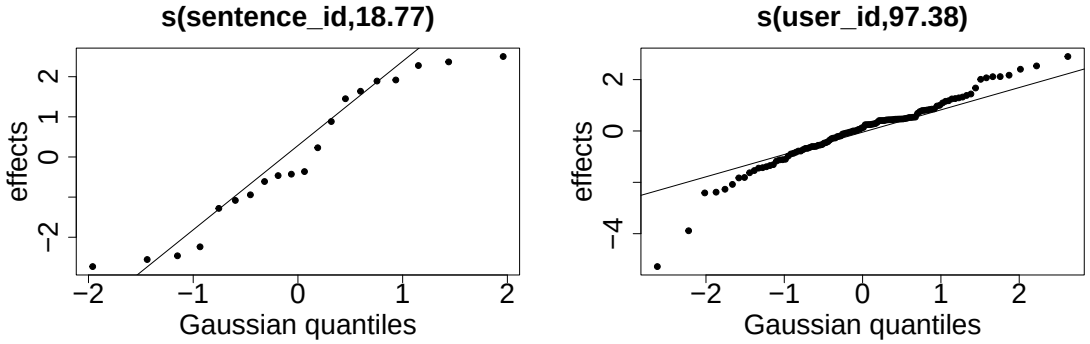
(c) Ethnicity

Figure 6.10: Fixed terms of the GAM model for the between-subject experiment. The combined, female, and black values for the partial effects are zero because they represent the reference level. In contrast, the values for the specific coefficients, e.g., list, local, male, and white, vary since they represent the estimation of their deviation from the intercept (reference values). Lastly, the dotted lines present the interval of the Std. error.

6.2.2 Between-Subject

In addition to executing a within-subject analysis, we aim to investigate direct differences between interpretability layouts, which enables us to understand if an interpretability layout asserts more influence on participants than others.

The results illustrated in Figure 6.10 and detailed in Table 6.7 indicate that no particular treatment (interpretability layout) influences participants' responses more than the others. Furthermore, the other fixed terms, gender and ethnicity, similarly do not affect the ratings across the different layouts. In the context of the smooth terms, Figure 6.11 and Table 6.8 demonstrate that the differences in classification ratings can be attributed to the specific user and sentence, aligning with the findings in our within-subject analysis.



(a) Sentence Id

(b) User Id

Figure 6.11: Smooth terms of the GAM model for the between-subject experiment. The Y-axis (effects) refers to the estimated slope, which is a consequence of the effect being modeled as a random slope. The Gaussian quantiles reflect the values of a standard normally distributed variable, while the Y-axis shows the predicted values of the random intercept. Consequently, the deviations from the line indicate the deviation of the data points from the expected normal distribution.

Factors	Estimate	Std. Error	z Value	$\Pr(>\|z\|)$
Intercept	4.1595	0.5041	8.25	$2e^{-16}$
Treatment - List	-0.0202	0.3267	-0.06	0.95
Treatment - Local	-0.2558	0.3265	-0.78	0.43
Gender - Male	-0.4674	0.2666	-1.75	0.08
Ethnicity - White	-0.3523	0.2666	-1.32	0.19

Table 6.7: Comparison of three interpretability layouts. The parametric coefficients have values for Estimate, Standard Error, z-value, and $\Pr(>|z|)$. The z-value relates to the estimation's mean, representing the number of standard deviations from this mean. Lastly, the $\Pr(>|z|)$ column depicts the p-value for the coefficients. Specifically, it indicates the probability of obtaining a value of z bigger than our calculated absolute z-value.

Smooth Terms	EDF	Ref.df	Chi.sq	P-value
User Id	97.4	109	2918	$< 2e^{-16}$
Sentence Id	18.8	19	2071	$< 2e^{-16}$

Table 6.8: Approximate significance of smooth terms considering the between-subject experiment. With Effective Degrees of Freedom (EDF), Reference Degrees of Freedom (Ref.df), Chi.sq, and p-value. EDF represents the complexity of the smooth. For instance, an EDF of 1 indicates a straight line. Higher EDFs represent more wiggly curves. The Ref.df column contains the maximum degrees of freedom for each term used in calculating the p-value. The Chi.sq represents the test statistic to determine the overall significance of the smooth. Lastly, the p-value is the result of the test.

6.2.3 Qualitative Evaluation

The previous two sections address the statistical analysis of our user study. Differently, here we investigate the qualitative aspect of this experiment, aiming to gain insights from participants' comments regarding what they consider relevant while evaluating the sentences and performing the classification task. We provide an optional box for open-text comments in our questionnaire to collect this data. Although most of the 114 comments do not convey any relevant message to our investigation, we shall discuss the comments that complement our statistical analysis in the rest of this section. Specifically, we analyze comments that fit into three categories: 1) familiarity with the task, comprising four comments; 2) questionnaire design, comprising five comments; and 3) model accuracy, comprising four comments. Next, we discuss the most representative instances within each category.

Familiarity with the Task. An interesting insight regarding participants' perception of how they used information from the interpretability layouts and their familiarity with the hate speech classification task is present in the following comment: "*I did not find that the AI-generated significance values affected my perceptions of whether statements were misogynistic or racist, but rather confirmed my thoughts, with a couple of exceptions that read more benign (something you'd read in a history book, e.g., the 'royal strippers' example) that the AI represented were more than I believed.*" This observation supports our reasoning that, in the context of hate speech - a common type of violation that people are exposed to in online interactions, participants tend to have a clear view of what constitutes hate speech before they are presented with interpretability information. Concretely, this comment highlights two scenarios: 1) our interpretability information already aligns with what participants expect of hate speech, i.e., the ML model correctly learns these kinds of violations; 2) if the interpretability information does not align with participants' definition of hate speech, then it is difficult to change people's views on what they consider hate speech when they evaluate the sentences present in our study (such as the presence of sentences with "royal strippers"). The following comment provides further insight into participants' familiarity with the hate speech task: "*...Maybe it went so quick [the time needed to complete the questionnaire] because the language used was very obvious for me in choosing an answer.*" This observation highlights that the participant has a clear view of what constitutes hate speech. Moreover, the sentences evaluated in our study clearly contain terms related to hate speech that people are familiar with.

Questionnaire Design. Another relevant observation relates to the questionnaire design, as highlighted in the following comment: "*This survey was a little unpleasant, as stated in the consent form. Thank you for not making this survey any longer than it is now.*" Our initial goal was to maintain survey conciseness, strictly including only the required number of sentences to achieve our predetermined statistical power. The comment's observation supports our initial decision, which is especially relevant since we are handling offensive language that could be hurtful for the readers, especially when they are numerous. As such, we strongly recommend that future research tackling sensitive topics like hate speech follow in our footsteps and carefully consider the number of violations presented in questionnaires, conducting statistical tests and comparing with existing literature to avoid exposing participants to sentences that may adversely affect their mental health. In this context, our work provides further metrics based on statistical analysis to assist in future questionnaire design processes (q.v. Section 6.1).

Model Accuracy. In this context, participants expressed concerns about the relevant terms identified by our interpretability approach. The following comment illustrates this concern: "*The highlighting seemed strangely unfocused in some instances...*" This observation highlights an important aspect in the ML field: the discrepancy between the ML model and human perception. Thus, we argue that if interpretability is not used to influence or correct a user's behavior, such

layouts can be useful to identify these discrepancies and trigger user feedback that would help correct the outputs of the ML model. Concretely, the existence of a contrast between participants' views and the model's output supports incorporating interpretability layouts to assist the evaluation of the model's behavior beyond conventional performance metrics (e.g., recall, F1-score, and precision). Moreover, given that our multi-scenario approach focuses on norm violation detection within contexts characterized by small labeled datasets, there might be instances where the model fails to capture terms commonly associated with offensive content. Consequently, this may lead to the model's inability to identify specific sentences as violating behavior, impacting its accuracy. For instance, the following comment is a representative case: "*The AI is unaware of racial nuances behind certain phrases such as 'big lipped' which was not even highlighted.*" This observation illustrates potential limitations in the detection capabilities of our model, especially when the model must handle terms that have limited representation in our training dataset (for instance, one occurrence). Integrating interpretability enhances the transparency of ML models by enabling people to understand the relevant terms that these models consider. This is particularly significant considering the dynamic nature of online interactions, where new sentences expressing hate speech emerge over time within a single community or as we incorporate data from diverse communities.

6.3 Summary

This chapter presented a user study to assess the potential influence of three interpretability layouts when participants evaluate hate speech in text. Specifically, our study focused on two hate speech classes: "Misogyny" and "Racism." With statistical and qualitative analysis of questionnaire responses, we aimed to provide empirical evidence in the context of interpretability for norm violation detection within online communities. Concretely, this investigation explored the impact of interpretability information on participants: do they influence people's views?

The three interpretability layouts considered in this work comprised local, the sum of relevance scores (referred to as "list"), and a combined approach, as illustrated in Figure 6.1. The Integrated Gradients (IG) algorithm provided interpretability data for the study (q.v. Section 2.5.2). Moreover, we used the model trained with the CAL framework (q.v. Chapter 5) to obtain the interpretability data.

Beyond exploring interpretability layouts, our study was interested in assessing the potential impact of other factors on participants' views of what constitutes a violation. Thus, our statistical model included terms related to participants' gender and ethnicity (q.v. Equation 6.2). Including these demographic characteristics was motivated by our focus on two violation classes: "Misogyny" and "Racism." Specifically, our statistical approach took into consideration that gender and ethnicity might influence perceptions of misogynistic and racist behavior(Center, 2020; Kirkman & Oswald, 2020), respectively. Thus, by considering information related to participants' characteristics in our statistical analysis, we ensured that these factors did not incorrectly impact our results.

This user study comprised both within-subject and between-subject analyses. Additionally, it covered a qualitative evaluation, in which we used participants' comments to complement our understanding of the statistical findings. Participants were randomly allocated into three groups, corresponding to the three distinct interpretability layouts. Following a power analysis, we determined a sample size of 38 participants per group. Each of these 38 participants had to evaluate 20 text sentences, equally distributed between the two violation classes. Participants performed two evaluations for each sentence: one without any information from the interpretability layout, and another with interpretability data from one of the three layouts. This study design yielded 40 responses from each participant, collected on a 7-point Likert scale. To estimate participants' ratings in classifying a text containing hate speech, we employed the Generalized Additive Model (GAM) (q.v. Section 2.6). By fitting this model to the response data, we analyzed the influence of

interpretability layouts, gender, ethnicity, individual sentences, and individual participants on the response (q.v. Section 6.2).

Results of our statistical analysis indicated that none of the interpretability layouts influenced participants' views regarding the classification of hate speech within the context of the "Misogyny" and "Racism" classes. Furthermore, in the comparative assessment between the different interpretability layouts, no layout demonstrated a larger influence over the others. The qualitative analysis, which integrated insights from participants' comments, leveraged our comprehension of these results. Two key points explain why interpretability layouts did not significantly influence participants' views: First, the participants' familiarity with hate speech, including the presence of explicit hate speech terms. Second, the alignment between the ML model's output and participants' views, where the interpretability layout identified relevant terms similar to those considered by the participants. Moreover, comments on the model's accuracy contributed valuable insights into the impact of incorporating ML interpretability and feedback mechanisms into our approach. First, interpretability addresses transparency by allowing people to understand the relevant terms considered by the ML model, while feedback enables people to correct misalignment between their views and the model's output. This is especially important in contexts where new instances of hate speech may emerge over time within a single community or with the inclusion of data from diverse communities. Second, interpretability layouts can provide insights into evaluating the ML model's behavior beyond traditional performance metrics (e.g., recall, F1-score, and precision), which is especially relevant in scenarios where understanding the rationale behind an ML model's decision is essential for optimal implementation.

Finally, the next chapter describes works related to our research. While it presents related works in the field of norm violation, ensemble and incremental learning, cross-community learning, and interpretability, it also analyses existing user studies and compares their results to ours.

Chapter 7

Literature Review

This chapter presents related work to the research reported in this thesis. First, we cite literature focusing on detecting detrimental behavior in online communities using Machine Learning (ML) models (q.v. Section 7.1). The idea is to present different approaches to this issue in diverse domains, highlighting the importance of research in the field. Second, we describe works that build an ensemble of classifiers trained using incremental learning approaches (q.v. Section 7.2). Subsequently, Section 7.3 presents the literature on cross-community learning, where works leverage the performance of a model using data from diverse sources, including solutions that employ transformer-based models, like Pre-Trained Language Models (PLMs) and adapters (q.v. Section 2.4). In Section 7.4, we describe works that handle interpretability in tabular and text classification tasks, demonstrating how this technique can improve human interaction in different contexts. Lastly, we investigate the literature that conducts experiments on how humans work with interpretability information (q.v. Section 7.5).

7.1 Norm Violation Detection

Detecting norm violations in online communities is one of the main challenges in this thesis. Moreover, this subject is relevant to previous research across diverse domains. Thus, in addition to the specific context of our investigation, which focuses on norm violation detection in Wikipedia article editing, we review previous research that has tackled similar challenges in online platforms, including Stack Overflow and various social media sites. Concretely, this review describes solutions, ML strategies, and datasets previously explored in these domains. Additionally, it compares our multi-scenario approach to existing solutions, emphasizing the contributions of this thesis.

Starting with existing works focusing on the Wikipedia online community to detect norm violations, Anand and Eswari (2019) apply Deep Learning (DL) to classify a comment as abusive or not based on a dataset from the talk page edit. Also, our initial work uses the Logistic Model Tree to learn to detect norm violations (Freitas dos Santos et al., 2022a).[1] 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. (2017) present a work that explores ML to detect norm violations in the Stack Overflow community. As in our work, they use specific data about the context of the community to train the ML models. In this case, instead of article edits, they analyze comments posted on

[1] We cite our initial work in the literature review instead of including it as part of our contributions. This decision is based on the fact that this initial work does not explore the same frameworks we investigate in this thesis. Notably, only the domain is similar.

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. (2017) focus on using a recommendation system to detect and recommend alternatives to the community members' posts. Continuing their work in (Cheriyan et al., 2021), the authors expand their solution to incorporate the detection of offensive language in four software engineering communities. They consider three violation classes: personal attack, racial discrimination, and swearing. Other ML techniques were also evaluated, ranging from Random Forest and Support Vector Machines to PLMs, which present the best classification performance. Both works (Cheriyan et al., 2017, 2021) use TF-IDF[2] Vectorizer, an automated method to obtain features from text data. TF-IDF ensures a consistent approach to feature extraction,[3] enables handling large amounts of text, and avoids using a manual process, which can be time-consuming. This differs from FeDAL, where the community provides the attributes (q.v. Section 3.5), allowing the incorporation of the community view to define the relevant features and the inclusion of contextual information in a domain-specific manner (creating a detailed set of features to formalize the data). Additionally, the differences between their PLM approach and ours are as follows: 1) LaMAL presents a continuous fine-tuning process to detect norm violations in binary and multi-label classification tasks (q.v. Section 4.3), and 2) CAL incorporates adapters, updating the network parameters using data from diverse communities to start the training process, which is especially useful when learning in newly created (or low-resource) communities with limited labeled data (q.v. Chapter 5).

Different researchers use transformer-based language models for violation detection in text classification tasks. Markov et al. (2022) create an ensemble using PLMs, Support Vector Machine, and feature information. Since their application considers the Dutch language, BERTje was used, which is explicitly trained on top of Dutch text sentences (W. De Vries et al., 2019). The authors used data from comments on Facebook and Twitter to evaluate their approach. While they mix text and features to solve violation classification tasks, our multi-scenario approach tackles them separately, aiming to avoid the need to have a featurization process when a community provides text data. For instance, LaMAL and CAL address hate speech detection without defining a set of attributes that encodes a text sentence. Consequently, these frameworks do not require the community members to create these attributes.

Intending to detect aggression and misogyny, Samghabadi et al. (2020) use BERT (Kenton & Toutanova, 2019) 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. This solution differs from CAL, as our framework adopts adapters for each violation class, incorporating data from source communities to initially train the transformer-based model.

Muslim et al. (2021) 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 sub-tasks: 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 their work, which uses cost-sensitive learning to evaluate the costs of mistakes made by the model, LaMAL employs binary focal loss to assess errors in the calculation of the loss function during the training process.

The work in (Akhter et al., 2021) compares conventional ML and DL methods. The main emphasis of their investigation is on low-resource language, particularly Urdu. The scarcity of resources for this language represents a challenge for DL methods to detect abusive language. Within this scope, they investigate four DL models: Convolutional Neural Network, Long Short-Term Memory, Bidirectional Long Short-Term Memory, and Convolutional Long Short-Term Memory.

[2]Term Frequency-Inverse Document Frequency (TF-IDF) (W. Zhang et al., 2011)

[3]This extraction process avoids possible variations introduced by different people in a manual process.

The authors argue that these are useful for capturing long-term dependencies in text sentences. YouTube is the community of interest. Like our work, they must also learn in context with an imbalanced dataset. However, they do not investigate transformer-based models. Furthermore, our research focuses on continuously adapting and interpreting elements contributing to norm violation detection. In contrast, their presentation focuses on comparing DL methods among themselves and against traditional ML models such as Naive Bayes, Instance-Based Learning, Support Vector Machine, Logistic Regression, and a rule-based approach. In conclusion, results show that the Convolutional Neural Network outperforms the other DL methods and that, within the specific language under investigation, DL methods perform better than ML methods.

Another research focusing on detecting norm-violating behavior within an online community is described by Hajibabaee et al. (2022). The authors explore the Twitter community. Their methodology comprises embedding approaches like TF-IDF, Word2Vec (Church, 2017), and FastText (Joulin et al., 2016) to obtain the embedding of text sentences. After obtaining this representation, the authors investigate the application of eight different ML methods. Furthermore, the classification task categorizes text sentences into hate speech, offensive language, or neither. Like our work, their dataset is annotated by crowdsourcing workers. Their findings describe an empirical study identifying that the best-performing methods are Adaboost, Support Vector Machine, and Multi-layer Perceptron, with TF-IDF embedding. Lastly, it is important to highlight that their work differs from the scope of this thesis in several aspects. Most notably, our research explicitly addresses concept drift, multi-scenario approaches, and interpretability.

