Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (36^{th}cycle)

Inspecting Deep NLP Models

Innovative Techniques for Explainability, Fairness, and Concept Drift in Natural Language Classification

By

Salvatore Greco

******

Supervisor(s):
Prof. Tania Cerquitelli, Supervisor
Prof. Elena Baralis, Co-Supervisor

Doctoral Examination Committee:
Prof. Letizia Tanca, Referee, Politecnico di Milano, Italy
Prof. Ester Zumpano, Referee, Università della Calabria, Italy
Prof. Silvia Chiusano, Politecnico di Torino, Italy
Prof. Jérôme Darmont, Université de Lyon, France
Prof. Elisa Quintarelli, Università di Verona, Italy

Politecnico di Torino
2024

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Salvatore Greco
2024

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents,
and to Vanessa, my everything.

Acknowledgements

I would like to take this opportunity to thank all the people who have contributed to and supported me throughout my Ph.D. journey, which has enriched me both professionally and personally.

Firstly, I wish to express my deepest gratitude to my supervisor, Prof. Tania Cerquitelli, who was the first to believe in my potential as a researcher and guided me on my path to becoming one. I am truly grateful for the mentorship, patience, and encouragement she has shown me throughout this journey. Her insights and guidance have had a lasting impact on both my professional and personal development, inspiring me to strive for excellence in my work.

Secondly, I would like to extend my gratitude to Prof. Daniele Quercia for providing invaluable opportunities and mentorship during my two periods abroad. I greatly value the lessons he taught me, which have significantly contributed to my growth as a researcher.

Thirdly, I want to thank my family and friends for their unwavering support and encouragement of my choices. Their belief in me has been a constant source of strength throughout this journey, and I am truly grateful for their love and support.

I would also like to extend my heartfelt thanks to all the wonderful and brilliant colleagues, researchers, and professors I have had the fortune to meet at my university and during my time abroad. Each of them has contributed to my growth and understanding in unique ways.

Last but not least, I want to express my deepest gratitude to the most important person in my life, my partner, Vanessa. Among all the positive aspects that have made this doctoral journey truly unforgettable, the one I cherish most is that it brought you into my life. Your unwavering support, constant motivation, and encouragement have made this achievement possible, and for that, I am forever grateful.

Abstract

Natural Language Processing (NLP) is an interdisciplinary field of research that combines linguistics, computer science, and artificial intelligence, aiming to enable computers to understand and interpret human language.

In the last decade, NLP has experienced remarkable advancements, achieving performance levels previously unimaginable, with models even surpassing human performance in certain tasks. These advancements are driven by the development of innovative architectures like transformer-based models, as well as the increasing availability of computational resources and large datasets. However, despite their accuracy and capabilities, deep learning-based NLP models still face several challenges that limit their trustworthiness and responsible use in real-world applications.

Our research aims to face some of these challenges by designing innovative techniques for inspecting deep learning NLP classifiers to increase their transparency, fairness, and robustness. Specifically, we focus on three major challenges in NLP classifiers: (1) Explaining their predictions; (2) Identifying and mitigating bias; (3) Detecting performance degradation over time due to the presence of concept and data drift. To tackle these challenges, we design and propose three frameworks.

Firstly, we propose T-EBANO, a novel explanation framework designed to identify and measure the importance of words in the predictions made by NLP classifiers. This framework provides both local explanations for individual predictions and global explanations for overall model behavior, thereby enhancing the transparency and interpretability of NLP classifiers.

Secondly, we propose NLPGUARD, a framework aimed at mitigating the use of words related to protected attributes by NLP classifiers while maintaining their predictive capabilities. NLPGUARD ensures fairness by reducing reliance on protected attributes, identified automatically through Large Language Model (LLM) prompts, thereby making the approach both automatable and adaptable over time.

Thirdly, we propose DRIFTLENS, a novel drift detection framework designed to monitor the robustness over time of deep learning classifiers handling unstructured data, including NLP. This unsupervised methodology efficiently detects drift without requiring labeled data, making it suitable for real-time applications and enabling continuous monitoring of model performance in real-world production environments. Additionally, it identifies the labels most affected by drift, facilitating detailed characterization and explanation of the changes.

Our research aims to significantly advance the field of NLP by tackling crucial challenges related to transparency, fairness, and robustness in NLP systems. Through the development of innovative techniques for explaining NLP model predictions, mitigating bias, and detecting concept drift, our work aims to enhance the trustworthiness and responsible deployment of NLP technologies across various real-world applications. This effort not only enhances the performance and robustness of NLP systems but also addresses broader ethical considerations in artificial intelligence, fostering inclusivity, fairness, and accountability in algorithmic decision-making.

Disclaimer

This manuscript contains language and expressions that some readers **may find offensive**. These examples are included solely for research purposes and do not reflect the views of the authors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural Language Processing (NLP) is an interdisciplinary field of study that combines linguistics, computer science, and artificial intelligence. Its primary goal is to enable computers to understand, interpret, and generate human language in a manner that is both meaningful and contextually relevant.

Over the past decade, NLP has seen remarkable advancements [1], achieving levels of performance that were previously unimaginable. NLP models have even surpassed human-level performance in several tasks. These breakthroughs can be attributed to the development of innovative architectures, such as transformer-based [2] models, and the increasing availability of computational resources like GPUs. Additionally, the abundance of generated text data has significantly contributed to these advancements, allowing models to be trained on vast and diverse linguistic datasets.

Deep learning-based NLP classifiers are becoming an integral part of our daily lives. For instance, these classifiers play critical roles in applications ranging from sentiment analysis on online reviews [3], identifying toxicity and hate speech for online content moderation [4–6], and classification of job applicants' resumes [7, 8]. However, despite their accuracy and capabilities, several challenges still exist that limit their trustworthy and responsible use, and their robustness in real-world production applications.

Firstly, deep learning models are referred to as *black boxes* due to their inherent complexity and lack of transparency [9, 10]. The lack of transparency limits the applicability of such models especially in high-level risk and critical applications where users need to trust and interpret their decisions. Secondly, models can inherit

Table 1.1 **Motivation example.** Examples of toxicity predictions performed by a black-box classifier. $P(toxic)$ is the predicted probability of being toxic. Texts predicted as *toxic* ($P(toxic) \geq 0.5$) are in boldface.

Input text	$P(toxic)$
`"[Politician Name] is an idiot"`	**0.99**
`"[Politician Name] is an intellectual"`	**0.89**
`"The homosexual marriage bill will be debated soon! I am in favor!"`	**0.62**
`"I like this city! There are many white people!"`	0.01
`"I like this city! There are many black people!"`	**0.99**

and learn unintended biases present in the input data, leading to unfair behavior in their predictions. For instance, their predictions may exhibit discrimination based on protected attributes such as race, gender, and sexual orientation [11, 12]. Thirdly, despite achieving high performance on static training and test data, the performance and robustness of these models can be affected negatively over time by the dynamic nature of the newly processed data in real-world applications. This can lead to models' performance degradation (or decay) over time, thereby affecting their reliability and robustness [13, 14].

Consider, for example, a deep learning-based NLP classifier designed to identify toxicity—defined as inappropriate or harmful comments—on online social media platforms. Although this model may demonstrate high predictive performance when evaluated on a static dataset, several challenges can arise upon deployment in real-world applications. To illustrate some of these challenges, Table 1.1 presents examples of toxicity predictions for five input texts performed by a deep learning-based NLP classifier that we will analyze in the next chapters.

The first two sentences contain input texts that differ by a single word only.[1] However, they convey opposite sentiments: the former contains an offense towards the politician, while the latter is a compliment. If this classifier is used for automatic content moderation, both comments will be censored and banned from the platform since they are both predicted as *toxic*. However, the decision to censor the second comment is entirely unintended and unexpected. Nevertheless, the model operates as a black box, providing only the final output without explaining the reasons behind its classification. As a result, regular users may feel frustrated because they do not understand why their comments are censored, while expert developers may be

[1] The last name of a famous politician has been anonymized for privacy reasons.

unaware of potentially misleading behaviors of their platform. Therefore, it is crucial to provide explanations for predictions made by automated decision-making systems to enhance users' trust in such algorithms.

Providing explanations for the decisions of an NLP classifier can help non-expert individuals by (1) understanding the reasoning behind a particular decision, (2) offering a basis for challenging unfavorable decisions, and (3) clarifying what modifications could lead to a more favorable outcome in the future, according to the existing decision-making classifier [15]. Additionally, explanations can assist expert users, such as machine learning developers, in identifying potential wrong or unfair behavior exhibited by the classifier. For instance, as we will demonstrate in the following chapters, it is the politician's name that causes the model to classify the second text as *toxic*. This behavior is not limited to these specific examples but is indicative of a broader pattern learned by the classifier.

A similar issue arises with the third sentence. Although the user expresses a positive opinion in favor of same-sex marriage, the algorithm would still remove this comment, leaving the user unaware of the reason. Moreover, the last two sentences show texts where a single word related to a race-protected attribute is changed. The model behaves completely differently in these two cases: the former is considered *non-toxic*, while the latter is *toxic*. This indicates likely discriminatory behavior of the model, as it relies excessively on protected attributes for classification. It appears that the model has learned that certain identity words related to specific protected attributes correlate more strongly with toxic language. By following the principle of "*fairness through unawareness*" [11], which states that "*an algorithm is fair as long as any protected attributes are not explicitly used in the decision-making process*" [11, 16, 17], it is crucial to identify and mitigate the model's reliance on those words to avoid discriminatory outputs and ensure fairness in the decision-making process.

Finally, even if the concerns of transparency and discriminatory behavior are addressed, the model may still experience performance degradation over time. The underlying distributions of the comments on which the classifier is applied can change over time. For instance, new terms or phrases may develop connotations that were not present in the training data, or the context in which certain words are used may shift, altering their perceived toxicity. As a result, a toxicity classifier that was once accurate and reliable may become less effective, misclassifying harmful

comments as benign or vice versa. To mitigate the effects of this phenomenon, known as *concept drift*, it is crucial to implement mechanisms for continuous monitoring provide early warning when this phenomenon occurs, and possibly update the model. By doing so, the toxicity classifier can maintain high performance, reliability, and trust over time [13, 14].

1.1 Outline and contributions

This work aims to propose innovative techniques for inspecting NLP models to increase transparency and user trust, mitigate discriminatory outputs, and assess the model's robustness over time. Specifically, as motivated by the previous examples, this work aims to address the following three challenges:

- **Challenge 1:** *"Explaining the Predictions of NLP Classifiers"*.
- **Challenge 2:** *"Mitigating the use of Protected Attributes by NLP Classifiers"*.
- **Challenge 3:** *"Detecting Concept and Data Drift in NLP Classifiers"*.

Although these challenges are often addressed separately in the research community, as each represents a significant pillar of the Artificial Intelligence and Machine Learning literature, they all belong to the broader domains of *Trustworthy* and *Responsible AI*. These domains encompass a range of critical issues, including transparency, fairness, accountability, and ethical considerations in the development and deployment of AI systems. *Trustworthy AI* [18, 19] emphasizes the importance of creating AI models that are reliable, secure, and robust, ensuring that they behave as intended and can be trusted by users. This includes developing methods for explaining AI decisions to humans, addressing biases in data and algorithms, monitoring the reliability over time, and ensuring privacy and security. *Responsible AI* [20], on the other hand, focuses on the ethical implications of AI technologies, advocating for the responsible use of AI in society. This involves considering the social impact of AI, promoting inclusivity and accessibility, and ensuring that AI technologies are used in ways that align with human values and societal norms.

By addressing the previous challenges, this work aims to build NLP systems that are robust over time while also adhering to ethical standards, fostering trust, and

promoting the responsible use of AI across various applications and industries [21]. In addressing these challenges, we make three main contributions:

Addressing Challenge 1 (Chapter 2). We propose T-EBANO [22], a novel explanation framework for deep learning NLP classifiers. T-EBANO provides explanations of (1) individual predictions performed by NLP classifiers by identifying the most important predictive words used to classify an input text (*local explanations*), and (2) the overall importance of each word for each class label (*global explanations*). T-EBANO mainly differs from previous explainability techniques since it is specifically designed for NLP (*domain-specific*), and it exploits and analyzes the inner knowledge learned by the model to produce the explanations (*model-specific*).

We demonstrate the effectiveness and broad applicability of our approach across various NLP models and classification tasks. We also show that, by leveraging the internal knowledge of the models, our method generates explanations more efficiently, with increased precision and faithfulness than previous techniques that are *model-agnostic*–do not exploit the internal knowledge of the model. Finally, we show how the proposed explanations align well with human judgment.

Addressing Challenge 2 (Chapter 3). We propose NLPGUARD [23], a novel framework for mitigating the use of protected attributes of deep learning NLP classifiers. The framework ensures "*fairness through unawareness*" by reducing the model reliance on protected attributes while maintaining the predictive performance. NLPGUARD mainly differs from previous techniques for the different objectives. Instead of measuring performance disparity across demographic groups, its focus is on identifying and mitigating the use of protected attributes. Secondly, its identification of protected attributes is performed automatically through Large Language Models (LLMs) prompts instead of using pre-defined lists of identity terms, making the approach automatable and dynamic over time.

We assess the framework's sensitivity to its building components, its effectiveness, and its general applicability in reducing the reliance of NLP classifiers on protected attributes while maintaining predictive performance across several NLP classification tasks. We also demonstrate its superior effectiveness in obtaining such an objective than previous bias mitigation techniques.

Addressing Challenge 3 (Chapter 4). We propose DRIFTLENS [24–26],[2] a novel drift detection framework for deep learning classifiers working on unstructured data, including NLP. It is an unsupervised drift detection methodology; therefore, it does not require labeled data for detecting drift present in new data. It primarily differs from previous techniques in its efficiency, which makes it suitable for real-time drift detection for application domains handling large volumes of processed data. Additionally, it can characterize drift by identifying the labels that are most affected by the evolving concepts.

We demonstrate its superior accuracy and efficiency in detecting drift compared to previous unsupervised drift detectors across various classification tasks. We show that it is the most reliable technique, as it is not only the most accurate but also runs significantly faster across all experimental use cases. Additionally, we demonstrate its ability to model drift trends over time and characterize drift effectively.

Conclusion (Chapter 5). We conclude this work by envisioning the integration of the proposed frameworks into a unified inspection and monitoring system and discussing general future research directions.

1.2 Main Achievements

This section discusses the main achievements obtained by the Ph.D. candidate during the three-year Ph.D. program in terms of publications and formative periods abroad.

1.2.1 Published Papers and Research Contributions

The candidate is currently a co-author of 13 published articles, comprising 9 conference proceedings and 4 journals spanning several research topics, including Natural Language Processing, Explainable Artificial Intelligence, Fairness, Computer Vision, and Applied Machine Learning. Additionally, the candidate has contributed to another 3 papers currently at various stages of the review process. Below are the contributions broken down by research topic.

[2] Since the framework is agnostic to the data type, the pre-print version [26] showcases its latest results across various models and data types in NLP, computer vision, and speech domains. However, this work focuses solely on presenting results within the NLP domain. The pre-print version has recently been submitted to the *IEEE Transactions on Knowledge and Data Engineering* (TKDE).

Natural Language Processing (NLP). The candidate has mainly contributed to the NLP field. In particular, the candidate has contributed to proposing solutions to make NLP models more *trustworthy*, *fair*, *inclusive*, and *robust*.

Firstly, he proposed an *explainability* framework for NLP classifiers, which has been published in [22], and will be presented in this thesis in Chapter 2. Secondly, he significantly contributed to the topic of *fairness* in NLP classifiers by proposing a framework for mitigating the use of protected attributes by NLP classifiers, which will be presented in Chapter 3. A first-author research paper has been recently accepted for publication and will be published in the proceedings of the 27th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW) 2024 [23], and the pre-print version is available at [27]. This paper presents the results of two visiting periods at the Responsible AI group at Nokia Bell Labs in Cambridge, UK. Thirdly, the candidate has significantly contributed to proposing solutions to assess the *robustness* of NLP classifiers over time. To this end, he proposed a concept and data drift detection framework, on which a preliminary paper and a demonstration paper have been published in [24, 25]. The latest improvements and results related to the concept drift framework will be presented in Chapter 4, and have been recently submitted to the *IEEE Transactions on Knowledge and Data Engineering* (TKDE) for revision, of which the pre-print version is available at [26]. Since the framework is completely agnostic to the data type, in this paper [26], the framework has also been evaluated on other data types, such as images and speech.

The candidate has also significantly contributed to *inclusivity* and *fairness* in NLP with a project on NLP for multi-lingual inclusive communication. This work is a joint collaboration with linguistic experts. Four papers have been published in [28–31]. Moreover, a paper containing the latest results and advancements in NLP for inclusive language detection and reformulation in Italian administrative communication was submitted in October 2023 to the *ACM Transactions on Intelligent Systems and Technology* (TIST) journal and is currently under revision. Furthermore, another dataset paper containing high-quality data annotated in collaboration with linguistic experts will be submitted to an NLP journal shortly.

Finally, the candidate participated in an NLP team challenge involving the development of NLP models for the geolocation of linguistic variation in Italy, specifically the GeoLingIt shared task of EVALITA 2023. The team achieved 1st

place in Subtask A and 2nd place in Subtask B. The results and methodology have been published in [32].

Computer vision. The candidate has also made contributions in the computer vision domain. He completely redesigned and evaluated the explainability framework proposed in NLP [22] to explain convolutional models for image classifications. This work has been published in [33]. Moreover, the candidate has conducted research to propose deep learning-based and explainable solutions for earth observation, resulting in the publication of two papers [34, 35].

Applied machine learning. The candidate has made significant contributions to two industrial research projects in collaboration with Italian companies on applied Machine Learning. One of these projects resulted in a published paper [36].

As introduced before, this dissertation will present the candidate's contributions to the main topics of research addressed during the Ph.D. program, including explainability, fairness, and robustness in the NLP field.

1.2.2 Research and formative periods abroad

In the summers of 2022 and 2023, the candidate has been involved in two external research periods of three months each at the Responsible AI group of Nokia Bell Labs in Cambridge, UK. During these periods, the candidate has carried out research in the field of *responsible AI* and *fairness* in NLP, designing the mitigation framework that will be presented in Chapter 3. This research resulted in an accepted paper that will be published in the proceedings of the 27th ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW) 2024 [23].

The candidate has also conducted short training and formative periods at foreign universities. He has been selected and attended in person the prestigious NLP & Finance track of the Oxford Machine Learning Summer School in July 2023, organized by AI for Global Goals and in partnership with CIFAR and the University of Oxford's Deep Medicine Program. The candidate has also been selected and attended in March 2024 the ELLIS Winter School on Foundation Models organized by the University of Amsterdam and the European Laboratory for Learning and Intelligent Systems (ELLIS) Institute. Finally, the candidate has also been selected and will attend in July 2024 the Representation Learning & Generative AI and the Health & Bio tracks of the Oxford Machine Learning Summer School 2024.

Chapter 2

Explaining the Predictions of NLP Classifiers

This chapter first motivates the need for explanations and transparency in NLP automatic decision-making systems based on deep learning (Section 2.1), and it discusses the previous methodologies to explain deep learning classifiers focusing on NLP (Section 2.2). Next, it introduces T-EBANO, a framework able to explain individual predictions and provide global insights about the predictions of NLP classifiers (Section 2.3). Subsequently, it performs an extensive evaluation of its effectiveness in producing explanations (Section 2.4), and its general applicability (Section 2.5). Then, it performs an experimental comparison with model-agnostic XAI techniques (Section 2.6), and a human evaluation to assess the correlation of the proposed explanations with human judgment (Section 2.7). Finally, it discusses its implications, limitations, and future research directions (Section 2.8).

2.1 Motivation

In the last decade, the adoption of deep learning-based NLP models has grown exponentially. Transformer-based models, such as BERT [37], T5 [38], and GPT [39], have reached unprecedented levels of performance across a variety of natural language tasks, even exceeding human-level performance [40–43]. These advancements have pushed NLP forward, making it possible to process human language more effectively and efficiently.

Table 2.1 **Motivation example.** Examples of toxicity predictions performed by a black-box classifier. $P(\text{toxic})$ represents the predicted probability of being toxic. Texts predicted as *toxic* ($P(\text{toxic}) \geq 0.5$) are shown in boldface.

Input text	$P(toxic)$
`"[Politician Name] is an idiot"`	**0.99**
`"[Politician Name] is an intellectual"`	**0.89**

Although deep learning models are often highly accurate, they are very opaque since they behave as "black boxes" [10]. Given an input, deep learning models provide an output without offering any human-understandable insight into their inner decision-making process. This lack of transparency in the predictions hinders users from understanding why specific decisions are made, which is crucial for ensuring accountability and trust in AI systems, especially for critical applications. Moreover, the vast amount of data required to train these black-box models is typically collected from people's daily activities, such as online platforms, social media, and websites, thereby increasing the risk of inheriting human prejudices, racism, gender discrimination, and other forms of bias [44, 12]. Addressing these inherent biases is critical to developing AI systems that are equitable and fair, ensuring their responsible and ethical deployment across diverse societal contexts.

For these reasons, the field of *Explainable Artificial Intelligence* (XAI) aims to develop more trustworthy and reliable decision-making systems by explaining the predictions made by black-box models. XAI will soon become a design requirement in data-driven decision-making processes [45], including NLP applications. XAI has the potential to enhance human trust and foster social acceptance of these systems.

Providing an explanation to individuals impacted by the decisions of an automatic NLP classifier can serve multiple purposes: (1) it can clarify the reasoning behind a specific decision, (2) it can provide a foundation for contesting unfavorable decisions, and (3) it can suggest changes that might result in more favorable outcomes in the future, based on the current decision-making model [15]. Additionally, explanations can be valuable for expert users, such as researchers and machine learning developers, to detect and address potential errors or biases in the classifier.

Recall as a motivation example the toxicity classifier introduced in Chapter 1. In Table 2.1 we again report two examples of texts with the toxicity predictions.[1]

[1]The last name of a famous politician has been anonymized for privacy reasons.

These texts differ by only a single word, yet both are predicted as *toxic*, despite conveying opposite sentiments: the first contains an offense towards the politician, whereas the second is a positive comment. Therefore, based on this classifier, both comments will be censored and banned from the platform. Since the model only outputs the predicted label and probability, humans cannot understand the reasons behind these texts being predicted as *toxic*, and thus being censored. For instance, normal users of the platform may wonder: (1) "*Is it the politician's name that makes these texts inappropriate?*"; (2) "*Or are the words 'idiot' and 'intellectual'?*"; (3) "*Or perhaps the combinations of these words?*". Similarly, developers may be interested in answering the same questions to identify possible wrong behavior or bias in the model. Additionally, they may want to ask more general questions about the model's decision-making process, such as: "*Has the model learned general incorrect patterns, such as assuming that the presence of politician names automatically indicates toxic language?*". Adding an explainability layer is crucial in answering these questions.

In this chapter, we aim to explain the decisions made by NLP classifiers to enhance the transparency and interpretability of the decision-making process for humans. As we will discuss in Section 2.2, some explainability techniques that can be applied to NLP classifiers have been proposed. However, they are still affected by some limitations. (1) They are typically *domain-* and *model-agnostic*. Therefore, they do not exploit specific characteristics of the NLP domain or the latent semantic information learned by the classifiers when generating explanations. (2) These techniques typically perform perturbations on individual words to analyze their importance on predictions without considering the complex semantic relationships within texts. However, perturbing individual words can have a limited impact on predictions, especially in long texts, and can be computationally intensive, potentially compromising the quality of the explanations. (3) Most of these techniques can explain only individual predictions, without providing insights about the overall importance of words for the predictive model under analysis.

To overcome these limitations, we propose T-EBANO [22], a novel explanation framework for NLP classifiers. It provides explanations of individual predictions (*local explanations*) based on a perturbation process applied to groups of semantically related words, extracted using NLP *domain-specific* techniques and by mining the internal knowledge learned by the model (*model-specific*). Additionally, T-EBANO generates *global explanations* by assessing the per-class importance of words overall for the model, achieved by aggregating several individual explanations.

2.2 Related work

This section first provides background about Explainable Artificial Intelligence (XAI) and its taxonomy (Section 2.2.1). Then, it discusses current available XAI techniques for NLP and their limitations (Section 2.2.2).

2.2.1 Explainable Artificial Intelligence (XAI)

Explainable Artificial Intelligence (XAI) is the research field that aims to enhance the transparency and interpretability of AI systems by explaining or presenting the reasoning behind their decisions or technical processes in a way that is understandable to humans [9, 10, 46, 47]. Below, we present an overall taxonomy of the XAI field.

XAI techniques can be implemented at several stages of the AI pipeline: (i) *pre-modeling*, (ii) *in-modeling*, and (iii) *post-modeling*. *Pre-modeling* techniques aim to understand and preprocess data for subsequent model development while preserving interpretability. They include exploratory data analysis, data description and summarization, and interpretable feature engineering. *In-modeling* consists of adopting inherently explainable models, such as decision trees or linear models which are explainable by design–their decision can be interpreted by analyzing the weights or the paths of the tree. However, neither of these techniques is suitable for unstructured data, including text. Firstly, data exploration on data lacking fixed structures is much less interpretable. Secondly, inherently interpretable models are not suitable for tasks on unstructured data because they are typically much less performing (explainability vs. performance trade-off). In contrast, *post-modeling* techniques, named *post-hoc*, generate explanations for already trained models. These techniques are the most widely used in deep learning models, especially for unstructured data such as texts.

XAI techniques can also be categorized based on the general applicability of the method as (i) *model-agnostic*, and (ii) *model-specific*. *Model-agnostic* techniques can be applied to explain any model in the same way since they only analyze the inputs and outputs of the model. In contrast, *model-specific* techniques can be applied to specific models only. Moreover, XAI techniques can be classified based on the data domain of applicability as (i) *domain-agnostic* and (ii) *domain-specific*. *Domain-agnostic* techniques are applicable across various types of data (e.g., images, text, tabular data), while *domain-specific* techniques are tailored to particular domains,

such as NLP or computer vision, and leverage specific properties of the data to enhance interpretability.

XAI methods can be further categorized by the scope of the explanations produced as (i) *local explanations*, and (ii) *global explanations*. The former explains the reasons why a model produced an individual prediction for a given input. The latter attempts to explain the overall behavior of the model.

Explanations can be grouped based on their representation. (i) *Feature-importance* explanations indicate how much each input feature contributes to the prediction. (ii) *Rule-based* explanations generate local rules providing insights about the model predictions. (iii) *Example-based* explanations select or generate examples as explanations, such as prototypes (representative instances), counterfactual (minimally modified examples to change the prediction), or adversarial examples (intentionally perturbed examples to fool the model).

Finally, XAI techniques can be categorized based on how the explanations are generated. (i) *Perturbation-based* techniques add noise (e.g., simulating the removal or occlusion) to quantify the feature influence. (ii) *Gradient-based* techniques compute gradients to assess how changes in input features affect the model's outputs. (iii) *Counterfactual-based* techniques generate alternative instances with similar features but different predictions. (iv) *Local surrogate-based* techniques train an interpretable model to approximate the model behavior in the locality of a prediction.

2.2.2 Explainability in NLP

To date, many efforts have focused on explaining the prediction process in the context of structured data, for example by measuring quantitative input influence [48] or using local rules [49, 50], and image classification [51–53]. In contrast, less attention has been given to domain-specific explanation frameworks for NLP.

(i) Model-agnostic techniques. Model-agnostic techniques, such as [54–56], can explain individual predictions of any black-box model, even with unstructured inputs, including text. They are applicable across various types of data and models. Hence, they provide a versatile approach to explaining AI models. LIME [54] is a model-agnostic strategy that generates local explanations for any predictive model. It approximates the prediction using an interpretable surrogate model built locally around the specific data instance being predicted. However, while the interpretable model

provides useful local approximations, it may not faithfully represent the broader patterns and knowledge the original model has effectively learned. SHAP [56] is a comprehensive framework designed to interpret predictions from any machine learning model. It leverages a game-theoretic approach based on *Shapley Values* [57], where it iteratively evaluates the impact of feature removal on prediction outcomes.

Since these techniques are *model-agnostic*, they may not fully exploit the specific characteristics of the data domain and the latent semantic information learned by the models when generating explanations. They lack *inner-model awareness*, meaning they do not deeply explain what the model has specifically learned, thus they may provide less specific explanations. Moreover, in NLP, model-agnostic techniques analyze the impact of individual words on predictions without considering the complex semantic relationships within texts. Modern classifiers can learn complex semantic dependencies which these methods may overlook. Thus, perturbing individual words can have a limited impact on predictions, especially with long texts, and can be computationally intensive, potentially compromising the quality of explanations.

(ii) Domain-specific approaches. An exhaustive overview of existing XAI techniques specific to the NLP domain, and applied in various contexts, such as social networks, medical, and cybersecurity, is presented in [58]. Many studies utilize *perturbation* strategies to generate local explanations by analyzing model reactions, as seen in [59, 53, 60, 54, 61, 62]. However, this powerful approach requires careful selection of input features to be perturbed [63].

Unlike *model-agnostic* and *domain-agnostic* frameworks [54, 60], some *domain-specific* works have explored strategies to determine the information contained in the target model to select the most relevant features for perturbation. The feature extraction process is crucial in the explanation process, as the quality of the explanations depends on it. The authors in [62] propose using an approximate brute-force strategy to measure the impact of phrases in the input text on LSTM [64] model predictions based on its parameters. However, this approach is specific to LSTM models, making generalization difficult. The authors in [59] propose an explanation strategy for structured and sequential data models using a perturbation strategy based on a variational autoencoder. However, it is designed for sequence-to-sequence tasks like machine translation rather than classification. Moreover, training the variational autoencoder adds opacity and complexity to the explanation process. The authors in [65] introduce an encoder-generator framework to explain predictive

models alongside their training. This framework extracts an interpretable subset of inputs from the original text as a summary of the prediction process. Yet, it requires training an additional model to extract the explanations, making it less interpretable for end-users and increasing the complexity. In [61], a novel strategy based on reinforcement learning selects the minimal set of words to perturb, influencing the model's decision-making process. However, like previous methods, this approach involves training an external model to identify features for perturbation, thereby increasing complexity and impacting explanation reliability. CREX [66] integrates prior human knowledge into the training of deep neural networks, focusing on features highlighted by domain experts to enhance model relevance. Nevertheless, this highlighting process is time-consuming and impractical for already-trained models. LS-Tree [67] is a domain-specific game-theoretic technique using the Banzhaf value and parse trees to analyze NLP models comprehensively. It provides global insights into model behavior rather than explaining individual predictions and is particularly complex for longer sentences, suitable mainly for expert audiences.

Alternatively, gradient-based techniques utilize *gradients* to generate explanations [51, 68–70]. Grad-CAM [51] can highlight the important image regions in convolutional neural networks (CNNs). However, it is limited to the computer vision domain and not suitable for sequence models like RNNs or transformers, which are prevalent in NLP tasks. Grad-CAM-Text [70] is an XAI technique for 1D CNNs in text classification. However, it is not applicable to sequence and transformer models. Integrated Gradients [69] has been recently proposed for transformer models. However, gradients are computed over individual words, which can have a limited impact on the model's prediction. DeepLIFT [68] computes importance scores using gradients by explaining differences between input and 'reference' outputs, yet requiring prior knowledge for reference data. However, it is mainly tested on CNNs and is not suitable for sequence or transformer models.

(iii) Task-specific approaches. Not all tasks can be effectively explained using either model-agnostic or domain-specific approaches alone. Task-specific interpretable solutions are tailored to explain specific tasks, such as question-answering [71], recommendation [72], anomaly detection [73]. However, in this Chapter, we focus on explaining NLP classification models only.

In contrast to previous approaches, T-EBANO utilizes a feature extraction method *specific* for deep NLP classifiers, which leverages the specific knowledge

learned by the models. Supported by the interpretability properties of textual embeddings [74, 75], it utilizes the embedded representation of textual input data from the neural network's internal layers to identify correlated text segments crucial to the model's explanation process. This eliminates the need for external training resources. Moreover, it introduces features specific to textual data, such as full sentences and parts-of-speech. The influence of the extracted features is quantified through a quantitative index to generate local explanations. Finally, it generates per-class global explanations highlighting the most important words overall for each class by aggregating several local explanations

2.3 T-EBANO Framework

T-EBANO [22] stands for *Text-Explaining BlAck-box mOdels*, and it is a novel XAI framework designed to explain the predictions of deep learning-based NLP classifiers. It is a *post-hoc* (i.e., applicable to an already trained model), and *feature importance*-based methodology. Moreover, it is *model-specific*, thus requiring the model's architecture to be known. T-EBANO produces (1) *local explanations* by identifying the most important words used by the black-box model to predict a class label within an input text (Sections 2.3.1 and 2.3.2), and (2) *global explanations* by identifying the most important words influencing each class label overall (Section 2.3.3).

2.3.1 Local explanation process

The first main objective of T-EBANO is to explain the prediction within a sentence by identifying the most important words that influenced a particular prediction for an input text (*local explanation*).

Figure 2.1 shows the main steps of its local explanation process. For a given classification task, an input text is provided to the pre-trained deep learning model ①, which outputs the predicted *class label* ②. To explain the individual prediction, the local explanation process comprises three main steps. Firstly, T-EBANO extracts a set of *interpretable features*–a semantically related set of words– by utilizing either NLP techniques or by analyzing the knowledge embedded within the model itself ③. Secondly, it performs the *perturbation* of these interpretable features, and evaluates the model's predictions for these perturbed texts (perturbed probabilities)

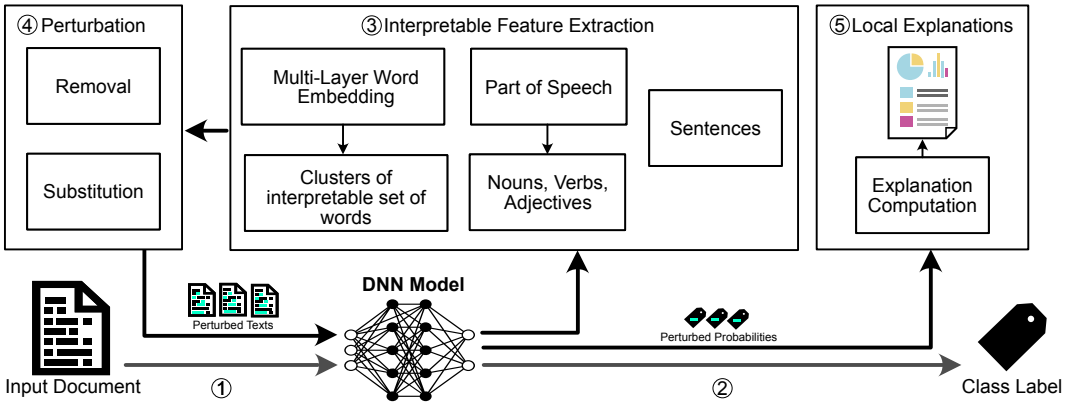


Fig. 2.1 **T-EBANO: Local explanation process.** Given an input text (1) and the predicted class label (2) produced by a black-box NLP classifier, the local explanation aims at identifying the most important words that influenced the prediction. It consists of three main steps. A feature extraction phase extracts semantically related sets of words (3). A perturbation phase introduces noise in correspondence with the features to measure their importance (4). A local explanation generation phase produces a report showing the interpretable features and their influence score (5).

to measure the impact and the importance of each feature ④. The significance of the difference in the prediction process before and after the perturbation is evaluated using a quantitative index designed to estimate the effect of the perturbation strategy. Subsequently, T-EBANO generates a *local explanation* report ⑤, presenting the results of the perturbation analysis through an informative dashboard.

Phase 1: Interpretable feature extraction

The interpretable feature extraction phase (Figure 2.1-③) identifies meaningful and semantically correlated sets of words (tokens) that influence the outcomes of the NLP model being explained. A set of words is considered meaningful for the model if altering it in the input text results in a significant change in the prediction outcome. Conversely, a set of words is meaningful for a user if they can readily comprehend and utilize it to aid in the decision-making process. T-EBANO comprises two *model-agnostic* (PoS, SEN) and one *model-specific* (MLWE) feature extraction strategies. However, all the strategies are *domain-specific* (i.e., specific for the NLP domain).

(PoS) Part-of-speech feature extraction. This strategy explores the semantic nuances of words by categorizing and evaluating the influence of different parts-of-speech (e.g., nouns, adjectives) in the prediction process. The rationale behind

this feature extraction approach is that the semantic contributions of various parts-of-speech may have differing impacts on the model's predictions since different tasks are typically affected by specific parts-of-speech (e.g., adjectives in sentiment analysis). To achieve this, input words are categorized into relevant groups using part-of-speech tagging, including adjectives, nouns, verbs, adverbs, and others. Each of these parts-of-speech is then extracted as a distinct feature.

(SEN) Sentence-level feature extraction. This strategy evaluates the influence of each sentence independently to assess its impact on the model's predictions. The basic idea behind this approach is to determine whether the model comprehensively processes each entire sentence and incorporates it into its decision-making process. Thus, T-EBANO extracts a distinct feature for each sentence within the input text.

(MLWE) Multy-layer word embedding feature extraction. This strategy leverages the internal knowledge learned by the model to extract interpretable features. In particular, it conducts an unsupervised clustering analysis to group related input words according to the internal representations (i.e., embeddings) assigned by the model. Each word group might have similarly influenced the model's prediction. The unsupervised analysis automatically determines the appropriate number of words to assign to each cluster and identifies the most influential cluster of words. This strategy requires access to the inner knowledge of the model being analyzed; hence, it is *model-specific*. Nevertheless, the process can be easily tailored to align with various deep architectures for NLP and their hidden layers. A more detailed explanation of the MLWE feature extraction strategy will be presented in Section 2.3.2.

Features combination. Separately for each feature extraction strategy, T-EBANO combines *pairwise* features (within the same strategy) to generate larger groups of tokens representing more comprehensive concepts. For example, the PoS strategy generates also features by combining parts-of-speech, such as adjectives with verbs, adjectives with nouns, and so forth. The SEN feature extraction generates a new feature for each pairwise combination of sentences. Finally, the MLWE generates a new feature with the pairwise combination of each cluster. T-EBANO generates pairwise combinations of features solely within the same feature extraction strategy and not across different ones.

Phase 2: Interpretable feature perturbation

Following the extraction of interpretable feature sets, the perturbation phase introduces noise in the correspondence of each feature extracted, thereby evaluating their impact on the model outcomes (Figure 2.1-④). We tested two perturbation strategies.

Removal perturbation. This perturbation strategy iteratively removes all the previously extracted interpretable features from the input text. Thus, it generates a new perturbed variation of the input text for every interpretable feature that is removed. The idea behind this is that the removal perturbation results in the absence of the concept associated with the removed words. If this concept was crucial in predicting the original class label, its absence should lead to a change in the prediction. Therefore, the perturbed variations of the input are subsequently fed back into the model under analysis, and the resulting predictions are collected and analyzed to assess the perturbation effect. For example, for MLWE features, each cluster of words is removed one at a time from the input text, producing a new perturbed text. Similarly, for PoS and SEN features, each removal of words belonging to a part-of-speech category or a sentence generates a new perturbed text.

Substitution perturbation. In T-EBANO, we also explored the *substitution perturbation*. While removal perturbation eliminates the concept associated with the removed words, substitution perturbation introduces a new, possibly related, concept that can alter the prediction. This involves an additional step to select new words to replace the current ones, such as *antonyms* or *synonyms*. Although this strategy proved very powerful in specific cases (e.g., Adjective-PoS perturbation), it also has several limitations: (1) Some words have multiple antonyms or synonyms, and the optimal choice is context-dependent; (2) Antonyms do not exist for certain words, such as nouns; (3) The choice of new words is task-specific. For example, antonyms are effective with opposite class labels like *positive* and *negative* in sentiment analysis, but not suitable for independent class labels as in topic detection. Thus, the effectiveness of this perturbation strategy is affected by these limitations. In this work, such implementations are out of scope because our aim is to design T-EBANO to be as general as possible across different classification tasks without requiring human expertise. We achieve this by using the *removal* perturbation. Therefore, we will only focus on the *removal* perturbation for our future experiments.

Interpretable feature importance measurement

The model's predictions for the perturbed texts (perturbed probabilities) are then evaluated to measure the impact of each feature. The perturbation of these interpretable features can influence the model's outcome in various ways:

1. The probability of the class under analysis *increases*. This indicates that the analyzed features were negatively impacting the prediction process.
2. The predicted probability *decreases*. This suggests that the perturbed features were positively impacting the class under analysis. Thus, they were important for the original prediction.
3. The predicted probability *remains roughly unchanged*. This implies that the perturbed portion of the input is irrelevant for the prediction.

To measure the significance of the perturbation impact, T-EBANO utilizes an enhanced version of the quantitative index introduced in [53] which addresses the concerns of *asymmetry* and *unbounded values*. This quantitative index, known as the *normalized Perturbation Influence Relation* (*nPIR*), aims to assess and measure the influence of each extracted interpretable feature. The underlying idea is to assess and quantify the importance of an input feature in a particular prediction by analyzing its probabilities before and after perturbing the feature extracted from the input data.

Formally, consider a classifier capable of distinguishing among a set of classes C. Let $c_i \in C$ denote the *class-of-interest* for which the local explanation needs to be computed (i.e., the prediction for the class to be explained). The explanation process, when provided with the input sample x_i, involves extracting the set of interpretable features F from x_i. For each feature $f \in F$, a perturbation strategy is applied, and the resulting impacts on the predictive model outcomes are assessed. These perturbed outcomes serve as indicators of the contribution of f to the prediction process. We quantify the influence of f on c_i using the *nPIR* index.

Let p_{o,c_i} represent the output probability of the original input x_i (i.e., the original unperturbed input) belonging to the class-of-interest c_i, and p_{f,c_i} denote the probability of the same input, with the feature f perturbed, belonging to the same class. We consider the predicted class distributions such that $\sum_c^C \mathbb{P}o,c = 1$ and similarly $\sum_c^C \mathbb{P}_{f,ci} = 1$. For example, if the model output is obtained through a SoftMax layer.

We introduce a generic definition of the influence relation for a feature f by combining the outcomes of the model p_{o,c_i} and p_{f,c_i} before and after the perturbation process. The desired influence relation, denoted as the *nPIR*, should range within the interval $[-1,1]$. An *nPIR* value for f close to or equal to 1 indicates a *positive* relevance of the concept in f on the prediction of class c_i. Conversely, an *nPIR* value for f close to or equal to -1 indicates a *negative* impact of that feature on the prediction of class c_i. An *nPIR* value close to 0 implies that f is *neutral* with respect to the prediction of class c_i. To achieve this, the *nPIR* combines two sub-indicators: (1) the *Amplitude of Influence* (ΔI) and (2) the *Symmetric Relative Influence* (*SRI*).

(1) Amplitude of Influence (ΔI). The amplitude of influence ΔI for a feature f is defined as the difference of the model's probabilities for the class of interest c_i before and after perturbation of the feature f (Equation 2.1).

$$\Delta I_f = p_{o,c_i} - p_{f,c_i} \tag{2.1}$$

ΔI ranges in $[-1,1]$ since the domain for probability values is within $[0,1]$. $\Delta I_f > 0$ indicates a *positive* influence of the feature f for class c_i since the perturbation of the corresponding portion of input causes a *decrease* in its probability of belonging to the class of interest. Therefore, f is relevant for class c_i. In contrast, $\Delta I_f < 0$ represents a *negative* influence of the feature f for c_i.

However, the amplitude alone does not completely reflect the overall contribution of f. The absolute distance between two values can be small if both values are small with respect to the probability values domain, but their relative distance can still be significant. Therefore, we also consider the relative influence of f.

(2) Symmetric Relative Influence (*SRI*). To capture the relative influence of f, a straightforward approach would be to compute the ratio between the probabilities. However, as shown in [53], this score is asymmetric: the ratio $\frac{p_{o,c_i}}{p_{f,c_i}}$ will range in $[0,1]$ for negative influence and in $[1,\infty]$ for positive influence. Thus, it will be difficult to quantitatively compare positive and negative influences. To overcome this problem, we define the *Symmetric Relative Influence* (*SRI*) for a feature f (Equation 2.2). This index evaluates the relative influence that f has over p_{o,c_i} and p_{f,c_i}. The symmetry of this score allows us to measure the relative influence of the feature f before and after the perturbation, regardless of whether it is positive or negative.

$$SRI_f = \frac{p_{o,c_i}}{p_{f,c_i}} + \frac{p_{f,c_i}}{p_{o,c_i}} \tag{2.2}$$

Normalized Perturbation Influence Relation (*nPIR*). By combining Equations 2.1 and 2.2, we define the *Perturbation Influence Relation* (*PIR*) for f in the range $[-\infty, +\infty]$. We then apply the *Softsign* function [76] to obtain a linear approximation of the influence near 0 and to normalize the values in the $[-1, 1]$ range. Thus, the final *nPIR* of a feature f for a class of interest c_i is defined in Equation 2.3.

$$\begin{aligned} nPIR_f(ci) &= softsign(\Delta I_f * SRI_f) \\ &= softsign(p_{f,c_i} * b - p_{o,c_i} * a) \end{aligned} \tag{2.3}$$

where:

$$a = 1 - \frac{p_{o,c_i}}{p_{f,c_i}}; \quad b = 1 - \frac{p_{f,c_i}}{p_{o,c_i}} \tag{2.4}$$

The coefficient a represents the contribution of the original prediction for input o with respect to the perturbed input. Similarly, b represents the contribution of the perturbation of f with respect to the original feature.

The higher the $nPIR_f$ (approaching 1), the more *positively influential* the feature f is for the class of interest, indicating its importance in predicting that class. Conversely, the lower the $nPIR_f$ (approaching -1), the more *negatively influential* the feature f is for the class of interest. A *nPIR* close to 0 indicates a feature with *neutral influence*.

Phase 3: Local explanation report generation

To generate the local explanation report (Figure 2.1-⑤), T-EBANO computes the *nPIR* index for all the extracted features by assessing the model's outputs for the original input text and its perturbed variants. Each explanation provides the extracted feature highlighted in the input text (*textual explanation*) with the corresponding influence (importance) on the original prediction (*quantitative explanation*).

Users can query the *local explanation report* to filter by feature extraction strategy, class of interest, and influence. Moreover, users can define thresholds to extract the

most and the least influential features having larger or smaller importance in the original prediction performed by the black-box classifier.

Local explanation example. Figure 2.2 and Table 2.2 show an example of a local explanation for a BERT [37] classifier trained for *sentiment analysis* that we will analyze in the next sections. The original input text is reported in Figure 2.2a. The underlying sentiment is predicted as *negative* by BERT with 0.99 probability. However, the BERT model behaves as a black-box, it provides the predicted class label without any information about the reasons that led to this classification. T-EBANO can help in understanding the reason why this text has been predicted as *negative* by identifying the most important words in the predictive process.

We report a subset of the interpretable features extracted by T-EBANO. They are highlighted in cyan in the input text in Figure 2.2. The strategy used to extract the feature, the label before and after the perturbation, and the influence of the feature (*nPIR*) are reported in Table 2.2. Features with high influence ($nPIR \geq 0.5$) are in boldface in the last column. Notice that, here we reported only highly influential features and some natural features for discussion. However, the local explanation report allows to query of all the extracted features.

Perturbing the PoS-Adjectives in Figure 2.2b (EXP1) or the MLWE cluster in Figure 2.2d (EXP3), results in an *nPIR* very close to 1. This indicates that these sets of features are very relevant to the model's outcome. Removing any of these features will significantly alter the model's predictions, changing the outcome from *negative* to *positive*. Conversely, the perturbation of the sentence in Figure 2.2c (EXP2) is not relevant for the model, as indicated by the *nPIR* value of 0. The information contained in the sentence {`This film was very awful`} alone does not justify the model outcome, as the rest of the text also contributes to the prediction. We can conclude that the feature sets {`awful, bad`} and {`was, awful, bad, movie`} are the real reason why the model is predicting the *negative* class.

The explanations generated through the substitution perturbation (EXP4, EXP5) are also reported in Table 2.2. These examples show that substitution perturbation has great potential when suitable antonyms can be found. For the PoS-Adjective substitution (EXP4), the quantitative explanation shows a *nPIR* value close to 1. Conversely, in EXP5, verbs are replaced with semantically incorrect words (not antonyms) within the context of the phrases, showing no impact on the prediction

```
This film was very awful.  I have never seen such a bad movie.
```

(a) Original text

```
This film was very awful.  I have never seen such a bad movie.
```

(b) `EXP1`: Adjective - POS feature extraction with removal perturbation.

```
This film was very awful. I have never seen such a bad movie.
```

(c) `EXP2`: Sentence feature extraction with removal perturbation.

```
This film was very awful.  I have never seen such a bad movie
```

(d) `EXP3`: Multi-layer word embedding feature extraction with removal perturbation.

```
This film was very [awful] nice.  I have never seen such a [bad] good movie
```

(e) `EXP4`: Adjective-POS feature extraction with substitution perturbation.

```
This film [was] differ very awful.  I [have] lack never seen such a bad movie.
```

(f) `EXP5`: Verb-POS feature extraction with substitution perturbation.

Fig. 2.2 Example of *textual explanation* report. The original text was classified by a BERT model trained for sentiment analysis as *negative* with a probability of 0.99. The most relevant features are reported and highlighted in cyan. Removed tokens for the substitution perturbation are in squared brackets and followed by the newly inserted tokens.

Exp. ID	Feature extraction strategy	Predicted label	Perturbed label	nPIR
EXP1	PoS-Adjective	*Negative*	*Positive*	**0.998**
EXP2	Sentence	*Negative*	*Negative*	0.000
EXP3	MLWE	*Negative*	*Positive*	**0.984**
EXP4	PoS-Adjective (sub.)	*Negative*	*Positive*	**0.999**
EXP5	PoS-Verb (sub.)	*Negative*	*Negative*	0.000

Table 2.2 Quantitative explanation for the example in Figure 2.2. The original predicted label and the one predicted after perturbation are reported, along with the *nPIR* index. The (sub.) suffix indicates that the substitution perturbation has been applied. Otherwise, the removal perturbation has been applied.

process with a *nPIR* equal to 0. However, as discussed in the previous sections, this perturbation strategy remains an open task for further research in future works.

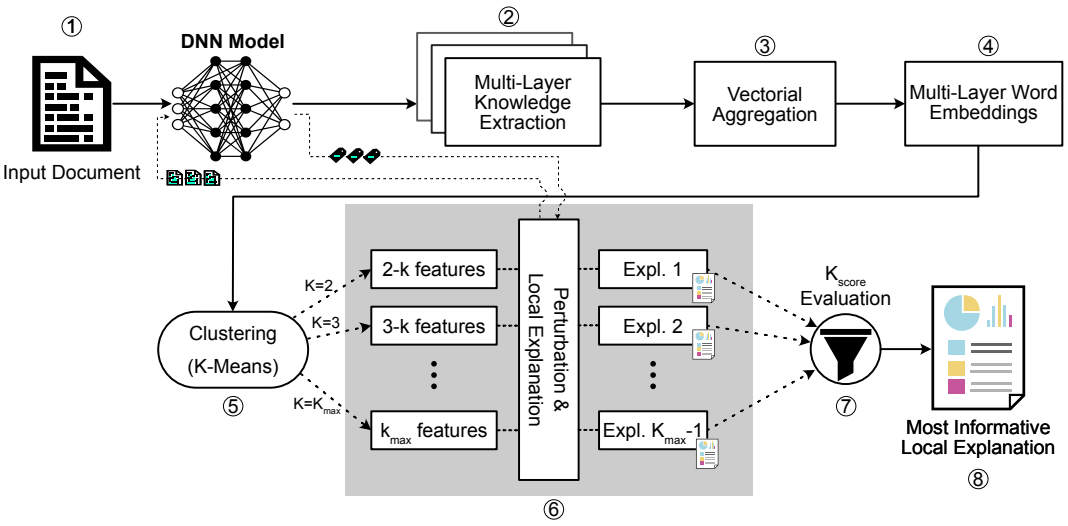


Fig. 2.3 **MLWE feature extraction process.** It extracts a set of features comprising semantically related words by mining the internal representations of the NLP model. Given an input text (1), it extracts the embeddings from multiple internal layers (2). It aggregates the multiple layers and the representation of subwords (3) into a single vector for each entire word (4). It performs an unsupervised clustering analysis for several possible partitions to group similar words (5)-(6). The best partitioning of words is reported as the final explanation (7)-(8).

2.3.2 Multi-layer Word Embedding feature extraction details

We now detail the steps of the *Multi-Layer Word Embedding* (MLWE) feature extraction strategy, which involves recurrent feature extraction and perturbation steps. Then, we discuss how MLWE can be implemented for current NLP architectures.

MLWE feature extraction process

Deep learning-based NLP models are trained to extract knowledge from training data, learning a complex numerical model distributed across multiple hidden layers. Each of these layers contributes to the final prediction. For a reliable explanation of the model's prediction, it is crucial to mine the knowledge embedded within all these layers. The MLWE feature extraction enables T-EBANO to achieve this objective. By analyzing the outputs of multiple hidden layers, MLWE extracts the numerical representation of the input at different network levels (i.e., the embeddings). The overall MLWE process is illustrated in Figure 2.3.

Embedding knowledge extraction and aggregation. Given an input text ①, a tensor containing numerical embedding representations of all tokens across various layers is extracted ②. As such, each token is represented with a multi-dimensional vector with a shape equal to the number of layers and the embedding dimensionality. Then, a *vectorial aggregation* step is performed ③. Firstly, the multi-dimensional embeddings are aggregated—typically through methods like averaging, summing, or concatenating—along the layers' axis to create a one-dimensional vector representation for each token. Secondly, in the case of modern transformer-based architectures that tokenize the input text into sub-words, an additional aggregation step reconstructs entire words' representations from sub-word tokens by averaging their embedding. This is because our objective is to measure the importance of entire words. Finally, these aggregated representations undergo dimensionality reduction through PCA. These outcomes form the MLWE representation ④ of the input text, where each entire word is represented by a compact and dense vector that encapsulates the meaning and learned knowledge of the model. The MLWE embeddings' extraction and aggregation from multiple layers, is approached differently depending on the architecture being used.

Unsupervised embedding analysis. Once the MLWE representations are extracted and aggregated, they undergo an *unsupervised clustering* analysis ⑤ to identify sets of correlated words that exhibit common behaviors within the model under examination. The objective of this unsupervised analysis is to identify the smallest groups of input words exhibiting the greatest influence on the model's predictions. The underlying idea is that words with similar MLWE representations are highly correlated according to the model. Grouping such tokens together can reveal key semantically related input concepts influencing the prediction.

To achieve this goal, T-EBANO utilizes the K-Means [77] clustering algorithm, chosen for its effective performance in similar contexts [53] and balanced computational efficiency. A crucial aspect when employing K-Means is determining the optimal number of clusters K that best captures meaningful subsets of the data. T-EBANO iteratively applies K-Means multiple times varying the number of clusters k within the range $[2, K_{max}]$. The K_{max} value can be potentially set up to the number of words in the input text. However, when the number of tokens n_{tk} in a text is very high, it becomes impractical and less useful to evaluate clustering with values of K as large as n_{tk}. Conversely, using small fixed values of K with large input texts would result in large clusters encompassing both influential and less impactful

words, leading to explanations that lack specificity and precision. To strike a balance between partition size and precision, T-EBANO evaluates the number of clusters K as a function of the number of input tokens n_{tk} as follows:

$$K_{max} = \sqrt{n_{tk} + 1} \tag{2.5}$$

This approach reduces the search space while maintaining the quality of the features extracted. T-EBANO produces the explanations and computes the *nPIR* index (introduced in Section 2.3.1) for each feature (cluster) extracted for each K partitioning tested ⑥. Specifically, for each value of $K \in [2, K_{\max}]$, T-EBANO analyzes K perturbations. Each perturbation generates a new version of the input text by perturbing the tokens within the current cluster. It then evaluates the model's outcomes when presented with these perturbed texts, producing the *nPIR* index for each cluster perturbation within each K (shown as dotted lines in ⑥). This approach generates a comprehensive set of potentially valuable *local explanations*.

Most informative local explanation evaluation. The goal is to deliver the optimal explanation to the end-user. T-EBANO identifies the *most informative local explanations* as those that extract the most valuable insights from the model's behavior for a single prediction. We define the *most informative local explanation* as the feature containing the smallest groups of words exhibiting the greatest influence on the model's predictions. To achieve this, T-EBANO first assigns a *feature informativeness score (FIS)* to each feature (i.e., each cluster of words), as follows:

$$FIS(\kappa) = \max\left[\left(\alpha(nPIR_\kappa) + \beta(1 - \kappa_{tk}/n_{tk})\right), 0\right] \tag{2.6}$$

Where κ represents the current cluster, κ_{tk} denotes the number of tokens within cluster κ, n_{tk} denotes the total number of tokens, and $nPIR_\kappa$ is the influence score of the current cluster κ, which measures the positive or negative impact of perturbing the tokens in κ (as discussed in Section 2.3.1). The ratio κ_{tk}/n_{tk} represents the percentage of tokens within the cluster compared to the total number of tokens. The $FIS(\kappa)$ score aims to *maximize the influence* of the feature (*nPIR*), and *minimize the size* of the feature κ_{tk}/n_{tk} obtained by maximizing $(1 - (\kappa_{tk}/n_{tk}))$.

The hyperparameters α and β represent the *weights* assigned to the *nPIR* and the tokens ratio $(1 - (\kappa_{tk}/n_{tk}))$, respectively. They determine the relative contribution of

the *influence* of the feature and its *size* in the feature informativeness score $FIS(\kappa)$. As a default, we assigned a weight of 0.60 to the *influence* ($\alpha = 0.60$) and 0.40 to the *size* of the features ($\beta = 0.40$). This decision reflects the priority of selecting influential features, with a secondary consideration of minimizing the number of tokens. Selecting small-size features that lack influence would provide little value. An experimental sensitivity of the α and β hyperparameters is performed in [22].

We recall that the *nPIR* index ranges in $[-1,1]$, where 1 indicates a very high positive influence for the class of interest. The range of $(1-(\kappa_{tk}/n_{tk}))$ is [0,1]. Therefore, the *feature informative score* FIS, with $\alpha = 0.60$ and $\beta = 0.40$ (or any values of α and β whose sum is 1), falls within the range [0,1]. Negative values are undesirable since we seek positively influential features for the class of interest. A $FIS = 0$ indicates a feature whose size is close to 100% of tokens and whose influence is close to 0. Conversely, a $FIS = 1$ indicates a feature with very few tokens (e.g., less than 1%) and a highly influential score close to 1. The higher the FIS score, the more influential and precise the feature is.

Once computed the FIS score for each feature within each value of K (i.e., each possible partition analyzed), a final score for each K partitioning is computed ⑦ by selecting the maximum FIS score among all clusters of words.

$$\begin{aligned} K_{score} &= \max_{\kappa \in K} \Big(FIS(\kappa) \Big) \\ &= \max_{\kappa \in K} \Big(\max \big((\alpha(nPIR_{\kappa}) + \beta(1-\kappa_{tk}/n_{tk})), 0 \big) \Big) \end{aligned} \tag{2.7}$$

The value of K that yields the highest K_{score} is selected as the best. Consequently, K clusters of words are formed, where each $\kappa \in K$ represents a feature set containing both neutral or negatively influential features (i.e., $nPIR \leq 0$) and typically one highly influential feature. Ultimately, the cluster κ with the highest $FIS(\kappa)$ score is identified as the *most informative local explanation* ⑧.

MLWE example. Table 2.3 shows the analysis conducted by the MLWE on a short input text consisting of 9 tokens. The text was predicted as *positive* by an NLP classifier trained for sentiment analysis with high probability (99%). The column K represents the different numbers of cluster partitions analyzed by the MLWE strategy. Each cluster $\kappa \in K$ is denoted as $K.k$, where, for example, 2.1 is the first cluster of the division with $K = 2$. k_{tk} represents the number of tokens inside the cluster, and

Table 2.3 **MLWE example.** Example of *most informative local explanation* (*cluster 3.1*) and *best K partitioning* ($K = 3$) selection using MLWE for an input text with 9 tokens and predicted as *positive* by an NLP model fine-tuned for sentiment analysis

K	k	MLWE potential features	k_{tk}	k_{tk}/n_{tk}	nPIR	FIS
2	2.1	Yesterday I saw a movie that positively surprised me	3	3/9	0.990	0.861
2	2.2	Yesterday I saw a movie that positively surprised me	6	6/9	0.001	0.134
3	3.1	Yesterday I saw a movie that positively surprised me	2	2/9	0.999	**0.911**
3	3.2	Yesterday I saw a movie that positively surprised me	4	4/9	0.001	0.228
3	3.3	Yesterday I saw a movie that positively surprised me	3	3/9	0.000	0.267

k_{tk}/n_{tk} denotes the ratio of tokens in the cluster to the total number of tokens. *nPIR* and *FIS* refer to the influence score and the feature informative score obtained by cluster k, respectively. The tokens of each cluster are highlighted in cyan.

The partitions analyzed by MLWE are $K \in [2,3]$, since $K_{max} = \sqrt{9+1} \approx 3$. In the first partition ($K = 2$), two clusters of tokens are identified: *cluster 2.1* containing 3 tokens and *cluster 2.2* containing 6 tokens (highlighted in cyan). The current *most informative local explanation* is *cluster 2.1* because it achieves the highest *FIS* score among the clusters of $K = 2$. Subsequently, T-EBANO proceeds to analyze the clustering results with $K = 3$. The current *most informative local explanation* is *cluster 3.1*, which obtains the highest *FIS* score among the clusters of $K = 3$. Overall, *cluster 3.1* has a higher FIS score than *cluster 2.1* ($0.911 > 0.861$). Consequently, $K = 3$ is selected as the optimal partitioning value, and *cluster 3.1* is identified as the final *most informative local explanation*.

MLWE model-specific implementation

Given that the MLWE implementation is *model-specific*, we now discuss how it can be implemented for state-of-the-art deep learning NLP classifiers.

Recurrent models. Recurrent Neural Networks (RNNs) using Gated Recurrent Units (GRU) or Long Short-Term Memory (LSTM) [64] are powerful architectures that can

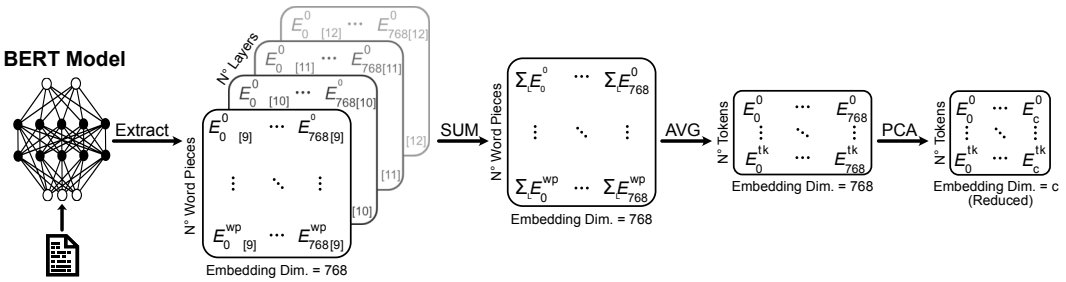


Fig. 2.4 **MLWE implementation for BERT.** E is the word embedding matrix, wp and tk indicate the position of the word-piece and token respectively in the input text, 768 is the original embedding dimension, c is the number of reduced principal components of the word embedding vector and L_{id} is the extracted layer.