Like CAL, Mutanga et al. (2020) use DistilBERT (Sanh et al., 2019) to detect violations in an imbalanced setting. Their findings show that DistilBERT outperforms other algorithms (e.g., BERT, RoBERTa (Y. Liu et al., 2019)) in terms of F-measure. The advantage of transformer-based models in this study is that they efficiently capture long-range dependencies and parallelization. Similar to our reasons for adopting DistilBERT, they highlight its small size, faster training time, and performance. Furthermore, they acknowledge the suitability of transformer-based models for learning from source data. However, in contrast to CAL, they do not investigate this aspect.

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 (Campos et al., 2013; Dell'Anna et al., 2022; Morales et al., 2015). For instance, Dell'Anna et al. (2022) propose the Data-Driven Norm Revision approach that relies on previously obtained knowledge of whether individual actions violate a norm or not. Their solution requires access to labeled data acquired from automated (or manual) classification processes, which is precisely the information our approach provides. In this context, our frameworks (LaMAL and CAL) complement their strategy by enabling the identification of multiple classes of norm violations occurring simultaneously and handling text-based scenarios.

7.2 Ensemble and Incremental Learning

Our multi-scenario approach aims to detect norm-violating behavior in domains with concept drift (i.e., changes in community members' views about what constitutes a norm violation) and imbalanced datasets (i.e., regular behavior occurs more frequently than violations). We accomplish that by adopting ensemble (q.v. Section 2.1) and incremental learning (q.v. Section 2.3). Thus, this section investigates existing works that adopt these techniques, comparing our multi-scenario approach to existing solutions to emphasize the contributions of this thesis. Concretely, we present solutions that tackle concept drift by employing incremental learning to continuously update the base ML model as data is made available. We argue that this approach is the most suitable to be deployed in a system supporting an online community due to the streaming characteristic of interactions. Additionally, we present solutions that employ ensemble learning to provide an efficient approach to learning in a context with imbalanced datasets.

Regarding the use of incremental learning in a setting with a class distribution that is highly imbalanced, the work in (Lebichot et al., 2021) builds a solution capable of detecting credit and debit card fraud. Like our use case, these transactions have a sequential nature, are highly imbalanced, and present concept drift. The authors report better results than when employing traditional offline learning approaches. To enhance incremental learning, they use ensemble learning to reduce variance and improve stability. Furthermore, they incorporate transfer learning to deal with information learned in a distinct task. One difference compared to FeDAL is that we use an active process (explicitly detecting the change in data distribution and when that change happens (Losing et al., 2017)) to detect concept drift, which is better suited to deal with major changes that happen in a single moment in time. On the other hand, they apply a passive strategy (continuous update of the ML model without explicit knowledge of concept drift (Losing et al., 2017)) since, in their case, several concept drifts happen daily. Another major difference is the way to deal with an imbalanced dataset. While FeDAL uses an ensemble, their work uses parameter tuning of a Dense Neural Network model. In this case, the ensemble's models are independently trained, and the final output is the average of the probability scores.

Z. Li et al. (2020) present an incremental learning approach that emphasizes misclassified instances in the update procedure of the ensemble's models. Another interesting characteristic of their approach is that it keeps a limited number of classifiers in the ensemble, ensuring efficiency. Like FeDAL, they use an ensemble to handle data imbalance without needing to access past data. Moreover, they employ SMOTE as the oversampling technique (Chawla et al., 2002), while we oversample by replication. Section 3.6 presents ablation studies comparing both techniques. In our context, these techniques have similar performance values before introducing concept drift. However, after introducing concept drift, the replication strategy outperforms SMOTE. Specifically, our results indicate that oversampling by replication is the most suitable strategy when handling online communities with class overlap, feedback noise (re-label data), and limited labeled datasets. One significant difference 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.

Different from FeDAL, which builds ensembles, Idrees et al. (2020) focus on discovering the best type of classifier over time in a domain with concept drift. This approach is particularly interesting when dealing with purely online learning since it is harder to know which base classifiers would be better for an ensemble from the beginning of the training procedures (because we have few data points). Thus, as new data becomes available, finding the best base classifiers can leverage the solution. If the best type of classifier changes due to concept drift, online ensemble learning should also be able to automatically identify which classifiers are best for the new situation. By using a heterogeneous framework, the authors aim to combine different classifiers to improve performance.

H. Zhang et al. (2019) present an ensemble framework to handle concept drift in an imbalanced dataset context, the Resample-based Ensemble Framework for Drifting Imbalance Stream (RE-DI). Their approach uses a resampling buffer to keep instances of the minority class to handle the class distribution over time. Also, ensemble members that perform worst in the minority class receive smaller weights. RE-DI maintains a long-term static classifier to handle gradual change and a set of dynamic classifiers to handle sudden concept drift, focusing only on recently received data. In the case of the dynamic classifiers, their weights incrementally decrease as time goes by while they are dynamically created and replaced. The goal is that the classifiers learn the latest concepts by the end of the training. Because the dynamic classifiers only exist for a period, RE-DI creates them using a block-based method, which has more initialization power and can be updated online. Unlike our frameworks, where we use oversampling to emphasize the minority class and undersampling to decrease the influence of the majority class in the training procedure, RE-DI uses a buffer (making use of past data), in which the last added data is used more than the older ones.

Addressing the same challenges as our work regarding imbalanced datasets and online learning, H. Du et al. (2021) introduce a cost-sensitive online ensemble learning algorithm. To build this ap-

proach, the authors use different equalization methods, from initially constructing base classifiers to calculating the weights of these initial classifiers. To evaluate their approach, the authors investigate the intrusion detection use case. Their approach shows better performance, including improvements in reducing the missing alarm rate and the false alarm rate. Unlike FeDAL, which uses all classifiers in an ensemble to make the final decision (voting scheme), they propose a weighted voting scheme, where only a subset of classifiers, selected based on specific criteria, contribute to the final decision. Their methodology relies on bagging ensemble learning and employs decision trees as the base classifiers, which differs from FeDAL's use of neural networks. Additionally, while they calculate classifier weights based on different misclassification costs, FeDAL focuses on presenting distinct data points to the classifiers, adopting a window approach to determine which data points are processed by each classifier (q.v. Section 3.2).

Abbasi et al. (2021) propose the ElStream approach. ElStream aims to build an ensemble of classifiers to detect concept drift. Their approach employs the majority voting scheme, allowing only the best classifiers to cast a vote. ElStream creates the ranking based on the confidence level. Unlike our multi-scenario approach, they focus on big data streams. Furthermore, they execute experiments in real and simulated datasets, emphasizing fake news detection. Specifically, their work investigates three ensemble classifiers: Decision Tree, Random Forest, and Extra Tree. In contrast to obtaining the set of features from community members,[4] ElStream uses feature extraction to identify the most relevant features related to fake news detection, extracting these features directly from text content. Lastly, unlike our highly imbalanced datasets, their datasets do not present this characteristic, so they are unconcerned with this issue.

Abedin et al. (2022) address the challenge of detecting credit risk in small businesses. Their methodology combines weighted SMOTE and ensemble learning, adding weights to SMOTE's new instances. This approach is similar to FeDAL's strategy to handle imbalanced datasets, as they also create balanced datasets from the original set of instances. Their findings include the beneficial impact of data-sampling techniques on classification performance. Specifically, results demonstrate that SMOTE, particularly combined with weighting, outperforms non-sampling techniques. Like FeDAL's results (q.v. Section 3.6), their findings also indicate that ensemble and data-level solutions yield the highest performance metrics, while cost-sensitive classifiers exhibit the lowest performance. In this context, weighted SMOTE and Random Forest present the best combination. However, unlike our Neural Network approach, which allows for incremental updates using stochastic gradient descent, their base classifier is a Decision Tree.

7.3 Cross-Community Learning

In this thesis, we aim to work with low-resource (or newly created) communities (q.v. Section 1.1.3), where low-resource implies working with a limited labeled set of norm-violating actions (Huang et al., 2022). To tackle this challenge, we adopt cross-community learning, which incorporates data from different sources (communities) to improve the performance of ML models in a new target community. This approach also includes solutions that employ transformer-based models and adapters (q.v. Section 2.4). Concretely, CAL is the framework that adopts this strategy, which this section compares with existing solutions to emphasize our contributions.

Chandrasekharan et al. (2019) present Crossmod, which uses cross-community learning through an ensemble of classifiers to assist moderators in detecting violations within different Reddit communities. Crossmod enables moderators to oversee the decision-making process of ML models and to deal with the scarcity of labeled data. Unlike CAL, they do not focus on understanding changes in

[4]Similar to the advantages when compared with these works (Cheriyan et al., 2017, 2021) in Section 7.1, using the set of features provided by the community allows the incorporation of the community view to include contextual information in a domain-specific manner (creating a detailed set of features to formalize the data).

the view of communities and on incorporating adapters to handle new violation classes dynamically (q.v. Section 5.1).

Like with CAL, Subramanian et al. (2022) use adapters to facilitate creating a solution to identify offensive comments on YouTube. Specifically, they consider low-resource languages, characterized by a scarcity of labeled data and language models (Ishmam & Sharmin, 2019; A. Sharma et al., 2022). Their results indicate that adapter-based fine-tuning is more effective than full fine-tuning PLMs while updating fewer parameters. Lastly, our approaches diverge in that we focus on understanding changes in community views (within a single community or across domains, q.v. Section 5.3) regarding norm-violating behavior.

Pamungkas and Patti (2019) propose an approach for detecting abusive language that combines DL techniques with a multilingual lexicon. Like CAL, their solution addresses the detection challenge from a cross-domain perspective. In particular, they incorporate a domain-independent multilingual lexicon of abusive words. It is worth noting that their work explores the availability of word embeddings across diverse languages to leverage cross-lingual learning, conducting experiments involving four languages. In contrast to CAL, which entails fine-tuning transformer-based models with target data without employing a lexicon, their experiments for cross-domain classification rely on training using only source data and testing directly on target data. By adopting this approach, they focus on evaluating the abusive lexicon's impact on the task, where results indicate better performance values. Furthermore, CAL differs from their approach in tackling the emergence of new violation classes. Lastly, similar to our findings that indicate abusive terms change from one community to another (q.v. Section 5.3), their results also show that the nature of swearing may vary from one domain to another.

Glavas et al. (2020) introduce an approach to address the challenges of cross-domain and cross-lingual learning. The authors conduct experiments considering three domains. Furthermore, they involve translation from English to five other languages. It is worth noting that they also investigate PLMs, including RoBERTa (q.v. Section 2.4). One interesting finding relevant to our research is that data augmentation through the combination of training instances from different domains can lead to detrimental performance, especially when dealing with abusive language domains that exhibit significant differences. The authors tackle a binary classification task, directly fine-tuning the PLMs and focusing on multiple languages. Consequently, their work differs from LaMAL and CAL, which detect multi-label violations and can use more than one source dataset to fine-tune adapters (depending on the violation class, as described in Section 5.2), respectively.

Jin et al. (2022) evaluate the potential benefits of continually adapting PLMs to emerging data, focusing on its application for two dataset configurations: 1) one derived from diverse domains, including different topic areas, such as Computer Science and Biology, along with data streams from Twitter, which spans several years; and 2) one derived from a single domain but including changes over time, similar to our use case of evolving community members' views on norm violations. However, unlike CAL, they do not investigate adapters' performance on classification tasks but rather on relation extraction[5] and named entity recognition.[6] Specifically, in the case of the Twitter community, the task revolves around predicting hashtags and emojis. Relevant to our work, their results show that adapters consistently achieve competitive performance across different tasks, comparing adapter-based methods against alternative techniques, including sequential pertaining, memory replay, and distillation-based approaches.

Q. Lu et al. (2021) focus on the biomedical case to introduce an architecture that employs adapters, independently training them on diverse sources. Their goal is to explore domains that benefit from knowledge derived from a set of source tasks. It is important to note that their framework differs

[5]Relation extraction refers to the task of identifying the association that exists between set of entities (e.g., people, objects, concepts) (Nasar et al., 2021).

[6]Named entity recognition refers to the task of identifying references to entities in text, specifically information like names of people and countries (Nadeau & Sekine, 2007).

from our CAL approach, as CAL's independent adapters are directly associated with specific violation classes, while they propose an adapter architecture that relates to the diverse data sources. Furthermore, CAL allows source communities to provide data for various violation classes, highlighting their role in training distinct and independent adapters. Their findings demonstrate the advantage of adopting adapters to improve performance. Like CAL, the trained adapters in their context can be deployed in different applications, exploring previously obtained knowledge from a source task to a novel target task. Consequently, this allows for an easy extension to new source data or even replacing existing sources. To conclude, they show that adapters are useful not only in a violation detection context but also in different tasks, including question answering, natural language inference, and named entity recognition.

While Le et al. (2021) use adapters for multi-lingual speech translation, Barbieri et al. (2022) present an approach that uses Twitter data from a multi-lingual setting for sentiment analysis. Both approaches use data in different source languages to leverage learning in a context with big datasets (the first with hundreds of hours of speech and the second with around 200M Twitter entries). This contrasts with our solution, which focuses on small datasets and low-resource communities (q.v. Section 1.1.3). Their results show that adapters can improve performance in a target task, even for cases considering distinct source languages.

Two interesting works (Malik et al., 2023; R. Zhang et al., 2021) employ adapters to address unsupervised domain adaptation. This task refers to the process of learning from a source domain (or community) and applying it effectively to a novel, *unlabeled* target domain (or community) (T. Xu et al., 2021). First, like CAL, R. Zhang et al. (2021) propose an approach requiring fine-tuning adapters that adopt the bottle-neck architecture. Particularly, they use this architecture to achieve domain adaptation. However, unlike CAL, which relies on a one-step procedure for adapter training (q.v. Algorithm 5.1), they follow a two-step approach. The initial step requires training on a dataset combining source and target data to reduce differences between the source and target domains. Subsequently, the second step requires fine-tuning using the source data while testing the model's performance on the target data. This approach presents an important difference from CAL since our main assumption is that CAL does not have access to the target data at the beginning of the training process (q.v. 5.2).

Second, Malik et al. (2023) use an approach that stacks two adapters, the first trained to mitigate domain divergence and the second specific for the task with labeled data. They also present an alternative approach, which does not stack adapters but simultaneously mitigates domain divergence and performance loss. Notably, they differ from R. Zhang et al. (2021) in that they do not use data from the target during the training process. Furthermore, they differ from CAL in that we focus on the incremental adaptation of the adapter based on new feedback from community members (or a new community).

In addition to the bottle-neck architecture (q.v. Section 2.4), different approaches exist for adapters. Pfeiffer et al. (2020) present AdapterFusion, which involves learning from multiple source tasks and combining their representations via a fusion layer. This approach aims to combine multiple adapters to solve a single target task. In the future, we shall explore this architecture to handle multiple cross-community definitions for the same violation class. Other approaches aim to optimize the parameter efficiency of adapters. First, S. He et al. (2022) introduce a pruning-based approach that reduces the number of trainable parameters. Second, Cai et al. (2022) focus on training fewer and smaller adapters at the top layers.

Lastly, besides NLP tasks, cross-community learning with adapters has been explored for computer vision. Huang et al. (2022) investigate the face anti-spoofing task, training adapters on diverse data sources and evaluating their ability to detect unseen instances in a new target task.

7.4 Interpretability

Following the training of ML models, we are interested in providing community members with an explanation of which features or words of their actions have contributed most to detecting norm violations. As such, in our specific domain, interpretability aims to provide information that assists community members in evaluating a model's decision on whether an article edit is correctly classified. It is worth noting that the utility of interpretability extends beyond our use case and is explored to solve different tasks (e.g., highlighting depression marks in text, evaluating adversarial attack situations, and identifying words related to political polarization).

Sarzynska-Wawer et al. (2021) apply the Local Interpretable Model-Agnostic Explanations (LIME) tool to the psychiatry use case. The authors use LIME to explain the predictions of a neural network, specifically the Embeddings from Language Models (ELMo). The main idea is to detect schizophrenia symptoms. Unlike our work with FeDAL that applies LIME to obtain the relevant features, they use the text data directly. Thus, in their case, LIME does not output feature scores. Instead, it outputs the words (parts of the texts) most relevant for the classification. One interesting point about healthcare domains is that interpretable models play an important role since healthcare professionals can make assertive decisions when assisted by automatic methods (ElShawi et al., 2021).

Novikova and Shkaruta (2022) use BERT to detect depression marks in text. In this scenario, 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. Also considering the healthcare case, ElShawi et al. (2021) focus on quantifying the quality of interpretability tools. Their work compares different tools, and the result indicates that no technique outperforms the others, even when considering different metrics.

Focused on maintaining a safe online space, Mahajan et al. (2021) compare different models to automatically moderate content, especially those that express online abuse. To do that, their work explores three datasets depicting this behavior (one is from the comments page on Wikipedia). The authors also treat comments as a series of features created employing TF-IDF and word embeddings (GloVE and FastText). Like our work, they conclude that LIME can help in the explanation process, providing additional information to evaluate an ML model's performance. Our approaches differ in the training strategy. Specifically, while we focus on comparing incremental learning approaches (q.v. Section 3.6), their work compares different ML models within the same training strategy.

LIME can also be applied to explain the results of DL models (Aluru et al., 2020). They focus on hate speech detection in a multi-language domain with different data sources. The authors build embeddings for the text being classified. Subsequently, the words in this text receive the LIME score, indicating how they contribute to the model's output. Although it differs from our LIME approach with FeDAL (q.v. Section 3.5), which considers the changing aspects of a single data source (Wikipedia article edits) over a certain period, their work is similar to our IG approach with LaMAL (q.v. Section 4.3) and CAL (q.v. Section 5.2) for text-related tasks. The main difference is that LaMAL and CAL use IG, a model-specific approach that allows extracting rules from PLMs, while LIME follows a model-agnostic architecture, which allows the creation of explanations for different types of ML models (e.g., an ensemble of neural networks, random forests, support vector machines) with no need to explore the inner workings of these models. Furthermore, they use the LIME scores for the embeddings, showing differences depending on the model. Specifically, they demonstrate that while one model focuses on hateful keywords, another focuses on the whole context in which the keyword is present.