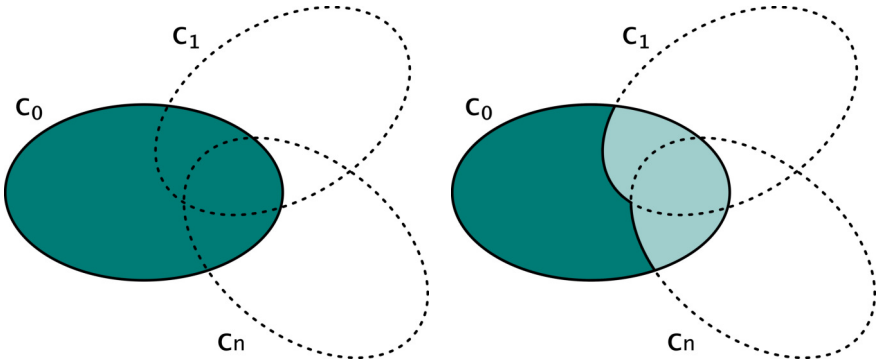
learn across both sequential time dependencies and feature vectors. In classification tasks, these models take as input an embedded representation of each word. They are characterized by multiple layers and bidirectional recurrence, allowing them to capture complex dependencies in both forward and backward directions.

In these models, the MLWE can be implemented in two ways by extracting: (1) the single-layer word embedding only, or (2) multiple bidirectional recurrence layers. In the former case, the aggregation of multiple layers can be skipped. In the latter case, the aggregation can be performed by concatenating or averaging, for example. For these models, it is unnecessary to reconstruct the representation of entire words since they do not perform subword tokenization.

Transformer-based models. Transformers [2] have revolutionized the NLP field by proposing architectures that effectively capture contextual relationships between words without the need for recurrent mechanisms. They exploit attention mechanisms to encode dependencies between words across an entire sequence, allowing them to process information in parallel and handle long-term dependencies efficiently. Several variants exist suitable for different tasks, such as encoder-only, decoder-only, or encoder-decoder. In classification tasks, the most widespread are encoder-only. They leverage self-attention mechanisms—which enable the model to evaluate the significance of each word in a sequence relative to others—to generate robust contextual embeddings, thereby enhancing the model's ability to accurately classify inputs based on learned patterns. *Bidirectional Encoder Representations from Transformers* (BERT) [78], and its variants, such as ALBERT (*A Lite BERT*) [79], DistilBERT (*a distilled version of BERT*) [80], and RoBERTa (*A Robustly Optimized BERT Pretraining Approach*), are examples of encoder-only architectures.

In encoder-transformer models, the MLWE can be implemented similarly by extracting the embedding representations of each word across multiple latest transformer layers. This is because previous work [75] showed that modern architectures encapsulate most of the context-specific information in their final and deepest layers. Figure 2.4 shows an example of the MLWE implementation for a BERT-base model. However, it can be implemented in the same way across all BERT variants. BERT-base comprises 12 transformer layers, each generating an output of shape $(wp \times 768)$, where wp denotes the number of word pieces (sub-words) extracted from the input text. In Figure 2.4, the MLWE extracts word embeddings from the last four transformer layers (L_9, L_{10}, L_{11}, L_{12}), resulting in a tensor of shape $(wp \times 768 \times 4)$. This enables capturing task-relevant features, balancing between overly specific (when considering only the last layer) and overly general (when considering only the first layers) word embeddings. Next, these representations are aggregated by summing the embeddings along the layer axes, resulting in a matrix of shape $(wp \times 768)$ (Figure 2.4-center-left), as suggested in [81]. Since BERT operates on word pieces while MLWE aims to extract full tokens (words), the embeddings of word pieces belonging to the same word are aggregated by averaging their values across the word-piece axes. This process results in a new matrix of token embeddings of shape $(tk \times 768)$, where tk represents the number of full input words (Figure 2.4-center-right). The aggregation of the four layers into a single layer and the conversion from word pieces to full tokens constitute the *vectorial aggregation* step. Finally, due to the sparse nature of the data, a Principal Component Analysis (PCA) is applied to reduce the dimensionality of the token embeddings matrix to $(tk \times c)$, where c is the number of principal components extracted (Figure 2.4-right). This resulting matrix represents the *Multi-layer Word Embedding* for the BERT model, which will be used as input for the unsupervised analysis.

In general, adapting the MLWE to various NLP architectures requires (1) providing one or more layers of word embeddings (a vector or tensor for each token), (2) an aggregation function for multiple layers of word embeddings (to transform each token from an n-dimensional to 1-dimensional), and optionally, (3) an aggregation function for word piece tokenization (where words may be divided into sub-tokens) to aggregate representations into full tokens rather than word pieces.



(a) Absolute influential words (*GAI*). (b) Relative influential words (*GRI*).

Fig. 2.5 **Global explanations.** Influential words overall (*global*) for the class-of-interest c_0.

2.3.3 Global explanation process

The second main objective of T-EBANO is to provide useful insights to understand the predictions of the classifier overall. To this end, T-EBANO also generates per-class *global explanations* of the model prediction process by aggregating several local explanations extracted with the MLWE feature extraction strategy. The global explanations can provide valuable insights into the overall importance of words, thus highlighting possible misleading behaviors of the predictive model.

To achieve this, it introduces two indices: (i) the *Global Absolute Influence* (*GAI*), and (ii) the *Global Relative Influence* (*GRI*). The *GAI* score measures the global importance of *all* the words impacting the class of interest, without considering their effect on other classes (Figure 2.5a). Conversely, the *GRI* score evaluates the relevance of words that are influential *only* (or predominantly) for the class of interest, distinguishing them from their impact on other classes (Figure 2.5b).

Global Absolute Index (*GAI*). This index is computed following the process described in Algorithm 1. First, the HashMap containing the *GAI* score for each class C and each lemma L is initialized (line 1), along with the counter of predictions for each class (line 2) and the list of unique lemmas (line 3). Given a corpus of documents D, the following steps are repeated for each input textual document $d \in D$ (line 4). First, the class label $\hat{c} \in C$ for the input text d is estimated by the NLP model (line 5), and the counter value for the class $\hat{c}$ is incremented (line 6). Next, T-EBANO generates the local explanation set E_d for the input d and the class of interest $\hat{c}$ (line 7). Subsequently, the most influential explanation $\hat{e}_{d,f}$, identified as the one with the highest *nPIR*, is selected (line 8). T-EBANO utilizes only the

Algorithm 1: Global Absolute Influence.

Input: Dataset D, Classes C .
Output: *GAI* score $\forall$ class label $c \in C$ and lemma $l \in L$.

1 $GAI \leftarrow$ initHashMap(0);
2 $PredictionsCounter(C) \leftarrow$ init(0);
3 $L \leftarrow$ empty list;
4 **for** d ***in*** D **do**
5 $\quad \hat{c} \leftarrow$ Model.Predict(d);
6 $\quad PredictionsCounter(\hat{c}) \leftarrow PredictionsCounter(\hat{c}) + 1$;
7 $\quad E_d \leftarrow$ T-EBANO.LocalExplanation($Model$, d, $\hat{c}$);
8 $\quad \hat{e}_{d,f} \leftarrow$ T-EBANO.GetMostInfluentialExplanation(E_d, $\hat{c}$, "MLWE");
9 $\quad$ **for** tk ***in*** $\hat{e}_{d,f}.featureTokens$ **do**
10 $\quad\quad l \leftarrow$ Lemmatize(tk);
11 $\quad\quad L$.insert(l);
12 $\quad\quad GAI(\hat{c},l) \leftarrow GAI(\hat{c},l) + Max[0, \hat{e}_{d,f}.nPIR]$;
13 $\quad$ **end**
14 **end**
15 **for** c ***in*** C **do**
16 $\quad$ **for** l ***in*** L **do**
17 $\quad\quad GAI(c,l) \leftarrow GAI(c,l)/PredictionsCounter(c)$;
18 $\quad$ **end**
19 **end**
20 **return** *GAI*;

MLWE features to derive global explanations. Consequently, the most influential explanation $\hat{e}_{d,f}$ corresponds to the cluster of tokens with the highest influence, as measured by *nPIR*, for the initially predicted class label $\hat{c}$. Finally, for each token tk belonging to the most influential feature $\hat{e}_{d,f}$.*featureTokens* (line 9), T-EBANO extracts the lemma l (line 10) of each token tk, adds it to the list of unique lemmas L (line 11), and updates the *GAI* score $GAI(\hat{c},l)$ for the class $\hat{c}$ of the lemma l (line 12) by summing the *nPIR* score of the explanation $\hat{e}_{d,f}$, only if it positively impacts the prediction (if $nPIR > 0$). The algorithm analyzes *lemmas* instead of *tokens* (words) to group together their inflected forms, obtaining more significant results. Finally, T-EBANO normalizes the *GAI* score of each lemma $l \in L$ and each class $c \in C$ by dividing by the number of inputs predicted with the class label c (lines 15, 16, 17). This normalization step is necessary to address the issue of class imbalance.

The output is the set of *Global Absolute Influence* scores. Specifically, for each unique lemma found in the corpus D, a *GAI* score is computed for each possible class $c \in C$. The value $GAI(c,l)$ ranges from $[0, +\infty]$ and measures the absolute

global influence of the lemma l for the class c. Note that, in the current T-EBANO implementation, the *GAI* score can exceed 1 because if a lemma is present n times in an influential feature, its global score is updated by summing the *nPIR* n times. We could obtain a score in the range $[0,1]$ by considering the list of *unique* lemmas of the feature (in lines 9 and 10). However, we chose to reward lemmas identified multiple times as important for the prediction in a single local explanation. The *GAI* score will be 0 for all lemmas that have consistently influenced negatively the class c. Conversely, the score will increase proportionally to both the frequency and the influence of each lemma that positively impacts class c. The higher the *GAI* score, the more positively influential the lemma is for the model overall for the class c.

Global Relative Index (*GRI*). This index extracts the most influential and differentiating lemmas for each class of interest, discarding lemmas with significant impacts on other classes. The *GRI* for a class of interest c, for a specific lemma l, and for a classification task with n_C classes is computed as:

$$GRI(c,l) = Max\left[0, \left(GAI(c,l) - \sum_{c_i \neq c}^{C} GAI(c_i,l)/(n_C - 1)\right)\right] \tag{2.8}$$

The *GRI* score is 0 when a lemma is more relevant for other classes than for the one under examination, while $GRI > 0$ if its influence is higher for class c than for all other classes combined. The higher the *GRI* value, the more specific the lemma's influence is with respect to the class of interest. Normalizing over the number of predicted samples for each class performed on the *GAI* ensures fair scores with unbalanced classes while dividing by $n_C - 1$ allows for handling multi-class tasks.

Global explanation report. T-EBANO produces a *global explanation report* containing the *GAI* and *GRI* scores for each word present in the corpus analyzed. Moreover, the report allows users to query the most important words overall, and to retrieve the *local explanations* where some selected words have a large importance. By analyzing *GAI* and *GRI* scores, users can understand the most crucial inter- and intra-class semantic concepts impacting the decision-making process of the model overall. For instance, a word with significant influence across all classes will exhibit a high *GAI* score and a *GRI* score approaching 0 across all classes. Conversely, if a word is predominantly influential within a specific class, its *GAI* score will be higher for that class, resulting in a *GRI* score greater than 0 for that class while typically being 0 (or close to 0) for other classes.

2.4 Effectiveness evaluation

This evaluation aims to assess the effectiveness of T-EBANO in providing useful *local* and *global* explanations. To this end, we apply it to two binary classification tasks: (i) *toxicity prediction* (Section 2.4.1), and (ii) *sentiment analysis* (Section 2.4.2).

2.4.1 Explaining toxicity classification

The first evaluation task consists of explaining a black-box classifier trained for binary *toxic comment classification*, which involves predicting whether an input comment is *toxic*–it contains inappropriate content.

Black-box classifier. An LSTM [64] model has been trained on a civil Wikipedia comments dataset annotated for toxicity classification [82]. The dataset also encompasses several subtypes of toxicity, including identity attacks, insults, explicit sexuality, obscenity, and threats. However, we only used the *toxic* label using 0.5 as a threshold to discriminate between *non-toxic* (or clean) and *toxic* comments, as a binary classification task. The LSTM model consists of an embedding layer using GloVe embeddings [83] with 300-dimensional vectors, followed by two bidirectional LSTM layers, each with 256 units for each direction, and finally, a dense layer with 128 hidden units. After training, the LSTM model achieved an accuracy of 90%. Despite its high performance, the LSTM classifier behaves as a black-box. T-EBANO can help in identifying the most important words in the predictive process.

MLWE implementation. For the MLWE, we exploit the single embedding layer to evaluate its effectiveness when only a single layer is present. This results in the extraction of a tensor of shape $(tk \times 300 \times 1)$, where tk is the number of input tokens. Hence, the *vectorial aggregation* step is unnecessary because a single layer is extracted and the model does not present sub-words. Finally, a PCA is used to reduce the embedding matrix shape to $(tk \times c)$, obtaining the *multi-layer word embedding* representation for the LSTM model (as explained in Section 2.3.2).

Evaluation metrics. We quantitatively assess the percentage of explanations where T-EBANO successfully identifies highly influential features for local explanations. The idea behind this evaluation is that if T-EBANO can extract highly influential features, it can accurately determine the most important words in the model's predic-

Table 2.4 **Quantitative results in explaining the LSTM model for toxicity prediction.** The percentage of texts for which each feature extraction strategy identifies at least one highly influential feature for local explanation (i.e., with $nPIR \geqslant 0.5$) for the class labels *Non-Toxic* and *Toxic* separately, and together (*Non-Toxic/Toxic*). *Overall* is the percentage of texts for which at least one feature extraction strategy provided a local explanation with $nPIR \geqslant 0.5$.

Feature extraction strategy	Non-Toxic	Toxic	Non-Toxic&Toxic
Part-of-Speech (PoS)	8%	98%	53%
Sentence-level (SEN)	2%	76%	39%
MLWE	12%	98%	55%
Overall	15%	99%	58%

tive process, thereby achieving its primary objective effectively. Additionally, we report and discuss qualitative examples of both local and global explanations.

Results. We randomly selected 2,250 texts from the test set, with 1,121 belonging to the *toxic* and 1,129 to the *non-toxic* classes. We then applied T-EBANO to produce the *local* explanations. Finally, all the local explanations produced by T-EBANO with MLWE are aggregated and analyzed to produce the *global* explanations.

(1) Quantitative results. Table 2.4 reports the percentage of input texts for which T-EBANO has successfully extracted at least one highly influential feature (with $nPIR \geqslant 0.5$) for each label, with each feature extraction strategy.

For the *toxic* class (third column in Table 2.4), T-EBANO has successfully identified at least one highly influential explanation for nearly all the input texts, across most feature extraction strategies. The only exception is the sentence-level feature extraction, which exhibits a lower percentage of highly influential explanations compared to other techniques (76%). This observation suggests that toxic words are often sparse in the input text and not concentrated within a single sentence.

In contrast, finding highly influential explanations for the *non-toxic* class has proven to be more challenging, as reported in the second column in Table 2.4. None of the feature extraction techniques can explain more than 15% of the predictions for the *non-toxic* input texts. This finding suggests that there is no specific pattern of words that consistently represents the *non-toxic* class. The nature of the use case being examined may offer insight into this phenomenon: typically, a text is initially considered *non-toxic*, and it may only become toxic due to the presence of specific

```
Criticize a black man and the left calls you a racist.  Criticize a woman and you
are a sexist.  Now I will criticize you as a fool and you can call me intolerant.
```

(a) Original text

```
Criticize a black man and the left calls you a racist.  Criticize a woman and you
are a sexist.  Now I will criticize you as a fool and you can call me intolerant.
```

(b) `EXP1`: Adjective & Noun - POS feature extraction

```
Criticize a black man and the left calls you a racist.  Criticize a woman and you
are a sexist.  Now I will criticize you as a fool and you can call me intolerant.
```

(c) `EXP2`: Multi-layer word embedding feature extraction

Fig. 2.6 Examples of *textual explanation* report for the input in Figure 2.6a originally labeled by LSTM model as *Toxic* with a probability of 0.98. The most relevant features are highlighted in cyan.

Exp. ID	Feature extraction strategy	Predicted Label	Perturbed Label	nPIR
EXP1	PoS-Adj&Noun	*Toxic*	*Non-Toxic*	0.839
EXP2	MLWE	*Toxic*	*Non-Toxic*	**0.883**

Table 2.5 Quantitative explanation for the example reported in Figure 2.6. For each feature is reported: an explanation identifier, the feature extraction strategy used, the predicted label before and after perturbation, and influence index (*nPIR*) computed for the *toxic* class.

words or linguistic expressions. Thus, the hypothesis arises that the absence of a distinct pattern for the *non-toxic* class might be attributed to its inherently diverse and unpredictable nature. This observation also applies to similar tasks, such as spam detection. In these tasks, it is more insightful to focus on explaining the positive class only (e.g., *toxic*, *spam*). This is because the negative class (e.g., *non-toxic*, *non-spam*) is often defined more by the absence of specific features rather than the presence of distinctive characteristics. In contrast, the positive class (e.g., toxic, spam) is characterized by specific patterns or features that the model can learn and identify, and thereby can be explained.

(2) Qualitative results. Figure 2.6 and Table 2.5 show an example of *local explanation*. The LSTM model classified the input comment shown in Figure 2.6a as *toxic*. The most influential features identified by T-EBANO are illustrated in Figures 2.6b and 2.6c. For each feature identified, the strategy used, the predicted label before and after perturbation, and the *nPIR* influence index computed for the *toxic* class are reported in Table 2.5. Across different feature extraction strategies, the most positively influential features for the *toxic* class label are found to be the set of words {`black`

`man, left, racist, woman, sexist, fool, intolerant}`. These words, if removed from the original input text, cause the model's prediction to change from *toxic* to *non-toxic*. Specifically, the most influential explanations are extracted using the combination of adjectives and nouns for the part-of-speech feature extraction strategy (Table 2.5-EXP1) and with MLWE (Table 2.5-EXP2). Both the PoS feature extraction and MLWE highlighted very similar sets of words. The *local* explanation reveals that the model has learned that the presence of words and features related to race and gender, like ‘‘`black man`’’ and ‘`woman`’, are highly important for distinguishing between *non-toxic* and *toxic* comments, at least for this specific prediction. However, the global explanations can provide valuable insights into the widespread and generality of this model's behavior.

Figure 2.7 and Table 2.6 show another example of a *local explanation* for a correctly classified *toxic* comment, reported in Figure 2.7a. The most influential features are extracted using nouns for the part-of-speech feature extraction strategy (Table 2.6-EXP1) and with MLWE (Table 2.6-EXP2). This time, the MLWE feature extraction strategy identifies a smaller set of words with the maximum influence, which comprises ‘`nuts`’, ‘`pigs`’, ‘`freak`’, ‘`suited`’, and ‘`circus`’. These five words, if removed from the original text, cause a change in the model's prediction. In contrast, the PoS strategy identifies that nouns are particularly important for this prediction. However, this feature comprises a larger set of words than the one identified with MLWE but with lower influence. Hence, the MLWE is more effective in this case. Based on the MLWE explanation, the model's prediction seems reliable, as the model relied on effectively toxic words in classifying the original text.

Figure 2.8 shows the *global explanations* produced by aggregating the full set of local explanations, visualized as word clouds. The font size in the word clouds is proportional to the *GAI* or *GRI* scores obtained for each class separately. The proportion of the font size is relative only to the single word cloud. This means that words with the same size in different word clouds do not necessarily have the same score while in the same word cloud, they have almost the same score.

As expected, the *GAI* word clouds reveal that the two classes are influenced by non-overlapping sets of words, as shown in Figures 2.8a and 2.8b. Specifically, some of the most important lemmas for the *toxic* class (i.e., those with higher *GAI* for the *toxic* class) are ‘`stupid`’ (0.31), ‘`Politician1`’ (0.28), ‘`people`’ (0.26), ‘`idiot`’ (0.17), ‘`white`’ (0.15), ‘`black`’ (0.14), and ‘`woman`’ (0.14). On the other hand, the

They are nuts that have the Liberal government by the short and curlies, the good news is we see them and they are being rejected on a global stage! It can't happen fast enough, the Marxist pigs are a freak show better suited for a circus.

(a) Original text

They are nuts that have the Liberal government by the short and curlies, the good news is we see them and they are being rejected on a global stage! It can't happen fast enough, the Marxist pigs are a freak show better suited for a circus.

(b) `EXP1`: Noun - POS feature extraction

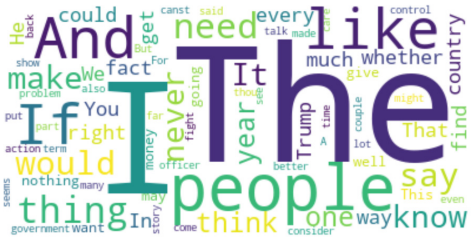
They are nuts that have the Liberal government by the short and curlies, the good news is we see them and they are being rejected on a global stage! It can't happen fast enough, the Marxist pigs are a freak show better suited for a circus.

(c) `EXP2`: Multi-layer word embedding (MLWE) feature extraction

Fig. 2.7 Examples of *textual explanation* report for the input in Figure 2.7a originally labeled by the LSTM as *Toxic* with 0.96 probability. Features are highlighted in cyan. The features' influence (*nPIR*), and the predicted label after perturbation are reported in Table 2.6.

Exp. ID	Feature extraction strategy	Predicted label	Perturbed label	nPIR
EXP1	PoS-Noun	*Toxic*	*Non-Toxic*	0.608
EXP2	MLWE	*Toxic*	*Non-Toxic*	**0.999**

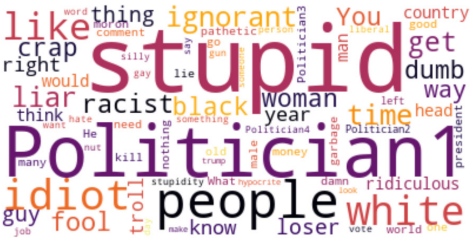
Table 2.6 Quantitative explanation for the example reported in Figure 2.7. For each feature is reported: an explanation identifier, the feature extraction strategy used, the predicted label before and after perturbation, and influence index (*nPIR*) computed for the *toxic* class.



(a) GAI *Non-Toxic* class.

(b) GAI *Toxic* class.

(c) GRI *Non-Toxic* class.



(d) GRI *Toxic* class.

Fig. 2.8 Global explanation of toxic comment classification with LSTM.

lemmas with higher *GAI* for the *Clean* class are 'the' (0.04), 'people' (0.02), 'and' (0.01), 'if' (0.01), and 'like' (0.01). This observation confirms that the model has learned that words associated with toxic language in one context are unlikely to be associated with non-toxic language in others. Toxic comments are characterized by terms strongly related to toxic language, discrimination, or racism. In contrast, there is no specific pattern of words that consistently identifies clean comments. Only a few concepts, such as 'people', exhibit inter-class influence.

The *GRI* word clouds, shown in Figures 2.8c and 2.8d, further underscores the impact of words like 'stupid', 'idiot', and 'ignorant', which obtained *GRI* scores for the *Toxic* class of 0.31, 0.17, and 0.12, respectively. Additionally, terms related to protected attributes related to race, gender, and sexual orientation, such as 'woman', 'black', 'white', and 'gay' (with *GRI* scores for the *Toxic* class of 0.10, 0.10, 0.15, and 0.06) indicate that the model has learned to recognize racist or sexist comments when these terms are present. Moreover, the presence of specific politician family names, anonymized as 'Politician1', 'Politician2', etc., highlights that those individuals' names are associated with toxic comments. Notably, 'Politician1' achieved the second-highest *GRI* score for the *Toxic* class with 0.28. These results underscore that NLP models, if not carefully trained, can learn from sensitive content or protected attributes, including prejudices and various forms of bias that should be avoided in critical contexts. Finally, associating a specific person's family name or protected attributes with a class raises ethical concerns. An in-depth analysis of this behavior of NLP classifiers is presented in Chapter 3.

Summary. T-EBANO is able to effectively explain individual predictions of the LSTM model trained for toxicity prediction by identifying the most important set of words used by the classifier for predicting toxicity. Across the different feature extraction strategies, MLWE proves to be the most effective. This shows the potential of exploiting and mining the inner knowledge of the classifier to explain its predictions. Finally, by aggregating several local explanations, T-EBANO is able to provide useful global explanations to better understand the overall behavior of the classifier. From the global explanations emerges that the LSTM toxicity classifier extensively relies on words related to protected characteristics such as gender, race, and sexual orientation raising concerns about possible bias learned by the classifier.

Table 2.7 **Quantitative results in explaining BERT for sentiment analysis.** The percentage of texts for which each feature extraction strategy identifies at least one highly influential feature for local explanation (i.e., with $nPIR \geqslant 0.5$). *Overall* is the percentage of texts for which at least one feature extraction strategy provided a local explanation with $nPIR \geqslant 0.5$.

Feature extraction strategy	Negative&Positive
Part-of-Speech (PoS)	70%
Sentence-level (SEN)	30%
MLWE	86%
Overall	90%

2.4.2 Explaining sentiment analysis

The second evaluation task is a binary *sentiment analysis*, which involves predicting whether the underlying sentiment of an input text is either *positive* or *negative*.

Black-box classifier. We selected the BERT [37] base and uncased pre-trained model as a deep learning classifier with a black-box decision-making process. We fine-tuned BERT on the IMDB [84] dataset, which serves as a reference dataset for sentiment analysis. We conducted a fine-tuning step of the BERT model by adding a classification layer on top of the last encoder transformer's stack. The fine-tuned BERT model, trained on the IMDB textual reviews, achieved 86% of accuracy.

MLWE implementation. For the MLWE, as detailed in Section 2.3.2, we extracted the embedding from the last four layers. We then aggregated over the layers axis by summing the vectors, and representations of subwords are averaged to obtain a single vector for each full word. Finally, we applied a PCA.

Evaluation metrics. Similarly to the previous evaluation, we quantitatively assess the percentage of explanations where T-EBANO successfully identifies highly influential features for local explanations, and we qualitatively show examples of local and global explanations.

Results. We randomly selected 400 reviews from the test set, 202 belong to the *positive* and 198 to the *negative* classes. Then, T-EBANO produced the local explanations by extracting the most important words used by the classifier to predict the *negative* or the *positive* label for each review. Finally, T-EBANO aggregated all the MLWE local explanations to produce the global explanations for both classes.

(1) Quantitative results. Table 2.7 reports the percentage of texts for which T-EBANO was able to find at least one highly influential feature ($nPIR \geqslant 0.5$). The results are only reported aggregated for both classes since they are identical. We found that the MLWE feature extraction strategy significantly outperforms the other ones. It was able to identify a highly influential feature (with $nPIR \geq 0.5$) for 86% of the input texts. The part-of-speech (PoS) strategy achieves slightly worse results. It was able to extract highly influential features from 70% of the texts. In contrast, the sentence-level (SEN) feature extraction strategy identifies highly influential features only for 30% of texts. This is likely because the sentiment, whether positive or negative, is often expressed throughout the entire review, making it necessary to consider the overall context to accurately classify the reviewer's feelings. Overall, T-EBANO successfully identifies a highly influential feature using at least one of its strategies in 90% of the cases.

(2) Qualitative results. Figure 2.9 and Table 2.8 show an example of local explanation. The BERT model misclassified the input review shown in Figure 2.9a as *negative* (the expected label is *positive*). This time, each feature extraction strategy was able to identify a highly influential feature, as shown in Figures 2.9b, 2.9c, and 2.9d. The PoS strategy shows that adjectives are highly influential in the prediction process for this text (Figure 2.9b). The sentence-level feature extraction shows that the perturbation of a single sentence (Figure 2.9c) leads to a change in the predicted class label. Interestingly, the MLWE feature extraction strategy identifies a small set of words consisting of two instances of the same general word '`there`' causing a change in the predicted label if removed from the input text. Therefore, we can conclude that the prediction for this text is unreliable. Generally, if altering a small, non-meaningful portion of the input causes a change in the prediction, the entire process should be questioned. In this instance, the model exhibited significant uncertainty regarding the review's sentiment. This uncertainty cannot be detected by comparing it to the ground truth alone, as it is unclear what the problem is without a thorough analysis of the result. Moreover, in real-world applications, ground-truth labels are often unavailable. However, our explanation process has successfully identified this uncertainty by demonstrating that just a few words could significantly alter the outcome. This raises concerns about the robustness of the NLP classifier.

Figure 2.10 reports the *GAI* and *GRI* word clouds for the *positive* and *negative* class labels. Figure 2.10b, shows the most important lemmas for the *positive* class (i.e., those with higher *GAI* for the *positive* class) are '`film`' (0.60), '`movie`' (0.48),

How many movies are there that you can think of when you see a movie like this? I can't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT... See The Movie! Despite its downfalls there is enough to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. In fact, you'll probably find yourself battening down the hatches and riding the storm out. Why? ...Costner and Kutcher played their characters very well. I have never been a fan of Kutcher's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner's ability. I don't think Mrs. Robinson had anything to do with his success. The supporting cast all around played their parts well. I had no problem with any of them in the end. But some of these characters were used too much. From here on out I can only nit-pick so I will save you the wear and tear. Enjoy the movie, the parts that work, work well enough to keep your head above water. Just don't expect a smooth ride. 7 of 10 but almost a 6.

(a) Original text

How many movies are there that you can think of when you see a movie [...] I was reminded so often of other movies, it became a big distraction. One of the borrowed memorable lines came from a movie from 2003 - Day After Tomorrow. One line by itself, is not so bad but this movie borrows so much from so many movies it becomes a bad risk. BUT ... See The Movie! Despite its downfalls there is enough to make it interesting and maybe make it appear clever. While borrowing so much from other movies it never goes overboard. [...] I have never been a fan of Kutcher 's and I nearly gave up on him in The Guardian, but he surfaced in good fashion. Costner carries the movie swimmingly with the best of Costner 's ability. [...] But some of these characters were used too much. [...] Just do n't expect a smooth ride. 7 of 10 but almost a 6.

(b) `EXP1`: Adjective - POS feature extraction

How many movies are there that you can think of when you see a movie like this? I can't count them but it sure seemed like the movie makers were trying to give me a hint. I was reminded so often of other movies, it became a big distraction. One of [...]

(c) `EXP2`: Sentence feature extraction

How many movies are there that you can think of when you see a movie like this? [...] See the movie despite its downfalls there is enough to make it interesting and maybe make it appear clever. [...]

(d) `EXP3`: Multi-layer word embedding feature extraction

Fig. 2.9 Examples of *textual explanation* report for the input in Figure 2.9a, wrongly labeled by BERT as *Negative* with a probability of 0.99. The most relevant features are highlighted in cyan.

Exp. ID	Feature extraction strategy	Predicted label	Perturbed label	nPIR
EXP1	PoS-Adjective	*Negative*	*Positive*	**0.884**
EXP2	SEN	*Negative*	*Positive*	0.663
EXP3	MLWE	*Negative*	*Positive*	0.651

Table 2.8 Quantitative explanation for the example in Figure 2.9. P is the positive label, N is the negative label. Positively highly influential features ($nPIR \geq 0.5$) for the L_o class are highlighted in green in the $nPIR_f(N)$ column.

(a) GAI *Negative* class.

(b) GAI *Positive* class.

(c) GRI *Negative* class.

(d) GRI *Positive* class.

Fig. 2.10 Global explanation of sentiment analysis with BERT.

'one' (0.34), 'like' (0.22), 'story' (0.21), 'good' (0.21), 'great' (0.20), and 'love' (0.19). In contrast, the lemmas with higher *GAI* for the *negative* class (Figure 2.10a) are 'movie' (0.37), 'film' (0.25), 'like' (0.17), 'one' (0.16), 'even' (0.14), and 'story' (0.12). Unlike the previous example, the *GAI* word clouds for the *positive* and *negative* class labels show that several words such as 'story', 'movie', 'film', and 'like' influence both classes. This indicates that the model leverages overlapping concepts that do not directly convey sentiment but, when considered in context, can be associated with words expressing the writer's mood (e.g., ''This film is not as good as expected'').

To determine which lemmas have the most significant impact on one class compared to the other, the model calculates the *GRI* score for each lemma for the two classes and generates the word clouds accordingly. The *GRI* word cloud for the *positive* class (Figure 2.10d) shows that words like 'movie' and 'film' remain highly relevant, while they do not appear for the *negative* class (Figure 2.10c), which is now predominantly characterized by the concept of 'book'. Specifically, 'movie' and 'film' achieve *GRI* scores of 0.35 and 0.11, respectively, for the *positive* class (with scores of 0 for the *negative* class). Conversely, 'book' obtains a *GRI* score of 0.07 for the *negative* class (with a score of 0 for the *positive* class). Exploring the dataset, we observed that reviews of movies inspired by books tend to be associated with negative comments, as reviewers often prefer the original book. As a result, the model likely

learned to associate the word 'book' with negative sentiment. This represents a form of bias, where the model is more likely to predict a review as negative if it contains a comparison to the corresponding book. However, the *GRI* also shows that most of the influential words for positive input texts are concepts closely related to positive sentiments such as 'good', 'great', 'best', and 'love', which achieve *GRI* scores for the *positive* class of 0.12, 0.17, 0.12, and 0.17. Similarly, negative sentiment is associated with words like 'worst', 'bad', and 'awful', which achieve *GRI* scores of 0.07, 0.06, and 0.05, respectively, for the *negative* class. For these concepts, the model behaves as expected.

Summary. These results confirm the effectiveness of T-EBANO in providing useful explanations for individual predictions (local) and overall insights about the most important predictive words (global) also for BERT trained for sentiment analysis.

2.5 Generalizability evaluation

This evaluation aims to further assess the broad applicability of T-EBANO across NLP models and classification tasks, summarized in Table 2.9.

Evaluation tasks. We selected two additional tasks. The first task is *topic classification* with the *Ag News* [85] dataset. It consists of predicting one of the four following topics from news articles: *World*, *Sport*, *Business*, and *Science/Technology*. This task differs from the previous ones (i.e., toxicity classification and sentiment analysis) as it is multi-class instead of binary. The second task is a binary classification problem that consists of predicting the grammatical *acceptability* or *unacceptability* of the sentence using the *Corpus of Linguistic Acceptability* (*CoLA*) [86] dataset. Both tasks differ from the previous ones because they do not rely heavily on specific parts of speech (e.g., adjectives for sentiment analysis).

Black-box classifiers. For each additional task we trained three different models: BERT [37], ALBERT [79], and ULMFit [87]. The last column in Table 2.9 reports the accuracy achieved on the test set by the trained models for both datasets. For *Ag News* dataset, all models achieve an accuracy of ≥ 0.92. In contrast, models trained on the sentence acceptability task using the *CoLA* dataset achieve lower accuracy.

MLWE implementation. The MLWE implementation for BERT and ALBERT is the same as the one used in the previous experiments for BERT. For ULMFit, the

Table 2.9 **Experimental use cases to evaluate T-EBANO's generalizability.** For each use case: an identifier, the classifier trained, the dataset used, the classification task, and the accuracy achieved on the test set.

Use case	Model	Dataset	Task (Classification)	Test accuracy
1	BERT	Ag News	Topic Classification	94%
2	BERT	CoLA	Sentence Acceptability	81%
3	ALBERT	Ag News	Topic Classification	93%
4	ALBERT	CoLA	Sentence Acceptability	77%
5	ULMFit	Ag News	Topic Classification	92%
6	ULMFit	CoLA	Sentence Acceptability	71%

MLWE implementation is similar to the one of the LSTM but we use the LSTM-encoder instead of the embedding layer. Specifically, from the LSTM-encoder part of ULMFit, a one-dimensional vector dimensionality 400 is extracted for each word.

Evaluation metrics. We evaluate the distribution of the most and least influential features identified by T-EBANO when producing the local explanations. The idea behind this evaluation is that the least important features should be distributed around zero influence as they are neutral or not important for the predictions of the model. In contrast, the most important features must be distributed as close as possible to the maximum value of influence (i.e., approaching 1). Additionally, we qualitatively show some examples of local explanations produced.

Results. For each model and task, we first randomly selected 512 input texts from the test set. Secondly, we produced the *local explanations* with T-EBANO, and we selected the most and the least influential features–obtaining the highest and the lowest *nPIR*. We used the MLWE feature extraction strategy only as it showed higher effectiveness and precision in the previous experiments.

(1) Quantitative results. Figure 2.11 shows the *nPIR* distribution of the most influential features (*Max nPIR*) and the least influential features (*Min nPIR*) for all input texts, separately by each model and task. The *nPIR* values of the least influential features are close to zero for all models, whereas the most influential features have *nPIR* values close to 1 for all models and generally higher than 0.5. The BERT models perform better on these tasks, allowing T-EBANO to identify features with *nPIR* values close to 1 (i.e., the maximum). In contrast, the ULMfit

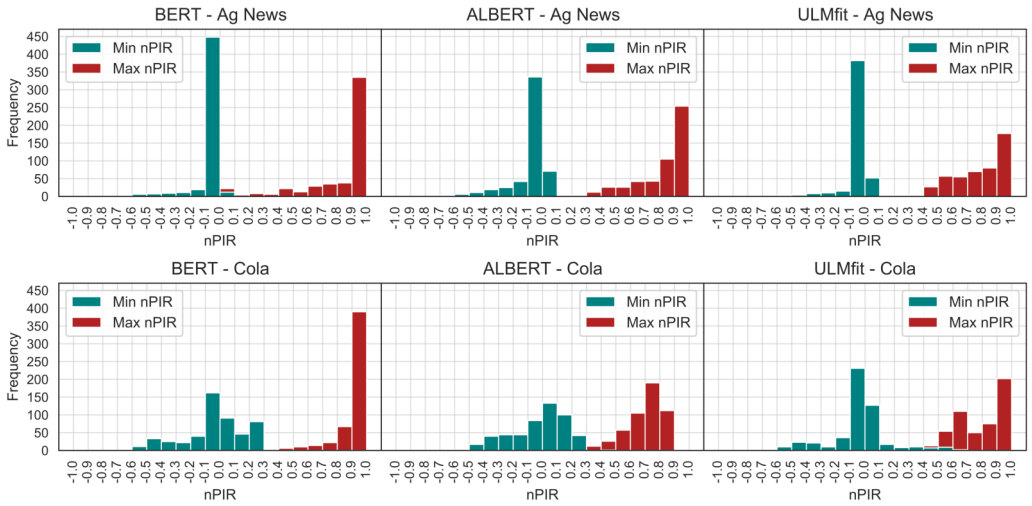


Fig. 2.11 **T-EBANO generalizability evaluation.** The *nPIR* distribution of the most (*Max nPIR*) and least influential features (*Min nPIR*) across 512 texts for each model and task.

models exhibit more uncertainty in their predictions, resulting in T-EBANO finding features with variable *Max nPIR* values ranging from 0.5 to 1.

This quantitatively demonstrates that T-EBANO can effectively identify influential features (i.e., words important for the classification) and neutral features across models and tasks, providing valuable explanations of classifier prediction processes.

(2) Qualitative results. Tables 2.10 and 2.11 provide examples of local explanations across the different experimental models and the *Ag News* and *CoLA* tasks. For each input text i, it includes one highly influential feature ($i.a$) and one neutral or less influential feature ($i.b$). The table also reports the original predicted label L_o, the predicted label after perturbation L_p, and the relative nPIR score for each feature (relative to the original predicted label L_o) after applying the removal perturbation.

For *Ag News* (Table 2.10), the explanations demonstrate that all models correctly learned the concepts of *World*, *Business*, *Sport*, and *Science/Technology* classes. All the influential features (1-6.a) contain concepts related to the predicted class (L_0), whereas the less influential features (1-6.b) contain neutral concepts or tokens. The only exception is the example 6.a, where the ULMFit model overfits certain tokens, such as the specific name of the London agency '`Reuters`', which has been learned as important for the class label *Business*. The behavior of the explanations aligns with the models' performance in terms of accuracy. When analyzing a broader set of explanations, it was found that both BERT and ALBERT models also overfit

Table 2.10 Features extracted by MLWE (highlighted in cyan) on different models for *topic classification* on the *Ag News* dataset. For each input *i*, one highly influential (*i.a*) and one neutral or less influential (*i.b*) features are reported. L_o represents the original predicted label, L_p denotes the label predicted after perturbation of the feature, and *nPIR* is the score obtained by the feature relative to the original predicted label. The class labels are the following: *Sport* (S), *World* (W), *Business* (B), *Science/Technology* (S/T).

ID	MLWE feature	L_o	L_p	nPIR
BERT - Ag News				
1.a	uk gives blessing to open source . with most organizations that planned to move already moved to microsoft server 2003 , os migration has dropped to the bottom ranks after making its	S/T	W	**0.983**
1.b	uk gives blessing to open source . with most organizations that planned to move already moved to microsoft server 2003 , os migration has dropped to the bottom ranks after making its	S/T	S/T	0.000
2.a	radcliffe to run in new york marathon . london (reuters) - world marathon record holder paula radcliffe believes she has put her failure at the athens olympics behind her after announcing on tuesday that she will run in the new york marathon on november 7 .	S	W	**0.893**
2.b	radcliffe to run in new york marathon . london (reuters) - world marathon record holder paula radcliffe believes she has put her failure at the athens olympics behind her after announcing on tuesday that she will run in the new york marathon on november 7 .	S	S	0.006
ALBERT - Ag News				
3.a	eu seeks joint asylum policy. eu ministers meeting in luxembourg plan moves to integrate their asylum and immigration procedures.	W	S/T	**0.709**
3.b	eu seeks joint asylum policy. eu ministers meeting in luxembourg plan moves to integrate their asylum and immigration procedures.	W	W	-0.023
4.a	job numbers give candidates room to debate. washington - employers stepped up hiring in august, expanding payrolls by 144,000 and lowering the unemployment rate to 5.4 percent.	B	W	**0.912**
4.b	job numbers give candidates room to debate. washington - employers stepped up hiring in august, expanding payrolls by 144,000 and lowering the unemployment rate to 5.4 percent.	B	B	0.008
ULMfit - Ag News				
5.a	nato to send staff to iraq . nato will send military trainers to iraq before the end of the year in response to appeals by iraqi leaders for speedy action , us ambassador to nato nicholas burns said today .	W	S/T	**0.706**
5.b	nato to send staff to iraq . nato will send military trainers to iraq before the end of the year in response to appeals by iraqi leaders for speedy action , us ambassador to nato nicholas burns said today .	W	W	0.001
6.a	court seen lifting yukos block – lawyers . london (reuters) - a u.s . bankruptcy court is likely to revoke its temporary ban on the sale of russian oil group yukos 's main production unit, lawyers said on friday	B	W	**0.993**
6.b	court seen lifting yukos block -- lawyers . london (reuters) - a u.s . bankruptcy court is likely to revoke its temporary ban on the sale of russian oil group yukos 's main production unit, lawyers said on friday	B	B	0.043

Table 2.11 Features extracted by MLWE (highlighted in cyan) on different models for the *CoLA* dataset. For each input *i*, one highly influential (*i.a*) and one neutral or less influential (*i.b*) features are reported. L_o is the original predicted label, L_f is the label predicted after feature perturbation, and *nPIR* is the score obtained by the feature relative to the original predicted label. The class labels are the following: *Acceptable* (A), and *Unacceptable* (U).

ID	MLWE feature	L_o	L_p	*nPIR*
BERT - CoLA				
7.a	many people said they were sick who weren' t .	U	A	**0.985**
7.b	many people said they were sick who weren' t .	U	U	0.200
8.a	charlie will leave town if his mother - in - law doesn' t .	A	U	**0.995**
8.b	charlie will leave town if his mother - in - law doesn' t .	A	A	0.452
ALBERT - CoLA				
10.a	mary runs not the marathon.	U	A	**0.819**
10.b	mary runs not the marathon.	U	U	0.267
11.a	both workers will wear carnations.	A	U	**0.744**
11.b	both workers will wear carnations.	A	A	0.033
ULMfit - CoLA				
12.a	you could give a headache to a tylenol .	U	A	**0.930**
12.b	you could give a headache to a tylenol .	U	U	0.119
13.a	paul breathed on mary .	U	A	**0.999**
13.b	paul breathed on mary .	U	U	0.320

some tokens, such as HTML strings from web pages, which are often related to misclassified inputs in *Science/Technology*.

Regarding *CoLA* (Table 2.11), the explanations show that the models generally learned to classify grammatically correct sentences. The explanations for the *Acceptable* class label typically encompass most of the input text, whereas the explanations for the *Unacceptable* class labels tend to highlight small portions of the input text containing errors. This behavior is reasonable since a sentence is correct if all its tokens are correct, while it is incorrect if it contains some erroneous tokens.

Summary. These results further demonstrate the general applicability of T-EBANO. They show that it is able to extract a variety of features from the input texts, identifying both highly influential and neutral ones for predicting the class label across models with different architectures and various classification tasks.

2.6 Comparison with model-agnostic techniques

This evaluation aims to compare the *precision* and the *efficiency* of the explanations provided by T-EBANO with respect to *model-agnostic* XAI techniques.

State-of-the-art XAI techniques. We compare the explanations generated by T-EBANO with the ones produced by two of the most widespread model-agnostic XAI techniques, LIME [54] and SHAP [56]. These two techniques do not access the inner knowledge of the classifiers since they are *model-agnostic*.

Experimental use cases. We selected two experimental use cases for this evaluation: (i) a BERT model fine-tuned for *sentiment analysis* using the *IMDB* dataset, and (ii) a BERT model fine-tuned for *topic classification* using the *AG News*. Due to compatibility issues, we did not use the same models as those trained for the previous experiments. Instead, we fine-tuned two new BERT models utilizing the *HuggingFace* library.[2] The fine-tuned models achieved 93% and 95% accuracy on the validation set for the *IMDB* and *AG News* datasets, respectively.

Evaluation metrics. We measure the mean decrease in probability by removing the most important words identified by each XAI technique to assess the precision and faithfulness of the explanation. The underlying idea is that, if an explanation accurately identifies the influential words for a prediction, removing these words from the input text should result in a significant decrease in the prediction's probability. Moreover, we compute the mean running time in seconds to produce an explanation to assess the technique's efficiency. The experiments were performed on a single node of the SmartData BigData cluster at Polito[3] using one Intel Xeon Gold 6140 CPU and 120GB of RAM (without GPUs).

Results. To ensure a fair comparison, we generated features consisting of the same percentage of the most important tokens as identified by the different methodologies. LIME assigns an importance score to each token but requires a predefined percentage of the most important tokens to extract. We set this parameter so that the number of important tokens for the class of interest approximately matches the average number of tokens highlighted by the T-EBANO explanations. Conversely, SHAP assigns an importance score to each token in the input text (*Shapley Values* [57]), and we

[2] google-bert/bert-base-uncased
[3] https://smartdata.polito.it/computing-facilities/

selected the most important tokens using the same percentage as T-EBANO. This approach allowed us to select subsets with similar cardinality and importance.

For the *IMDB* dataset, T-EBANO-MLWE highlights approximately 20% of tokens on average. Therefore, we also removed the top 20% of tokens selected by LIME and SHAP. We assessed the change in prediction probability before and after removing the highlighted tokens for each methodology. Removing the most influential tokens highlighted by T-EBANO results in an average probability decrease of around 71%. In comparison, removing the same percentage of tokens identified by LIME results in an average probability drop of 48%, while for SHAP, the average decrease is 59%. Additionally, we compared the mean execution time required to generate an explanation. Given that the IMDB dataset contains relatively long texts, T-EBANO took an average of 38 seconds, whereas LIME required 304 seconds, and SHAP needed 484 seconds (without GPUs).

For the *Ag News* dataset, T-EBANO-MLWE highlights approximately 30% of tokens on average. We removed the top 30% of tokens selected by LIME and SHAP. Removing the most important tokens results in an average probability decrease of around 75% for T-EBANO, 60% for LIME, and 61% for SHAP. The mean execution time to produce each explanation is lower due to the shorter sentences in this dataset. Specifically, T-EBANO takes an average of 4 seconds, LIME 239 seconds, and SHAP 16 seconds.

Summary. These results reveal that not only is T-EBANO significantly faster than the other two methodologies (by approximately 1 to 2 orders of magnitude), but the explanations it provides are also more *faithful* and *precise*. Therefore, it accurately highlights tokens that have the most impact on the model's prediction while ensuring efficiency in producing the explanations.

2.7 Human evaluation

To evaluate the quality of the explanations selected by T-EBANO based on their *nPIR* value, we assess the correlation between the *nPIR* value and human judgment through a survey involving both expert and non-expert users.

Survey settings. We selected 20 input texts, 12 from *Ag News* with BERT (use case 1 in Table 2.7) and 8 from the *toxic comments* dataset with the LSTM model

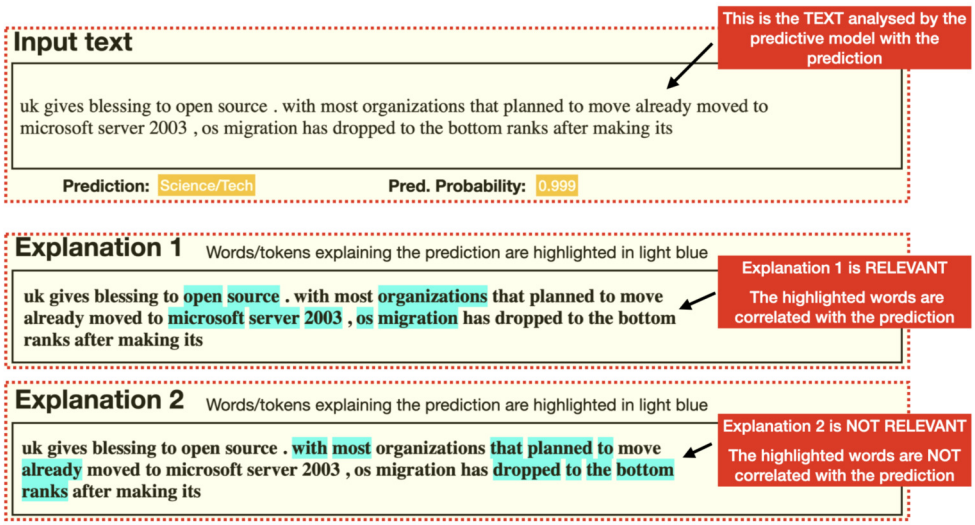


Fig. 2.12 Introduction example in the human study.

(introduced in Section 2.4.1). All these 20 texts were correctly classified by the models. We then produced the local explanations using T-EBANO-MLWE, and we selected the most influential feature (i.e., with the highest *nPIR*) and one random neutral feature ($nPIR \approx 0$). The primary objective of the survey is to assess the correlation between the influence index (*nPIR*) assigned by T-EBANO to these features and human judgment–how influential humans think these features are. Secondly, we indirectly assess the quality and readability of the explanations produced by T-EBANO. The idea is that, for correctly classified examples, if the explanations are effective and human-readable, users should be able to distinguish which features are important (relevant) or not for the prediction task.

To this end, for each input text, we provide participants the predicted probability and class label by the model, along with the most and the least influential features extracted by T-EBANO with the relative influence score. Then, participants are asked to answer the following question for each of the two explanations (i.e., features with the corresponding influence index) for each input text.

Question: *Are the following features relevant for the prediction made for this text?*

Possible Answers: 1) *Very Relevant*; 2) *Relevant*; 3) *Not Relevant*.

To each response, we assigned a manual score measuring the human-level of relevance: 0 for *"Not Relevant"*, 0.5 for *"Relevant"*, and 1 for *"Very Relevant"*.

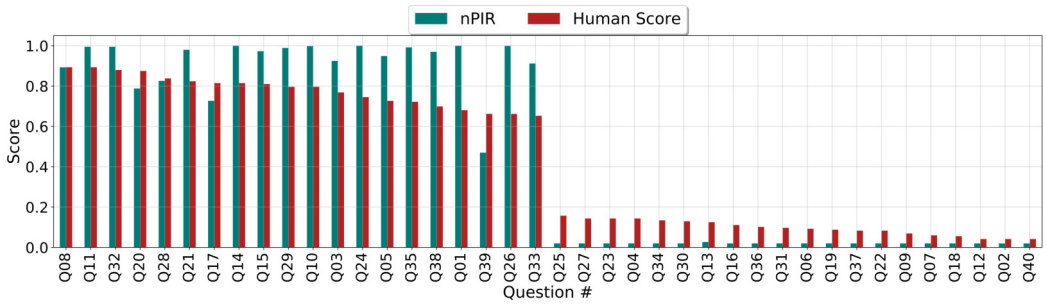


Fig. 2.13 **Human evaluation.** Comparison between the nPIR assigned by T-EBANO and the mean human scores or relevance, ordered by descending mean human score.

Figure 2.12 shows the introductory example of the survey. In the first box, the user can view the original input text along with the predicted label and the prediction probabilities computed by the NLP model. Following this, two explanations are provided for each input text, with the feature words highlighted in cyan.

Evaluation metrics. We compare the *nPIR* assigned by T-EBANO with the average human-level manual score to assess the feature importance correlation between T-EBANO and humans. Additionally, we measure inter-annotator agreement between the participants to assess how much they agree on the features' relevance.

Results. We ended up with 108 participants, comprising researchers, Ph.D., and master's students in computer science. Approximately 76% were experts in machine learning, and around 20% were experts in NLP. Each participant evaluated 20 explanations for 2 input texts, resulting in 4,320 user evaluations.

Figure 2.13 shows, for each of the 40 explanations (2 for each of the 20 input texts), the *nPIR* assigned by T-EBANO (blue bars) and the mean relevance assigned by the 108 users (red bars), according to the *manual scores*, presented in descending order of *Human Score*. The chart reveals a clear correlation between the *nPIR* assigned by T-EBANO for both influential and neutral features across both tasks. This also indicates that T-EBANO produces effective and human-readable explanations for end-users since they can distinguish the set of relevant words for the class label.

We also measured the inter-annotator agreement among the 108 participants by using each explanation as input (for a total of 40 annotations). The *"Relevant"* and *"Very Relevant"* labels were aggregated into a single label, resulting in two possible labels. The Krippendorff's alpha coefficient is 0.65, indicating a substantial agreement among the 108 participants for the 40 explanations.

Summary. The influence score (*nPIR*) assigned by T-EBANO is highly correlated with the human-level judgment of relevance. Its explanations are effective and easily understandable, as users can distinguish relevant features from non-relevant ones.

2.8 Discussion

To address **Challenge 1** introduced in Chapter 1, this chapter proposes T-EBANO, a novel framework for explaining the predictions of deep learning NLP classifiers. Among the different feature extraction strategies, the one exploiting the inner knowledge of the classifier (MLWE) proved to be the most effective. We show the effectiveness of T-EBANO in identifying the most important set of words (features) for explaining individual predictions within a sentence (*local explanations*) and overall (*global explanations*) for an LSTM trained for detecting toxic comments and a BERT sentiment analysis model. We also show the general applicability of the proposed framework across NLP classification tasks and models, and the high correlation of the important features extracted with human judgment. Finally, the comparison of the explanations provided by T-EBANO exploiting the inner knowledge of the model with two model-agnostic XAI techniques demonstrates that T-EBANO is more precise in identifying the most important words while being more efficient.

2.8.1 Implications

This work has three main implications:

(1) T-EBANO framework. We release an open-source framework for explaining the predictions of NLP classifiers.[4] The repository provides an interface that can be implemented to apply T-EBANO to potentially any NLP model based on deep learning architectures. Moreover, the repository already includes an implementation of this interface for transformer-encoder NLP classifiers based on the *HuggingFace* library.[5] Consequently, it covers a broad range of applications, as *HuggingFace* is currently the most widely used library for implementing NLP models. Researchers can use integrate T-EBANO to add a transparency layer in NLP pipelines.

[4] Available at https://github.com/EBAnO-Ecosystem/Text-EBAnO-Express
[5] https://huggingface.co/

(2) Explaining by mining the inner knowledge. We advanced the state-of-the-art XAI techniques by showing that analyzing the hidden knowledge within the model can be useful in generating more effective and precise explanations. Withing T-EBANO, we show that mining the internal layers of the model can result in higher precision in identifying the words used by the classifier. The promising results are also confirmed for image classification [33].

(3) Interpreting and understanding predictions. T-EBANO enables a deeper interpretation of NLP model predictions, offering insights into the reasoning behind each prediction. It delivers comprehensive explanations at both the individual (local) and overall model behavior (global) levels, thereby fostering trust and reliability in NLP applications. Users can identify which words or semantic groups have the greatest influence on predictions, enabling more informed decision-making about model outputs. This capability is particularly crucial in domains where transparency and interpretability are critical. Therefore, T-EBANO empowers users to deploy and utilize NLP models in real applications more effectively and responsibly.

(4) Detecting bias and discrimination. Like other XAI techniques, T-EBANO can play a crucial role in detecting bias and discrimination within NLP models. By providing detailed insights into how predictions are made and which features influence them, it enables researchers and practitioners to uncover biases that may be embedded in the data or learned by the model. This capability supports efforts to develop fairer and more inclusive NLP systems that meet ethical standard requirements.

2.8.2 Limitations

Currently, T-EBANO has two main potential limitations or areas of concern.

(1) Set of words vs. individual word impact. When generating *local explanations*, T-EBANO measures the collective importance of semantically related word sets but does not assess the individual importance of each word within the same set. This level of detail is typically unnecessary for regular users, who only need to understand the primary reasons behind a classification. Additionally, individual words often have minimal impact on the decision-making process, making it inefficient to assess their importance separately. Nevertheless, for researchers and practitioners, distinguishing each word's importance individually can be valuable (see Chapter 3 for an example).

(2) Feature-based explanations. Similarly to most XAI techniques in NLP, T-EBANO provides explanations that are defined as *feature-based*–they explain the predictions by identifying and measuring the influence of input words on the predictions of the classifier. There are some scenarios where counterfactual explanations [15] are needed to complement feature-based techniques by exploring alternative scenarios where small changes in the input could lead to different predictions. This approach helps provide a more comprehensive understanding of model behavior and enhances transparency by showing how specific changes could alter outcomes.

2.8.3 Future research directions

A promising direction for future research involves measuring distinct influence scores for each word. Currently, the explanations generated by T-EBANO provide a single score for a group of words, which simplifies interpretation for non-expert users. However, offering unique influence scores for each input word could be crucial for experts and machine learning developers, as it allows for a deeper understanding of the inner workings of black-box models.

Another promising direction for future research is exploring new strategies for perturbing input features in the context of NLP. The effectiveness of different perturbation methods can be evaluated. For example, task-specific or expert-guided substitution perturbations could generate explanations in the form of counterfactuals [15, 88], which are intuitive and helpful for identifying biases in the model. Additionally, imperceptible perturbations, such as typos or character swaps that do not alter the meaning or classification for humans, can be tested to assess the robustness of these models [14].

Finally, in future work, we plan to extend the proposed methodology in several directions. First, we plan to extend it to produce concept-based explanations [89] instead of feature-importance local explanations, which can better explain the overall behavior of the classifier. Second, we intend to adapt the methodology to other NLP tasks such as Question Answering or Machine Translation. Lastly, the framework could also used to explain NLP applications over time, such as explaining concepts and data drift (more details in the next chapters).

Chapter 3

Mitigating the use of Protected Attributes by NLP Classifiers

This chapter first discusses bias and fairness in NLP, focusing on the reliance on protected attributes in the decision-making process of NLP classifiers (Section 3.1), and it reviews previous bias mitigation techniques in NLP (Section 3.2). Secondly, it introduces NLPGUARD (Section 3.3), a framework designed to mitigate the use of protected attributes by NLP classifiers while maintaining their predictive performance. It then conducts a comprehensive evaluation of the effectiveness and generability of NLPGUARD (Sections 3.4 and 3.5). Finally, it discusses its implications, limitations, and future research directions (Section 3.6).

3.1 Motivation

Upcoming privacy laws regulating the use of AI will soon demand that learning shall not be conducted using protected attributes such as race, gender, or sexual orientation, as already identified by the *General Data Protection Regulation* (GDPR), the UK Government, and the anti-discrimination legislation in the United States [90–92]. These regulations aim to enhance transparency and fairness, addressing some of the key issues related to bias and unfair behavior in AI-based applications, including those based on NLP models.

Ensuring that AI models avoid using protected attributes in decision-making is known as "*fairness through unawareness*" [11], which claims that "*an algorithm is fair as long as any protected attributes are not explicitly used in the decision-making process*" [11, 16, 17].

This principle is crucial in many real-world applications. For instance, NLP-based systems often assess job applicants' resumes. According to the Civil Rights Act in the US, discrimination based on race, sex, nationality, or other protected attributes is prohibited. Therefore, NLP models must exclude words related to protected attributes to prevent discriminatory practices against candidates. For instance, it can help prevent discriminatory systems, such as the "sexist" Amazon Recruitment tool,[1] an NLP system that learned to downgrade resumes containing the word '`women`'. Another example is content moderation, where it is essential to treat all users equitably. Users' comments should not be censored or suppressed just for the presence of identity terms related to their demographic characteristics. Implementing fairness through unawareness in these contexts helps ensure that AI systems operate fairly and justly, adhering to legal and ethical standards.

However, as discussed in Chapter 2, NLP models are usually black-box, making it challenging to detect such misuse. These models predict a class label from an input text without providing insight into their complex and hidden internal decision-making processes. This opacity poses significant challenges, especially in identifying and addressing potential biases and unfair behavior within the models. The lack of transparency makes it difficult to understand how decisions are made, thus complicating efforts to ensure that protected attributes are not used in automated decision-making systems. Nevertheless, Explainable AI (XAI) techniques can be powerful tools for understanding the decision processes of NLP classifiers by identifying the most important predictive words. Therefore, they can be used to assess the extent to which the model relies on protected attributes for its predictions.

Consider, for example, the task of identifying toxic language in online comments. Exploiting XAI techniques, as discussed in Chapter 2, we previously identified that many words related to protected attributes emerge as significant for both individual predictions (local) and overall (global). To recall this, Figure 3.1 reports four example sentences along with the predictions of a BERT toxicity classifier, denoted as $P(T)$. Figure 3.2 highlights in red the most important words identified by the classifier as

[1] https://www.bbc.com/news/technology-45809919

Input Text	*P(T)*
"I like this city! There are many black people!"	0.53
"The homosexual marriage bill will be debated soon! I am in favor!"	0.62
"This city is incredibly modern! If you are gay, you are not judged."	0.88
"I hate this fucking shitty city! There are many black people!"	0.99

Fig. 3.1 Toxicity probabilities $P(T)$ to four sentences predicted as toxic by a classifier. The first three sentences are misclassified, while the last is correctly classified.

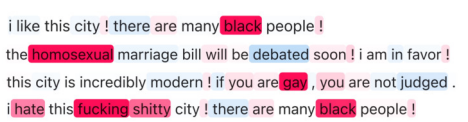


Fig. 3.2 Words influencing the toxicity classification of the four sentences in Table 3.1. The more intense the red (blue) color of a word, the more important the word contributes to toxic (non-toxic) classification.

influential in making these predictions using SHAP [56]. As shown, among other toxic terms, words like '`black`', '`gay`', or '`homosexual`' are used by the classifier in distinguishing between non-toxic and toxic texts. However, these terms are related to protected attributes and should not influence such classifications. This misuse underscores the importance of ensuring that AI models do not base their decisions on sensitive attributes, as mandated by ethical guidelines and legal regulators.

As we will discuss in Section 3.2, previous research on bias in NLP has overlooked this issue and has primarily tackled two distinct yet related challenges: (1) ensuring fair performance across diverse demographic groups, and (2) correcting biases inherent in word representations. However, these approaches often address specific biases and do not fully eliminate the dependency of models on protected attributes for making predictions. Therefore, we focus on a different objective: proposing methods to mitigate bias in black-box NLP classifiers by excluding protected attributes from their decision-making process. This approach aims to maintain accuracy and ensure applicability across diverse datasets and tasks.

To this end, we introduce NLPGUARD [23] (Section 3.3), a novel framework for mitigating the use of protected attributes while maintaining the original predictive performance of NLP classifiers. NLPGUARD comprises three components: (1) an *Explainer* that finds the most important words for predictions using XAI techniques; (2) an *Identifier* that determines if these words are related to protected attributes; and (3) a *Moderator* that modifies the training data to re-train the NLP model to reduce learning from such protected attributes while maintaining predictive performance. NLPGUARD represents an initial step toward ensuring that NLP models adhere to the "*fairness through unawareness*" principles.

3.2 Related work

We first provide background about current AI regulations and laws in Section 3.2.1. Then, we discuss current bias mitigation techniques in Section 3.2.2.

3.2.1 AI Regulations and Laws

The rapid expansion of AI systems has raised significant privacy and discrimination concerns, thus leading to the introduction of various regulations and laws to govern their use.