In their work, Xiang et al. (2021) introduce an approach to enhance the interpretability of PLMs. Unlike IG with LaMAL and CAL, 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, they conduct a user experiment, discovering that the explanations generated by their method outperform those produced by inherently interpretable models (e.g., Logistic Regression). Compared to our user study (q.v. Chapter 6), the difference is that we aim to directly understand how participants' views about norm-violating behavior change due to the interpretability tool. Lastly, future work shall evaluate the differences between IG and their proposal, focusing on analyzing how the relevant terms for norm violation differ depending on the interpretability algorithm.

The relationship between interpretability and PLMs can also benefit low-resource languages, characterized by a scarcity of labeled data and language models (Ishmam & Sharmin, 2019; A. Sharma et al., 2022). Karim et al., 2021 present a study that applies an interpretability approach to investigating 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 technique to obtain interpretations when hate speech is detected.

Sabry et al. (2022) contribute to exploring the understanding of hate speech by investigating five tasks. Their approach employs IG to tackle interpretability. However, unlike our work, they aim to identify why their Bidirectional Long Short-Term Memory model made mistakes, especially considering instances where errors are prevalent. On a different domain, Sáenz and Becker (2021) investigate interpretability to obtain the relevant words associated with political polarization in the context of anti/pro-vaccination concerning COVID-19. Specifically, the authors focus on attention weight for word relevance, which differs from our work with IG. Furthermore, while they focus on identifying the relevant words for different classes (similar to LaMAL), CAL's analysis focuses on describing the differences in relevance between communities.

Some researchers also use interpretability to simulate and evaluate how ML models behave in an adversarial attack situation (Alsmadi et al., 2021; Y. Li et al., 2022; P. Yang et al., 2020), 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. (2021) 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 norm-violating behavior from different communities and understanding how the definition of what constitutes a violation evolves.

7.5 User Study

This thesis proposes an approach incorporating interpretability tools to present which action elements, such as a set of features or words of a text sentence, contribute to norm violation detection. Considering the tools employed by our multi-scenario approach, it is possible to present interpretability information using three different layouts: local interpretability, a list with the sum of relevance scores, and a combination of both. Since these layouts provide different information, it is worth conducting a user study to evaluate whether any of these layouts can influence people's views when evaluating sentences containing hate speech (q.v. Chapter 6), aiming to provide empirical evidence regarding the effective use of ML-model interpretability and to investigate how these layouts can help mitigate norm-violating behavior online. Thus, this section presents works investigating interpretability tools and how people interact with them.

The survey in (Rong et al., 2022) offers important guidelines for user studies in Explainable Artificial Intelligence (XAI) contexts, exploring interesting works in this domain. Their discussion regarding the effectiveness of explanations in increasing participant trust and usability of ML models is particularly relevant to our research. However, it is important to note that existing literature presents studies with different results. Consequently, we emphasize the importance of our work in

providing additional empirical evidence in the field of assessing the impact of interpretability. Furthermore, we adopt an approach that measures the participants' perception ratings using a Likert scale (Mohseni et al., 2021), directly inquiring participants about their views when evaluating a text sentence classified as a hate speech violation.

Schuff et al. (2022) investigate how people understand and interpret explanations provided by the word salience in text sentences. Their methodology shares similarities with the information in Figures 5.2 and 5.3, where salience indicates the terms that exhibit the most influence on the model's decision-making process. Specifically, they execute a study to evaluate the impact of different factors (e.g., word frequency, word length, display index) on a participant's interpretation of the explanation. Like our study, they use crowdsourcing workers and employ a GAM-based approach to analyze the data. However, certain differences are important to note. First, they compare bar charts to heatmap-based salience visualizations, while we compare salience (local interpretability), list, and a combination of both. Second, their study focuses on textual data elements as factors in the model, while we incorporate demographic variables as influencing factors. Lastly, instead of asking about the participants' views regarding the model's classification, they inquire how relevant a specific word is to the model's output (also on a Likert scale).

Arora et al. (2022) consider the domain of hotel reviews, where the trained model must differentiate between authentic and fake reviews. Their user study asks participants to simulate the model's behavior on new reviews after having access to explanations. The goal is to assess whether interpretability information helps humans predict the model output on unseen instances. Additionally, the participants are asked to engage in a manipulation task by editing the review to change the classification output. This step evaluates if participants can lower the model confidence towards the original predicted class (i.e., lower the classification probability). Like our findings demonstrating that interpretability information does not influence participants' views, their results indicate that local interpretability fails to improve participants' capacity to replicate the model's decision-making process. However, the authors conclude that participants can affect the model's confidence by leveraging information from the global attributions tool, built using a linear model that emulates a PLM (specifically, a BERT model). These general explanations provide insights into common input-output associations that the models exploit. Lastly, they employ IG as the interpretability algorithm and use mixed effects models to analyze participants' interactions.

Unlike our work that evaluates interpretability in the scope of textual data, Chu et al. (2020) address saliency-based interpretability techniques applied to image data, specifically in the context of age prediction tasks. Their study measures whether effective model explanations enhance human accuracy (i.e., predict age better) while quantifying whether flawed explanations decrease human trust (i.e., quantify the difference between people's answers and the model's output). The findings indicate that different types of explanations do not significantly influence human accuracy or trust, which is similar to our work's conclusion, where different interpretability tools fail to impact participants' perception ratings (q.v. Section 6.2). They use a mixed-effects model to estimate error, measuring the difference between participants' guesses about a person's age and the true label. Additionally, they estimate trust, a measurement of the difference between the participants' guesses and the model's prediction. Also working with image data, Alqaraawi et al. (2020) find that the saliency-based approach helps participants comprehend specific image features (i.e., the relevant pixels of an image) that contribute to the output of a ML model solving an image classification tasks, such as identifying objects. However, it is worth noting that their results indicate that this approach is limited in assisting participants in anticipating the model's predictions for new images.

To assess the factors contributing to the interpretability of explainability information, Lage et al. (2019) investigate three tasks people can perform when engaging with an ML system. The tasks comprise simulating system responses, verifying suggested responses, and evaluating whether the correctness of a suggested response changes when the input data changes. In summary, they aim to present the types of explanations participants could effectively employ for these tasks. The scope of their work involves decision sets in three ways that can affect how easy it is for humans

to interpret them: explanation length, number and presentation of newly introduced concepts, and the number of repeated terms (i.e., input conditions). Their results indicate that the nature of complexity matters, with newly introduced concepts having more impact on task performance than the inclusion of repeated terms in the explanation. The performance metrics employed for evaluation are accuracy, response time, and subjective satisfaction, with participants providing ratings on a Likert scale.

Similar to our combined layout (q.v. Section 6.1), Radensky et al. (2022) conduct a user study to assess the effectiveness of combining both local and global information in comparison to relying solely on a single interpretability layout. Their findings indicate that the combination is better at assisting participants to comprehend how a recommender system can improve. However, it is worth noting that this result differs from our findings (q.v. Section 6.2), as in our domain, the combined layout does not present a higher impact outcome compared to other layouts, with no significant influence on participants' perception concerning hate speech ratings. Furthermore, they focus on improving recommendations while employing a Likert scale for collecting participant responses.

X. Wang and Yin (2021) conduct an empirical investigation that compares different XAI methods in AI-assisted decision-making. Their study describes three essential properties that should be present in AI explanations to make them helpful from a human perspective: model understanding, uncertainty recognition, and trust calibration. Like our study, they also use a crowdsourcing platform for the user experiment. Their findings indicate that the nature of the decision-making task significantly impacts the efficacy of the employed XAI techniques. However, the results also show that when participants lack domain knowledge, the XAI methods do not satisfy the investigated properties. Consequently, they recommend building XAI methods specific to these cases, incorporating alternative techniques to provide interpretability information. It is worth noting that this recommendation differs from our study in terms of the specific task, as we focus on evaluating two classes of hate speech, a domain where participants generally have some prior knowledge (q.v. Section 6.2).

Lastly, regarding user studies carried out on crowdsourcing platforms (like ours), Shank (2016) present concepts, patterns, and suggestions for exploring these platforms in online research. Their review showcases several papers where researchers successfully acquire reliable and quality data in diverse contexts such as psychology and economics. Additionally, one essential feature of these platforms relevant to our study is their ability to easily target specific demographic characteristics within a particular population. In our use case, we aim to target participants based on ethnicity and gender (q.v. Section 6.1).

7.6 Summary

This chapter reviewed works related to the challenges addressed in this thesis. Our first challenge was to detect norm violations in online communities, focusing on the Wikipedia article editing use case. The literature review began by investigating norm violation on diverse online platforms, examining existing ML solutions and datasets used in these domains. Then, we focused on a complex issue in this context: identifying norm-violating behavior in environments characterized by concept drift and imbalanced datasets. Specifically, we examined solutions that employed ensemble and incremental learning techniques. Additionally, this chapter reviewed research on cross-community learning, incorporated into our solution to leverage data from diverse sources to enhance a model's performance in low-resource (or newly created) online communities. Moving beyond violation detection, the literature review investigated works emphasizing interpretability to provide insights into the features or words that affect norm violation detection by a ML model, describing solutions that employ interpretability tools such as LIME and Integrated Gradients (IG), both of which used by our solution. Lastly, we closed the literature review by presenting works exploring how humans interact with ML models using interpretability information. This is

relevant to our thesis since we conducted a user study to provide empirical evidence on whether ML model interpretability influences participants' views when evaluating sentences containing hate speech.

In the remainder of this section, we present a concise analysis of the identified gaps in the literature, outlining each topic investigated in this chapter. We describe how our proposed multi-scenario approach, comprising FeDAL, LaMAL, and Cross-Community Adapter Learning (CAL), addresses these gaps. Concretely, our approach is designed to continuously learn from the interactions and agents' feedback on what constitutes norm-violating behavior in online communities.

Section 7.1 presented significant gaps in existing research regarding norm violation detection in online communities. First, beyond our use case of the Wikipedia community, also explored in previous work, the literature examined various online platforms, providing different solutions to detect violations in this context (e.g., DL classifiers, Logistic Model Tree, transformer-based models). However, they lacked the mechanisms to cope with concept drift, did not incrementally learn as interactions unfolded, and did not incorporate community feedback for ML model updates. In contrast, our approach tackled these gaps by employing ensemble and incremental learning techniques, facilitating continuous adaptation to concept drift. Moreover, it incorporated community feedback in the learning process, allowing model updating based on evolving community members' views. Second, regarding the data formats (e.g., set of features (tabular data), text sentences) used to train ML models, existing research demonstrated a tendency to address either textual or tabular datasets individually, with some attempts to manage both simultaneously. In contrast, our approach adopted a strategy to create different frameworks that separately handled text data (LaMAL and CAL) and the formalization of actions as a set of features (FeDAL). This strategy was relevant in our work because it allowed us to eliminate the need for a featurization step[7] when handling data collected as text sentences while still enabling deploying this solution in a multi-scenario context.

While certain studies adopted different ML classification algorithms (e.g., Decision Tree, Random Forests) to detect norm violations, our approach adopted neural networks and transformer-based models, enabling continuous model updates. In this context, we were interested in research employing ensemble and incremental learning techniques to learn with imbalanced datasets (q.v. Section 7.2). The existing literature presented various solutions that enhanced these methods by using parameter tuning, SMOTE, and resampling buffers. In contrast, our approach used oversampling by replication to leverage ensemble and incremental learning. Our solution performed better, especially in scenarios characterized by class overlap, feedback noise, and limited labeled datasets (q.v. Section 3.6.3), without the need for resampling buffers. Thus, we avoided dependence on past data (i.e., we used only the latest data provided by a community) to incorporate changes in communities' views on what constitutes a violation. Additionally, different from some works that employed a passive strategy to detect concept drift, we employed an active strategy, which is relevant for our use case to deal with major changes in a single moment by monitoring changes in class distribution.

Since this thesis focused on low-resource (or newly created) online communities, this chapter investigated the literature with solutions that incorporated data from different source communities to enhance the performance of a ML model in a new community with limited labeled data (q.v. Section 7.3). We identified a relevant gap in the literature: these solutions did not address changes in community views by using adapters to identify specific norm violations and dynamically creating them to handle the emergence of new violation classes. Thus, we created the CAL framework to fill this gap. Additionally, existing works emphasized the contextual relevance of community-specific data in training ML models to identify norm violations. Our solution adopted this idea and conducted experiments in a fine-tuning configuration with the new community data (q.v. Section 5.2). Lastly, a major contribution to the literature introduced by CAL was incorporating an

[7]The featurization step refers to creating a set of features to formalize an action in an online community.

interpretability component to understand the differences in community members' views across domains, especially after initially training with source data and fine-tuning with the new community data.

Different works presented the advantages of using interpretability to understand the output of ML models (q.v. Section 7.4). The literature investigated solutions that employed LIME to explain the predictions of a neural network within the psychiatry context and to discern signs of depression. However, unlike our solution with LIME, they did not focus on obtaining a set of features relevant for norm violation detection but rather on obtaining the words' relevance scores. Other domains (e.g., political polarization, content moderation, addressing adversarial attacks) also benefited from interpretability to assist humans. However, these works did not incorporate interpretability to analyze changes in ML models when training using incremental learning approaches with data from one or more online communities. This was especially relevant in our context since these changes in the model reflect evolving community members' views, allowing us to analyze how detrimental behavior changed over time and across domains. Additionally, while the majority of approaches explored in the literature review tended to employ either model-agnostic (LIME) or model-specific (IG) interoperability tools, our multi-scenario approach employed both. This was important because it allowed us to obtain the relevant set of features or the words that contribute to norm violation detection by a ML model (e.g., an ensemble of neural networks, transformer-based models).

The main challenge addressed by our user study, as discussed in Section 7.5, was providing statistical and qualitative analyses of whether three different interpretability layouts could impact participants' views on hate speech sentences. This study was important because existing literature presented divergent conclusions on the impact of interpretability of ML models. For instance, some works suggested that the manner in which interpretability information was presented could influence people in different activities (e.g., understanding recommender systems, lowering a ML model classification probability). In contrast, other works indicated that interpretability information did not significantly affect how people executed tasks (e.g., simulate a model's behavior on new reviews, predict age). Regarding our user study, the findings indicated that no interpretability layout significantly impacted participants' views on hate speech sentences. Our experiment aligned with established guidelines for conducting user studies in Explainable Artificial Intelligence contexts and contributed additional empirical evidence with insights into evaluating ML interpretability. Moreover, while previous works focused on textual elements (e.g., word frequency, word length) as factors that could impact participants' interpretation of an explanation, our study focused on incorporating demographic variables (e.g., gender, ethnicity) as factors that could impact participants' perception ratings of a text sentence since they evaluated hate speech sentences within these topics (q.v. Section 6.1).[8] Lastly, existing works using crowdsourcing platforms presented how researchers successfully obtained reliable and high-quality data across diverse domains, including psychology and economics.

Finally, in the next chapter, we present the conclusions of our research, including directions for future work.

[8]Including these demographic variables in our statistical analysis allowed us to evaluate their potential influence on hate speech classification. We aimed to isolate the variable of interest (interpretability layout) while considering confounding variables (gender and ethnicity), ensuring accurate conclusions.

Chapter 8

Conclusions and Future Work

This thesis addressed dynamic online communities, where agents (in this work referred to as community members) with diverse backgrounds interact. As interactions unfold, these communities continuously establish relevant norms to guide their interactions and define what behavior constitutes a violation, which may evolve over time or across communities. This process involves incorporating feedback from the interacting agents, reflecting changes in community members' views on norm-violating behavior. For instance, previously accepted behavior may become frowned upon, or vice versa (such as the shift in a community's perspective to start allowing previously prohibited terminology). In this context, a normative system that can adapt to the changing definitions of norm violations is crucial for regulating community behavior. Thus, we proposed a Machine Learning (ML) approach that supports normative systems in continuous learning from the interactions and agents' feedback. Our proposed learning process uses examples of actions labeled as violations in a multi-scenario context, effectively handling agent actions defined either as a set of features (tabular data) or as text sentences.

We illustrated learning what constitutes a norm violation by investigating the Wikipedia article editing use case (q.v. Section 1.1.3). Actions evaluated in this context were community members' attempts to edit articles, and we specifically focused on the "no vandalism" norm. This norm included the requirement to use proper writing style, refrain from removing content, avoid editing wars, and not engage in hate speech. Wikipedia used MTurk to identify and label edits as regular or violating behavior. Building upon this existing annotation, we extended it further by having the author of this thesis provide additional annotations for each violation instance containing offensive language, categorizing these into six different hate speech classes: Swear, Insult and Ableism, Sexual Harassment, Racism, LGBTQIA+ Attack, and Misogyny. Notably, a single article edit may exhibit multiple violation classes simultaneously, demonstrating the need for our multi-scenario approach to handle binary and multi-label classification tasks.

Moreover, we investigated the emergence of violation classes in low-resource or newly created online communities.[1] In these domains, the learning process occurs in a context with limited labeled data. Thus, we created a cross-community learning solution capable of using data from different sources (other communities) to improve the performance of ML models in a new target community, incorporating diverse views from community members across domains. When learning from other communities, we used data from three sources other than Wikipedia to help address emerging violation classes. This is particularly important because some violation classes had only a few labeled instances in our Wikipedia article editing dataset (q.v. Section 5.2). By learning using data from different communities, we demonstrated that normative systems equipped with our proposed ML models can adapt to changing views within a single community and across diverse

[1] Although Wikipedia is known for its extensive content repository, the labeled dataset used in this thesis has a limited size, comprising approximately 30.000 article edits labeled either as violation or not.

domains. These changing views may reflect the emergence of new violation classes or shifts in the community view (q.v. Section 5.3). In summary, our cross-community learning solution addresses the following challenges: limited *labeled* data, varied data types (text and tabular), concept drift (changing community views), and limited computational power.