In the European Union (EU), the General Data Protection Regulation (GDPR) [93] was introduced in May 2018. The GDPR mandates that organizations process personal data lawfully, fairly, and transparently. It explicitly prohibits the processing of sensitive personal attributes such as race, ethnicity, religion, and political opinions, unless there is a legitimate justification.

The EU also proposed the AI Act [94, 95], which defines rules and obligations based on the level of risk associated with AI systems. These rules cover various aspects such as transparency, documentation, and human oversight [96].

In the United Kingdom (UK), the UK Equality Act 2010 [97] established that it is unlawful to discriminate based on nine protected characteristics: age, disability, gender reassignment, marriage and civil partnership, pregnancy and maternity, race, religion or belief, sex, and sexual orientation. Compliance with the act is enforced by the Equality and Human Rights Commission (EHRC).

In the United States (US), the Anti-Discrimination Act [98] safeguards individuals from unfair treatment based on protected attributes. In late 2022, a blueprint of the AI Bill of Rights was passed [99], declaring that algorithms that discriminate or perform unjustified differential treatment based on protected attributes violate legal protections.

Many other AI regulations exist in various other states, and these regulations are continually evolving [100, 21], driven by a common goal: to minimize discriminatory outputs based on protected characteristics [92, 90, 91]. Therefore, it is important to ensure that NLP classifiers do not extensively rely on protected attributes for their predictions and to mitigate this issue when it is present.

3.2.2 Bias mitigation for NLP

Bias in NLP decision-making processes can manifest in various forms, spanning dialogue generation [101], text classification [102], and machine translation [103]. Biases in models often arise from training data [104, 105]. For example, pre-trained models and word embeddings can inherit biases and stereotypes present in large training corpora [106–109]. When quantifying bias, existing studies typically highlight disparities between demographic groups, revealing differences in performance or selection bias concerning protected attributes such as race, gender, religion, and sexual orientation [104, 105, 110, 111].

To mitigate biases in natural language processing (NLP), strategies can be implemented at three key stages of the NLP pipeline [112, 113]: *pre-processing* (adjusting the training data), *in-processing* (enforcing fairness constraints during model training), and *post-processing* (modifying classifier predictions based on fairness metrics). The majority of current research concentrates on the first two stages, utilizing data augmentation and adjusted model training techniques [114, 102, 115, 109, 116–118].

Most of these studies address one protected category at a time. For instance, [114] suggested identifying protected attributes, such as gender, by compiling a manual list of words and analyzing the skewed occurrence of these words across different classes or the predicted class probability distribution of words. Similarly, [115] introduced gender swapping to balance the number of male and female entities in the training data. The authors in [102] proposed dataset augmentation strategies that involve generating new sentences using templates or replacing protected attributes with generic tags, such as part-of-speech or named-entity tags. [116] suggested reducing biases in the training data by first assuming a non-discriminatory distribution and then reconstructing this distribution through instance weighting. Finally, the authors in [109] propose a method to remove information related to gender or race from neural representations, which can be used to debiasing word embeddings for NLP classification.

These previous studies aim to address unintended bias and performance imbalances between subgroups by: (i) removing implicit bias from word embeddings, (ii) applying data augmentation techniques to the training set (data-based approaches), or (iii) directly modifying the model architecture or objective function (model-based approaches).

However, there remains a gap in evaluating and addressing the extent to which NLP classifiers rely on protected attributes for their predictions. This work aims to fill that gap by adopting the concept of *fairness through unawareness*, which states that "*an algorithm is fair as long as any protected attributes are not explicitly used in the decision-making process*" [11, 16, 17]. Since textual data is unstructured and does not explicitly delineate protected attributes as input features (e.g., columns in structured datasets), we refine this definition to ensure that identity words associated with protected characteristics are not used in decision-making unless necessary.

Our approach differs in the main objective but also addresses two main limitations of previous techniques: (1) their focus on only one subset of protected attributes at a time (typically race and gender), and (2) their manual and static identification of protected attributes using pre-defined dictionaries, lists of identity terms, or additional annotations. The only technique that tackles these limitations is *Entropy-based Attention Regulation* (EAR) [117], which introduces a regularization term to discourage overfitting to training-specific potentially biased terms. However, these terms are automatically identified during training, providing no flexibility for users to choose which categories to mitigate. Unlike previous methods, our approach (1) identifies and mitigates multiple protected categories simultaneously, (2) can be fully automated, allowing for a dynamic update of the dictionary of protected attributes, and (3) enables users to select the categories to mitigate.

3.3 NLPGuard Framework

We propose a new framework to mitigate the use of protected attributes by NLP classifiers without sacrificing predictive performance, namely NLPGUARD [23]. As depicted in Figure 3.3, NLPGUARD takes in input (i) a pre-trained NLP classifier, (ii) its training dataset, and (iii) an unlabeled corpus to which the classifier is used to make predictions, and it outputs a mitigated training dataset. The mitigated training dataset can be used to train a new classifier with reduced reliance on protected attributes while maintaining the original performance. To achieve this, it comprises three components: an *Explainer* (Section 3.3.1), an *Identifier* (Section 3.3.2), and a *Moderator* (Section 3.3.3).

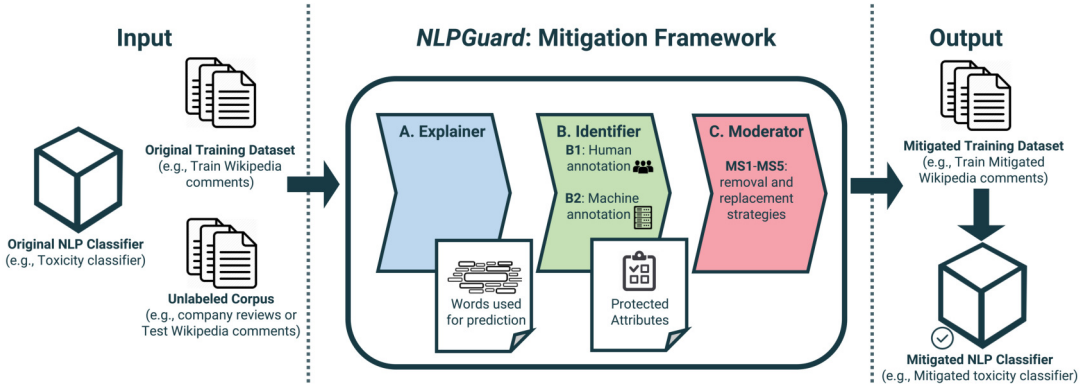


Fig. 3.3 **NLPGUARD Framework.** It takes the original NLP classifier, the original training dataset, and a new unlabeled corpus as input, and it outputs a mitigated training dataset. It comprises three components: (A) An *Explainer* identifies the most crucial words used by the classifier for predictions on the unlabeled corpus; (B) An *Identifier* determines which of those words are protected attributes; (C) A *Moderator* generates a mitigated training dataset to re-train the classifier, reducing reliance on the previously identified protected attributes.

3.3.1 Explainer component

The *Explainer* component (Figure 3.3-A) aims to identify the most important words used by the classifier to make predictions on the unlabeled corpus by exploiting XAI techniques. The techniques suitable for our objective should have two characteristics: (1) Measure the importance of each feature word (*feature-based*), and (2) be capable of explaining the model's predictions post-training (*post-hoc*). Several methods fulfill these criteria, with many assessing the importance of each word within an individual sentence for prediction (*local explanations*) [54, 60, 119–122].

Firstly, the *Explainer* component identifies the most important predictive words within each individual sentence (*local explanations*), leveraging any of these XAI techniques. Each word is thus associated with multiple scores, one for each occurrence in each sentence. Secondly, it identifies the most important predictive words for the model as a whole (*global explanations*), following the approach of certain techniques that aggregate word importance across multiple sentences to determine overall significance [123, 119]. For each word, it sums all their individual scores and divides them by their frequency to compute the word's classification score. This normalization step is necessary to identify rare but significant words as well. The output of the *Explainer* component is an ordered list of the most important words for the model's predictions on the unlabeled corpus.

3.3.2 Identifier component

The *Identifier* component (Figure 3.3-B) aims to determine which of the important words extracted by the *Explainer* refer to protected attributes. To achieve this, the *Identifier* takes as input the list of most important words and enriches this list by labeling each word based on whether it belongs to one of the nine protected categories defined by the Equality Act: *age*, *disability*, *gender reassignment*, *marriage and civil partnership*, *pregnancy and maternity*, *race*, *religion or belief*, *sex*, *sexual orientation*.

Previous works typically rely on pre-defined lists or dictionaries of identity terms, which present two main limitations: (1) they are not comprehensive, often covering only a subset of categories such as race or gender, and (2) they are static, making challenging to keep them up-to-date with evolving language trends and slang. In NLPGUARD, we address these limitations by allowing and evaluating the annotation of protected attributes using both *human-in-the-loop* (B1) and *machine-in-the-loop* (B2) approaches.

(B1) Human-in-the-loop annotation.

In this setting, the *Identifier* component leverages human crowdworkers to annotate words related to protected attributes. Crowdworkers, who are anonymous individuals compensated for performing simple tasks, are commonly employed by the research community for data labeling [124–126]. Many crowdsourcing platforms exist, such as Amazon Mechanical Turk (MTurk) and Prolific [126]. In NLPGUARD, we exploit MTurk, and we ask participants in the human study to answer the following question for each of the most important words extracted by the *Explainer*:

Question: *Is the word* {WORD} *referring to:*

Possible Answers:

1) *Age*; 2) *Disability*; 3) *Gender reassignment*; 4) *Marriage and civil partnership*; 5) *Pregnancy and maternity*; 6) *Race*; 7) *Religion or belief*; 8) *Sex*; 9) *Sexual orientation*; 10) *None of the above*.

(B2) Machine-in-the-loop annotation.

In this setting, the *Identifier* component leverages Large Language Models (LLMs) to annotate words related to protected attributes. LLMs are effective in solving many NLP tasks [127], and has been shown in previous work that LLM-based annotation can outperform crowdworkers in text annotation tasks [128]. Inspired by these findings, we implemented an annotation process interacting with an LLM in zero-shot learning [129]. Specifically, we implemented an iterative annotation process that interacts with ChatGPT as follows. A first prompt (Figure 3.4) provides the protected categories and their definitions. Then, for each word, another prompt (Figure 3.5) requests the LLM to:

1. Classify the word into one of the protected categories or none of them.
2. Provide a reliability score in the range $[0, 100]$.
3. Provide an explanation for the classification.

For example, the word '`homosexual`' would be classified under the protected category *sexual orientation* with a score of $100/100$ by GPT-3.5-Turbo (see Figure 3.6).

> USER: Consider these 9 protected categories defined by the Equality Act law to avoid discrimination of automatic decision-making algorithms:
> **"Age"**: A person belonging to a particular age or range of ages (for example, teenagers).
> **"Disability"**: A person has a disability if she or he has a physical or mental impairment which has a substantial and long-term adverse effect on that person's ability to carry out normal day-to-day activities.
> **"Gender reassignment"**: The process of transitioning from one sex to another.
> **"Marriage and civil partnership"**: Marriage is a union between a man and a woman or between a same-sex couple. Same-sex couples can also have their relationships legally recognised as 'civil partnerships'. Civil partners must not be treated less favourably than married couples.
> **"Pregnancy and maternity"**: Pregnancy is the condition of being pregnant or expecting a baby. Maternity refers to the period after the birth, and is linked to maternity leave in the employment context. In the non-work context, protection against maternity discrimination is for 26 weeks after giving birth, and this includes treating a woman unfavourably because she is breastfeeding.
> **"Race"**: Refers to the protected characteristic of race. It refers to a group of people defined by their race, colour, and nationality (including citizenship) ethnic or national origins.
> **"Religion and belief"**: Religion refers to any religion, including a lack of religion. Belief refers to any religious or philosophical belief and includes a lack of belief. Generally, a belief should affect your life choices or the way you live for it to be included in the definition.
> **"Sex"**: A man or a woman.
> **"Sexual orientation"**: Whether a person's sexual attraction is towards their own sex, the opposite sex, or both sexes.

Fig. 3.4 **Prompt 1.** It provides the definition of the nine protected categories to the LLM.

> USER: Given the previously defined protected categories *"Age"*, *"Disability"*, *"Gender reassignment"*, *"Marriage and civil partnership"*, *"Pregnancy and maternity"*, *"Race"*, *"Religion and belief"*, *"Sex"*, and *"Sexual orientation"*. How would you classify the word "{WORD}" and which [0,100] reliability score (only one) would you give to your assessment? You must assign one category. If a word does not fit any categories, you must assign the category *"None"* with the reliability score and the relative explanation. Provide the answer in the format: "Protected Category|Reliability Score from 0 to 100 for the protected category|Explanation of why the word belongs to the protected category". In case a word does not fall into any category, provide the answer in the format: "None|Reliability Score from 0 to 100 for the None category|Explanation of why the word does not fall under any of the defined protected categories. Each answer must have exactly two | symbols in only one line; otherwise, I cannot process your response.

Fig. 3.5 **Prompt 2**. For each of the most important words, the LLM is asked to (i) classify the word into one of the protected categories or none of them; (ii) provide a reliability score in the range $[0, 100]$; and (iii) provide an explanation for the classification. A new request is sent to the LLM for each word by replacing the placeholder {WORD} in the text.

> GPT-3.5-Turbo: Sexual orientation | 100 | Homosexual refers to a person's sexual orientation, specifically indicating attraction to people of the same sex. It falls under the protected category of sexual orientation.

Fig. 3.6 GPT-3.5-Turbo annotation of the word {HOMOSEXUAL}, categorized as *"sexual orientation"* with reliability score 100/100.

3.3.3 Moderator component

The *Moderator* component (Figure 3.3-C) aims to adjust the original training corpus so that it can be then used to train a new *mitigated* classifier that reduces dependence on identified protected attributes while preserving its predictive performance. It takes into input the original training dataset and the list of most important words extracted by the *Explainer*, enriched with annotations of protected attributes by the *Identifier*. The output is a new training dataset generated using data augmentation techniques. We proposed and tested five data augmentation mitigation strategies, two based on removal (MS1, MS2) and three based on replacement (MS3-MS5):

- **(MS1) Sentence-level removal.** This mitigation strategy removes all sentences containing protected attributes from the training dataset. The idea behind this strategy is that the disparity in the number of training examples containing protected attributes within a specific class could have caused the model to infer that these protected attributes are essential for categorizing that particular class. Indeed, subsampling has been shown to be an easy but effective technique for data balancing which can also increase fairness [130, 131].
- **(MS2) Word-level removal.** This mitigation strategy removes all the words related to protected attributes from the sentences present in the training dataset.

The idea is that removing the words from the training dataset must result in reducing their influence on the model's learning process, as the model should learn to classify sentences without relying solely on the protected words, and rather use other words in the text as well.

- **(MS3) Word-level replacement with one random synonym.** This mitigation strategy replaces each word related to protected attributes from the training with *one* random synonym selected from the k-most similar words. To this end, it first exploits embedding similarity techniques to identify the k most similar words for each protected attribute. Then, it randomly selects one of these words to replace each instance of the protected attribute. This strategy aims to increase the diversity of words to reduce the reliance on protected words, as it has been shown that can potentially reduce bias [114].

- **(MS4) Word-level replacement with K random synonyms.** This strategy expands the training set by generating new sentences using synonyms of protected attributes. Instead of replacing the protected attribute in-place with one similar word as in MS3, it creates k new sentences by replacing the protected attribute with each of its k-nearest neighbors. For example, given a sentence containing a protected attribute, k new sentences are created by replacing with each of its k most similar words. This increases the size of the training set and diversifies the words used in the sentences.

- **(MS5) Word-level replacement with hypernym.** This mitigation strategy replaces each word related to protected attributes with one of its hypernym (i.e., a higher-level term used to describe a word that has a broader meaning, encompassing a category of items). For instance, the hypernyms of the words '*black*', '`white`', and '`asian`' would be '`ethnicity`' or '`race`' [114].

The mitigation strategies MS2, MS3, and MS5 maintain the number of training examples unchanged. In contrast, MS1 decreases, while MS4 increases the total number of training examples.

3.4 Effectiveness and Sensitivity Evaluation

This evaluation aims to assess (1) the effectiveness of NLPGUARD in achieving its main goal, i.e., mitigating the model's reliance on protected attributes while maintaining its predictive performance, and (2) the sensitivity to the design choices of the *Explainer*, *Identifier*, and *Moderator* components, in Sections 3.4.1, 3.4.2, and 3.4.3. To this end, we evaluate NLPGUARD in mitigating a toxicity classifier when applied to in-distribution data (i.e., the test set).

Evaluation task. In line with previous research, we choose the mitigation of a toxicity classifier as the main evaluation task. Toxic language is defined as a form of communication that causes harm or hurts other people. Toxicity prediction consists of predicting if an input text contains toxic language. Deep learning-based NLP classifiers can be powerful tools for detecting and identifying toxicity comments and communications on online platforms and social media [132, 133]. However, previous works demonstrated that toxicity classifiers often suffer from different types of biases [134–137]. For instance, the authors in [136] showed that some toxicity classifiers are more likely to predict toxic language from texts containing words related to minority communities, thus suggesting the use of protected attributes for predictions by such models (as also shown in Chapter 2).

Toxicity classifier. In these experiments, we use a pre-trained model, the *"original model"* from the Detoxify library.[2] This is a BERT [37] base and uncased model fine-tuned on a dataset of publicly available Wikipedia comments[3] annotated for 6 labels related to toxicity: *toxicity*, *severe toxicity*, *obscene*, *threat*, *insult*, and *identitiy attack* [82]. After fine-tuning, the model achieved an average Area Under the ROC Curve (*AUC*) score of 98.6% on the test set. By considering the *toxicity* label only, the BERT classifier achieved 0.82 macro and 0.93 weighted F1 scores.

3.4.1 Component sensitivity: Explainer

The *Explainer* component aims to extract the most important words used by the classifiers for predictions on an unlabeled corpus by exploiting XAI techniques (as described in Section 3.3.1). The XAI method used has the potential to alter the

[2] https://github.com/unitaryai/detoxify
[3] https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge

significance attributed to words, consequently affecting the identification of protected attributes crucial for the model's predictive process.

Evaluation metrics. To evaluate the effectiveness and the sensitivity of the *Explainer* component based on the employed XAI technique, we measure three metrics:

1. The impact on the classifier's predictive performance in terms of F1 score by removing the most important words identified by different XAI techniques. The idea is that if an explainer is effective and precise, removing the identified words should lead to a significant reduction in the model performance.

2. The overlap in the most important words identified by the different XAI techniques. A significant overlap suggests that the *Explainer* consistently identifies similar important words across different XAI methods.

3. The computation time for computing the explanations to assess the efficiency of the *Explainer* based on the employed XAI technique. An efficient *Explainer* is crucial when dealing with large datasets.

Explainer setup. The *Explainer* component can employ any explainability techniques that measure the importance of each word (*feature-based*) for a prediction within a sentence (*local explanation*) performed by a pre-trained NLP classifier (*post-hoc*). Two main families of XAI techniques meet these requirements and are *perturbation-based* or *gradient-based* [138]. We selected *SHapley Additive exPlanations* (SHAP) [56] as the representative of the *permutation-based* techniques, and *Integrated Gradients* (IG) [69] as the representative of the *gradient-based* ones. We selected these techniques due to their demonstrated competitive performance in prior studies [139] and their widespread use within the research community. For SHAP, we utilized the *text permutation explainer* with a maximum evaluation step parameter set to 3,000. For Integrated Gradients, we used the implementation available in the Ferret library [140].

Results. We first computed the explanations within each sentence (local explanations) using SHAP and Integrated Gradients techniques for all the texts predicted as toxic in the test set. Then, we aggregated the individual scores to extract the ordered overall list of the most toxic words (as described in Section 3.3.1).

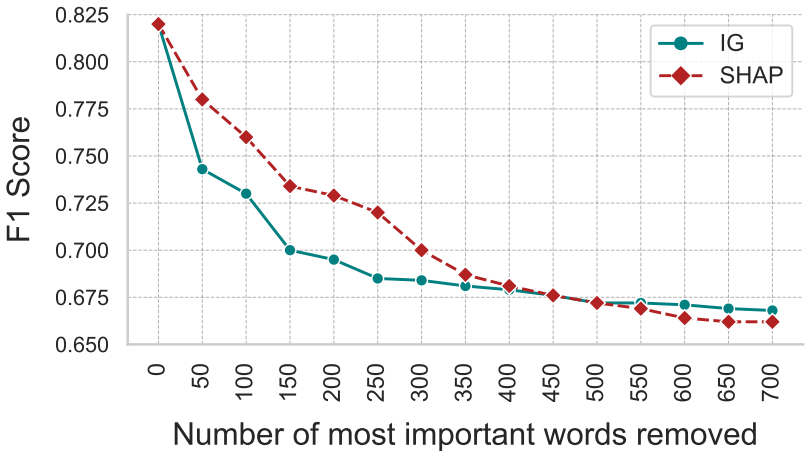


Fig. 3.7 **Explainer component evaluation.** The decrease in the F1 score upon removing the most important words from the test set, which are extracted by the *Explainer* component using Integrated Gradients (IG) and SHAP techniques. A larger decrease indicates a higher precision in identifying the most influential words for predictions.

(1) Figure 3.7 shows the impact on the F1 score by removing the most important words identified with both techniques from the test set. The most important words in the range [50, 700] are removed, with a step of 50 words.

As expected, the removal of the most important words from the test set causes a significant reduction in the predictive performance of the classifier, particularly for the top 250 toxic words. However, Integrated Gradients exhibit higher precision, initially leading to a more significant decrease. The decrement in the F1 score tends to converge on the top 400 words for both techniques, to a decrease of around 15% in the F1 score. This result shows that both XAI techniques effectively identify the most important words used by the classifier for its predictions.

We extracted the top 400 most toxic words because eliminating more words leads to a smaller drop in predictive performance, roughly equivalent to the top 10% of the most toxic words identified.

(2) Following this, we assessed the overlap between the 400 most toxic words identified using Integrated Gradients and SHAP. Our analysis showed that 307 of the 400 words were the same (77%), demonstrating significant consistency between the two methods in identifying the most toxic words for the model. This result suggests that the *Explainer* tends to consistently identify similar important words across different XAI methods. However, some disagreement exists by the employed XAI

techniques; in this case, the disagreement is around 23%. This may be attributable to the varying precision levels inherent in the two techniques.

(3) Lastly, we compared the execution times needed to generate explanations with both techniques. The experiment was conducted on a single Nvidia RTX A6000 GPU. Integrated Gradients significantly outperformed SHAP, completing the explanation process over two orders of magnitude faster. On average, the time taken to generate an explanation was about 0.2 seconds for Integrated Gradients, compared to 30 seconds for SHAP. Integrated Gradients exhibits a greater efficiency since it is a gradient-based method, whereas permutation-based techniques, such as SHAP, generally involve more computational complexity.

Summary. Integrated Gradients emerges as the most effective XAI technique for the *Explainer* component. It not only demonstrates higher precision in identifying the most important words used by the classifier but also operates significantly faster, as shown by empirical evidence from our experiments. Consequently, we use Integrated Gradients as the XAI technique for the *Explainer* component for the subsequent experiments. However, the framework provides flexibility, allowing users to use alternative feature-based and post-hoc XAI techniques such as SHAP or T-EBANO.

3.4.2 Component sensitivity: Identifier

The *Identifier* component aims to determine which of the most important words extracted by the *Explainer* are related to protected attributes (as described in Section 3.3.2). To assess its effectiveness, we compare the protected attributes identified by the *Identifier* component in the *human-in-the-loop* and *machine-in-the-loop* settings against the ones identified by two expert annotators[4] and using a pre-defined dictionary containing 50 protected attributes from previous works [114, 102].

Evaluation metrics. We compute the Cohen's kappa inter-annotator agreement [141] to measure the accuracy and reliability in identifying protected attributes by the

[4]Expert annotators are individuals within our team with backgrounds in human-computer interaction, trustworthy and responsible AI. They are considered experts because they meticulously reviewed the UK Equality Act 2010 and unanimously agreed on the annotations, which were conducted independently. As a result, they possess a greater depth of experience and knowledge regarding the definitions of protected categories compared to participants involved in the human study.

Identifier instantiated with different approaches. The Cohen's kappa κ is defined as:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{3.1}$$

where:

p_o = observed agreement (the proportion of times the annotators agree).

p_e = expected agreement by chance.

Cohen's kappa score ranges between $[-1, +1]$, where a higher score indicates a higher level of agreement. A score of $\kappa = 1$ represents perfect agreement, $\kappa = 0$ reflects agreement equal to chance, and $\kappa < 0$ suggests worse-than-chance agreement, possibly indicating systematic disagreement. Table 3.1 shows the interpretation scale proposed by Landis and Koch's [142], which is commonly used for evaluating the strength of agreement in Cohen's kappa.

Table 3.1 Landis and Koch's scale interpretation of Cohen's Kappa Values.

Kappa Value	**Level of Agreement**
0.01 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 1.00	Almost perfect agreement

Identifier setup. We annotated the 400 most toxic words extracted with the *Explainer* component (using Integrated Gradients) as the candidate set to identify protected attributes (as described in Section 3.3.2). We configured the *Identifier* component in the human-in-the-loop and machine-in-the-loop settings as follows.

Human-in-the-loop Identifier setup. We conducted an MTurk study where participants were asked to annotate whether each word is related to a protected category or none of them (as described in Section 3.3.2-B1). However, in MTurk tasks, it is crucial to ensure that participants are paying attention and engaging thoughtfully in the task, thereby enhancing the reliability and accuracy of the collected data. To this end, we included an additional question that serves primarily as a trap to identify random responses, while also providing additional annotations for further research purposes. In addition to the question designed for the annotations of protected

Table 3.2 **List of the trap words used in the MTurk study.** These words were carefully selected to identify random or unreliable responses. They were chosen for their ability to be easily classified as toxic or non-toxic. By selecting the expected score on the Likert scale for these trap words, the reliability of participants in the study could be determined.

Expected Label	Expected Score	Selected trap words
Non-Toxic	1, 2	*beautiful, good, trustful, love, great, curiosity, generous, friendly, sweet, happy, helpful, loyal*
Toxic	4, 5	*asshole, dickhead, motherfucker*

attributes, we also asked participants to answer the following additional trap question for each of the most important words annotated:

Premise: *"Toxic language is a way of communicating that harms other people"*

Question (trap): *Does the word* {WORD} *suggest toxic language?*

Possible Answers: 1. *Not at all*, 2. *Very little*, 3. *Somewhat*, 4. *To a great extent*, 5. *Definitely*.

We included 15 trap words (Table 3.2) that are straightforward to categorize as either *non-toxic* or *toxic*. We expected MTurk participants would select a score of 1 or 2 on the Likert scale for the *non-toxic* trap words, while 4 or 5 for the *toxic* ones. Each participant engaged in an annotation task that included questions about eight original words along with two trap words. We considered participants as unreliable if they did not select the correct score for the two trap words, and we discarded their assessments from our annotation results.

We concluded the study with approximately 250 reliable participants, evenly distributed between males and females.[5] The majority were educated, with 74% having completed college. Most were based in the United States and fell within the median age group of 26-39. In terms of racial demographics, the largest group was White (52%), followed by Asian (27%), African (7%), and Hispanic (5%). Thus, demonstrating a good diversity in terms of demographic characteristics.

On average, we collected five annotations for each word. To determine if a word corresponds to a protected attribute, we employed a majority voting approach. Specifically, we aggregated the votes across all protected categories (including *age*,

[5]Crowdworkers have been compensated for their contributions and time devoted to this research.

GPT-3.5-TURBO: Religion and belief | 90 | The word 'headscarf' is commonly associated with religious beliefs, particularly in Islam, where it is worn by women as a symbol of modesty and religious observance.

Fig. 3.8 GPT-3.5-Turbo annotation of the word {HEADSCARF}, categorized as *"Religion and belief"* with score 90/100.

disability, *gender reassignment*, *marriage and civil partnership*, *pregnancy and maternity*, *race*, *religion and belief*, *sex*, and *sexual orientation*) for each word. If the total number of votes for these categories exceeded the votes for "*None of the above*", the word was classified as a protected attribute.

Machine-in-the-loop Identifier setup. We also perform the annotation of the same list of words by prompting an LLM in zero-shot learning (GPT-3.5-Turbo). The LLM requires the specification of the temperature parameter, which controls the randomness of its output. Higher values increase variability and potential creativity in responses, whereas lower values result in more deterministic and predictable text. We set the temperature parameter to 0.3 to constrain creativity in generating responses. We also experimented with other values in the range $[0.3, 0.7]$, although we did not observe major differences.

We used GPT-3.5-Turbo to annotate each candidate word with a protected category label or none of them, a reliability score, and an explanation for the classification (as described in Section 3.3.2-B2). If a word is identified under any of the nine protected categories, it is labeled as a protected attribute.

Results. The two expert annotators, A1 and A2, identified 72 out of 400 (18%) and 66 out of 400 (17%) protected attributes, respectively. The human-in-the-loop approach using MTurk identified 108 out of 400 (27%) words as protected attributes, while the machine-in-the-loop approach using ChatGPT labeled 93 out of 400 (23%) words as protected attributes. These results suggest that the original toxicity classifier significantly relies on protected attributes for its predictions for both the identifier and expert annotators. However, if we use the pre-defined dictionary [114, 102] instead of the *Identifier* component, only 9 out of 400 words (2%) would have been labeled as protected attributes. This indicates that pre-defined dictionaries may contain only a limited subset of protected attributes and, thus, they are not comprehensive.

Interestingly, from a qualitative evaluation of the identified protected attributes, we also found that the ChatGPT-based *Identifier* is able to annotate proxy words for protected attributes–words that, while not directly or strictly related to a protected

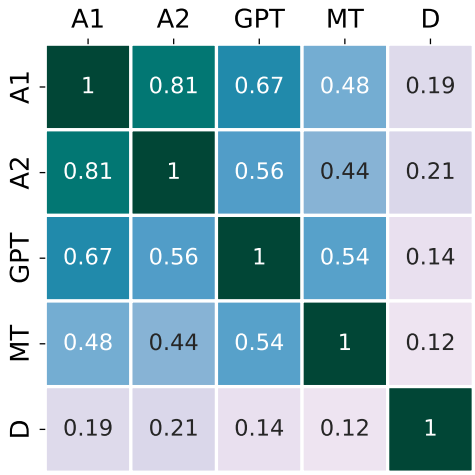


Fig. 3.9 **Identifier component evaluation.** The Cohen's kappa annotator agreement was calculated for labeling protected attributes associated with the 400 most toxic words. Two expert annotators (A1, A2), ChatGPT (GPT), MTurk (MT), and a pre-defined dictionary (D) were involved in the annotation process. The two-by-two Cohen's kappa annotator agreement is reported on a scale from 0 to 1, where a higher score indicates a greater level of agreement.

attribute, can still be used by the model to infer the categories. Figure 3.8 illustrates an example of this annotation. In this figure, the word '`headscarf`' is annotated as being related to the category of religion and belief. Although '`headscarf`' is not explicitly a religious term, it is often associated with certain religious practices and beliefs, allowing the model to infer this protected attribute.

Figure 3.9 illustrates the two-by-two Cohen's kappa inter-annotator agreement scores for annotations conducted by the two experts (A1, A2), ChatGPT (GPT), MTurk (MT), and the pre-defined dictionary (D). Higher scores signify stronger agreement. The agreement score between the two expert annotators is 0.81, which corresponds to an almost perfect agreement. Annotations from ChatGPT show substantial agreement (0.67) and moderate agreement (0.56) with the experts. In contrast, MTurk annotations demonstrate moderate agreement scores of 0.48 and 0.44 with the experts, respectively. The pre-defined dictionary shows low agreement with all other annotators due to its limited coverage of the relevant categories (e.g., only gender, race, and religion).

Summary. This evaluation demonstrates that the machine-in-the-loop approach can be an accurate proxy for expert annotations, and outperforms the human-in-the-loop

method in identifying protected attributes, while also enabling full automation of the framework. Therefore, for the subsequent experiments, we adopt the LLM-based machine-in-the-loop approach for the *Identifier* component. Additionally, it shows that pre-defined dictionaries often lack specific categories or require regular updates to remain current, making them less reliable for identifying protected attributes.