To create our multi-scenario approach, we integrated incremental learning in three frameworks: FeDAL (q.v. Chapter 3), LaMAL (q.v. Chapter 4), and CAL (q.v. Chapter 5). These frameworks addressed the challenge of continuously learning what constitutes norm violation as interactions unfold using different data types (tabular and text). Furthermore, the frameworks incorporated interpretability to provide evidence on which elements of an action (words of a sentence or set of features) contributed to detecting violating behavior. This ability aligns with the principles of responsible artificial intelligence, promoting transparency and facilitating community members' comprehension of what constitutes a violation. It also supports model debugging, enables analyzing how community views change over time or across communities, and creates the conditions for triggering collaborative feedback elicitation when community members agree there are discrepancies between the model's output and their view on norm-violating behavior.

We executed different experiments for the three frameworks. First, the Feedback-Driven Adaptive Learning (FeDAL) framework is an ensemble of Feed-Forward Neural Networks (FNNs) equipped with an incremental learning approach designed to handle tabular data with class distribution imbalance (q.v. Section 2.1). FeDAL employed a replication by oversampling to limit ensemble size, enhancing computational efficiency. Moreover, it implemented incremental learning using Stochastic Gradient Descent (SGD) to update a model's parameters as new interactions occur (q.v. Section 2.3.2). In practice, FeDAL handled the actions of an online community formalized as a set of features (X, y), allowing community members to understand the model's decisions through interpretability by employing the Local Interpretable Model-Agnostic Explanations (LIME) algorithm (q.v. Section 2.5.1). The experiments in Section 3.5 described the evaluation of FeDAL's performance in the Wikipedia use case. Our findings demonstrated FeDAL's effectiveness in detecting norm violations with imbalanced data and concept drift (q.v. Section 3.6). Additionally, we provided a comprehensive interpretability analysis of the differences between the learning techniques. Lastly, our ablation studies demonstrated the advantages of an ensemble approach (q.v. Section 3.6.3) compared to single models and of oversampling by replication compared to SMOTE (Chawla et al., 2002).

Second, the Language Model Adaptive Learning (LaMAL) framework employed Pre-Trained Language Models (PLMs) using an incremental learning approach (q.v. Section 2.4). Unlike FeDAL, LaMAL leveraged PLMs directly for handling actions specified as text sentences, reducing the need for an ensemble of classifiers. PLMs handle imbalanced datasets and enable superior performance by fine-tuning the classification head for specific use cases (such as violation detection in Wikipedia article editing). LaMAL efficiently tackled binary and multi-label text classification, identifying specific violation classes since it directly handled text input. We used Integrated Gradients (IG) to explain a PLM's behavior (q.v. Section 2.5.2), providing insights on which words contributed to norm violation detection. Experiments in Section 4.3 described the comparison between two PLMs, DistilBERT and RoBERTa, in identifying violations in Wikipedia article edits. Our results presented how the architecture of each PLM impacts the understanding of violating behavior, with different relevant words obtained from DistilBERT and RoBERTa (q.v. Section 4.4).

Third, the Cross-Community Adapter Learning (CAL) framework integrated PLMs and adapters using an incremental learning framework (q.v. Section 2.4). Unlike FeDAL and LaMAL, CAL enhanced our multi-scenario approach by incorporating data from different communities to improve ML model performance in new target communities with limited labeled data. By integrating adapters, CAL presented the following advantages: 1) it allowed for an efficient fine-tuning process for specific violation classes, dynamically creating adapters for emerging classes as interactions unfold; 2) it addressed catastrophic forgetting by having individual adapters for different classes;[2]

[2]Here, we refer to the loss of knowledge acquired from previous classification tasks when learning new information

and 3) it employed the IG algorithm to understand distinct views on norm violation across communities, a unique contribution of this framework. Experiments in Section 5.2 described using DistilBERT with an adapter and data from three communities. Our results demonstrated how CAL continuously learns changing community views on norm violation through community feedback (labeled data) (q.v. Section 5.3). Moreover, it demonstrated CAL's ability to incorporate data from different sources to address emerging violation classes while explaining the different views on norm-violating behavior across communities.

To close the research of this thesis, we conducted a user study to assess the impact of interpretability on influencing user views when evaluating sentences containing hate speech (q.v. Chapter 6). Participants answered an online questionnaire where they had to choose whether they agreed or disagreed with classifying a sentence given a violation class (Misogyny or Racism) using a 7-point Likert scale (q.v. Section 6.1). Three interpretability layouts were considered: the local interpretability, the list of relevance scores, and the combination of both. Participants evaluated each sentence twice, once without interpretability data and once with it. The study involved within-subject and between-subject analyses, employing the Generalized Additive Model (GAM) to obtain the estimate of the participants' confidence ratings (q.v. Section 2.6). Our statistical analysis indicated that none of the interpretability layouts significantly influenced participants' views regarding the classification of hate speech (q.v. Section 6.2). Despite that, our qualitative analyses contributed valuable insights into the impact of incorporating ML interpretability in our frameworks. First, interpretability can enhance people's understanding of words relevant to the ML model's output, which triggers them to provide corrective feedback in cases of discrepancies. Second, interpretability layouts can provide insights into evaluating the ML model's behavior beyond traditional performance metrics (e.g., recall, F1-score, and precision), which is especially relevant in scenarios where understanding the rationale behind an ML model's decision is essential for optimal implementation.

Future Work

In the future, we aim to investigate different lines of research that can benefit from our multi-scenario approach. One such area of interest is the combination of norms and indirect reciprocity in the context of evolving game theory (Nowak & Sigmund, 2005; Okada, 2020). In this case, people's perceptions of each other's actions influence indirect reciprocity. For instance, individuals may have diverse views regarding assisting defectors. Consider a community where researchers support each other in the peer-review process. In this example, we could use norms to penalize researchers who fail to contribute to the review process (resulting in fewer available reviewers) while still benefiting from having their work reviewed by others. Here, helping non-cooperating members could be detrimental to overall cooperation, as it might lead to extended wait times for cooperating members seeking timely peer reviews, affecting their engagement in the community (i.e., providing new reviews). However, imagine the same norms adopted by an online community that aims to support the elderly. Cooperation in this community has a different meaning since the elderly who do not offer assistance cannot be punished, and those who help them should be promoted. Thus, we argue that our solution has the potential to learn the concept of cooperation from interaction data and support normative systems by adapting to each community's definitions.

In this thesis, we conducted experiments to evaluate learning techniques after introducing concept drift, particularly when class labels are swapped. We aimed to assess how FeDAL adapts after receiving feedback. One limitation is that our current approach adopted a simulation strategy since we did not have access to real feedback from community members. Thus, we plan to address this limitation in future work by getting real feedback. This is not only interesting because of feedback collection but also from the point of view of how community members can collaboratively

from a different class (Pfeiffer et al., 2020).

deliberate and agree on what constitutes norm violation. Moreover, regarding the swap strategy,[3] we plan to conduct experiments where not all data points have a swap of class labels when concept drift is introduced. This experiment design aims to evaluate how FeDAL addresses potential inconsistencies in the labeling process that may arise when people interact online.

Additionally, as we envision people in control of defining their community norms and expect them to collectively agree on those norms through deliberation, we aim to use the interpretability information generated by our solution to support such deliberation in the future. This interpretability data will offer insights into actions by providing the words in a text sentence or the set of features that contribute to the model's output. This information will support community members in reviewing actions to help them understand what feedback should be provided to the learning tools to correct their models. For instance, when a community member performs an action, and the ML model detects it as violating behavior, our solution will provide the member with the reasons for such detection. If the member disagrees with the output, they can engage with other community members to evaluate the action with information about the relevant features (tabular data) or words of a sentence (text data), assessing the correctness of the ML model accordingly. As our solution focuses on low-resource communities,[4] the member's original action will not be relabeled until the community reaches a consensus, and only then will the system allow the relabeling of that action. This process will facilitate proactive, collective, and collaborative feedback elicitation.

Regarding the research on different interpretability techniques, future work shall evaluate the differences between Integrated Gradients (IG) (q.v. Section 2.5.2) and the solution created by Xiang et al. (2021), focusing on analyzing how the relevant terms for norm violation differ depending on the interpretability algorithm. Additionally, we will explore data augmentation techniques to improve the interpretability capabilities of PLMs, aiming at using interpretability information from source domains to create new perturbed instances in the target training process.

Exploring other adapter architectures to improve training efficiency in low-resource communities and tasks is also an interesting direction for future research. Within this research line, we plan to inject additional factual knowledge from source communities into adapters to improve their generalization ability in the target domain. By incorporating factual information about allowed terminology (or mapping prohibited terms) and their contextual usage, adapters can perform better across cases not covered by the limited labeled data used for training. Moreover, we plan to extend our approach by simultaneously incorporating multiple additional source community datasets into CAL, allowing our models to access a broader range of information. Thus, instead of relying on a single source community to provide data to an emerging violation class (q.v. Section 5.2), we plan to use data from multiple communities. In this context, employing AdapterFusion (Pfeiffer et al., 2020) is an interesting direction, as it represents a transfer-learning approach that integrates data from different sources (communities) simultaneously to improve its performance in a target domain (new community of interest). Lastly, we can explore the connection between Reinforcement Learning (RL) and PLMs to improve the training procedure. We aim to guide the RL approach with information derived from PLMs. This includes providing contextual state representations for RL agents and continuously updating these representations as interactions unfold. Additionally, we plan to use interpretability data from PLMs to assist RL agents in focusing on the most relevant components of a state representation.

As our multi-scenario approach focused on learning the relevant elements of an action associated with norm violation and detecting such violations, we plan to integrate this solution into a line of research that focuses on the continuous revision of norms as agents interact. Existing research aims to revise norms based on previously obtained knowledge about whether an action is a violation or not (Campos et al., 2013; Dell'Anna et al., 2022; Morales et al., 2015). As such, their approach relies on having access to the classification of an action derived from an automated or manual

[3]The swap strategy was implemented to simulate feedback that represented changing community views.

[4]Here, we highlight our focus on low-resource communities since open large communities might benefit from different feedback processes.

process. In other words, their works can not identify when a specific action violates a norm and require labeled data with such information (which our multi-scenario approach can provide). Concretely, our solution will complement their approach by providing labeled data to serve as a basis for norm revision. With this collaborative approach, we aim to assist norm revision in scenarios that require handling different classes of norm violations that may occur simultaneously (multi-label classification tasks) while addressing multiple scenarios (tabular and text data).

Lastly, future work shall investigate other norm violations unrelated to hate speech. This aims to address the limitation of our user study in evaluating a task (classification of a text sentence as racist or misogynistic) already familiar to participants (q.v. Section 6.2.3). Our goal is to investigate tasks in different communities, such as the interaction of people during online meetings, where norms might regulate the meeting's duration, the volume of messages exchanged, and when participants should interact. In this context, interpretability information describing expected behavior within a particular online community may assist novice members in understanding and adhering to established norms. For instance, if the volume of message exchange exceeds established norms, interpretability can provide information regarding the expected volume in this community. Again, this information can contribute to a future feedback elicitation process, where new definitions of the elements (i.e., the new expected volume of message exchange) that comprise violation behavior in this domain may emerge. A new user study can help assess whether community members use interpretability information to adapt their behavior (without human interventions) in online meetings. We believe the investigation in this use case has the potential to yield different results compared to the hate speech use case, as participants may not have established opinions about what constitutes a violation in this context. Additionally, when they have such notions, expected behavior may vary greatly depending on the individual views. As such, our assumption is that participants will demonstrate a higher tendency to adapt to the definitions provided by the interpretability tool.

Bibliography

Abayomi-Alli, O. O., Damaševičius, R., Maskeliūnas, R., & Misra, S. (2022). An ensemble learning model for covid-19 detection from blood test samples. *Sensors*, *22*(6), 2224.

Abbasi, A., Javed, A. R., Chakraborty, C., Nebhen, J., Zehra, W., & Jalil, Z. (2021). Elstream: An ensemble learning approach for concept drift detection in dynamic social big data stream learning. *IEEE Access*, *9*, 66408–66419.

Abedin, M. Z., Guotai, C., Hajek, P., & Zhang, T. (2022). Combining weighted smote with ensemble learning for the class-imbalanced prediction of small business credit risk. *Complex & Intelligent Systems*, 1–21.

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. *Advanced Information Networking and Applications: Proceedings of the 34th International Conference on Advanced Information Networking and Applications (AINA-2020)*, 1341–1354.

Adler, B. T., 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*, 277–288.

Afroz, S., Brennan, M., & Greenstadt, R. (2012). Detecting hoaxes, frauds, and deception in writing style online. *2012 IEEE Symposium on Security and Privacy*, 461–475.

Agrawal, R., Ajmeri, N., & Singh, M. P. (2022). Socially intelligent genetic agents for the emergence of explicit norms. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 1–7.

Ahishakiye, E., Wario, R., Mwangi, W., & Taremwa, D. (2020). Prediction of cervical cancer basing on risk factors using ensemble learning. *2020 IST-Africa Conference*, 1–12.

Akhter, M. P., Jiangbin, Z., Naqvi, I. R., AbdelMajeed, M., & Zia, T. (2021). Abusive language detection from social media comments using conventional machine learning and deep learning approaches. *Multimedia Systems*, 1–16.

Alam, K. S., Bhowmik, S., & Prosun, P. R. K. (2021). Cyberbullying detection: An ensemble based machine learning approach. *2021 third international conference on intelligent communication technologies and virtual mobile networks (ICICV)*, 710–715.

Allison, K. R., Bussey, K., & Sweller, N. (2019). 'I'm going to hell for laughing at this': Norms, humour, and the neutralisation of aggression in online communities. *Proceedings of the ACM on Human-Computer Interaction*, *3*(CSCW), 1–25.

Alqaraawi, A., Schuessler, M., Weiß, P., Costanza, E., & Berthouze, N. (2020). Evaluating saliency map explanations for convolutional neural networks: A user study. *Proceedings of the 25th international conference on intelligent user interfaces*, 275–285.

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. *CoRR*, *abs/2110.13980*.

Aluru, S. S., Mathew, B., Saha, P., & Mukherjee, A. (2020). Deep learning models for multilingual hate speech detection. *CoRR*, *abs/2004.06465*.

Anand, M., & Eswari, R. (2019). Classification of Abusive Comments in Social Media using Deep Learning. *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 974–977.

Anowar, F., & Sadaoui, S. (2021). Incremental learning framework for real-world fraud detection environment. *Computational Intelligence*, *37*(1), 635–656.

Arora, S., Pruthi, D., Sadeh, N., Cohen, W. W., Lipton, Z. C., & Neubig, G. (2022). Explain, edit, and understand: Rethinking user study design for evaluating model explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *36*(5), 5277–5285.

Arora, S., & Patro, A. (2021). Inclusivity and empowerment—grow and let grow. *Global Business and Organizational Excellence*, *41*(1), 21–30.

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*, 82–115.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *CoRR*, *abs/1607.06450*.

Baayen, R. H., & Linke, M. (2020). An introduction to the generalized additive model. *A Practical Handbook of Corpus Linguistics*, 563–591.

Babajide Mustapha, I., & Saeed, F. (2016). Bioactive molecule prediction using extreme gradient boosting. *Molecules*, *21*(8), 983.

Banks, D. L., & Fienberg, S. E. (2003). Statistics, multivariate. In R. A. Meyers (Ed.), *Encyclopedia of physical science and technology (third edition)* (Third Edition, pp. 851–889). Academic Press.

Barbado, R., Araque, O., & Iglesias, C. A. (2019). A framework for fake review detection in online consumer electronics retailers. *Information Processing & Management*, *56*(4), 1234–1244.

Barbieri, F., Anke, L. E., & Camacho-Collados, J. (2022). Xlm-t: Multilingual language models in twitter for sentiment analysis and beyond. *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 258–266.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, *58*, 82–115.

Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, *54*, 1937–1967.

Biber, J. K., Doverspike, D., Baznik, D., Cober, A., & Ritter, B. A. (2002). Sexual harassment in online communications: Effects of gender and discourse medium. *CyberPsychology & Behavior*, *5*(1), 33–42.

Bogart, K. R., & Dunn, D. S. (2019). Ableism special issue introduction. *Journal of Social Issues*, *75*(3), 650–664.

Brysbaert, M., & Stevens, M. (2018). Power analysis and effect size in mixed effects models: A tutorial. *Journal of cognition*, *1*(1).

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.

Cahyana, N., Khomsah, S., & Aribowo, A. S. (2019). Improving imbalanced dataset classification using oversampling and gradient boosting. *2019 5th International Conference on Science in Information Technology (ICSITech)*, 217–222.

Cai, D., Wu, Y., Wang, S., Lin, F. X., & Xu, M. (2022). Autofednlp: An efficient fednlp framework. *CoRR*, *abs/2205.10162*.

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.

Center, P. R. (2020). *Amid national reckoning, americans divided on whether increased focus on race will lead to major policy change* (tech. rep.). Pew Research Center. Retrieved Decem-

ber 7, 2023, from https://www.pewresearch.org/social-trends/2020/10/06/amid-national-reckoning-americans-divided-on-whether-increased-focus-on-race-will-lead-to-major-policy-change/

Chandrasekharan, E., Gandhi, C., Mustelier, M. W., & Gilbert, E. (2019). Crossmod: A cross-community learning-based system to assist reddit moderators. *Proc. ACM Hum.-Comput. Interact.*, *3*(CSCW).

Chatterjee, P., Damevski, K., Kraft, N. A., & Pollock, L. (2020). Software-related slack chats with disentangled conversations. *Proceedings of the 17th international conference on mining software repositories*, 588–592.

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*, 321–357.

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.

Cheriyan, J., Savarimuthu, B. T. R., & Cranefield, S. (2021). Towards offensive language detection and reduction in four software engineering communities. *Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering*, 254–259.

Chollet, F., et al. (2015). Keras.

Chu, E., Roy, D., & Andreas, J. (2020). Are visual explanations useful? A case study in model-in-the-loop prediction. *CoRR*, *abs/2007.12248*.