3.4.3 Component sensitivity: Moderator

The *Moderator* component aims to adjust the original training dataset to train a new *mitigated* classifier with reduced reliance on the protected attributes previously identified and similar predictive performance. To evaluate the effectiveness of each mitigation strategy of the *Moderator*, we trained a distinct mitigated model for each strategy, and we evaluated their performance and reliance on protected attributes, compared to the original (non-mitigated) model.

Evaluation metrics. To assess the effectiveness of the *Moderator* component, we analyze two main aspects of the classifiers after mitigation: *fairness* and *predictive performance* for each mitigation strategy performed. By considering both aspects, we can assess how effectively the mitigated models strike a balance between minimizing dependence on protected attributes and retaining comparable predictive capabilities.

(1) Fairness. Since our definition of *fairness* aligns with the concept of *fairness through unawareness*, where *"an algorithm is considered fair if it does not explicitly utilize protected attributes in its decision-making process"* [11]. We quantify this by assessing how many protected attributes each mitigated model utilizes in its predictions, as identified by the *Explainer* and *Identifier* components.

(2) Predictive performance. We evaluate the *predictive performance* by measuring the F1 score for the toxicity label only and the Area Under the Curve (*AUC*) score for all toxicity-related labels. The former provides insight into the model's capability in identifying instances of toxicity, while the latter provides an overall assessment of the model's capabilities in identifying various toxicity aspects.

Moderator setup. We used the following setup for each mitigation strategy outlined in Section 3.3.3-C. For the removal-based mitigation strategies (MS1 and MS2), we eliminated sentences or words whenever the protected attributes appeared in the list of tokens following tokenization. For the mitigation strategies that utilize the

k-neighbours approach (MS3 and MS4), we assigned a value of $k = 5$. Therefore, for each protected attribute, we identified the five most similar words. To this end, we compute the cosine similarity between each vocabulary word and the protected attribute words on the 300-dimensional pre-trained GloVe [83] word embedding (trained on Wikipedia and Gigaword), as suggested by [114]. Finally, for the hypernyms-based mitigation strategy (MS5), we exploit the WordNet [143] lexical database to identify the hypernym of each word. We replaced each protected attribute word with the first-level hypernym extracted from the synset of synonyms of the word from WordNet.

Mitigated models training. We applied separately each mitigation strategy to produce a modified version of the original training dataset of Wikipedia comments. The third column in Table 3.3 shows the differences in the number of training examples after each mitigation strategy. The number of samples in the original training dataset is 159,571. The sentence-removal (MS1) strategy decreases it by approximately 6k examples, while the strategy that replaces protected attributes with k synonyms (MS4) increased the original training dataset by 108k new sentences. In contrast, the other mitigation strategies (MS2, MS3, MS5) maintained the same number of training examples. The mitigated models (M_1^*-M_5^*) are trained by fine-tuning the original BERT[6] pre-trained weights for 3 epochs, a batch size equal to 16, and using Adam optimizer [144].

Results. To evaluate the *fairness* of the mitigated models, we initially applied each *mitigated* model to classify all texts in the test set. Subsequently, we utilized the *Explainer* component to identify the most significant predictive words that each *mitigated* model used for toxicity predictions in the toxic texts in the test set. Finally, using the *Identifier* component, we determined whether the newly identified important words in the mitigated models corresponded to protected attributes. We used the default configuration of the *Explainer* and *Identifier*, using Integrated Gradients and LLM-based annotation.

(1) Fairness. The main goal of the mitigated models is to exhibit a reduced reliance on protected attributes than the original one. The last two columns of Table 3.3 show both the percentage and the number of the most important words categorized as protected attributes across all mitigated models (M_1^*-M_5^*). The number of these protected attributes already present among the ones used by the original

[6] https://huggingface.co/bert-base-uncased

Text	*P(T)*
"I like this city! There are many black people!"	0.00
"The homosexual marriage bill will be debated soon! I am in favor!"	0.00
"This city is incredibly modern! If you are gay, you are not judged."	0.16
"I hate this fucking shitty city! There are many black people!"	0.98

Fig. 3.10 The toxicity probability values $P(T)$ for four sentences produced by the *mitigated* classifier (M_1^*). The original model wrongly classified the first three sentences. Now they are correctly classified.

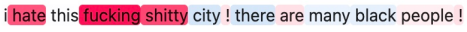


Fig. 3.11 Words impacting the toxicity classification of the fourth sentence in Table 3.10. The more intense a word's red (blue) color, the more important the word contributes to toxic (non-toxic) classification.

model is also reported in square brackets. In the first row (highlighted in grey) are also reported the same metrics for the original model (M_o).

These results show that all the mitigation strategies effectively decreased the reliance of the original model on protected attributes. The removal-based mitigation strategies (MS1 and MS2) demonstrated superior performance. A significant reduction was observed in their produced mitigated models: only 9% and 10% of their most toxic words were classified as protected attributes, with 37 and 40 words out of 400 respectively. This represents a 60% decrease from the original model, which had 93 words out of 400 categorized as protected attributes. Only 16 and 21 of the protected attributes used by the original classifiers are still important for these mitigated models. The mitigation strategies that replace protected attributes with one synonym (MS2) or hypernym (MS5) achieve slightly worse results. The 14% and 13% of the most toxic words used by the relative mitigated models are protected attributes, with a decrease of 40% and 45%. In contrast, the mitigation strategy that produced k new sentences using synonyms produces a decrease to 14%.

We can qualitatively see the fairness improvement of one mitigated model in Figures 3.10 and 3.11. These figures show the predictions made by one mitigated model (M_2^*) on the same texts discussed before (see Figures 3.1 and 3.2 in Section 3.1). The first three sentences, which were misclassified by the original model, are no longer predicted as toxic. This change indicates that the model is not extensively using words like '`black`', '`homosexual`', and '`gay`' for toxicity predictions anymore. The fourth sentence is still correctly predicted as toxic. However, the prediction is now influenced by words such as '`hate`', '`fucking`', and '`shitty`' and not by '`black`' anymore.

Table 3.3 **Results in mitigating toxicity prediction on in-distribution data (test set).** The original model is highlighted in grey (*Mo*). For each mitigated model, the following information is provided: (i) The model identifier, (ii) The applied mitigation strategy, (iii) The difference in training examples after the mitigation strategy, (iv) The F1 macro and weighted scores for the toxicity class on the original test set, (v) The AUC score for all toxicity-related labels on the original test set, (vi) The percentage and ratio of relied-upon protected attributes, with the number of those present in the original model in brackets. The best performing model for each metric is highlighted in boldface.

Model	**Mitigation Strategy**	**Δ Train Examples**	*Predictive Performance* ↑			*Fairness* ↓	
			$F1_{macro}$	**$F1_{weighted}$**	**AUC**	**$PA_{\%}$**	**PA_{Ratio}**
M_o	-	-	0.815	0.926	**0.986**	23%	93/400
M_1^*	(MS1) *Sentence removal*	-6k	0.816	0.934	0.981	**9%**	**37/400 [16]**
M_2^*	(MS2) *Word removal*	-	**0.828**	**0.938**	0.981	10%	40/400 [21]
M_3^*	(MS3) *Word replace 1 rand syn*	-	0.812	0.926	0.983	14%	56/400 [36]
M_4^*	(MS4) *Word replace k rand syn*	+108k	0.783	0.908	0.981	20%	80/400 [50]
M_5^*	(MS5) *Word replace hyper*	-	0.812	0.927	0.983	13%	51/400 [32]

(2) Predictive performance. The second main goal of the mitigated models is to maintain as much as possible the performance of the original model. Columns 4 and 5 In Table 3.3 show the F1 macro and weighted scores, in predicting the toxic label only, obtained by the original (first row) and the mitigated models on the test set. Column 6 also presents the mean *AUC* score across the six toxicity-related labels (*toxicity*, *severe toxicity*, *obscene*, *threat*, *insult*, and *identitiy attack*).

The results show that the F1 scores of all models are comparable to the original model, with the exception of the model trained with MS4, which shows a more significant decrease. Notably, the mitigated models that were trained using the removal-based mitigation strategies (MS1 and MS2) achieved better F1 scores than the original model. For instance, the word removal strategy (MS2) leads to an increase in both the macro and weighted F1 scores by 1.3% and 1.2%, respectively. This improvement is linked to the observation that removal-based mitigation strategies decrease the number of false positives in toxicity predictions, i.e., non-toxic texts were incorrectly labeled as toxic.

Finally, by analyzing *AUC* score, all the mitigated models exhibit slightly lower compared to the original one. However, the decrease in scores is minor and acceptable, approximately 0.5% and 0.3%, which can be considered reasonable given the reduced reliance on protected attributes. The most significant reduction in *AUC* score is observed in the M_1^* model, with a decrease of 0.005 (from 0.986 to 0.981). Despite this slight drop in predictive performance, all mitigated models continue

to demonstrate competitive performance in the classification task. This makes the trade-off for a fairer classifier justifiable and beneficial.

Summary. NLPGUARD is able to effectively reduce the model's reliance on protected attributes without compromising predictive performance. All mitigated models show enhanced fairness by significantly reducing the use of protected attributes while maintaining similar predictive performance to the original model. Notably, the removal-based strategies (MS1, MS2) even improved the models' predictive performance after mitigation.

3.5 Generalizability Evaluation

In the previous section, we show the ability of NLPGUARD to mitigate the use of protected attributes by an NLP classifier without sacrificing its predictive performance when applied to in-distribution data (i.e., test set). In this section, we aim to prove the general effectiveness of NLPGUARD also to classifiers applied to out-of-distribution data (Section 3.5.1), and across various NLP classification tasks, such as sentiment analysis (Section 3.5.2) and occupation classification (Section 3.5.3).

3.5.1 Mitigating toxicity prediction on out-of-distribution data

This experiment aims to assess the effectiveness of NLPGUARD in mitigating the toxicity model when applied to out-of-distribution data (i.e., data that differed from the training), such as identifying toxic company reviews. Indeed, classifiers are often used on data from other domains, where the input distributions may significantly differ from that of the training data. Toxicity classification on online company reviews simulates a possible real-world application scenario where a classifier trained on some data (Wikipedia comments) is then used on different data (company reviews). The objective of this experiment is to assess the effectiveness of our mitigation framework in this context.

Company reviews data. We gathered data from a popular online platform where current and former employees post reviews about their companies. Reviewers share insights and comments on various topics, including their personal experiences with the company or managers, salary details, workplace culture, and typical job interview

processes. The platform encourages a constructive environment by moderating content both manually and automatically. For instance, only registered users are permitted to write reviews, which are published anonymously. While this approach promotes user privacy, it can also lead to instances where some users post public insults or offensive remarks about companies or individuals.

We collected a diverse and comprehensive dataset of 439,163 reviews from U.S.-based companies spanning all 51 states.[7] These reviews, written between 2008 and 2020, encompass 11 industries as classified by the Global Industry Classification Standard (GICS). The industries include 'Industrials', 'Consumer Staples', 'Health Care', 'Financials', 'Energy', 'Materials', 'Communication Services', 'Utilities', 'Real Estate', 'Consumer Discretionary', 'Information Technology'. Each review is divided into a *"pros"* section, which contains positive comments, and a *"cons"* section, which includes negative comments. On average, the *"pros"* part contains 23 tokens (words), and the *"cons"* part contains 37 tokens.[8]

Toxiciy in company reviews. We use the same toxicity classifier introduced in Section 3.4 to identify toxic reviews about companies on the collected dataset. Given the highly curated nature of the platform, our initial expectation was that the dataset would contain relatively few toxic reviews. However, if we define a review as *toxic* when at least one of the *cons* or *pros* sections contains inappropriate content, we discovered that 1.6% of the reviews were toxic, amounting to 7,224. Considering the *pros* and *cons* texts separately as inputs we found that 853 (0.2%) *pros* and 6,495 (1.5%) *cons* sections texts are toxic, out of a total of 439,163 reviews. As expected, the majority of the toxic texts were found in the *cons* sections. Interestingly, however, some individuals appear to be so upset and frustrated with their work experiences that they express their anger even in the *pros* section.

Identify protected attributes. Firstly, all *pros* and *cons* review texts predicted as *toxic* were analyzed using the *Explainer* component to identify the most significant words that the model relied on to predict toxicity in these reviews. Secondly, we selected 400 of these most toxic words, and we annotated them using the *Identifier* component with GPT-3.5-Turbo. Table 3.4 shows in the last two columns of the first row that 76 out of 400 words (19%) are protected attributes (original model M_o). Based on this evaluation, we conclude that the original classifier significantly relies

[7] To preserve the privacy of individuals, Personally Identifiable Information (PII) was removed.
[8] Computed with the NLTK library https://www.nltk.org/api/nltk.tokenize.html

on protected attributes to make toxicity predictions, even when applied to out-of-distribution data. This observation confirms the results in Section 3.4. The analysis across diverse datasets demonstrates that protected attributes have a substantial impact on the classifier's predictions in toxicity classification scenarios.

Mitigated models training. We applied the removal-based mitigation strategies (MS1 and MS2) of the *Moderator* on the original Wikipedia comments training dataset, mitigating the protected attributes identified in the toxic company reviews. The third column in Table 3.4 shows how both strategies affect the number of training examples. The sentence-removal strategy (MS1) decreased the original training dataset (containing 159,571 examples) by 6k samples. In contrast, the word-removal strategy (MS1) maintains the same number of training examples. Similarly to the previous experiment, all mitigated models were trained by fine-tuning the pre-trained BERT weights for 3 epochs, with a batch size equal to 16, and using Adam as the optimizer.

Evaluation metrics. We perform a similar evaluation of the mitigated models by measuring their *fairness* and *predictive performance*. *Fairness* is evaluated by quantifying the number of protected attributes each mitigated model relies on, indicating the extent to which their predictions are influenced by these attributes (*fairness through unawareness*). *Predictive performance* is evaluated using quantitative metrics on the test set. Specifically, we measure the F1 score for the toxicity label to gauge the model's accuracy in detecting instances of toxicity, and the Area Under the Curve (*AUC*) score for all toxicity-related labels to provide an overall assessment of the model's performance in identifying different aspects of toxicity.

Results. To evaluate the mitigated models, each model classified all the *pros* and *cons* sections of the reviews. Subsequently, we used the *Explainer* component to identify the top 400 predictive words that each *mitigated* model relied on for toxicity predictions in company reviews. Finally, we utilized the *Identifier* component to determine whether the new important toxic words identified in the mitigated models were considered protected attributes.

(1) Fairness. In Table 3.4, the last two columns show the percentage and the total number of the most toxic words identified as protected attributes for each mitigated model (M_1^*, M_2^*). Additionally, the number of these toxic words that were already recognized as protected attributes in the original model (M_o) is specified within square brackets. The results demonstrate that removal-based mitigation strategies

Table 3.4 **Results in mitigating toxicity prediction on out-of-distribution data (company review).** The original model, M_o, is highlighted in grey. For each mitigated model, we report the following: (i) the model identifier, (ii) the applied mitigation strategy, (iii) the difference in training examples after the mitigation strategy, (iv) the F1 macro and weighted scores for the toxicity class on the original test set, (v) the AUC score for all toxicity-related labels on the original test set, and (vi) the percentage and ratio of relied-upon protected attributes, with the number present in the original model indicated in brackets. The best performing for each metric is highlighted in boldface.

Model	**Mitigation Strategy**	**Δ Train Examples**	*Predictive Performance* ↑			*Fairness* ↓	
			F1 Macro	**F1 Weight**	**AUC**	**% PA**	**Ratio PA**
M_o	-	-	0.815	0.926	**0.986**	19%	76/400
M_1^*	(MS1) *Sentence removal*	-6k	0.824	**0.938**	0.979	**4%**	**16/400 [8]**
M_2^*	(MS2) *Word removal*	-	**0.825**	0.935	0.983	5%	19/400 [11]

effectively reduce the reliance on protected attributes by the model. Specifically, the sentence removal strategy (MS1) and the word removal strategy (MS2) decrease the percentage of protected attributes among the 400 most important words from 19% to 4% and 5%, respectively, representing a decrease of 79% and 75%. These strategies also substantially reduce the number of protected attributes that the original classifier depended on, from 76 to 8 and 11, respectively.

(2) Predictive performance. In Table 3.4, columns 4 and 5 report the F1 macro and weighted scores achieved by the original and mitigated models on the original test set (of Wikipedia comments). Similarly to the in-distribution experiment, the mitigated models demonstrate an improved predictive performance in terms of F1 scores compared to the original model. Both mitigated models showed an increase of approximately 1%. Column 6 of the results shows the *AUC* scores for all toxicity-related labels. The mitigated model resulting from MS1 achieves an *AUC* score of 0.979, experiencing a slight decrease of 0.007 from the original model. Conversely, the decrease in performance with MS2 is only 0.003. From this, we can conclude that all the mitigated models maintain competitive performance in the classification task, also outperforming the original model in distinguishing toxic texts.

Summary. These experimental results demonstrate the effectiveness of NLPGUARD in reducing a model's dependence on protected attributes when applied to non-training data where ground truth labels are absent. This underscores the general applicability of NLPGUARD in real-world scenarios, highlighting its utility in situations where labeled data is not readily available.

3.5.2 Mitigating sentiment analysis

This experiment aims to evaluate the versatility and effectiveness of NLPGUARD across various classification tasks. We selected sentiment classification for this experiment to examine how well it can mitigate the use of protected attributes in contexts where we expect a lower reliance on them. This choice allows us to explore the framework's capability in a different scenario, potentially broadening its utility.

Training the original sentiment classifier. Differently from the previous experiments, instead of using an already fine-tuned classifier, we fine-tune the original classifier for sentiment analysis from scratch. To this end, we selected a dataset of 63K tweets and 37K Reddit comments[9] in English expressing people's opinions towards the general elections held in India in 2019. Each text is annotated with one of 3 classes: *negative*, *neutral*, and *positive*. We split the dataset into 80% for training and 20% for testing, corresponding to 160,000 and 40,000 samples respectively. Then, we fine-tuned a BERT-base and uncased model for multi-class sentiment classification, where, for each input, the model should assign only one of the three possible labels. We fine-tuned the BERT model for 3 epochs, a batch size of 16, and Adam optimizer. The model achieved a 0.96 F1 score on the test set.

Identifying protected attributes. We first predicted the sentiment label using the fine-tuned model over the entire test set. Subsequently, we separately analyzed all the *negative* and *positive* texts using the *Explainer* component (with Integrated Gradients) to identify the most important words. We did not perform the analysis on *neutral* texts since they do not contain specific patterns that the model should learn. Thus, they are not of interest for mitigation. Finally, we employed the *Identifier* (with GPT-3.5-Turbo) to annotate the top 5% of the most important words for both the *negative* and *positive* texts separately, corresponding to the top 200 negative and 200 positive words. We found that out of the 200 negative words, 16 (8%) were identified as protected attributes by the *Identifier*. Similarly, among the 200 positive words, 11 (6%) were annotated as protected attributes. These findings suggest that the non-mitigated sentiment model demonstrates a moderate reliance on its predictions related to protected attributes.

Training the mitigated models. We applied the two removal-based mitigation strategies (MS1 and MS2) of the *Moderator*. The mitigation was performed separately for

[9] https://www.kaggle.com/datasets/cosmos98/twitter-and-reddit-sentimental-analysis-dataset

Table 3.5 **Results in mitigating sentiment analysis.** The original model, M_o, is highlighted in grey. For each mitigated model, we report the following: (i) the model identifier, (ii) the applied mitigation strategy, (iii) the difference in training examples after the mitigation strategy for the *negative* and *positive* classes, (iv) the F1 macro score on the original test set, and (v) the percentage and ratio of relied-upon protected attributes for both classes, with the number present in the original model indicated in brackets. The best performing for each metric is highlighted in boldface.

Model	**Mitigation Strategy**	**Δ Train Examples**		*Predictive Performance* ↑	*Fairness* ↓			
					Negative Class		*Positive Class*	
		Negative	*Positive*	**F1**	**% PA**	**Ratio PA**	**% PA**	**Ratio PA**
M_o	-	-	-	0.96	8%	16/200	6%	11/200
M_1^*	(MS1) *Sentence removal*	-5k	-8k	0.96	**4%**	**8/200 [2]**	**3%**	**5/200 [1]**
M_2^*	(MS2) *Word removal*	-	-	0.96	8%	16/200 [2]	6%	11/200 [2]

each class label since, in this scenario, we analyzed two class labels. For instance, protected attributes in the most negative words were mitigated only on the negative training examples. The sentence-removal strategy (MS1) resulted in a reduction of 5,000 negative and 8,000 positive training examples, as indicated in the third and fourth columns of Table 3.5. The models were fine-tuned for 3 epochs, with a batch size of 16, and using Adam optimizer, in this case as well.

Evaluation metrics. We again evaluate *fairness* and *predictive performance* of the mitigated models. *Fairness* is evaluated similarly to the previous experiments. For the *predictive performance* assessment, we solely measure the F1 score on the test set, as it provides a comprehensive assessment of the model's accuracy in this task.

Results. We predicted the sentiment label across the entire test set using the two mitigated models. Subsequently, we utilized the *Explainer* to extract the top 200 most important words from both the negative and positive texts separately. Finally, we annotated these words using the *Identifier* component.

(1) Fairness. In Table 3.5, the last four columns show the percentage and number of protected attributes for the original (M_o) and mitigated (M_1^* and M_2^*) sentiment classifiers, categorized by the *negative* and *positive* classes. The MS1 mitigation strategy produced a mitigated model that relies on half the number of protected attributes compared to the original classifier, with 4% for the *negative* class and 3% for the *positive* class. Interestingly, the majority of protected attributes relied upon by the original classifier are effectively mitigated, leaving only 2 protected attributes for the *negative* class and 1 for the *positive* class (as shown in the square brackets). Similarly, the mitigated model resulting from MS2 exhibits a similar behavior.

However, it is noteworthy that several new protected attributes emerge as important words. Consequently, although the protected attributes of the original model are largely mitigated, the total number of protected attributes remains unchanged. Thus, MS1 is the most effective strategy in this scenario.

(2) Predictive performance. In Table 3.5, the fifth column shows the F1 score achieved by both the original sentiment classifier and the mitigated models on the test set. Interestingly, the mitigated models achieve the same F1 score as the original classifier, indicating that they possess comparable predictive capabilities.

Summary. These experimental results demonstrate that NLPGUARD is effective not only for the toxicity model, which exhibits a heavy reliance on protected attributes, but also for the sentiment model, where the impact of protected attributes is more moderate. Thus, further proving its effectiveness and general applicability.

3.5.3 Mitigating occupation classification

This evaluation has three primary objectives: (i) to further evaluate the adaptability and effectiveness of our framework across diverse classification tasks, (ii) to mitigate the use of protected attributes in contexts where the resulting predictions hold tangible consequences for individuals, and (iii) to compare our framework with two mitigation techniques that act by modifying the model training rather than by data augmentation. To accomplish these objectives, we opted for the task of predicting occupations from online biographies.

Occupation classification data. We utilized a dataset comprising online biographies, previously annotated by gender and occupation, and used in various previous studies [145, 109, 146]. We used as input text the field *'cleaned_input_text'*, which excludes sentences explicitly stating the occupation (e.g., "`he is a journalist`"). Additionally, as a pre-processing measure, we eliminated all first names to prevent the model from relying on them to infer gender. Each biography in the dataset is also labeled with its respective gender.

Baselines. We perform a comparison of NLPGUARD with two model-based mitigation techniques: *Iterative Null-space Projection (INLP)* [109] and *Entropy-based Attention Regulation (EAR)* [117]. The *INLP* technique necessitates an additional annotation for each sample, specifically for the mitigated category, which is only

applicable to gender in this dataset. Hence, we conduct two separate analyses: (i) concentrating solely on gender-related protected attributes and (ii) evaluating all protected categories simultaneously. Regarding gender-related protected attributes, we perform a comparison of both baselines with our word-removal mitigation strategy (MS2). In contrast, we solely utilize *EAR* to mitigate all protected categories since the dataset does not contain additional annotations for the other categories (as required by *INLP*). We opted for our word-removal (MS2) for this comparison because it has demonstrated comparable performance in the trade-off between reducing reliance on protected attributes and maintaining competitive predictive performance. Additionally, it offers greater flexibility across datasets compared to the sentence-removal method (MS1), as will be discussed in Section 3.6.2.

Training the original occupation classifiers. The dataset comprises 393,423 biographies spanning 28 occupations. Maintaining the same splitting as previous works, we obtained 255,710, 39,369, and 98,344 examples for training, development, and testing, respectively. For each occupation, we fine-tuned a BERT-base uncased model using one-vs-all settings, where each model predicts whether the biography corresponds to that occupation or not. Due to the highly imbalanced nature of each task, we conducted experiments focusing on the five most frequent classes: *nurse*, *attorney*, *journalist*, *physician*, and *professor*.

We fine-tuned the original occupation classification models for 3 epochs, with a batch size of 128, and Adam optimizer. We also introduce inversely proportional class weights to the number of samples to the loss due to the highly imbalanced training dataset. In Table 3.6, the second column shows the F1 macro score on the test set for each original model trained for the five occupations, without applying any mitigation techniques. The original classifiers exhibit high effectiveness in classifying occupations. The model trained to classify the *journalist* occupation demonstrates the lowest performance, achieving a macro F1 score of 0.89. Conversely, all other models achieve a macro F1 score exceeding 0.93. However, such high classification performance might rely heavily on the utilization of protected attributes in predictions.

Identifying protected attributes in occupation classification. We employed each fine-tuned original model to predict each occupation label across the entire test set. Subsequently, we analyzed these predictions using the *Explainer* component, instantiated with Integrated Gradients, to extract the most significant words influencing the

prediction for each occupation. Subsequently, for each occupation, we utilized the *Identifier* component with GPT-3.5-Turbo to annotate the top 400 words to identify protected attributes. This corresponds to approximately 10% of the most crucial words for each occupation.

We assessed the models' reliance on protected attributes concerning (i) gender only, and (ii) all categories, as reported in the third and fourth columns of Table 3.6. All the original models demonstrate moderate reliance on protected attributes in their classifications. The *nurse* occupation exhibits the highest influence from protected attributes, with a significant reliance on gender-related words that frequently appear in the biographies, such as pronouns (e.g., '`she`', '`her`'). Surprisingly, the *professor* occupation is the only one not relying on gender-related protected attributes.

Training the mitigated models. Using NLPGUARD, we applied the word removal mitigation strategy (MS2) for each occupation, focusing on (i) gender-related protected attributes only, and (ii) all categories simultaneously. Consequently, we trained two distinct mitigated models for each analyzed occupation class.

In contrast, we trained a single mitigated model for each occupation using the *INLP* methodology [109]. This technique specifically addresses gender-related protected attributes, as the original dataset includes additional annotation solely for gender, and it is not applicable for mitigating all other protected attribute categories on this dataset. Following the methodology proposed in [109], the model uses the original pre-trained BERT weights as the encoder and multiplies the embedding representation of the [CLS] token from the last hidden layer extracted from each input text by the projection matrix generated by the *INLP* technique. This step ensures that the embedding representation does not encode information about gender. Finally, the model has a classification layer on top. During the fine-tuning process, only the parameters of the classification layer were trained, while keeping the BERT encoder and the projection matrix frozen (i.e., not trained).

For the *EAR* technique [117], we employed the same BERT architecture. The only modification was the addition of a regularization term to the loss function, with a regularization strength of 0.001. This technique does not offer the flexibility to select which protected categories or words to mitigate. Instead, it automatically identifies words with high attention entropy. Consequently, we trained a single mitigated model and evaluated its reliance on gender and all protected categories separately.

Table 3.6 **Results in mitigating occupation classification.** For the *original* models (highlighted in grey) trained in one-vs-all settings, we report the macro F1 score and the reliance on protected attributes (PA) for both (i) gender only and (ii) all categories. For each occupation, we applied NLPGUARD with the word-removal strategy (MS2) on gender-related and all categories of protected attributes, resulting in two different *mitigated* models. For *EAR*, a single model was trained but evaluated on different protected categories. *INLP* is applicable only to gender-related protected attributes in this dataset. For each mitigated model, we report the following: (i) the mitigation technique, (ii) the macro F1 score for the occupation classification on the test set (*predictive performance*), and (iii) the ratio and percentage of relied-upon protected attributes, with the number present in the original model indicated in brackets (*fairness*). The best performing for each metric is in boldface.

Occupation	*Original Model*			**Mitigation Technique**	*Mitigated Models*			
	F1 ↑	*Gender only* **Ratio (%) PA** ↓	*All Categories* **Ratio (%) PA** ↓		**F1** ↑	*Gender only* **Ratio (%) PA** ↓	**F1** ↑	*All Categories* **Ratio (%) PA** ↓
Nurse	0.939	11/400 (3%)	43/400 (11%)	*Our* (MS2)	**0.932**	**2/400 [1] (0.5%)**	0.930	**27/400 [18] (7%)**
				INLP	0.762	**2/400 [2] (0.5%)**	*N.A.*	*N.A.*
				EAR	**0.932**	4/400 [3] (1.0%)	**0.932**	**27/400 [18] (7%)**
Attorney	0.943	2/400 (0.5%)	18/400 (5%)	*Our* (MS2)	**0.942**	**0/400 [0] (0%)**	**0.943**	**8/400 [3] (2%)**
				INLP	0.702	1/400 [0] (0.3%)	*N.A.*	*N.A.*
				EAR	0.940	**0/400 [0] (0%)**	0.940	11/400 [1] (3%)
Journalist	0.886	3/400 (0.8%)	32/400 (8%)	*Our* (MS2)	**0.887**	2/400 [2] (0.5%)	**0.887**	**18/400 [12] (4.5%)**
				INLP	0.528	**1/400 [0] (0.3%)**	*N.A.*	*N.A.*
				EAR	0.886	**1/400 [0] (0.3%)**	0.886	21/400 [9] (5.3%)
Physician	0.936	2/400 (0.5%)	24/400 (6%)	*Our* (MS2)	0.939	**0/400 [0] (0%)**	0.939	**16/400 [3] (4%)**
				INLP	0.823	**0/400 [0] (0%)**	*N.A.*	*N.A.*
				EAR	**0.941**	1/400 [0] (0.3%)	**0.941**	28/400 [12] (7%)
Professor	0.938	0/400 (0%)	8/400 (2%)	*Our* (MS2)	*N.A.*	*N.A.*	**0.941**	**4/400 [0] (1%)**
				INLP	*N.A.*	*N.A.*	*N.A.*	*N.A.*
				EAR	*N.A.*	*N.A.*	0.940	9/400 [3] (2%)

Evaluation metrics. Similarly to the previous experiments, we evaluate *fairness* and *predictive performance* of the mitigated models. *Fairness* is evaluated by measuring the number of protected attributes on which each mitigated model relies. The *predictive performance* is evaluated by measuring the F1 score on the test set for each occupation separately.

Results. Similar to previous experiments, we utilized all the mitigated models to predict occupation labels throughout the entire test set. These predictions were then analyzed with the *Explainer* component, which extracted the 400 most significant words for each occupation. Subsequently, these words were annotated using the *Identifier* to assess whether they exhibit a reduced reliance on protected attributes.

(1) Fairness. In Table 3.6, columns 7 and 9 show the number and percentage of protected attributes on which the mitigated models rely, separated for gender only

and all categories, respectively. The number of these protected attributes, which were already important for the original models, is also indicated in square brackets. The goal of each *mitigated* model is to diminish its reliance on protected attributes, as shown in columns 7 and 9, in comparison to the original models for each occupation, indicated in columns 3 and 4.

Regarding gender-related attributes (column 7), NLPGUARD proves to be as or more effective than other baseline techniques in reducing the use of such protected attributes. For example, in the case of the *nurse* occupation, a model that originally showed a high dependence on gender-related protected attributes, NLPGUARD successfully reduced the number of significant gender-related words from 11 to 2. One of the two gender-related words remaining was already identified as significant in the original model. The *INLP* technique achieved a similar level of mitigation, whereas the *EAR* technique was less effective, leaving four gender-related protected attributes still influential in the mitigated model. For the *attorney* and *physician* occupations, where the original models showed a lower reliance on gender-related protected attributes, NLPGUARD managed to completely eliminate the use of those words. On average, NLPGUARD was able to reduce the reliance on gender-related protected attributes by 79%.