Church, K. W. (2017). Word2vec. *Natural Language Engineering*, *23*(1), 155–162.

Cohen, J. (1992). Statistical power analysis. *Current directions in psychological science*, *1*(3), 98–101.

Cumming, G. (2014). The new statistics: Why and how. *Psychological science*, *25*(1), 7–29.

Curry, A. C., Abercrombie, G., & Rieser, V. (2021). Convabuse: Data, analysis, and benchmarks for nuanced abuse detection in conversational ai. *ArXiv*, *abs/2109.09483*.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and roc curves. *Proceedings of the 23rd International Conference on Machine Learning*, 233–240.

Davis, J. D. (2007). Real-world contexts, multiple representations, student-invented terminology, and y-intercept. *Mathematical Thinking and Learning*, *9*(4), 387–418.

De Boer, P.-T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, *134*, 19–67.

De Vries, P., Midden, C., & Bouwhuis, D. (2003). The effects of errors on system trust, self-confidence, and the allocation of control in route planning. *International Journal of Human-Computer Studies*, *58*(6), 719–735.

De Vries, W., van Cranenburgh, A., Bisazza, A., Caselli, T., van Noord, G., & Nissim, M. (2019). Bertje: A dutch bert model. *CoRR*, *abs/1912.09582*.

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. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 120–128.

de Freitas Barbosa, V. A., Gomes, J. C., de Santana, M. A., Albuquerque, J. E. d. A., de Souza, R. G., de Souza, R. E., & dos Santos, W. P. (2021). Heg. ia: An intelligent system to support diagnosis of covid-19 based on blood tests. *Research on Biomedical Engineering*, 1–18.

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*, 1549–1593.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Dong, X., Yu, Z., Cao, W., Shi, Y., & Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science*, *14*(2), 241–258.

Du, H., Zhang, Y., Gang, K., Zhang, L., & Chen, Y.-C. (2021). Online ensemble learning algorithm for imbalanced data stream. *Applied Soft Computing*, *107*, 107378.

Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. *Communications of the ACM*, *63*(1), 68–77.

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*, 1012–1031.

ElShawi, R., Sherif, Y., Al-Mallah, M., & Sakr, S. (2021). Interpretability in healthcare: A comparative study of local machine learning interpretability techniques. *Computational Intelligence*, *37*(4), 1633–1650.

Fahrmeir, L., Kneib, T., Lang, S., Marx, B., Fahrmeir, L., Kneib, T., Lang, S., & Marx, B. (2013). *Regression models*. Springer.

Fairwork. (2022). *Work in the planetary labour market: Fairwork cloudwork ratings 2022* (tech. rep.). Oxford. Retrieved December 7, 2023, from https://fair.work/wp-content/uploads/sites/17/2022/08/Fairwork-Cloudwork-Ratings-2022-FINAL-EN.pdf

Fernández, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, *61*, 863–905.

Forman, G. (2006). Tackling concept drift by temporal inductive transfer. *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 252–259.

Freitas dos Santos, T., Osman, N., & Schorlemmer, M. (2022a). Ensemble and incremental learning for norm violation detection. *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*.

Freitas dos Santos, T., Osman, N., & Schorlemmer, M. (2022b). Learning for detecting norm violation in online communities. *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*, 127–142.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(4), 463–484.

Gama, J., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, *46*(4).

Ging, D., & Siapera, E. (2018). Special issue on online misogyny.

Glavas, G., Karan, M., & Vulic, I. (2020). Xhate-999: Analyzing and detecting abusive language across domains and languages. *International Conference on Computational Linguistics*.

Gray, K. L. (2018). Gaming out online: Black lesbian identity development and community building in xbox live. *Journal of Lesbian Studies*, *22*, 282–296.

Hajibabaee, P., Malekzadeh, M., Ahmadi, M., Heidari, M., Esmaeilzadeh, A., Abdolazimi, R., & James Jr, H. (2022). Offensive language detection on social media based on text classification. *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 0092–0098.

Hakak, S., Alazab, M., Khan, S., Gadekallu, T. R., Maddikunta, P. K. R., & Khan, W. Z. (2021). An ensemble machine learning approach through effective feature extraction to classify fake news. *Future Generation Computer Systems*, *117*, 47–58.

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.

Hastie, T., & Tibshirani, R. (1987). Generalized additive models: Some applications. *Journal of the American Statistical Association*, *82*(398), 371–386.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Vol. 2). Springer.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

He, R., Liu, L., Ye, H., Tan, Q., Ding, B., Cheng, L., Low, J.-W., Bing, L., & Si, L. (2021). On the effectiveness of adapter-based tuning for pretrained language model adaptation. *ArXiv*, *abs/2106.03164*.

He, S., Ding, L., Dong, D., Zhang, M., & Tao, D. (2022). Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *ArXiv*, *abs/2210.04284*.

Hoi, S. C., Sahoo, D., Lu, J., & Zhao, P. (2021). Online learning: A comprehensive survey. *Neurocomputing*, *459*, 249–289.

Hong, S. R., Hullman, J., & Bertini, E. (2020). Human factors in model interpretability: Industry practices, challenges, and needs. *Proceedings of the ACM on Human-Computer Interaction*, *4*(CSCW1), 1–26.

Hossam, M., Le, T., Zhao, H., & Phung, D. (2021). Explain2attack: Text adversarial attacks via cross-domain interpretability. *2020 25th International Conference on Pattern Recognition (ICPR)*, 8922–8928.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., & Gelly, S. (2019). Parameter-efficient transfer learning for nlp. *International Conference on Machine Learning*, 2790–2799.

Huang, H.-P., Sun, D., Liu, Y., Chu, W.-S., Xiao, T., Yuan, J., Adam, H., & Yang, M.-H. (2022). Adaptive transformers for robust few-shot cross-domain face anti-spoofing. *European Conference on Computer Vision*, 37–54.

Iadarola, G., Martinelli, F., Mercaldo, F., & Santone, A. (2021). Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security*, *105*, 102198.

Idrees, M. M., Minku, L. L., Stahl, F., & Badii, A. (2020). A heterogeneous online learning ensemble for non-stationary environments. *Knowledge-Based Systems*, *188*, 104983.

Ishmam, A. M., & Sharmin, S. (2019). Hateful speech detection in public facebook pages for the bengali language. *2019 18th ICMLA*, 555–560.

Islam, R., Treves, B., Rokon, M. 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.

Jia, F., Li, S., Zuo, H., & Shen, J. (2020). Deep neural network ensemble for the intelligent fault diagnosis of machines under imbalanced data. *IEEE Access*, *8*, 120974–120982.

Jin, X., Zhang, D., Zhu, H., Xiao, W., Li, S.-W., Wei, X., Arnold, A., & Ren, X. (2022). Lifelong pretraining: Continually adapting language models to emerging corpora. *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4764–4780.

Joachims, T. (1996). *A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.* (tech. rep.). Carnegie-mellon univ pittsburgh pa dept of computer science.

Jones, A. J. I., & Sergot, M. (1994). On the characterization of law and computer systems: The normative systems perspective. In *Deontic logic in computer science: Normative system specification* (pp. 275–307). John Wiley; Sons Ltd.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., & Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. *CoRR*, *abs/1612.03651*.

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.

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. *2021 IEEE 8th DSAA*, 1–10.

Karunakaran, S., & Ramakrishan, R. (2019). Testing stylistic interventions to reduce emotional impact of content moderation workers. *Seventh AAAI Conference on Human Computation and Crowdsourcing*, *7*, 50–58.

Kavzoglu, T., & Teke, A. (2022). Predictive performances of ensemble machine learning algorithms in landslide susceptibility mapping using random forest, extreme gradient boosting (xgboost) and natural gradient boosting (ngboost). *Arabian Journal for Science and Engineering*, *47*(6), 7367–7385.

Kenton, J. D. M.-W. C., & Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of naacL-HLT*, 4171–4186.

Keum, B. T., & Miller, M. J. (2018). Racism on the internet: Conceptualization and recommendations for research. *Psychology of violence*, *8*(6), 782.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv*, *abs/1412.6980*.

Kirkman, M. S., & Oswald, D. L. (2020). Is it just me, or was that sexist? the role of sexism type and perpetrator race in identifying sexism. *The Journal of Social Psychology*, *160*(2), 236–247.

Krishna, K., & Murty, M. N. (1999). Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *29*(3), 433–439.

Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S., & Doshi-Velez, F. (2019). An evaluation of the human-interpretability of explanation. *ArXiv*, *abs/1902.00006*.

Larsen, K. (2015). Gam: The predictive modeling silver bullet. *Multithreaded. Stitch Fix*, *30*, 1–27.

Le, H., Pino, J., Wang, C., Gu, J., Schwab, D., & Besacier, L. (2021). Lightweight adapter tuning for multilingual speech translation. *ArXiv*, *abs/2106.01463*.

Lebichot, B., Paldino, G. M., Siblini, W., He-Guelton, L., Oblé, F., & Bontempi, G. (2021). Incremental learning strategies for credit cards fraud detection. *International Journal of Data Science and Analytics*.

Lenka, S. R., Bisoy, S. K., Priyadarshini, R., & Sain, M. (2022). Empirical analysis of ensemble learning for imbalanced credit scoring datasets: A systematic review. *Wireless Communications and Mobile Computing*, *2022*.

Li, T. C., Gharibshah, J., Papalexakis, E. E., & Faloutsos, M. (2017). Trollspot: Detecting misbehavior in commenting platforms. *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, 171–175.

Li, Y., Cheng, M., Hsieh, C.-J., & Lee, T. C. (2022). A review of adversarial attack and defense for classification methods. *The American Statistician*, 1–17.

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*, 105694.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Liu, N., Li, X., Qi, E., Xu, M., Li, L., & Gao, B. (2020). A novel ensemble learning paradigm for medical diagnosis with imbalanced data. *IEEE Access*, *8*, 171263–171280.

Liu, S., Wang, Y., Zhang, J., Chen, C., & Xiang, Y. (2017). Addressing the class imbalance problem in twitter spam detection using ensemble learning. *Computers & Security*, *69*, 35–49.

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*, *abs/1907.11692*.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *International Conference on Learning Representations*.

Losing, V., Hammer, B., & Wersing, H. (2017). Self-adjusting memory: How to deal with diverse drift types. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4899–4903.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346–2363.

Lu, Q., Dou, D., & Nguyen, T. H. (2021). Parameter-efficient domain knowledge integration from multiple sources for biomedical pre-trained language models. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 3855–3865.

Luengo, J., Fernández, A., García, S., & Herrera, F. (2011). Addressing data complexity for imbalanced data sets: Analysis of smote-based oversampling and evolutionary undersampling. *Soft Computing*, *15*, 1909–1936.

Lyu, Q., Apidianaki, M., & Callison-Burch, C. (2022). Towards faithful model explanation in nlp: A survey. *CoRR*, *abs/2209.11326*.

Mahajan, A., Shah, D., & Jafar, G. (2021). Explainable ai approach towards toxic comment classification. In A. E. Hassanien, S. Bhattacharyya, S. Chakrabati, A. Bhattacharya, & S. Dutta (Eds.), *Emerging technologies in data mining and information security* (pp. 849–858). Springer Singapore.

Malik, B., Kashyap, A. R., Kan, M., & Poria, S. (2023). Udapter - efficient domain adaptation using adapters. *Conference of the European Chapter of the Association for Computational Linguistics*.

Markov, I., Gevers, I., & Daelemans, W. (2022). An ensemble approach for dutch cross-domain hate speech detection. *International Conference on Applications of Natural Language to Information Systems*, 3–15.

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.

Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., & Roth, D. (2023). Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, *56*(2), 1–40.

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*, *abs/2301.13852*.

Mohammed, R., Rawashdeh, J., & Abdullah, M. (2020). Machine learning with oversampling and undersampling techniques: Overview study and experimental results. *2020 11th international conference on information and communication systems (ICICS)*, 243–248.

Mohawesh, R., Tran, S., Ollington, R., & Xu, S. (2021). Analysis of concept drift in fake reviews detection. *Expert Systems with Applications*, *169*, 114318.

Mohseni, S., Zarei, N., & Ragan, E. D. (2021). A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, *11*(3-4), 1–45.

Mollas, I., Chrysopoulou, Z., Karlos, S., & Tsoumakas, G. (2022). Ethos: A multi-label hate speech detection dataset. *Complex & Intelligent Systems*, 1–16.

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.

Morales, J., López-Sánchez, M., Rodríguez-Aguilar, J. A., Wooldridge, M., & Vasconcelos, W. (2015). Synthesising liberal normative systems. *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 433–441.

Morris, J., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., & Qi, Y. (2020). Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 119–126.

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*.

Muslim, F., Purwarianti, A., & Ruskanda, F. Z. (2021). Cost-sensitive learning and ensemble bert for identifying and categorizing offensive language in social media. *2021 8th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, 1–6.

Mutanga, R. T., Naicker, N., & Olugbara, O. O. (2020). Hate speech detection in twitter using transformer methods. *International Journal of Advanced Computer Science and Applications*, *11*(9).

Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, *30*(1), 3–26.

Nasar, Z., Jaffry, S. W., & Malik, M. K. (2021). Named entity recognition and relation extraction: State-of-the-art. *ACM Computing Surveys (CSUR)*, *54*(1), 1–39.

Niu, R., Wei, Z., Wang, Y., & Wang, Q. (n.d.). Attexplainer: Explain transformer via attention by reinforcement learning.

Nockleby, J. (2000). Hate speech. In L. Levy, K. Kenneth, & A. Winkler (Eds.), *Encyclopedia of the american constitution (vol 6.)* (pp. 1277–1279).

Novikova, J., & Shkaruta, K. (2022). Deck: Behavioral tests to improve interpretability and generalizability of bert models detecting depression from text. *ArXiv*, *abs/2209.05286*.

Nowak, M. A., & Sigmund, K. (2005). Evolution of indirect reciprocity. *Nature*, *437*(7063), 1291–1298.

Okada, I. (2020). A review of theoretical studies on indirect reciprocity. *Games*, *11*(3), 27.

Pamungkas, E. W., & Patti, V. (2019). Cross-domain and cross-lingual abusive language detection: A hybrid approach with deep learning and a multilingual lexicon. *Proceedings of the 57th annual meeting of the association for computational linguistics: Student research workshop*, 363–370.

Park, H. M. (2010). Hypothesis testing and statistical power of a test. *The University Information Technology Services (UITS) Center for Statistical and Mathematical Computing, Indiana University*.

Pega, F., & Veale, J. F. (2015). The case for the world health organization's commission on social determinants of health to address gender identity. *American journal of public health*, *105*(3), e58–e62.

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*, 171–182.

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., & Gurevych, I. (2020). Adapterfusion: Non-destructive task composition for transfer learning. *CoRR*, *abs/2005.00247*.

Pierse, C. (2023). Transformers interpret [[Online; accessed 17-October-2023]]. https://pypi.org/project/transformers-interpret

Prolific. (2023a). Prolific, crowdsourcing platform [[Online; accessed 17-October-2023]]. https://www.prolific.com

Prolific. (2023b). Prolific's payment principles [[Online; accessed 17-October-2023]]. https://researcher-help.prolific.com/hc/en-gb/articles/4407695146002-Prolific-s-payment-principles

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., & Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, *63*(10), 1872–1897.

R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

Radensky, M., Downey, D., Lo, K., Popovic, Z., & Weld, D. S. (2022). Exploring the role of local and global explanations in recommender systems. *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 1–7.

Rahman, J. (2012). The n word: Its history and use in the african american community. *Journal of English Linguistics*, *40*(2), 137–171.

Räuker, T., Ho, A., Casper, S., & Hadfield-Menell, D. (2023). Toward transparent ai: A survey on interpreting the inner structures of deep neural networks. *2023 IEEE Conference on Secure and Trustworthy Machine Learning*, 464–483.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Risch, J., & Krestel, R. (2020). Toxic comment detection in online discussions. In *Deep learning-based approaches for sentiment analysis* (pp. 85–109). Springer.

Rong, Y., Leemann, T., Nguyen, T.-t., Fiedler, L., Seidel, T., Kasneci, G., & Kasneci, E. (2022). Towards human-centered explainable ai: User studies for model explanations. *ArXiv*, *abs/2210.11584*.

Roshan, S. E., & Asadi, S. (2020). Improvement of bagging performance for classification of imbalanced datasets using evolutionary multi-objective optimization. *Engineering Applications of Artificial Intelligence*, *87*, 103319.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Russell, S., & Norvig, P. (2002). *Artificial intelligence: A modern approach*. Pearson Education.

Sabry, S. S., Adewumi, T., Abid, N., Kovács, G., Liwicki, F., & Liwicki, M. (2022). Hat5: Hate language identification using text-to-text transfer transformer. *2022 International Joint Conference on Neural Networks (IJCNN)*, 1–7.

Sáenz, C. A. C., & Becker, K. (2021). Interpreting bert-based stance classification: A case study about the brazilian covid vaccination. *Anais do XXXVI Simpósio Brasileiro de Bancos de Dados*, 73–84.

Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), e1249.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432.

Samghabadi, N. S., Patwa, P., Pykl, S., Mukherjee, P., Das, A., & Solorio, T. (2020). Aggression and misogyny detection using bert: A multi-task approach. *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, 126–131.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, *abs/1910.01108*.

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*, 114135.

Sayre, G. M. (2023). The costs of insecurity: Pay volatility and health outcomes. *Journal of Applied Psychology*, *108*(7), 1223.

Schuff, H., Jacovi, A., Adel, H., Goldberg, Y., & Vu, N. T. (2022). Human interpretation of saliency-based explanation over text. *2022 ACM Conference on Fairness, Accountability, and Transparency*, 611–636.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, *45*(11), 2673–2681.

Sechidis, K., Tsoumakas, G., & Vlahavas, I. (2011). On the stratification of multi-label data. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 145–158.

Shank, D. B. (2016). Using crowdsourcing websites for sociological research: The case of amazon mechanical turk. *The American Sociologist*, *47*, 47–55.

Sharma, A., Kabra, A., & Jain, M. (2022). Ceasing hate with moh: Hate speech detection in hindi–english code-switched language. *Information Processing & Management*, *59*(1), 102760.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, *6*(12), 310–316.

Shmargad, Y., Coe, K., Kenski, K., & Rains, S. A. (2022). Social norms and the dynamics of online incivility. *Social Science Computer Review*, *40*(3), 717–735.

Shojaee, S., Murad, M. A. A., Azman, A. B., Sharef, N. M., & Nadali, S. (2013). Detecting deceptive reviews using lexical and syntactic features. *2013 13th International Conference on Intellient Systems Design and Applications*, 53–58.

Siblini, W., Fréry, J., He-Guelton, L., Oblé, F., & Wang, Y.-Q. (2020). Master your metrics with calibration. In M. R. Berthold, A. Feelders, & G. Krempl (Eds.), *Advances in intelligent data analysis xviii* (pp. 457–469). Springer International Publishing.

Skopik, F., & Pahi, T. (2020). Under false flag: Using technical artifacts for cyber attack attribution. *Cybersecurity*, *3*, 1–20.

Steiger, M., Bharucha, T. J., Venkatagiri, S., Riedl, M. J., & Lease, M. (2021). The psychological well-being of content moderators: The emotional labor of commercial moderation and avenues for improving support. *2021 CHI Conference on Human Factors in Computing Systems*, 1–14.

Strøm, E. (2021). Multi-label style change detection by solving a binary classification problem. *CLEF (Working Notes)*, 2146–2157.

Subramanian, M., Ponnusamy, R., Benhur, S., Shanmugavadivel, K., Ganesan, A., Ravi, D., Shanmugasundaram, G. K., Priyadharshini, R., & Chakravarthi, B. R. (2022). Offensive language detection in tamil youtube comments by adapters and cross-domain knowledge transfer. *Computer Speech & Language*, *76*, 101404.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *34th International Conference on Machine Learning*, 3319–3328.

Suri, J. S., Bhagawati, M., Paul, S., Protogerou, A. D., Sfikakis, P. P., Kitas, G. D., Khanna, N. N., Ruzsa, Z., Sharma, A. M., Saxena, S., et al. (2022). A powerful paradigm for cardiovascular risk stratification using multiclass, multi-label, and ensemble-based machine learning paradigms: A narrative review. *Diagnostics*, *12*(3), 722.

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.

Taha, A. A., & Malebary, S. J. (2020). An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access*, *8*, 25579–25587.

Taherkhani, A., Cosma, G., & McGinnity, T. M. (2020). Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, *404*, 351–366.

Tang, W., Ding, Z., & Zhou, M. (2019). A spammer identification method for class imbalanced weibo datasets. *IEEE Access*, *7*, 29193–29201.

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*.

Vidgen, B., Thrush, T., Waseem, Z., & Kiela, D. (2021). Learning from the worst: Dynamically generated datasets to improve online hate detection. *Proceedings of EMNLP 2021*, 1667–1682.

Voigt, P., & Von dem Bussche, A. (2017). The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, *10*(3152676), 10–5555.

Wang, B., & Pineau, J. (2016). Online bagging and boosting for imbalanced data streams. *IEEE Transactions on Knowledge and Data Engineering*, *28*(12), 3353–3366.

Wang, H., & Abraham, Z. (2015). Concept drift detection for streaming data. *2015 international joint conference on neural networks (IJCNN)*, 1–9.

Wang, Q., Luo, Z., Huang, J., Feng, Y., & Liu, Z. (2017). A novel ensemble method for imbalanced data learning: Bagging of extrapolation-smote svm. *Computational intelligence and neuroscience*, *2017*.

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.

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.

Wang, W., & Sun, D. (2021). The improved adaboost algorithms for imbalanced data classification. *Information Sciences*, *563*, 358–374.

Wang, X., & Yin, M. (2021). Are explanations helpful? a comparative study of the effects of explanations in ai-assisted decision-making. *26th international conference on intelligent user interfaces*, 318–328.

Weiss, G. M. (2009). The impact of small disjuncts on classifier learning. In *Data mining: Special issue in annals of information systems* (pp. 193–226). Springer.

West, A. G., & Lee, I. (2011). Multilingual vandalism detection using language-independent & ex post facto evidence. *CLEF Notebooks*.

Wikipedia. (2023). Wikipedia, the free encyclopedia [[Online; accessed 02-Jun-2023]]. https://en.wikipedia.org/wiki/Wikipedia

Wilk, M. B., & Gnanadesikan, R. (1968). Probability plotting methods for the analysis for the analysis of data. *Biometrika*, *55*(1), 1–17.

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., ... Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45.

Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *73*(1), 3–36.

Woolson, R. F. (2007). Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials*, 1–3.

Wu, Z., Lin, W., & Ji, Y. (2018). An integrated ensemble learning model for imbalanced fault diagnostics and prognostics. *IEEE Access*, *6*, 8394–8402.

Xiang, T., MacAvaney, S., Yang, E., & Goharian, N. (2021). ToxCCIn: Toxic content classification with interpretability. *Proceedings of the 11th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 1–12.

Xu, J., Sun, X., Zhang, Z., Zhao, G., & Lin, J. (2019). Understanding and improving layer normalization. *Advances in Neural Information Processing Systems*, *32*.

Xu, T., Chen, W., Pichao, W., Wang, F., Li, H., & Jin, R. (2021). Cdtrans: Cross-domain transformer for unsupervised domain adaptation. *International Conference on Learning Representations*.

Yang, F., Huang, Z., Scholtz, J., & Arendt, D. L. (2020). How do visual explanations foster end users' appropriate trust in machine learning? *Proceedings of the 25th international conference on intelligent user interfaces*, 189–201.

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. *J. Mach. Learn. Res.*, *21*(43), 1–36.

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.

Zangerle, E., Mayerl, M., Specht, G., Potthast, M., & Stein, B. (2020). Overview of the style change detection task at pan 2020. *CLEF (Working Notes)*, *93*.

Zhang, H., Liu, W., Wang, S., Shan, J., & Liu, Q. (2019). Resample-based ensemble framework for drifting imbalanced data streams. *IEEE Access*, *7*, 65103–65115.

Zhang, R., Zheng, Y., Mao, X., & Huang, M. (2021). Unsupervised domain adaptation with adapter. *CoRR*, *abs/2111.00667*.

Zhang, T., Chen, J., Li, F., Zhang, K., Lv, H., He, S., & Xu, E. (2022). Intelligent fault diagnosis of machines with small & imbalanced data: A state-of-the-art review and possible extensions. *ISA transactions*, *119*, 152–171.

Zhang, W., Yoshida, T., & Tang, X. (2011). A comparative study of TF*IDF, LSI and multi-words for text classification. *Expert systems with applications*, *38*(3), 2758–2765.

Zhang, Y., Ding, X., & Gu, N. (2018). Understanding fatigue and its impact in crowdsourcing. *22nd IEEE International Conference on Computer Supported Cooperative Work in Design*, 57–62.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.

Appendix A

FeDAL – Friedman-Nemenyi Complete Test Results

Friedman Test P-Value	**0.0005**			
Critical Difference (CD) Value	0.6051			
Metric	Overall Recall			
	Method	Batch	Mini-Batch	Online
Original	Batch	1.0000	**0.0036**	0.5527
	Mini-Batch	**0.0036**	1.0000	0.0721
	Online	0.5527	0.0721	1.0000

Table A.1: P-values for the dataset with NO concept drift (original). Friedman-Nemenyi test comparing OVERALL recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0005**			
Critical Difference (CD) Value	0.6051			
Metric	Overall Recall			
	Method	Batch	Mini-Batch	Online
Concept Drift	Batch	1.0000	**0.0125**	**0.0010**
	Mini-Batch	**0.0125**	1.0000	**0.0125**
	Online	**0.0010**	**0.0125**	1.0000

Table A.2: P-values for the dataset with concept drift. Friedman-Nemenyi test comparing OVERALL recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0001**			
Critical Difference (CD) Value	0.6051			
Metric	Regular Recall			
	Method	Batch	Mini-Batch	Online
Original	Batch	1.0000	**0.0010**	0.1120
	Mini-Batch	**0.0010**	1.0000	**0.0010**
	Online	0.1120	**0.0010**	1.0000

Table A.3: P-values for the dataset with NO concept drift (original). Friedman-Nemenyi test comparing REGULAR recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0015**			
Critical Difference (CD) Value	0.6051			
Metric	Regular Recall			
	Method	Batch	Mini-Batch	Online
Concept Drift	Batch	1.0000	**0.0010**	0.0103
	Mini-Batch	**0.0010**	1.0000	**0.0317**
	Online	**0.0103**	**0.0317**	1.0000

Table A.4: P-values for the dataset with concept drift. Friedman-Nemenyi test comparing REGULAR recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0245**			
Critical Difference (CD) Value	0.6051			
Metric	Vandalism Recall			
	Method	Batch	Mini-Batch	Online
Original	Batch	1.0000	0.2138	**0.0222**
	Mini-Batch	0.2138	1.0000	**0.0010**
	Online	**0.0222**	**0.0010**	1.0000

Table A.5: P-values for the dataset with NO concept drift (original). Friedman-Nemenyi test comparing VANDALISM recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0032**			
Critical Difference (CD) Value	0.6051			
Metric	Vandalism Recall			
	Method	Batch	Mini-Batch	Online
Concept Drift	Batch	1.0000	0.9000	0.0014
	Mini-Batch	0.9000	1.0000	0.0010
	Online	0.0014	0.0010	1.0000

Table A.6: P-values for the dataset with concept drift. Friedman-Nemenyi test comparing VANDALISM recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0095**			
Critical Difference (CD) Value	0.6051			
Metric	Re-label Recall			
	Method	Batch	Mini-Batch	Online
Re-label	Batch	1.0000	0.5158	**0.0010**
	Mini-Batch	0.5158	1.0000	**0.0010**
	Online	**0.0010**	**0.0010**	1.0000

Table A.7: P-values for the dataset with concept drift. Friedman-Nemenyi test comparing re-label recall for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0245**			
Critical Difference (CD) Value	0.6051			
Metric	AUC-ROC			
	Method	Batch	Mini-Batch	Online
Original	Batch	1.0000	**0.0010**	**0.0068**
	Mini-Batch	**0.0010**	1.0000	0.7373
	Online	**0.0068**	0.7373	1.0000

Table A.8: P-values for the dataset with NO concept drift (original). Friedman-Nemenyi test comparing AUC-ROC for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0001**			
Critical Difference (CD) Value	0.6051			
Metric	AUC-ROC			
	Method	Batch	Mini-Batch	Online
Concept Drift	Batch	1.0000	**0.0010**	**0.0010**
	Mini-Batch	**0.0010**	1.0000	0.1673
	Online	**0.0010**	0.1673	1.0000

Table A.9: P-values for the dataset with concept drift. Friedman-Nemenyi test comparing AUC-ROC for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0001**			
Critical Difference (CD) Value	0.6051			
Metric	AUC-PR			
	Method	Batch	Mini-Batch	Online
Original	Batch	1.0000	**0.0010**	**0.0010**
	Mini-Batch	**0.0010**	1.0000	0.0971
	Online	**0.0010**	0.0971	1.0000

Table A.10: P-values for the dataset with NO concept drift (original). Friedman-Nemenyi test comparing AUC-PR for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Friedman Test P-Value	**0.0001**			
Critical Difference (CD) Value	0.6051			
Metric	AUC-PR			
	Method	Batch	Mini-Batch	Online
Concept Drift	Batch	1.0000	**0.0010**	**0.0010**
	Mini-Batch	**0.0010**	1.0000	0.1285
	Online	**0.0010**	0.1285	1.0000

Table A.11: P-values for the dataset with concept drift. Friedman-Nemenyi test comparing AUC-PR for the batch, mini-batch, and online learning approaches. Bold p-values indicate results below the critical value $\alpha = 0.05$.

Appendix B

LaMAL - Local Interpretation Examples for All Violation Classes

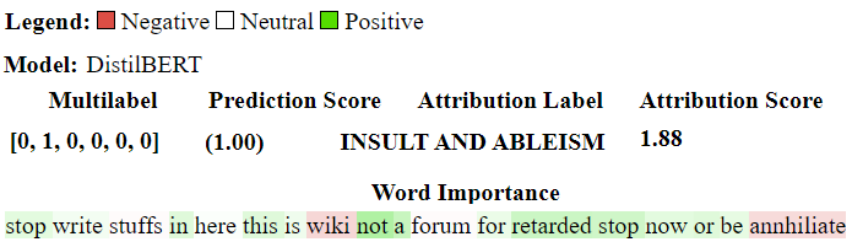


Figure B.1: The local interpretation of a specific edit considering the DistilBERT model in the multi-label case for LaMAL. The label considered is INSULT AND ABLEISM. The relevance score is calculated using Integrated Gradients (IG) (q.v. Section 2.5.2).

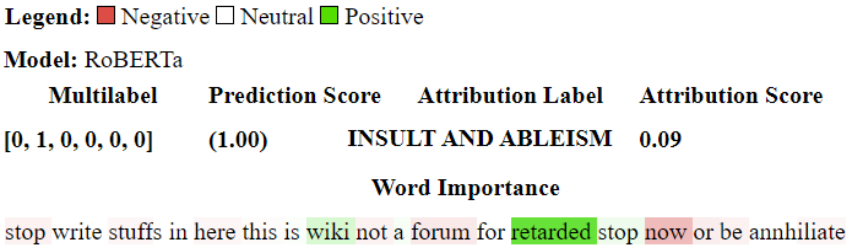


Figure B.2: The local interpretation of a specific edit considering the RoBERTa model in the multi-label case for LaMAL. The label considered is INSULT AND ABLEISM. The relevance score is calculated using IG (q.v. Section 2.5.2).

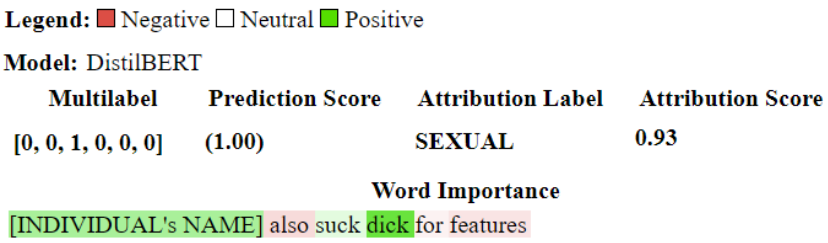


Figure B.3: The local interpretation of a specific edit considering the DistilBERT model in the multi-label case for LaMAL. The label considered is SEXUAL HARASSMENT. The relevance score is calculated using IG (q.v. Section 2.5.2).

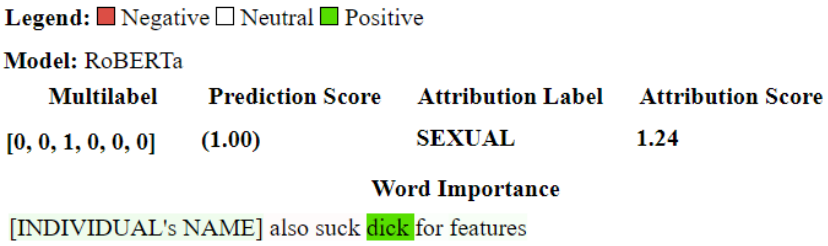


Figure B.4: The local interpretation of a specific edit considering the RoBERTa model in the multi-label case for LaMAL. The label considered is SEXUAL HARASSMENT. The relevance score is calculated using IG (q.v. Section 2.5.2).

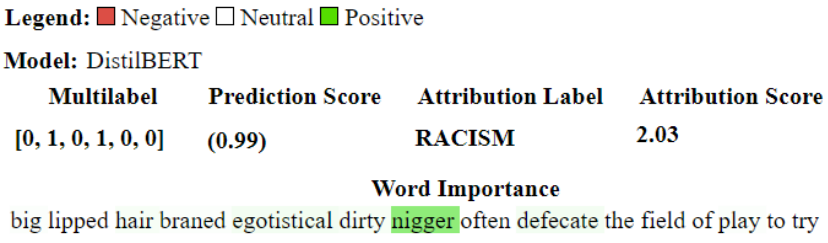


Figure B.5: The local interpretation of a specific edit considering the DistilBERT model in the multi-label case for LaMAL. The label considered is RACISM. The relevance score is calculated using IG (q.v. Section 2.5.2).

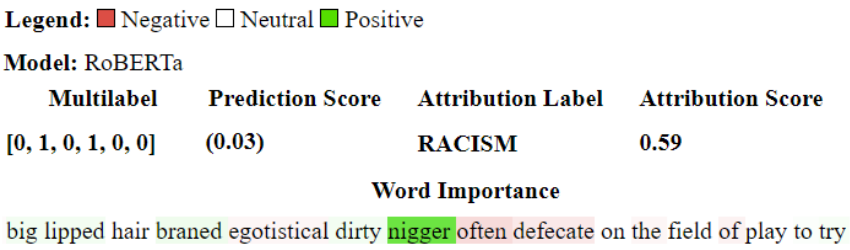


Figure B.6: The local interpretation of a specific edit considering the RoBERTa model in the multi-label case for LaMAL. The label considered is RACISM. The relevance score is calculated using IG (q.v. Section 2.5.2).

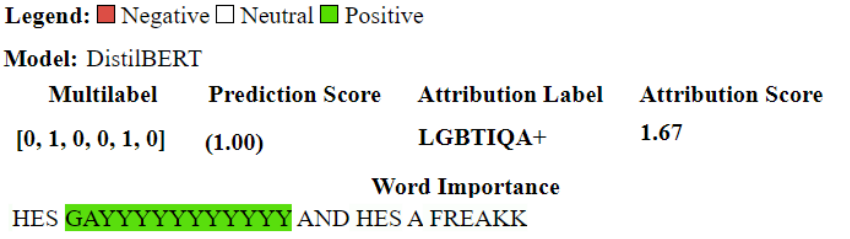


Figure B.7: The local interpretation of a specific edit considering the DistilBERT model in the multi-label case for LaMAL. The label considered is LGBTQIA+ Attack. The relevance score is calculated using IG (q.v. Section 2.5.2).