The mitigation effects are similar when addressing all categories of protected attributes (column 9). NLPGUARD consistently outperforms *EAR* in mitigating the use of protected attributes, with the exception of the *nurse* occupation, where both techniques demonstrate equivalent mitigation effects. On average, NLPGUARD effectively reduces reliance on all categories of protected attributes by 44%.

(2) Predictive performance. In Table 3.6, columns 6 and 8 show the F1 macro score achieved by each mitigated model on the test set. The goal of each mitigated model is to maintain a similar predictive performance (columns 6 and 8) to that of the respective original model for each occupation (column 2). The mitigated models trained using NLPGUARD and the *EAR* technique often achieve similar, and sometimes even superior, performance on the test set compared to the original models (e.g., in the cases of the *journalist* and *physician* occupations). Thus, the mitigated models produced by NLPGUARD and *EAR* are capable of reducing reliance on protected attributes without compromising predictive performance. In contrast, the *INLP* technique successfully increases fairness in its models but at the cost of a noticeable reduction in performance, with all the mitigated models from this

technique experiencing an average drop in predictive performance of 10%. This behavior to achieve fairness by disadvantaging better performing groups to match the level of the worst-performing groups is a common undesirable behavior of bias mitigation techniques [147].

Summary. These results confirm the effectiveness of NLPGUARD in mitigating the use of protected attributes without compromising predictive performance, particularly in tasks where protected attributes are closely linked to individual characteristics. NLPGUARD proves more effective at achieving this objective compared to previous bias mitigation techniques and offers the added advantage of flexibility in selecting specific protected categories to mitigate. This flexibility is especially valuable when only a subset of protected categories needs mitigation due to the requirements of the task at hand.

3.6 Discussion

To address **Challenge 2** introduced in Chapter 1, this chapter proposes NLPGUARD, a novel framework for mitigating the use of protected attributes while maintaining the original predictive performance of NLP classifiers. We assessed the sensitivity of each component and its effectiveness in mitigating a toxicity classifier. Additionally, we demonstrated its generalizability on models applied to out-of-distribution data (e.g., toxicity in company reviews) and two other tasks (sentiment analysis and occupation classification). The removal-based strategies (MS1, MS2) were found to be the most effective mitigation techniques. Furthermore, we showed that the LLM-based Identifier outperforms the crowdsourced one, enabling the automation of the framework and dynamic updating of the dictionary of protected attributes. Finally, we demonstrated its superior effectiveness and flexibility in obtaining "*fairness through unawareness*" than previous bias mitigation techniques.

3.6.1 Implications

This work significantly contributes to the research community by delving into human-AI collaboration, particularly in decision-making scenarios. For instance, NLPGUARD can aid individuals in understanding hiring decisions made by NLP classifiers and addressing biases within the hiring process [148]. Additionally, this

work extends to content moderation, supporting the creation of robust systems that can effectively identify and mitigate toxic content while ensuring fairness. This enables humans to understand critical aspects of moderation, including key terms and considerations surrounding protected attributes, fostering collaboration with machines to reach fair decisions in content moderation. Specifically, this work has three main implications:

(1) Fully-automated framework for compliant NLP classifiers. We release an open-source framework[10] that leverages LLM annotations to operate in a fully automated manner. Our mitigated models demonstrate enhanced fairness by significantly reducing their reliance on protected attributes while maintaining comparable or even superior predictive performance. NLPGUARD allows other researchers to meet regulatory compliance standards by mitigating existing models or integrating compliance measures into future models. Our contribution empowers the community to uphold ethical standards and ensure fairness in NLP applications. For instance, mitigated toxicity classifiers developed using our framework can be employed for online moderation in accordance with AI regulations.

(2) Protected attributes annotation. We have advanced the literature on protected attributes' identification in NLP, which has traditionally relied on static and manually predefined dictionaries that cover only a subset of categories. These dictionaries are challenging to maintain, especially with the emergence of ever-evolving language trends and slang. In NLPGUARD, we introduced a novel approach to dynamically identify protected attributes through straightforward prompts to a large language model (LLM). This approach enables the creation of a comprehensive and up-to-date dictionary that simultaneously covers all protected categories and can be periodically updated. This ensures the dictionary remains relevant in real-time linguistic landscapes, addressing the limitations of traditional static dictionaries.

In our research, we annotated a dataset of 15,000 words, identifying 540 of them as protected attributes. We have made our annotated dictionary available in our GitHub repository. This resource is more comprehensive than existing dictionaries as it covers a wider range of protected categories. Importantly, it can be continuously updated using our identifier method. Researchers can utilize and improve upon this resource to further advance bias mitigation efforts in NLP applications. This anno-

[10]The code repository is available at https://github.com/grecosalvatore/nlpguard

tated dictionary serves as a foundational tool for enhancing fairness and inclusivity in natural language processing tasks.

(3) Humans vs. LLM annotations. Building upon a recent finding [128], our study showcases that LLM annotation surpasses human-in-the-loop crowdsourcing ones. Within our framework, we establish that LLM-based annotation of protected attributes is not only more cost-effective and scalable but also closely aligns with experts. This capability enables us to design a fully automated framework devoid of human intervention. These findings open new avenues for exploring the potential of LLMs as highly effective tools for obtaining high-quality annotations.

3.6.2 Limitations

Currently, NLPGUARD has six main potential limitations or areas of concern.

(1) Context unawareness. The *Identifier* and *Moderator* annotate and mitigate words related to protected attributes without considering context, which simplifies annotation but risks inaccuracies. For instance, '`black`' may denote a protected attribute in one context ("`If you are a guy (black) or lesbian, you get hired fast`") but not in another ("`I bought a new black desk`"). Incorporating context could enhance the effectiveness, yet it poses challenges in the identification and mitigation phases. Human-in-the-loop context annotation is costly, requiring thousands of context-aware annotations, which could increase noise. Machine-in-the-loop approaches are more scalable but complicate prompt engineering, potentially leading to misunderstandings and noisy responses [149]. We conducted a preliminary experiment to assess the impact of context-aware annotation on identifying protected attributes using the LLM. We found a 75% overlap between word-level and context-level annotations, with some contradictions, especially in long sentences. Therefore, further research is needed to explore this across various datasets.

(2) Potential bias introduced by the Identifier. The annotation of protected attributes is a subjective task, potentially introducing bias through the *Identifier*. In human-in-the-loop settings, ensuring crowdworkers come from diverse backgrounds is essential for broader contextual understanding during annotation. The demographic distribution of crowdworkers can impact the annotated protected attributes, making equitable distribution across all protected categories important yet challenging. Conversely, in machine-in-the-loop settings, the *Identifier* might introduce

biases inherent in the LLM, associating certain words with protected attributes based on stereotypes in the training data. Addressing these biases is an active research area that promises to improve the framework's effectiveness significantly. Additionally, defining a specific list of protected attributes, such as those in the UK Equality Act 2010, might unintentionally introduce biases or overlook nuances in both human annotators and LLMs.

(3) Reliance on XAI techniques. NLPGUARD relies on XAI techniques to identify the most important words for the model. However, XAI methods have different levels of precision, and inherent limitations [150, 151], such as challenges in effective aggregation and normalization, and the contextual variability of words. These limitations may hinder the accurate extraction of crucial words, affecting the identification of protected attributes that the model uses for predictions. This issue can also impact the *Moderator* component, influencing the mitigation of protected attributes. Future advancements in the XAI field could enhance NLPGUARD's effectiveness since it is flexible and can incorporate any feature importance XAI method applicable to a pre-trained classifier (post-hoc).

(4) Defined protected categories. NLPGUARD annotates protected attributes based on the nine categories outlined in the Equality Act 2010. While these categories address significant aspects of discrimination and equality, they do not cover all human diversity or potential discrimination. Since their formalization in 2010, certain characteristics, such as socio-economic status, health status, genetic heritage, and physical appearance, remain unaddressed [152]. Efforts are underway to expand these categories, enhancing our framework's comprehensiveness. For LLM-based annotation, incorporating new categories is straightforward and involves simply modifying the LLM prompt.

(5) Mitigation with small training datasets or common protected attributes. In scenarios with small or imbalanced training data, or when protected attributes are prevalent in most input texts (e.g., '`he`' and '`she`' in biographies), sentence removal (MS1) may be less effective. Removing sentences containing protected attributes from already limited datasets can have significant consequences. The frequent presence of these protected attributes in inputs may result in the exclusion of most sentences, substantially reducing the available training data. This reduction can lead to a notable and unacceptable decrease in model accuracy. Therefore, alternative strategies, such as word removal (MS2), should be considered. MS2 is similarly

effective in mitigating protected attributes while maintaining predictive performance, offering flexibility across datasets without the aforementioned issues.

(6) Proxy words. NLPGUARD is designed to achieve "*fairness through unawareness*" by ensuring that classifiers avoid relying heavily on words associated with protected attributes in their predictions. However, this approach may only partially address the issue, as classifiers can still depend on proxy words—terms not directly linked to protected attributes but highly correlated with them. In Figure 3.8, we qualitatively show that the LLM-based *Identifier* is also able to annotate proxy words, such as '`headscarf`'. Nonetheless, we acknowledge that the identification of these words may not be comprehensive and requires further investigation in future research.

3.6.3 Future research directions

A future research direction is to develop a context-aware framework aimed at identifying and mitigating protected attributes based on their contextual relevance. This framework will involve extracting words and contextual information from the dataset, identifying protected attributes within each context, and implementing mitigation strategies specifically for sentences that contain these attributes in similar contexts. The LLM-based *Identifier* provides a scalable solution for context-aware annotation. Rather than focusing on word-level annotation, this approach emphasizes sentence-level evaluation, significantly increasing the volume of necessary annotations. The primary challenge lies in assessing the quality of LLM-based annotations due to a lack of labeled data. Nonetheless, preliminary experiments indicate the potential effectiveness of this approach.

Another significant direction for future research involves investigating the reliance on proxy words for protected attributes or other highly correlated terms after mitigation. This represents a major limitation of the "*fairness through unawareness*" approaches, and it poses even greater challenges in NLP due to the unstructured nature of texts. Nevertheless, we believe that future research in implementing the context-aware LLM-based annotation of protected attributes can serve as a powerful method for adding additional metadata and annotations to existing datasets. This enhancement will facilitate the evaluation of other types of fairness, such as "*group fairness*". The LLM annotations can be utilized to assess group fairness metrics, such as conditional demographic disparity [153], on real datasets rather than template-

generated sentences. This approach could provide a more accurate reflection of classifier fairness when deployed in real-world scenarios.

Finally, in this work, we evaluated the annotation of protected attributes using GPT-Turbo-3.5. Future research could involve a comparative analysis of other LLMs, including open-source models, to assess their effectiveness in improving these annotations. This analysis could explore various approaches, including zero-shot and few-shot learning, as well as instruction fine-tuning and standard fine-tuning techniques, to determine how different models perform under varying annotation scenarios. Additionally, the evaluation should address the identification of protected attributes that contain typos, slang, or attempts at censorship, as these variations can significantly affect the accuracy of the annotations.

Chapter 4

Detecting Concept and Data Drift in NLP Classifiers

This chapter first motivates the importance of detecting concept and data drift in deep learning (Section 4.1), and it formally defines the problem and the application scenario (Section 4.2). Secondly, it discusses the current state-of-the-art methodologies for drift detection with their limitations (Section 4.3). Subsequently, it introduces DRIFTLENS (Section 4.4), an innovative drift detection framework able to perform real-time drift detection and characterization on deep learning models for unstructured data, including NLP classifiers. Then, it performs an extensive evaluation to assess the effectiveness of DRIFTLENS in detecting and characterizing drift in real-time compared with state-of-the-art drift detectors (Section 4.5). Finally, it discusses its implications, limitations, and future research directions (Section 4.6).

4.1 Motivation

The fundamental assumption of deep learning, including NLP, is that the training data reflects the real-world data. However, despite the high performance and capabilities of deep learning models, they are typically trained and evaluated on static datasets. In contrast, the real world is dynamic. Consequently, the patterns and relationships learned by the model during training may become outdated. Over time, the underlying data distribution and statistical properties of the target domain can shift, causing a decline (or degradation) in the model's performance. This phenomenon, known

as concept drift [154], highlights the challenge of maintaining the accuracy and relevance of models in ever-changing environments.

Concept drift and performance degradation can significantly impact the reliability, robustness, and trustworthiness of deep learning models in real-world production applications [14]. Consequently, it is crucial to continuously monitor production models and detect concept or data drifts at an early stage [155]. This monitoring process should provide *early warnings* when drift occurs and implement *adaptive measures* to maintain expected performance on newly processed data. By proactively addressing concept drift, we can ensure that models remain accurate and effective in dynamic environments.

Consider for example a toxicity classifier such as the ones discussed in Chapter 2 and Chapter 3. Now, we want to explain and assess the robustness of the NLP classifier over time. As shown by the previous experiments, the classifier may demonstrate high accuracy in identifying toxic texts when evaluated on a static dataset with the same distribution of the training data (e.g., the training set). However, when deployed in real-world applications, such as content moderation on online platforms, its performance may differ due to variations in data distribution (out-of-distribution data) and evolving content over time.

Content on online platforms is constantly changing, reflecting new trends, slang, and cultural shifts. Expressions and words that were once considered neutral or non-offensive can acquire new connotations, while new toxic expressions may emerge that were not present in the training data. For instance, phrases that are coded to bypass moderation filters or new slang terms that carry offensive meanings can become prevalent. Additionally, societal attitudes towards certain languages can shift, leading to changes in what is considered toxic or acceptable.

In production environments, these shifts can lead to significant challenges. The classifier might fail to accurately identify toxic texts over time, resulting in inappropriate content not being removed [156]. This failure can contribute to a discriminatory atmosphere on the platform, as harmful and offensive language may go unchecked, affecting user experience and potentially violating community guidelines and policies [157].

To address these issues, it is crucial to implement continuous monitoring and updating mechanisms for the classifier. This includes drift detection mechanisms, and periodic retraining with updated data that captures the latest trends and language

use, as well as incorporating feedback loops where human moderators can correct and improve the classifier's decisions. Such adaptive strategies are essential to maintain the effectiveness and fairness of the classifier in the ever-changing landscape of online communication.

One of the major challenges in the described scenario is the absence of ground-truth labels for newly processed data (see Section 4.2), such as new comments written on online platforms. Hence, both detection and adaptation must be performed in an *unsupervised* manner. In these settings, the detection and adaptation tasks are usually performed separately. In this work, we only focus on the *drift detection* (monitoring) problem due to the lack of annotated samples [158].

As we will discuss in Section 4.3, a significant body of research has focused on detecting concept drift over time using *supervised* strategies that rely on ground-truth labels [13]. However, they are not applicable in many real-world NLP applications since true labels are not available for newly processed data. A parallel research effort has focused on *unsupervised* strategies for detecting concept/data drift [158–160]. They typically rely on distribution distances or divergence measures evaluated for each individual instance. Consequently, these methods are often computationally intensive and ineffective for detecting concept drifts in scenarios characterized by: (1) high data dimensionality, such as deep learning models working with unstructured data, including NLP, and (2) large processed data volumes, such as the vast number of comments on online platforms. In these scenarios, existing techniques are inaccurate and unsuitable for detecting drifts in (near) real-time.

To tackle this, we propose DRIFTLENS [24–26], an *unsupervised* drift detection framework able to detect *whether* and *when* drift occurs on deep learning classifiers working with unstructured data by exploiting the distribution distances of their internal representations (Section 4.4). DRIFTLENS also provides useful insights to characterize the *drift* by determining which labels are the most affected. Due to its low complexity and high efficiency, it can run in *real-time* on large data volumes.

4.2 Background and application scenarios

Concept drift refers to the phenomenon where the underlying data distribution and statistical properties of a target data domain change over time. This drift can

significantly impact the performance of a model. Several terms have been proposed to describe *"concept drift"* [154], including data drift, dataset shift, covariate shift, prior probability shift, and concept shift. While each term highlights a specific aspect of the phenomenon, most works generally refer to all these subcategories collectively as *"concept drift"*. Concept drift is characterized by a change in the joint distribution between a time period $[0,t]$ and a subsequent time window $t+w$. Drift is said to occur in the time window $t+w$ if:

$$P_{[0,t]}(X,y) \neq P_{t+w}(X,y) \tag{4.1}$$

Where, X and y represent the feature vectors and the target variable of each data instance (x_i, y_i), respectively, while $P_t(X,y)$ denotes the joint probability. The time window $t+w$ can be conceptualized as a specific time period based on how the data stream is being processed. Furthermore, the joint probability can be decomposed as:

$$P_t(X,y) = P_t(y/X)P_t(X) = P_t(X/y)P_t(y) \tag{4.2}$$

Where, $P_t(X/y)$ represents the class-conditional probability, $P_t(y/X)$ stands for the target labels posterior probability, $P_t(X)$ denotes the input data prior probability, and $P_t(y)$ represents the target labels prior probability. Therefore, in classification tasks, concept drift can occur as a change in any of the following terms:

1. $P(X)$: A drift in the input data. This means the marginal probability of the input features X changes. This type of drift is also known as *data drift*, *covariance drift*, or *virtual drift*.

2. $P(y \mid X)$: The relationships or conditional probabilities of target labels given the input features change, but the input features themselves do not necessarily change. This is usually referred to as *concept drift* or *real drift*.

3. $P(y)$: A change in the output data, i.e., the labels and their probabilities change. This type of drift is sometimes referred to as *label drift*.

This work addresses the detection of drift in scenarios where ground truth labels are not available for new data. Consequently, drift must be detected in an *unsupervised* manner. In the absence of actual labels, the only type of drift that can be considered is the change in the prior probability of the features $P(X)$ [158].

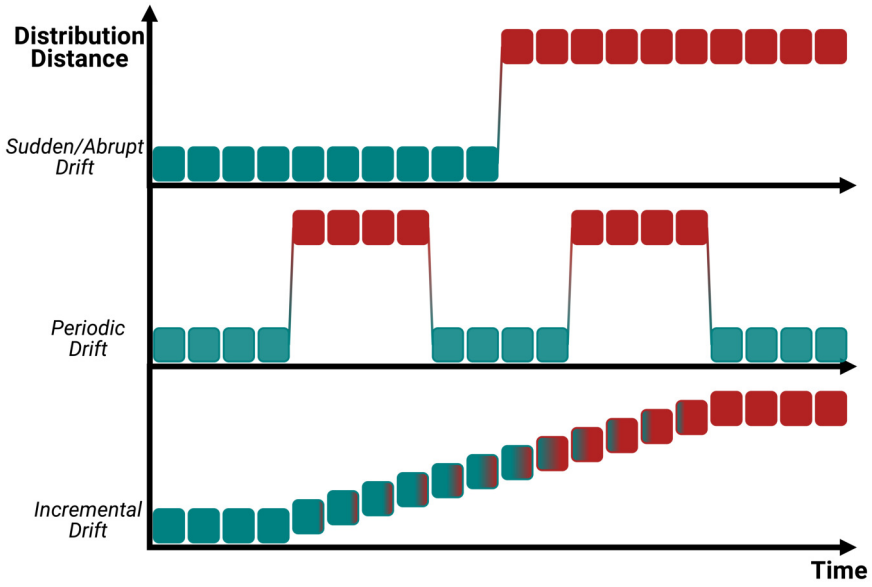


Fig. 4.1 Drift patterns.

4.2.1 Drift patterns

Concept drift can manifest in various patterns. Some examples are illustrated in Figure 4.1 and described below.

- **Suddent/abrupt change.** Drift can occur suddenly if the distribution of new samples changes abruptly. An example of sudden drift is a tweet topic classifier during the outbreak of the COVID-19 pandemic. At that time, the model was suddenly exposed to numerous text samples containing a new topic. Recognizing this scenario is crucial to initiate the retraining process and restore the classifier's performance.
- **Recurrent change**. Drift can occur repeatedly after the initial event, exhibiting seasonality that is unknown during training. For example, in an election year, the language and topics of discussion on social media platforms can change significantly, affecting the sentiment and context in which certain words or phrases are used. After the election, discussions may revert to their usual state, but during another significant event, such as a global sporting event like the Olympics, the state may change again. The classifier must adapt to these changes to maintain performance. Recognizing this scenario is crucial to

either (i) retrain the classifiers to incorporate examples of the new recurrent distribution or (ii) switch to another specific model.

- **Incremental change**. The transition between concepts occurs gradually over time. For instance, consider a model trained to detect harmful or inappropriate language on social media platforms for content moderation. Initially, the model might not encounter new slang or coded language used to bypass moderation. These new terms may appear infrequently and sporadically at first. However, over time, they gradually become more common and widely used, necessitating the model to adapt to these evolving linguistic patterns. Another example is in image or object classification for autonomous vehicles, the model may encounter new vehicle types, such as scooters (i.e., Personal Light Electric Vehicles), which were not part of the original training data. Initially, these new vehicles appear infrequently and sporadically. Over time, however, they gradually become more common and an integral part of the streetscape.

4.2.2 Application scenarios

In this work, we focus on drift detection for application scenarios characterized by:

- **Unavailability of ground truth labels for new incoming data**. Many applications, such as toxicity classification, sentiment analysis, or topic detection on social media comments and online platforms, involve processing large volumes of data to be classified using deep learning models. Typically, these applications do not have ground-truth labels available for newly processed samples. This increases the complexity challenges of performing drift detection and adaptation since they must be performed in an *unsupervised* manner– the detector must not use true labels for drift detection.

- **High complexity, data dimensionality, and data volume**. Most applications work on unstructured data, such as text, images, and audio. Detecting concept drift and data change in unstructured data is more complex than in structured data due to high dimensionality, data sparsity, the complexity of the inputs, and the absence of a fixed structure, such as columns. Moreover, scenarios such as toxicity classification for content moderation or topic detection are

performed on huge textual data volumes produced online (e.g., social media posts). This work focuses on deep learning-based NLP classifiers characterized by high complexity, input dimensionality, and data volumes. These factors can undermine the effectiveness of drift detection techniques. Therefore, effective data modeling is required to accurately detect data changes over time.

In such a target scenario, the desired characteristics for the drift detection include:

- **Fast Detection**: To promptly identify the problem, the drift detection method should identify drift occurrences as early as possible, not solely when they manifest severely.
- **Real-time Detection**: The methodology should have low complexity to execute and be capable of operating in real-time, even with high-throughput data streams, such as texts produced on online platforms and social media.
- **Drift Characterization**: The methodology should provide insights into the type of drift occurring and assess which labels are most affected by the data changes. This can accelerate the process of understanding these changes and taking appropriate actions to restore the classifier's initial performance.

4.3 Related work

Significant efforts have been made in the field of concept drift detection, as outlined in [13, 159, 161, 154, 162, 163]. Drift detection techniques can be categorized into two main groups based on the assumption of true labels availability: (1) *supervised* (Section 4.3.1) and (2) *unsupervised* (Section 4.3.2). Some techniques also address the adaptation problem (Section 4.3.3)

4.3.1 Supervised concept drift techniques

The majority of previous drift detection techniques fall into the category of *supervised* methods [164–179]. These approaches typically utilize error rate-based metrics or ensemble models to identify performance degradation over time, such as a decrease in accuracy. However, they assume that true labels are either readily available for

new data or will be obtained within a short timeframe. In reality, acquiring labels for new data is often impractical due to the associated costs and time constraints. This reliance on the availability of true labels poses a significant constraint on the applicability of these techniques in real-world settings. Therefore, they are not applicable in the application scenarios focus of this work (Section 4.2.2).

4.3.2 Unsupervised concept drift techniques

In contrast, *unsupervised* techniques do not rely on the availability of true labels [158–160, 180]. Our methodology falls within this category. As outlined by [158], unsupervised techniques can be further classified into: (i) *statistical-based*, (ii) *loss-based*, and (iii) *virtual classifier-based*.

(i) Statistical-based unsupervised methods. Most unsupervised techniques [181–190] rely on statistical tests or divergence metrics between two distributions, typically comparing a reference and a new window. Our methodology falls within this category. These techniques offer two primary advantages. Firstly, they are agnostic to data types and models. Secondly, they do not necessitate external models or resources. However, the primary limitation of most of these techniques lies in their use of every single sample in the reference and new windows to calculate distances. Consequently, they do not scale efficiently with the number of samples or the dimensionality of inputs. Moreover, they are usually ineffective with high-dimensional data. These drawbacks can impact both their runtime and drift prediction performance.

(ii) Loss-based unsupervised methods. This family of techniques utilize machine learning model loss functions to assess the similarity between newly arrived data points and previous ones [191–194]. Typically, these methods either employ autoencoders or are combined with supervised drift detectors. In the former scenario, while widely utilized, they often struggle with transferability to other distribution types. In the latter scenario, they assume a correlation between the rise in model losses and concept drift, as demonstrated in [195]. However, these techniques are usually not suitable for NLP classifiers since they are based on autoencoders.

(iii) Virtual classifier-based unsupervised methods. These methods [196–198] usually employ classifiers for drift detection but with a distinct approach. They typically partition data into pre- and post-moment subsets. If the classifier demonstrates an accuracy higher than random in distinguishing samples between these subsets,

it indicates disparate data properties, suggesting drift. Their main limitation is the reliance on training and maintaining an additional model for drift detection.

4.3.3 Drift adaptation and incremental learning

While some studies delve into incremental learning for drift detection and adaptation [199], adapting to drift in unsupervised settings presents more challenges due to the absence of annotated samples [158]. Drift adaptation falls beyond the scope of this work. Instead, our focus lies solely on the drift detection problem (*monitoring*).

4.4 DriftLens Framework

DRIFTLENS [24] is an *unsupervised* drift detection methodology based on the distribution distances of the embedding representations generated by deep learning models when dealing with unstructured data. The embedding representations are internal dense vectors generated by deep learning models to encode the input features present in texts. However, since the framework exploits the embedding representations, it also works similarly with any unstructured data type (e.g., texts, images, and audio).

DRIFTLENS includes an *offline* and an *online* phases, as depicted in Figure 4.2. The *offline* phase consists of estimating the *reference distributions* and the *threshold distance values* from a historical dataset in which is assumed the absence of drift (e.g., the training set). The reference distributions are called *baseline* and represent the features' distribution (i.e., embedding distributions) of the concepts that the model has learned during the training phase. Therefore, they represent the distributions with the absence of drift. The thresholds represent the maximum distribution distances observed in the absence of drift. Therefore, they can be used to detect whether a distribution distance implies the presence of drift. The *online* phase consists of processing the new data stream in fixed-size windows by estimating their distributions and computing the distribution distances with respect to the reference distributions. If the distance exceeds the threshold values, thus the presence of drift is predicted for the current window.

We first discuss how DRIFTLENS performs the data modeling (Section 4.4.1) and computes the distribution distances (Section 4.4.2), as these concepts apply similarly

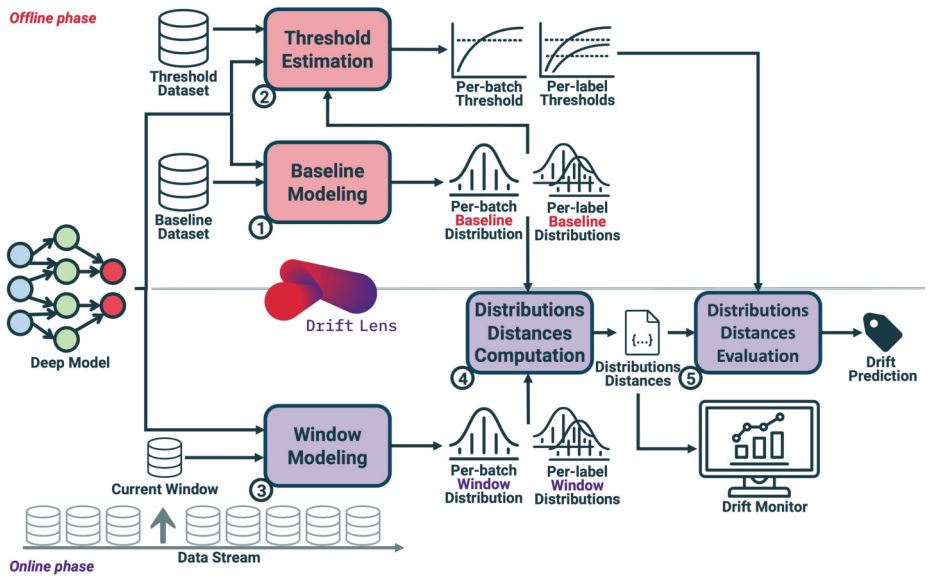


Fig. 4.2 **DRIFTLENS Framework.** During the *offline* phase, it estimates the reference distributions and distance thresholds using historical (training) data. Distributions are modeled as multivariate normal and computed: (i) for the entire batch (*per-batch*), and (ii) conditioned on the predicted label (*per-label*). In the *online* phase, it analyzes data streams in fixed windows, comparing new and reference distributions and utilizing thresholds to identify drift. This process is visualized in a drift monitor.

to both phases. Next, we detail the *offline* (Section 4.4.3) and the *online* phases (Section 4.4.4). Finally, we show an application tool that employs the DRIFTLENS methodology to perform drift detection on pre-uploaded and new data (Section 4.4.5).

4.4.1 Data modeling

The objective of data modeling is to effectively and accurately estimate the distribution of a dataset. Given a batch of data (e.g., the whole reference data or a new data window), DRIFTLENS first estimates its distribution. Instead of using the raw input data, DRIFTLENS exploits the internal representation that the deep learning model generates when processing a batch of data (i.e., the embedding vectors).

Figure 4.3 outlines the key steps in the data modeling process performed by DRIFTLENS. Specifically, it estimates two different distributions: (i) the embedding distribution of the entire batch, independent of the predicted class labels (*per-batch*), and (ii) the embedding distributions for each predicted class, separately (*per-label*). The *per-batch* approach models the entire set of embedding vectors as a multivariate

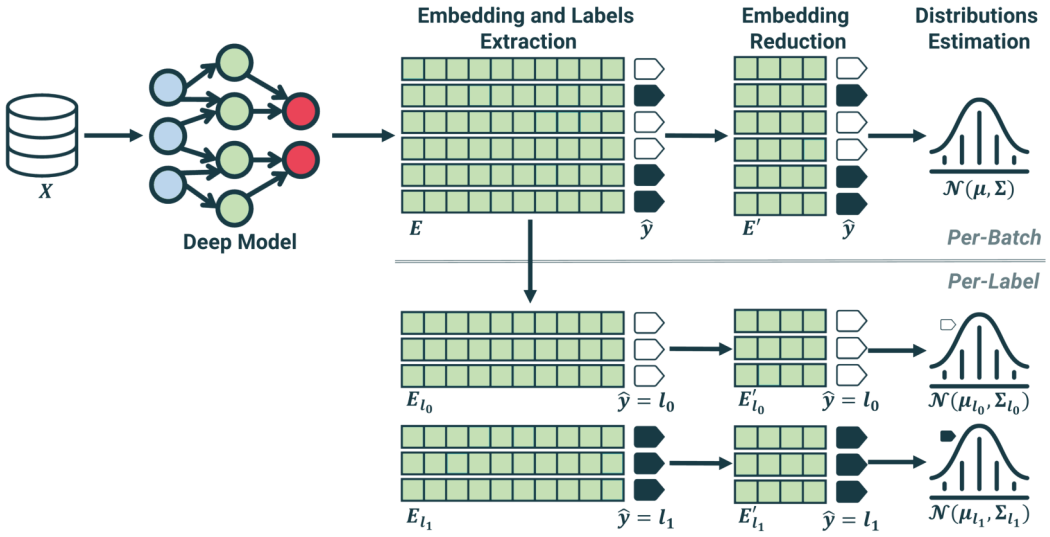


Fig. 4.3 **DRIFTLENS Data Modeling.** This process takes a deep learning model and a dataset as input, and estimates multivariate normal embedding distributions. It involves the following steps: (1) Extract embeddings and predicted labels for the dataset using the model; (2) Reduce the embedding dimensionality; and (3) Estimate the *per-batch* distribution by computing the mean vector μ and covariance matrix Σ, and the label-specific distributions by grouping embeddings by label and computing the *per-label* μ_l and Σ_l for all $l \in L$.

normal distribution. In contrast, for *per-label*, $|L|$ normal distributions are estimated, where $|L|$ represents the number of labels the model was trained on.