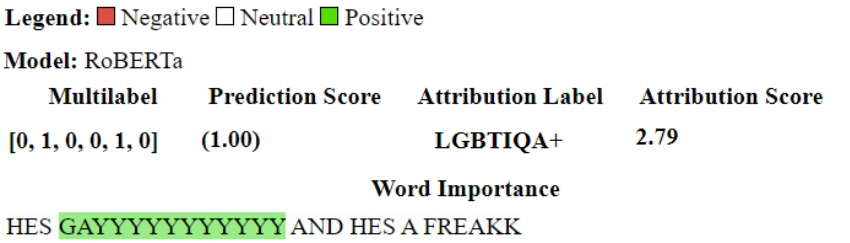


Figure B.8: The local interpretation of a specific edit considering the RoBERTa model in the multi-label case for LaMAL. The label considered is LGBTQIA+ Attack. The relevance score is calculated using IG (q.v. Section 2.5.2).

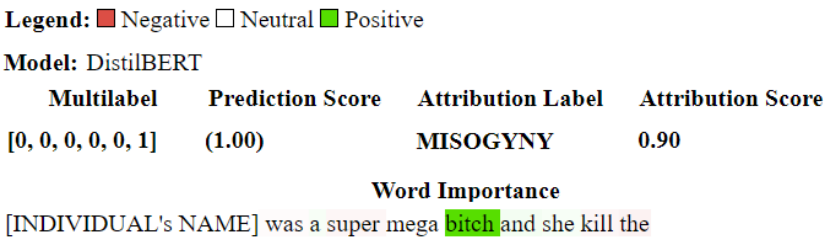


Figure B.9: The local interpretation of a specific edit considering the DistilBERT model in the multi-label case for LaMAL. The label considered is MISOGYNY. The relevance score is calculated using IG (q.v. Section 2.5.2).

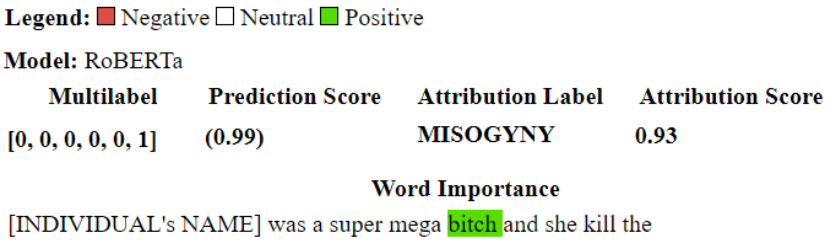


Figure B.10: The local interpretation of a specific edit considering the RoBERTa model in the multi-label case for LaMAL. The label considered is MISOGYNY. The relevance score is calculated using IG (q.v. Section 2.5.2).

Appendix C

LaMAL - Sum of Relevance Scores for All Violation Classes

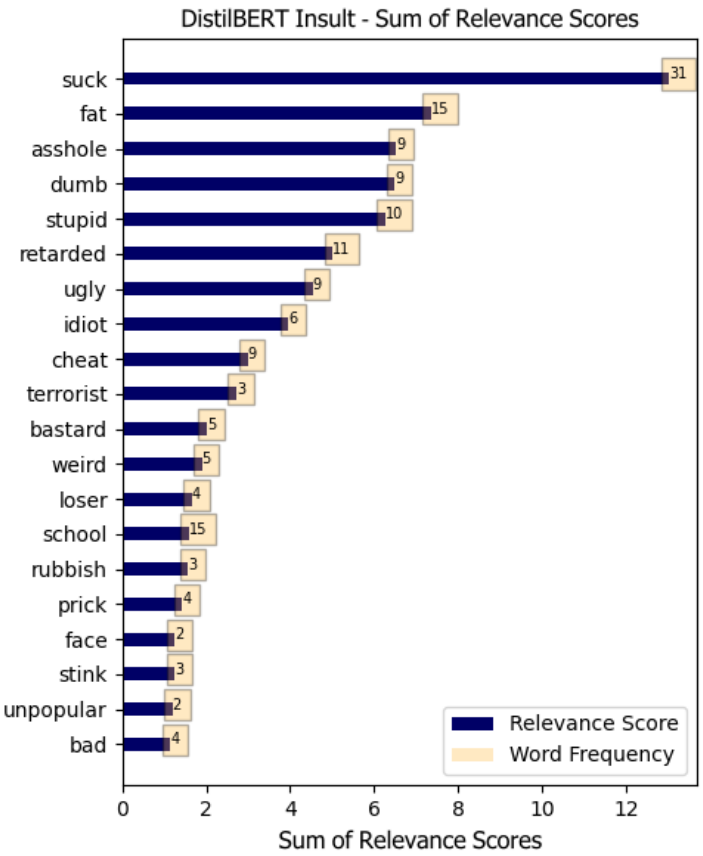


Figure C.1: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model in the multi-label case for LaMAL. The label considered is INSULT AND ABLEISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using Integrated Gradients (IG) (q.v. Section 2.5.2).

s

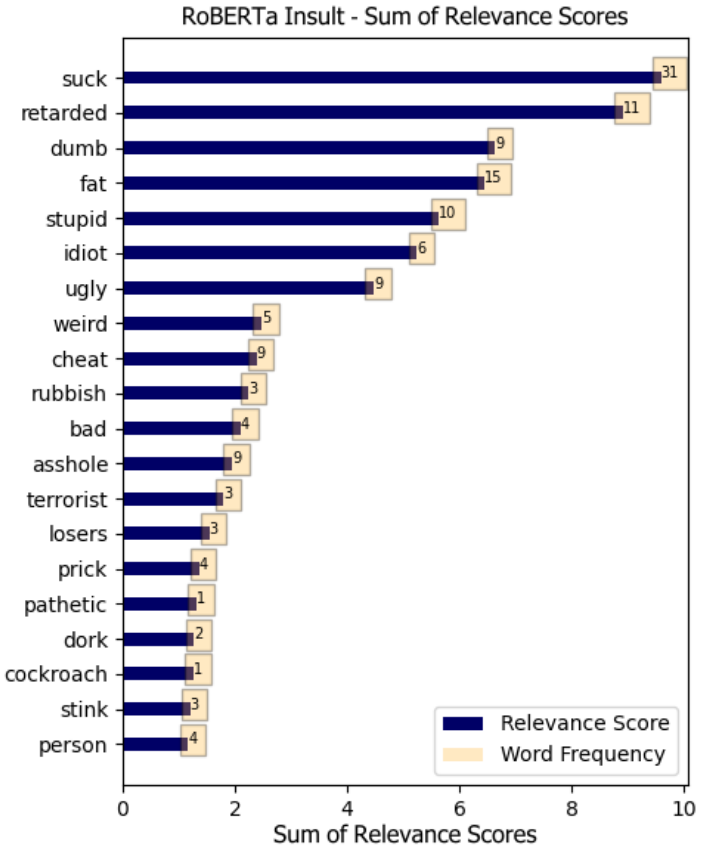


Figure C.2: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is INSULT AND ABLEISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

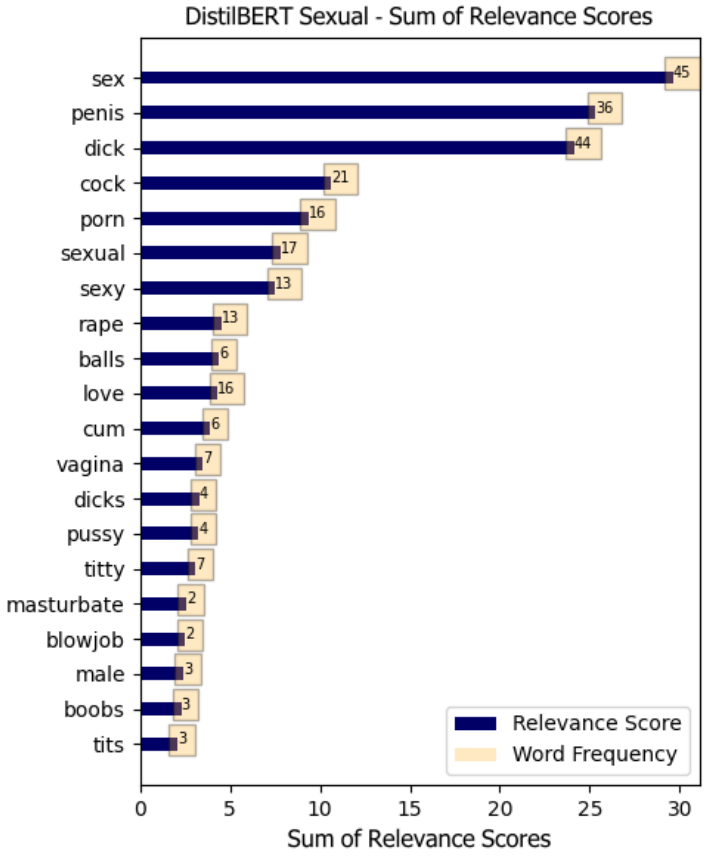


Figure C.3: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model in the multi-label case for LaMAL. The label considered is SEXUAL HARASSMENT. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

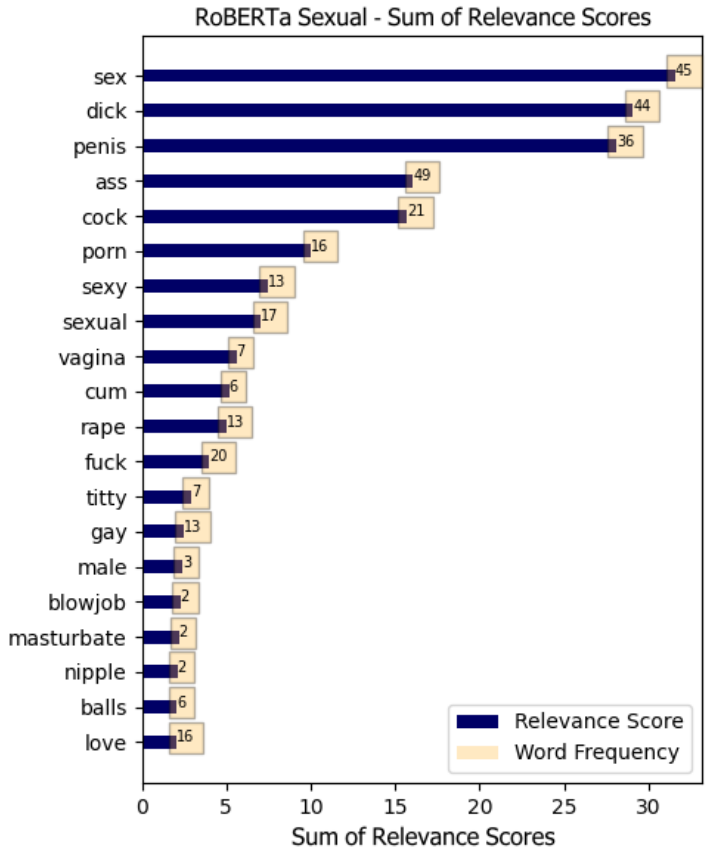


Figure C.4: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is SEXUAL HARASSMENT. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

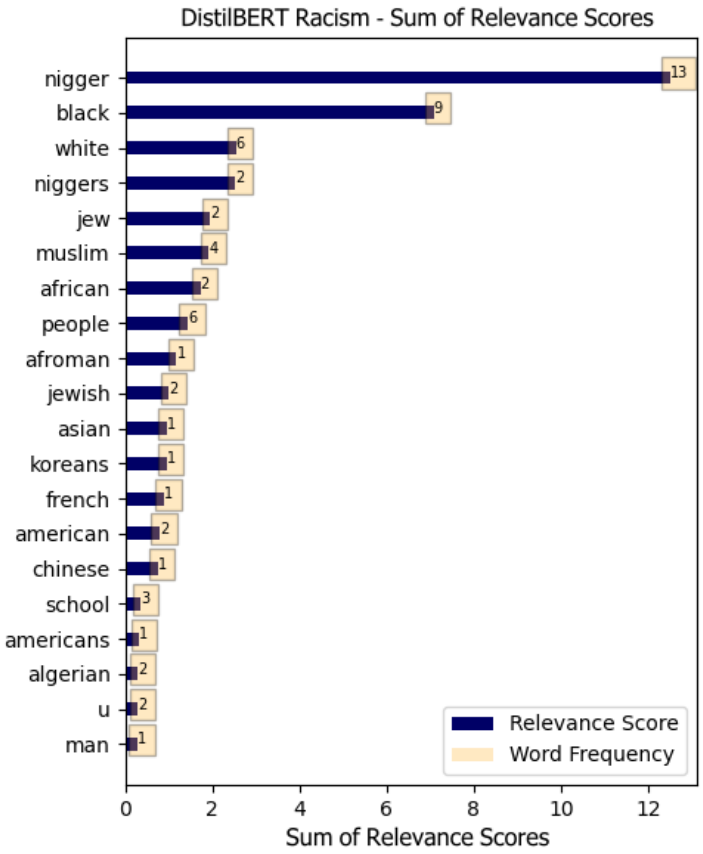


Figure C.5: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model in the multi-label case for LaMAL. The label considered is RACISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

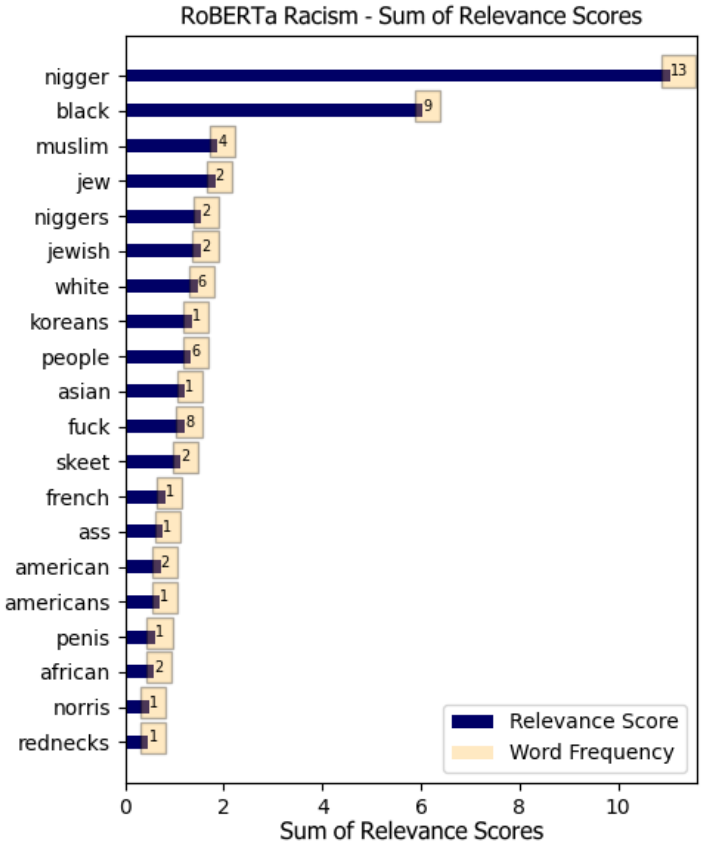


Figure C.6: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is RACISM. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

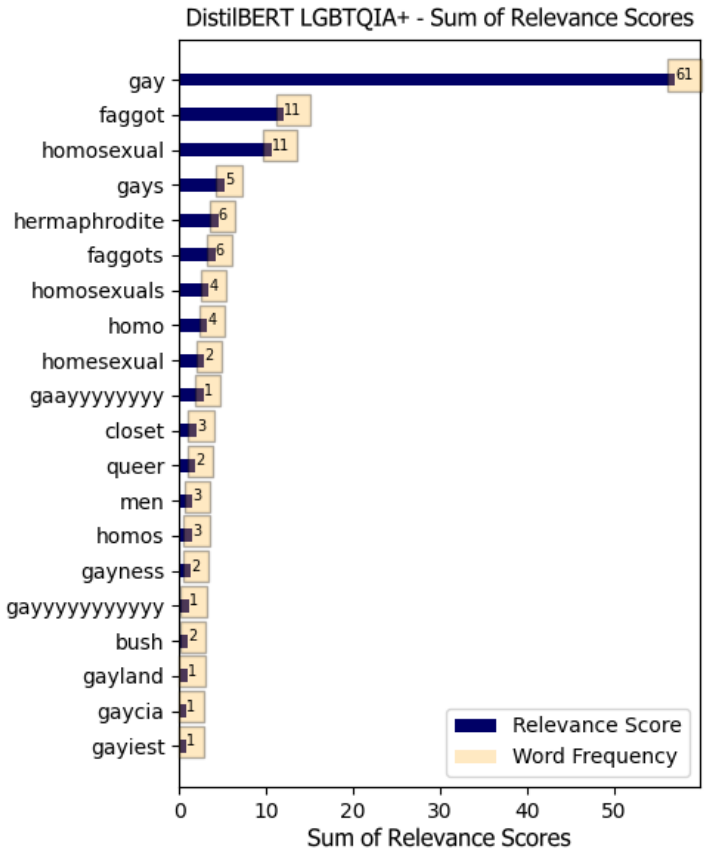


Figure C.7: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model in the multi-label case for LaMAL. The label considered is LGBTQIA+ Attack. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

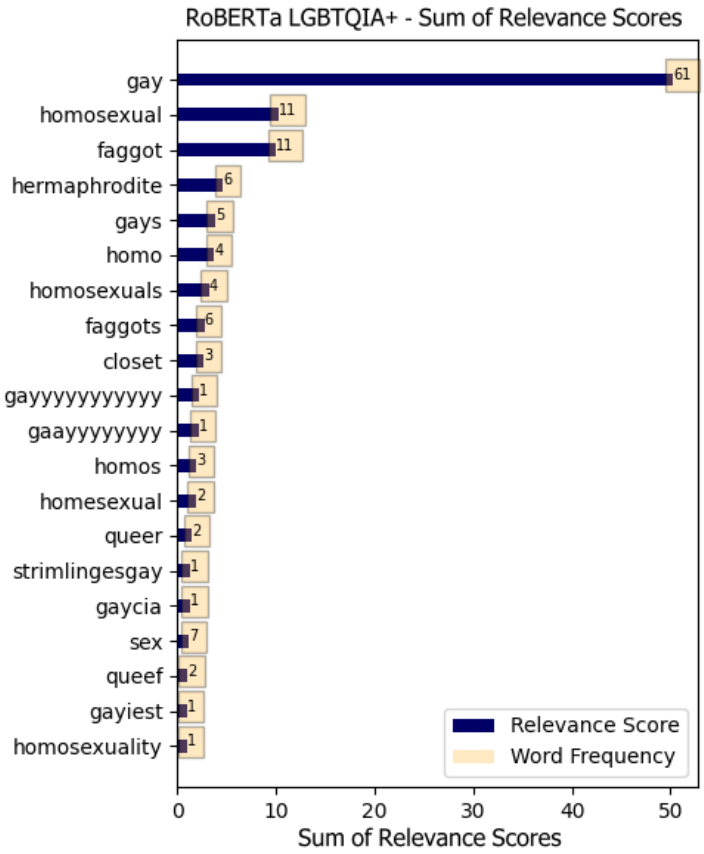


Figure C.8: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is LGBTQIA+ Attack. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

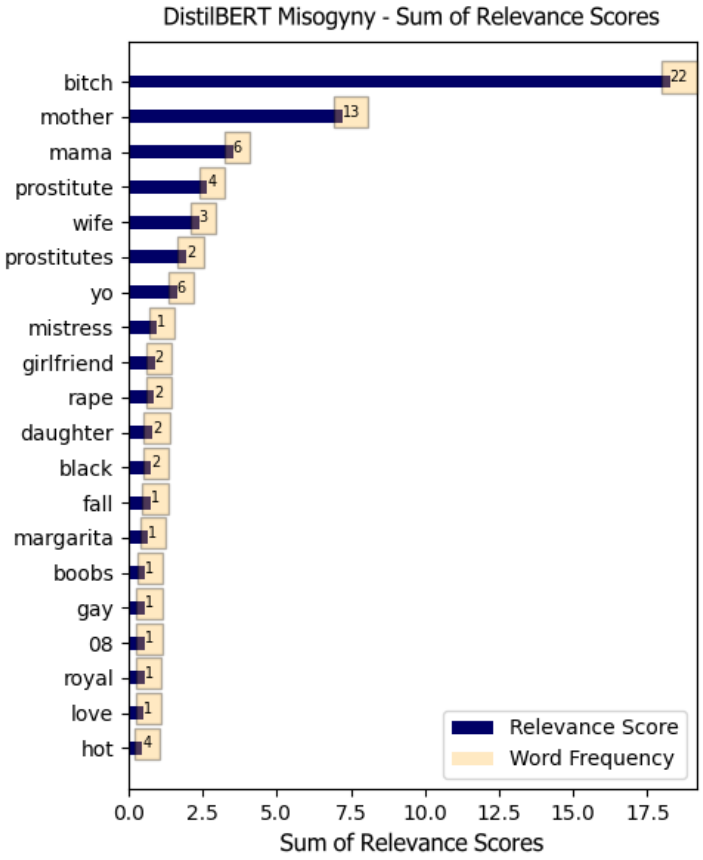


Figure C.9: The list with the sum of relevance scores for the top 20 words considering the DistilBERT model in the multi-label case for LaMAL. The label considered is MISOGYNY. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

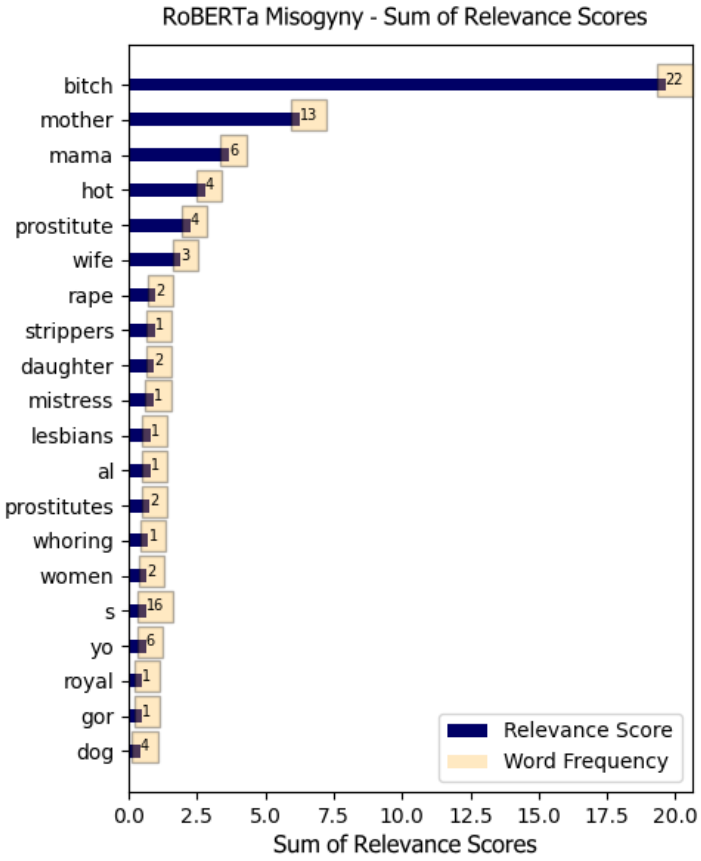


Figure C.10: The list with the sum of relevance scores for the top 20 words considering the RoBERTa model in the multi-label case. The label considered is MISOGYNY. Besides, we also present the frequency in which a word appears in the dataset used for training. The relevance score is calculated using IG (q.v. Section 2.5.2).

Appendix D

CAL - Sum of Relevance Scores for All Violation Classes

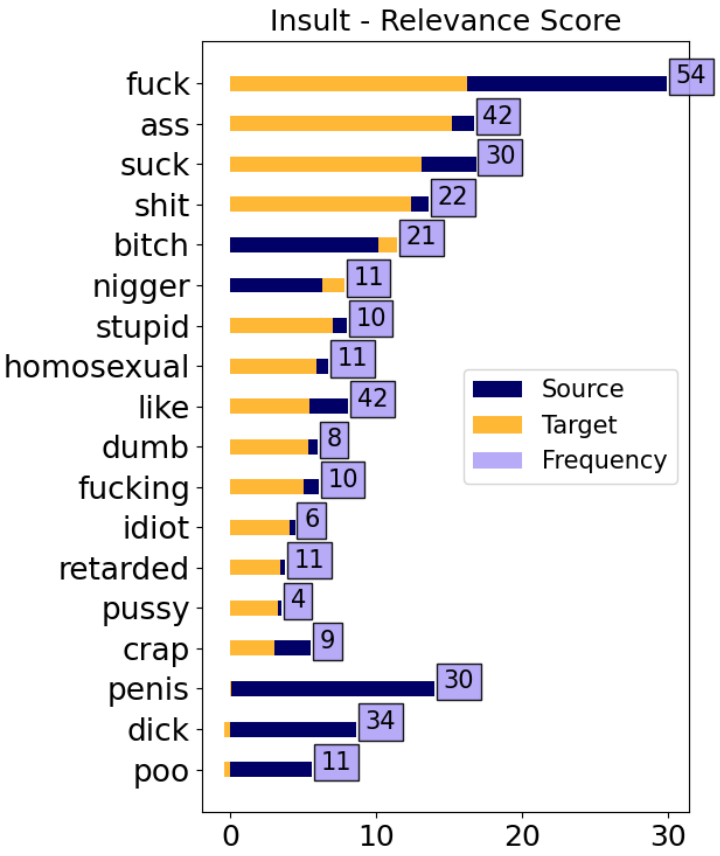


Figure D.1: The sum of relevance scores for the INSULT AND ABLEISM class. The source task refers to the model trained using the source community data, while the target task is the result after fine-tuning the model with Wikipedia data. "Frequency" refers to the number of occurrences of a word in the training dataset. Here, we present the increase (or decrease) of the relevant words for the violation classification. We show the difference between the two communities and how norm violations are defined.

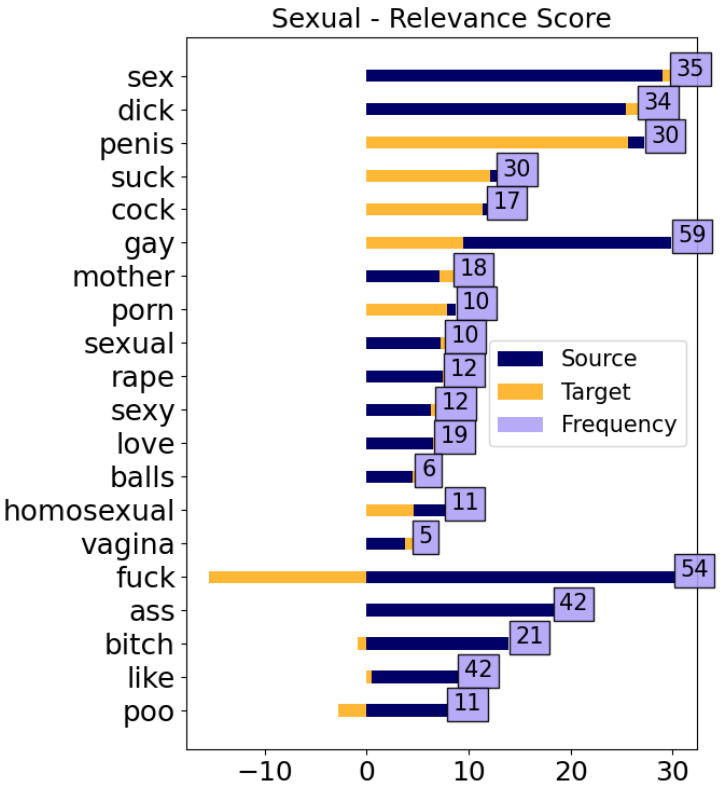


Figure D.2: The sum of relevance scores for the SEXUAL HARASSMENT class. The source task refers to the model trained using the source community data, while the target task is the result after fine-tuning the model with Wikipedia data. "Frequency" refers to the number of occurrences of a word in the training dataset. Here, we present the increase (or decrease) of the relevant words for the violation classification. We show the difference between the two communities and how norm violations are defined.

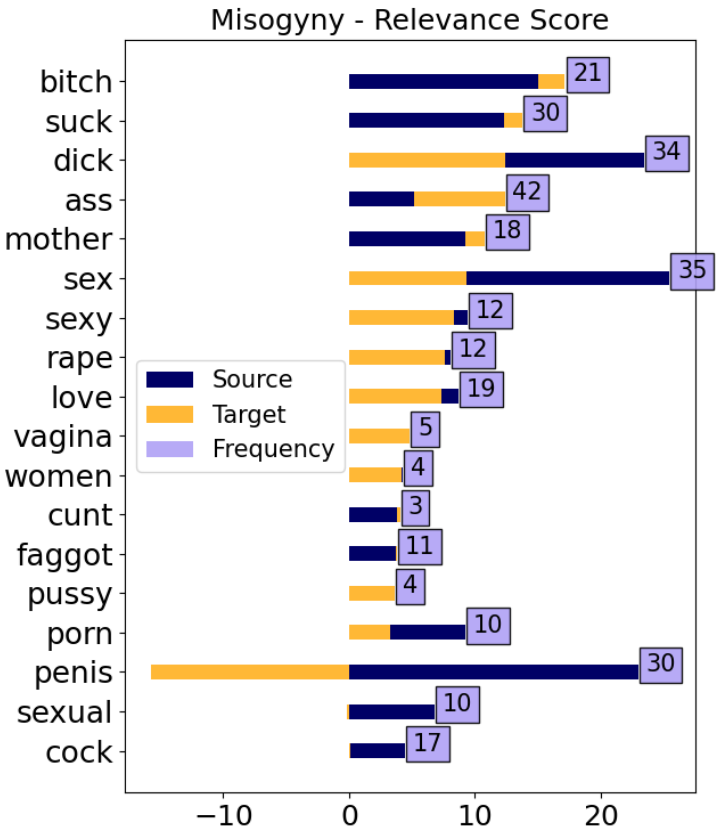


Figure D.3: The sum of relevance scores for the MISOGYNY class. The source task refers to the model trained using the source community data, while the target task is the result after fine-tuning the model with Wikipedia data. "Frequency" refers to the number of occurrences of a word in the training dataset. Here, we present the increase (or decrease) of the relevant words for the violation classification. We show the difference between the two communities and how norm violations are defined.

Appendix E

User Study - Sentences and Examples for Interpretability Layouts

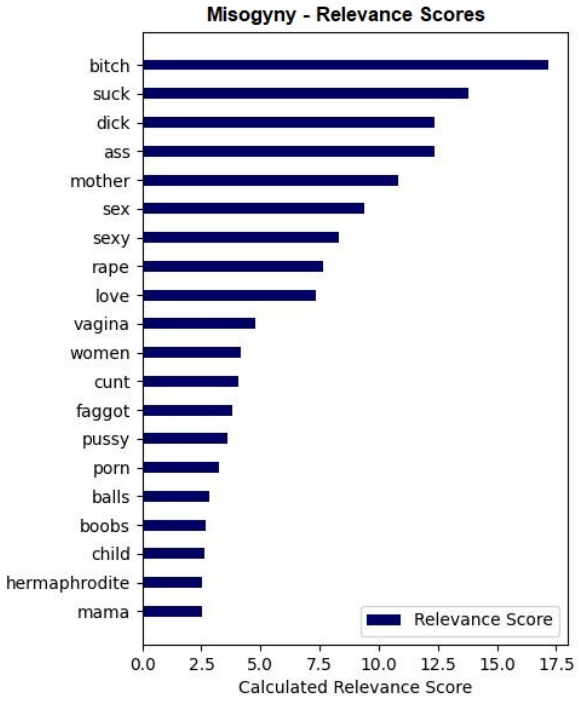
Misogyny
His daughter is hot. Hey, Massi.
She is very hot and I would love to give her one.
My stupid ex-girlfriend claims she was at this concert and everybody know she was not.
Jacob's step mama best blowjob.
Two of his royal strippers were Alana and Margarita.
Who was also employed previously as a prostitute.
Local ginger that has won the award of town bitch.
CENSORED in a woman's *CENSORED* and then *CENSORED* her.
I'm making it rain on them bitches.
I did your mother last night in every possible position.

Racism
What is up, "N-Word"?
Which as we call it black people music.
Fuck you all "N-Word."
All the "N-Word" are allowed to stay after school and smoke pot.
The big lipped hair-brained egotistical dirty "N-Word."
Linked to the troll organization gay "N-Word" association of america or gnna.
I hate "N-Word," I hate you black white bastard.
Small children or ass african american.
The american government to kill african americans second.
The toronto FC "N-Word."

Table E.1: All the 20 Sentences evaluated by the participants in the User Study, 10 for each violation class (Misogyny and Racism). We provide this table with some censored terms. It is possible to access the uncensored list of sentences in the following link:

Regarding the same sentence previously seen, now let us consider the information provided by the AI tool (list of relevance scores). After seeing this new information, what is your opinion on the following statement: The sentence below contains MISOGYNISTIC content.

Local ginger that has won the award of town bitch.



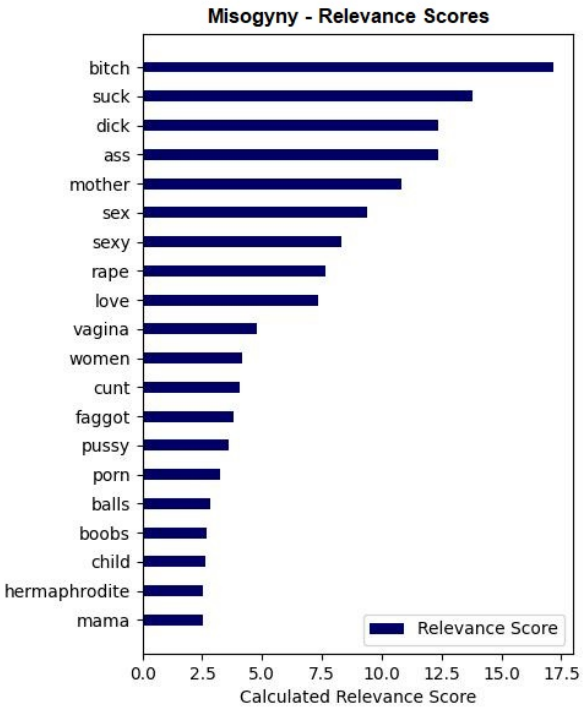
- ◯ Strongly disagree
- ◯ Disagree
- ◯ Somewhat disagree
- ◯ Neither agree nor disagree
- ◯ Somewhat agree
- ◯ Agree
- ◯ Strongly agree

Figure E.1: List with the sum of relevance score layout. This question inquires participants about their views on misogyny while considering a list with the sum of relevance scores. Participants must answer this question right after responding to the question about the same sentence with no interpretability data (q.v. Figure 6.2). Participants must choose whether they agree or disagree with classifying this text as misogynous using the 7-point Likert scale.

Regarding the same sentence previously seen, now let us consider the information provided by the AI tool (highlighted words and list of relevance scores). After seeing this new information, what is your opinion on the following statement: The sentence below contains MISOGYNISTIC content.

Legend: Misogynistic Neutral Not-Misogynistic

Local ginger that has won the award of town bitch.



- ◯ Strongly disagree
- ◯ Disagree
- ◯ Somewhat disagree
- ◯ Neither agree nor disagree
- ◯ Somewhat agree
- ◯ Agree
- ◯ Strongly agree

Figure E.2: Combined layout. This question inquires participants about their views on misogyny while considering the combination layout (highlighted words and list with the sum of relevance scores). Participants must answer this question right after responding to the question about the same sentence with no interpretability data (q.v. Figure 6.2). Participants must choose whether they agree or disagree with classifying this text as misogynous using the 7-point Likert scale.