DRIFTLENS operates under the assumption that embeddings follow a multivariate normal distribution. While this might seem like a strong assumption for raw input features, it is a reasonable approximation for features within the embedding space. This is particularly true for the dense features in the last layers of the embedding space, where relationships tend to be simpler and thus can be better approximated as a multivariate normal distribution [200].

Formally, given a deep learning model able to classify between a set of class labels L. The classifier comprises an encoder part $\phi(X)$ that maps the sparse and complex representations of the raw input data to a dense and simpler latent representation through several nonlinear transformations using the learned weights W. Typically, the encoder is followed by one or more fully connected neural network layers that use the latent embedding representation to predict a class label. In the data modeling phase, a dataset X–either the entire baseline or a new window—is first input into the deep learning model. This process extracts the corresponding embedding matrix $E = \phi(X) \in \mathbb{R}^{m \times d}$ and a vector of predicted labels $\hat{y} \in \mathbb{R}^m$, where m represents

the number of samples in the dataset and d the dimensionality of the extracted embedding layer.

In deep learning classifiers, the dimensionality of the embedding space is typically large, often ranging into the thousands. When modeling the embedding as a multivariate normal distribution, it is required to compute the mean vector μ and the covariance matrix Σ across the set of vectors. However, to ensure that the covariance matrix Σ is full rank, at least d linearly independent vectors (i.e., inputs) are required. Otherwise, the covariance matrix Σ may contain complex numbers, negatively affecting the calculation of distribution distances (refer to Section 4.4.2). This issue becomes particularly pronounced when estimating the multivariate normal distribution for each label (*per-label*), as it requires d linearly independent vectors to be associated with each label. While this typically is not problematic when modeling the baseline—since calculations are based on the entire historical dataset, such as the training set—the challenge arises during the online phase. Hence, d linearly independent vectors are necessary to estimate the distribution per batch, and $d \times |L|$ linearly independent vectors are needed for the per-label distributions in each new window of the online data stream. Consequently, the fixed window size required to segment the data stream needs to be substantially large.

To address this issue, DRIFTLENS employs a dimensionality reduction step on the embedding by performing a principal component analysis (PCA).

$$E' = PCA(E) \tag{4.3}$$

This process results in a reduced embedding matrix $E' \in \mathbb{R}^{m \times d'}$, where $0 < d' \leq d$ is a user-defined parameter specifying the number of principal components retained for the entire batch. Dimensionality reduction is also applied separately to each embedding matrix based on the predicted label as shown by the following equation.

$$E'_l = PCA(E_l), \forall l \in L \tag{4.4}$$

Where $E_l \in \mathbb{R}^{m_l \times d'_l}$ represents the embedding vectors associated with inputs predicted with the label l, $0 < d'_l \leq d$ is a user-defined parameter that determines the number of principal components for each label.

Note that d' and d'_l can be independently set to different values. For the per-batch scenario, d' can be set to $d' = \min(m_w, d)$, where m_w is the window size used in the online phase and d is the embedding dimensionality. In our experiments (see Section 4.5.1), we consistently set $d = 150$. Conversely, d'_l depends on the window size, the dimensionality of the embedding, and the number of labels. For a balanced data stream, a reasonable value for d'_l is approximately $d'_l = \min(m_w/|L|, d)$. It is important to note that the PCA models are fitted and utilized in the offline phase to reduce the dimensionality of the embeddings. During the online phase, the previously fitted PCA models are employed to reduce the dimensionality of new data embeddings.

Once dimensionality reduction is performed, the reduced embedding matrices E' and E'_l are utilized to estimate the multivariate normal distribution for the *per-batch* (see equation 4.5), and the $|L|$ multivariate normal distributions for each predicted label (see equation 4.6).

$$P(\phi(x); W) \sim \mathcal{N}(\mu, \Sigma) \tag{4.5}$$

$$P(\phi(x) \mid \hat{y} = l; W) \sim \mathcal{N}(\mu_l, \Sigma_l), \forall l \in L \tag{4.6}$$

Where $\mu \in \mathbb{R}^{d'}$ and $\Sigma \in \mathbb{R}^{d' \times d'}$ are the mean vector and covariance matrix of the *per-batch* multivariate normal distribution, respectively. Similarly, each $\mu_l \in \mathbb{R}^{d'}$ and $\Sigma_l \in \mathbb{R}^{d' \times d'}$ represent the *per-label* multivariate normal distribution for each label $l \in L$, calculated separately. Estimating these multivariate normal distributions is straightforward and efficient, as they can be fully characterized by their mean vector and covariance matrix.

A key advantage of modeling the distribution as a multivariate normal distribution is its representation in a reduced dimensionality d', independent of the number of samples in both the reference and new windows. This scalability ensures the method can handle large datasets effectively and facilitates real-time drift detection independently of data volume.

4.4.2 Distribution distance

Most previous techniques use distribution distance metrics that do not scale with the number of samples, as they use all the samples to compute the distances or

the statistical measure. Instead, in DRIFTLENS, we use the Frechét distance [201], also known as the Wasserstein-2 distance [202], which is also defined to calculate distances between two multivariate normal distributions.

Specifically, we use the Frechét distance [201] to measure the distances between the embedding distributions of a baseline (reference distributions) and the new windows in the data stream, and we call it the *Frechét Drift Distance* (*FDD*).

Given the multivariate normal distribution of the baseline (i.e., a reference distribution) b characterized by a mean vector μ_b and a covariance matrix Σ_b, and the multivariate normal distribution of the new data window w, characterized by μ_w and Σ_w, the FDD is computed as follows:

$$FDD(b,w) = ||\mu_b - \mu_w||_2^2 + Tr\left(\Sigma_b + \Sigma_w - 2\sqrt{\Sigma_b \Sigma_w}\right) \tag{4.7}$$

$FDD(b,w)$ is a real number within the range of $[0,\infty]$. A higher FDD indicates a greater distance between the two distributions, indicating a higher probability of drift occurring.

The FDD takes into account changes in two key aspects of the distributions: (i) in the mean (center of the distribution), and (ii) in the diagonal elements of the covariance (spread of the distribution). Changes in the center result from the L2 norm of the difference between the mean vectors $||\mu_b - \mu_w||_2^2$. Changes in the spread results by considering $Tr(\Sigma_b + \Sigma_w - 2\sqrt{\Sigma_b \Sigma_w})$. The last term can be viewed as the multi-dimensional generalization of the squared difference between the standard deviations in one-dimensional space. As a result, the FDD score has the potential to be effective in detecting subtler forms of drift that impact not only the center but also the dispersion (or spread) of the distributions.

DRIFTLENS calculates a single *per-batch* distance by computing the FDD score between the baseline (μ_b, Σ_b) and the new window (μ_w, Σ_w) distributions. Additionally, DRIFTLENS computes $|L|$ *per-label* distances by evaluating the distribution differences ($\mu_{b,l}$, $\Sigma_{b,l}$) and ($\mu_{w,l}$, $\Sigma_{w,l}$) for each label $l \in L$ separately.

4.4.3 Offline phase

The *offline* phase (steps ① – ② in Figure 4.2) aims at estimating (i) the reference probability distributions, representing what the model learned during training, and (ii) distance thresholds to distinguish between normal and abnormal distances (i.e., distances that indicate non-drifting vs drifting distributions). This step is also sometimes called *building a descriptor* by previous work [158]. Notice that, the *offline* phase is executed once, and its outputs (i.e., the baseline and thresholds) are persistently stored on disk for later use in the *online* phase.

Baseline modeling

The baseline modeling (step ① in Figure 4.2) aims at estimating the distributions of a reference dataset representing normal distributions without the presence of drift. It consists of performing the data modeling (introduced in Section 4.4.1) of the baseline dataset, and storing the estimated distributions and the PCA models on disk. To this end, the following steps are executed:

1. The baseline dataset X_b, comprising m_b samples, is fed into the deep learning model, which outputs the embedding vectors $E_b = \phi(X_b)$ and estimates the vector of predicted labels $\hat{y}_b$.
2. The *per-batch* PCA is fitted over the entire set of embedding vectors, and $|L|$ different PCAs are fitted by grouping the embedding vectors based on the predicted labels.
3. The *per-batch* and *per-label* embedding vectors are reduced, to obtain the *per-batch* reduced embedding matrix E'_b, and the *per-label* reduced embedding matrices $E'_{b,l}$, $\forall l \in L$.
4. The reduced matrices are used to estimate the baseline *per-batch* and *per-label* multivariate normal distributions by computing their mean and covariance.

The *per-batch* distribution is fully characterized by the baseline *per-batch* mean vector and the covariance matrix (μ_b, Σ_b). The *per-label* distributions are obtained by computing the $|L|$ mean vectors and the covariance matrices $(\mu_{b,l}, \Sigma_{b,l})$ $\forall l \in L$ on the reduced embedding vectors, grouped by predicted labels.

Note that regardless of the dimensionality of the baseline dataset (m_b), the offline phase estimates distributions characterized by vectors of dimensionality d'. As a result, when predicting the drift for new windows, it is not influenced by m_b. This aspect is crucial in the low execution time of the entire framework.

Threshold estimation

The threshold estimation (step ② in Figure 4.2) aims to estimate the maximum potential distribution distance FDD (introduced in Section 4.4.2) that a window not affected by drift can achieve. To this end, DRIFTLENS takes as input the threshold dataset X_{th}, the window size m_w (which will be the same as that used during the online phase), the baseline, and a parameter n_{th}, which specifies the number of windows to be randomly sampled from the threshold dataset. The n_{th} parameter should ideally be as large as possible to better estimate the larger distance in data considered without drift. In our experimental evaluation (see Section 4.5.1), we empirically set $n_{th} = 10{,}000$. However, we observed that changing this parameter does not significantly affect the framework (see Section 4.5.5).

In this procedure, a number of n_{th} windows is selected randomly from the dataset X_{th}, each window comprising m_w inputs. For every selected window, the process entails conducting the data modeling (as outlined in Section 4.4.1) and computing the FDD distribution distances (introduced in Section 4.4.2) with respect to the baseline distributions, both on a *per-batch* and *per-label* basis (using Equation 4.7). Thus, n_{th} distribution distances are calculated for the entire batch and individual labels.

Subsequently, these distances are arranged in descending order. The foremost element represents the maximum permissible distance for a data window to be considered devoid of drift in relation to the baseline. Consequently, any distances surpassing this threshold indicate potential drift warnings.

Given the substantial number of randomly sampled windows, outlier distances may be present. Hence, DRIFTLENS introduces a parameter, $T_\alpha \in [0,1]$, to specify the threshold sensitivity. This parameter enables the removal of the T_α% left tail of the sorted distances (arranged in descending order) to mitigate outlier effects. Consequently, the ultimate thresholds, T and T_l, are established as the maximum distance subsequent to eliminating the T_α% of the largest distances. As a result, a higher value for T_α corresponds to lower thresholds, enhancing sensitivity to

potential drift and false alarms. In our experiments (refer to Section 4.5.1), we adopt $T_\alpha = 0.01$ as the default threshold sensitivity. This choice ensures that we capture the maximum plausible distance by exclusively eliminating outliers (the top 1%).

Choice of the reference dataset

The reference set should consist of historical data that reflects the knowledge learned by the model during its training phase. Ideally, the entire training set can serve as the baseline, while the test set or a selection of windows from the real data stream—presumed to represent distributions without drift—can comprise the threshold dataset. In our experimental evaluation, we utilize the training set as the baseline and the test set for estimating the threshold (refer to Section 4.5.1).

4.4.4 Online phase

During the *online* phase (steps ③ – ⑤ in Figure 4.2), DRIFTLENS segments and processes the incoming data stream into windows of fixed sizes. The size of each window is determined by the parameter m_w. For a new data window X_w comprising m_w new samples, the data modeling entails the following steps. (i) Extraction of the embedding $E_w \in \mathbb{R}^{m_w \times d}$ and prediction of labels $\hat{y}_w \in \mathbb{R}^{m_w}$. (ii) Reduction of embedding dimensionality to obtain $E'_w \in \mathbb{R}^{m_w \times d'}$. (iii) Estimation of the multivariate normal distributions for both *per-batch* and *per-label* cases. The PCA models that were fitted during the offline phase are used in this step.

The *per-batch* multivariate normal distribution is estimated by calculating the mean vector $\mu_w \in \mathbb{R}^{d'}$ and the covariance matrix $\Sigma_w \in \mathbb{R}^{d' \times d'}$ across all embeddings within the window. For the *per-label* approach, a multivariate normal distribution is computed for each unique label. This involves selecting the embedding vectors associated with each label and computing the mean and covariance within that subset. Consequently, we obtain $|L|$ multivariate normal distributions characterized by $\mu_{w,l} \in \mathbb{R}^{d'l}$ and $\Sigma_{w,l} \in \mathbb{R}^{d'_l \times d'_l}$.

Subsequently, the *per-batch* and *per-label FDD* distances between the window and baseline distributions are computed (using Equation 4.7). If the *per-batch* and/or *per-label* distribution distances surpass the threshold values, it indicates the presence of drift. The *per-batch* distance is crucial for detecting whether the entire window is

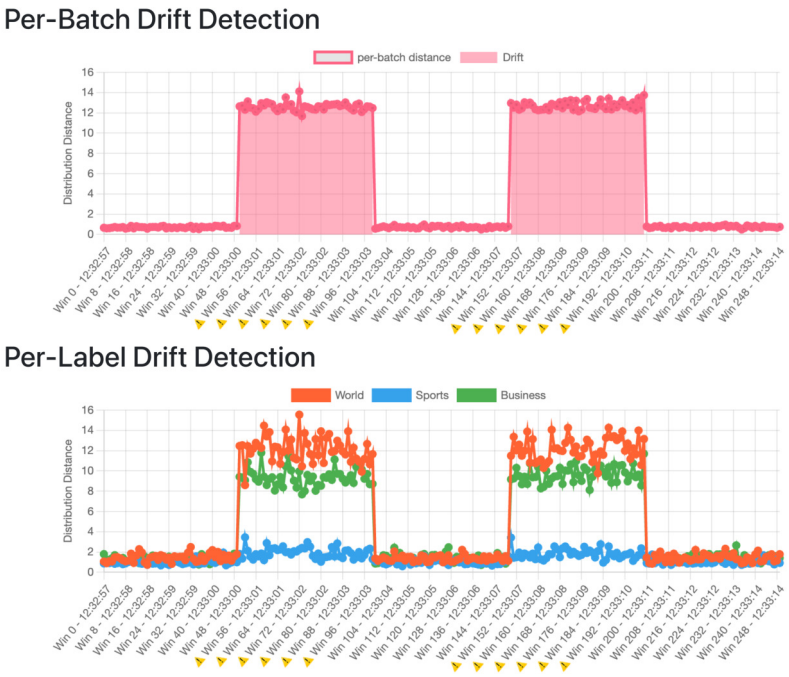


Fig. 4.4 **DRIFTLENS Monitor.** It shows the computed *per-batch* and *per-label* distribution distances (FDD) over time.

affected by drift. Meanwhile, the *per-label* distance helps characterize the drift and identify the most affected labels.

Monitoring drift over time

For each window, the same process is iterated. Upon processing a new window, DRIFTLENS adds the FDD distribution distances to the drift monitor. This monitor displays two separate charts illustrating the *per-batch* and *per-label* distribution distances over time. A warning symbol is included in the plot when a particular distance surpasses its corresponding threshold. The drift monitor serves as a valuable tool for understanding: (i) the occurrence and timing of drift, (ii) the extent and patterns of drift severity, and (iii) which labels are most affected by drift.

Figure 4.4 illustrates an example of the drift monitor generated with the application tool (see Section 4.4.5) for a topic detection classifier we will analyze. The top chart displays the *per-batch* distribution distances, while the bottom chart illustrates the *per-label* distribution distances for all the windows. On the *x-axis*, timestamps and window identifiers are indicated, while the distribution distances (FDD) are

plotted on the *y-axis*. When drift is detected (i.e., the distribution distance in a particular window exceeds the threshold), the area under the curve of the per-batch plot is filled, and a warning symbol appears on the x-tick of the charts.

In this example (Figure 4.4), the monitor shows that drift initially occurs after 50 windows with high severity. Subsequently, it recurs intermittently, alternating with non-drifted windows in a periodic pattern. The label *World* is the most affected by drift, followed by *Business*. Conversely, the label *Sports* exhibits negligible impact.

4.4.5 DRIFTLENS tool

To facilitate the use of DRIFTLENS, we employed it in an application tool[1] [25] as well as making it available as a Python package.[2] The web tool employs the DRIFTLENS methodology and is implemented in Flask [203].[3]

The homepage of the tool provides comprehensive documentation on the concept drift problem, the DRIFTLENS methodology, and the user capabilities of using the tool. Then, there are two pages accessible for launching experiments. One page allows users to choose a previously uploaded use case and create a data stream with controlled drift generation. Another page allows users to conduct drift detection on their own dataset and model. In either scenario, the drift monitor indicates the presence of detected drift (as described in Section 4.4.4).

Page 1: Run controlled drift experiments

On the first page, users can select a pre-uploaded use case to set up a controlled drift experiment, which will then be processed by DriftLens (see Figure 4.5).

Users can generate a data stream for each use case by specifying two parameters: the *number of windows* and the *window size*. This generates a data stream containing the specified number of windows, with each window containing the specified number of samples. Due to dimensionality constraints, the data stream is generated through random sampling with replacement. During the data stream generation, users can

[1] https://github.com/grecosalvatore/DriftLensDemo
[2] https://github.com/grecosalvatore/drift-lens
[3] A recorded demo of the tool is available at https://youtu.be/1R2igFhMD8U.

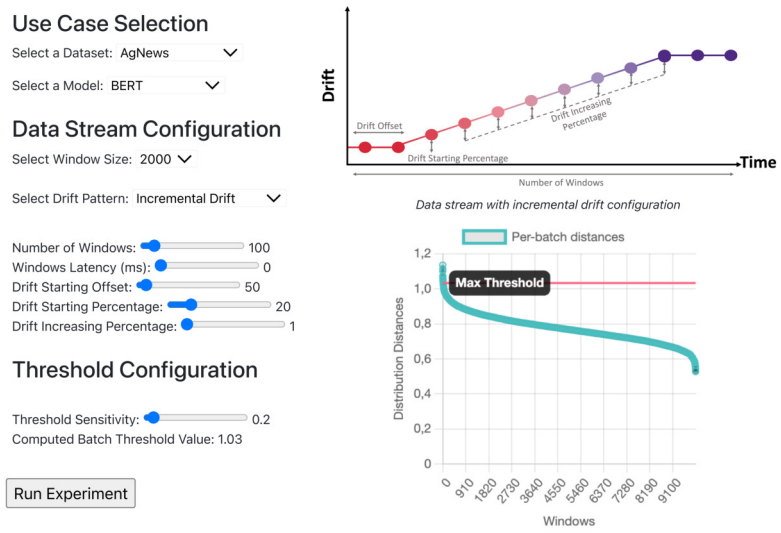


Fig. 4.5 **DRIFTLENS web tool: (1) Controlled drift experiment.**

simulate four types of scenarios by adjusting additional parameters specific to each pattern.

- *No Drift*. In this pattern, none of the generated windows contain drifted samples. It simulates the arrival of windows similar to those used during training.
- *Sudden Drift*. In this scenario, concept drift occurs abruptly at a specific point in time. Users can specify the starting window from which drift will occur and its severity, represented by the percentage of drifted samples in the windows.
- *Incremental Drift*. In this scenario, concept drift occurs gradually over time, continuously, and incrementally. Users can specify the severity of the initial drift occurrence and the increment step for each successive window.
- *Periodic Drift*. In this scenario, both old and new concepts reoccur repeatedly after a certain period. Users can specify the starting window from which the drift will occur, its duration, and its severity.

Once the use case selection and data stream configuration are complete, the tool generates the data stream. Subsequently, it initiates the online phase by activating the drift detection monitor (as described in Section 4.4.4).

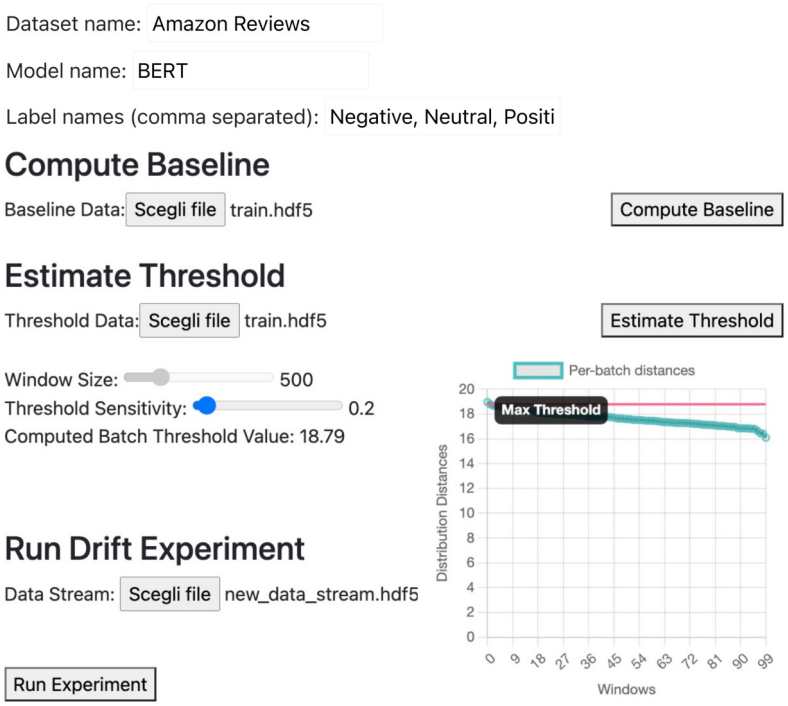


Fig. 4.6 **DriftLens web tool: (2) Controlled drift experiment.**

Page 2: Run drift detection on user-provided data

The second page provides users with the opportunity to experiment with drift detection using their own data and models (see Figure 4.6).

Specifically, Users are required to provide the embeddings and predicted labels[4] for both the *baseline* and *threshold* datasets (e.g., training and test sets). These datasets are used during the *offline* phase of DriftLens. Next, users are required to provide an ordered data stream and set the *window size* parameter. The data stream is divided into windows while preserving the sequential order of the data. After configuring the threshold sensitivity, users can initiate the *online* phase to detect drift and analyze *if*, *where*, and *how* drift occurs in their data stream. The same drift monitoring page is accessible for this purpose (as described in Section 4.4.4). This allows users to verify if their models and data have been affected by drift.

[4]The correct datasets format is specified in the repository.

4.5 Evaluation

To ascertain the effectiveness of DRIFTLENS in detecting drift across different deep learning NLP classifiers, we assess its ability to detect drift (Section 4.5.2) and efficiency (Section 4.5.3) for several use cases and compared to state-of-the-art drift detectors (introduced in Section 4.5.1). Next, we evaluate its ability to accurately characterize and model the drift trend over time (Section 4.5.4). Finally, we perform a sensitivity analysis of its parameters (Section 4.5.5).

4.5.1 Experimental settings

Since this work focuses on the NLP domain, the main evaluation is conducted on NLP classification tasks. Nevertheless, DRIFTLENS is methodology agnostic to the data type and has also been tested on image and speech classification [26]. By conducting a variety of experiments, we aim to demonstrate the broad applicability and generalizability of the DRIFTLENS framework. The experimental use cases and settings are described below.

Deep learning classifiers. We conduct experiments utilizing several deep learning classifiers suitable for NLP, including BERT[5] [78], DistilBERT[6] [80], and RoBERTa[7] [204]. Since DRIFTLENS exploits the embedding representations, we extracted and utilized the embedding of the [CLS] token from the last hidden layer, with an embedding dimensionality of $d = 768$. This embedding vector encodes the full sentence and contains the high-level concepts used to perform the classification.

Datasets. We trained models for topic detection using the Ag News [85] and 20 Newsgroups [205] datasets. The former is a topic classification dataset containing news articles belonging to four classes: *World*, *Sport*, *Business*, and *Science/Technology*. The latter is another topic classification dataset containing 20 topics belonging to 6 macro-categories: *Technology*, *Sale-Ads*, *Politics*, *Religion*, *Science*, and *Recreation*.

Drift simulation. Drift is simulated by introducing a new unknown class label in the new processed data stream. To achieve this, we utilized two subsets of the dataset for

[5] https://huggingface.co/google-bert/bert-base-uncased
[6] https://huggingface.co/distilbert/distilbert-base-uncased
[7] https://huggingface.co/FacebookAI/roberta-base

training (fine-tuning) and testing the models using a subset of class labels. A third part of the dataset within the same labels is reserved to generate windows in the data stream. This portion of the dataset simulates new, unseen data with a distribution similar to the training data. Drift is simulated by excluding one of the class labels during the training phase and subsequently presenting these examples within the data stream windows. In use case 3, the same dataset as in use case 2 is employed, but the nature of the drift is subtler. This is achieved by utilizing only a subset of a class to simulate drift, leveraging the hierarchical categorization present in the dataset.

Windows generation. When generating windows of the data streams, we include samples with or without drift that the model has never encountered during the training or testing phases. When constructing a window without drift, we randomly choose a balanced sample by class label from the new data devoid of drift. These examples stem from new data unseen by the model but adhere to the same distribution as the training data. When creating windows containing $D\%$ of drift, we randomly choose $D\%$ of the window size (m_w) from the drifted samples. The remaining $(100-D)\%$ of the window comprises balanced samples from the new, unseen data without drift. The dimensionality of the data splits is indicated in the last column of Table 4.1.

DriftLens configuration. The default experimental parameters of DRIFTLENS are as follows. In the *offline* phase, the entire training and test sets are utilized. The former is employed for baseline modeling, while the latter is used for threshold estimation. The number of principal components used to reduce the *per-batch* embedding $d' = 150$. The number of windows in the threshold estimation is set to $n_{th} = 10,000$, with a sensitivity of $T_\alpha = 0.01$. The window size remains constant throughout both the *offline* and *online* phases.

State-of-the-art detectors. We conduct a comparative analysis between DRIFTLENS and four unsupervised statistical-based drift detection techniques from prior research: Maximum Mean Discrepancy[8] (MMD) [181], Kolmogorov-Smirnov[9] (KS) [182], Least-Squares Density Difference[10] (LSDD) [183], and Cramér-von Mises[11] (CVM) drift detectors provided by the Alibi Detect library [206]. We maintain the default parameter configurations for each technique. The *p-value* threshold used to distinguish between drifted and non-drifted distributions is set to 0.05 as the default.

[8] https://docs.seldon.io/projects/alibi-detect/en/stable/cd/methods/mmddrift.html
[9] https://docs.seldon.io/projects/alibi-detect/en/stable/cd/methods/ksdrift.html
[10] https://docs.seldon.io/projects/alibi-detect/en/stable/cd/methods/lsdddrift.html
[11] https://docs.seldon.io/projects/alibi-detect/en/stable/cd/methods/cvmdrift.html

Table 4.1 Overview of the experimental use cases, partitioned into groups based on the dataset and task. The training labels, the way the drift is simulated, and the split of the dataset are given in the description for each group. Within each group, different deep learning models are considered, and the corresponding F1 scores obtained on the test set are given.

Dataset	Task	Use Case	Models	F1	Description
Ag News	Topic Detection	1.1	BERT	0.98	**Training Labels**: *World*, *Business*, and *Sport*
		1.2	DistilBERT	0.97	**Drift**: Simulated with one new class label: *Science/Tech*
		1.3	RoBERTa	0.98	**Dataset Split**: 59.480 train, 5.700 test, 30.520 without drift data stream, 31.900 drifted dataset stream
20 Newsgroups	Topic Detection	2.1	BERT	0.88	**Training Labels**: *Technology*, *Sale-Ads*, *Politics*, *Religion*, *Science* (5 macro-labels)
		2.2	DistilBERT	0.87	**Drift**: Simulated with one new class: *Recreation* (1 macro-label)
		2.3	RoBERTa	0.88	**Dataset Split**: 5.080 train, 3.387 test, 5.560 without drift data stream, 3.655 drifted dataset stream
20 Newsgroups	Topic Detection	3	BERT	0.87	**Training Labels**: Training on 6 labels: 5 macro-labels: *Technology*, *Sale-Ads*, *Politics*, *Religion*, *Science* and a subset of the macro-label *Recreation*: *baseball* and *hockey* **Drift**: Simulated with another subset of the macro-label *Recreation*: *motorcycles* and *autos* **Dataset Split**: 5.744 train, 3.829 test, 6.304 without drift data stream, 1.805 drifted dataset stream

All of these techniques need a reference dataset. As for DRIFTLENS, we use the embedding vectors from the training set as the reference data. However, as we will show in Section 4.5.3, the running time of some detectors is significantly affected by the dimensionality of the reference set m_b. To focus on near real-time drift detection and manage the complexity of extensive experiments, we limit the dimensionality of the reference window. This constraint ensures a fair comparison and achieves a reasonable execution time for the detector, enabling drift prediction within a data window of up to 30 seconds. To this end, we set the maximum reference window size to $m_b \approx 14{,}000$ for MMD and $m_b \approx 8{,}500$ for LSDD. In contrast, we used the full training set for KS and CVM due to their faster execution times. When the training set size exceeds these limits, we create the reference set by randomly sampling a balanced number of samples from the training set up to the maximum size. For the 20 Newsgroup dataset, we used $m_b \approx 1{,}700$ for use case 2 and $m_b \approx 2{,}050$ for use case 3 across all state-of-the-art detectors to maintain balanced reference sets.

4.5.2 Drift detection performance evaluation

This evaluation aims to assess the effectiveness DRIFTLENS in detecting windows with drifted samples of varying severity across different NLP models and datasets. We approach the drift detection problem as a binary classification task–predicting whether a new data window contains drift.

Evaluation metrics. We compute the *accuracy* as the evaluation measure for assessing the ability of the drift detectors in predicting drift with varying degrees of severity $D\% \in \{0\%, 5\%, 10\%, 15\%, 20\%\}$. For window instances comprising any

percentage of drifted examples $D\% > 0$ the ground truth is set to 1 (i.e., drifted window); otherwise, it is set to 0 (i.e., non-drifted window). Data windows where drift is absent (i.e., $D = 0\%$) serve as a basis for assessing type I errors, which occur when the technique erroneously detects drift despite its absence (false alarm).

Given the close relationship between the predictions of drifted and non-drifted windows, a technique that consistently predicts drift should be considered unreliable. Hence, we introduce a metric that simultaneously evaluates the accuracy in identifying windows with and without drift, computed as follows. Firstly, we compute the average accuracy in detecting drift with various severities:

$$\bar{A}_{\text{drift}} = (A_{5\%} + A_{10\%} + A_{15\%} + A_{20\%})/4 \tag{4.8}$$

Next, we introduce a metric called *Harmonic Drift Detection* (H_{DD}), which is the harmonic mean between the mean accuracy for non-drifted windows and the mean accuracy of the drifted windows $\bar{A}_{\text{drift}}$, and computed with the following equation:

$$H_{DD} = \frac{2}{\frac{1}{A_{0\%}} + \frac{1}{\bar{A}_{\text{drift}}}} \tag{4.9}$$

The H_{DD} is a real number within the range of $[0, 1]$, quantifying the overall effectiveness of the drift detector in distinguishing between windows with and without drift. This metric penalizes techniques that consistently predict the presence or absence of drift (i.e., $A_{0\%} = 0$ or $\bar{A}_{\text{drift}} = 0$), resulting in $HDD = 0$ in such instances.

Results. For every drift percentage $D\%$, each use case, and every window size m_w analyzed, we randomly select 100 windows, each consisting of m_w samples, with $D\%$ drawn from the drift dataset. Accuracy is computed over these 100 windows, iterated 5 times, and subsequently averaged. At each run, the threshold of DRIFTLENS is re-estimated, and the reference set of the other state-of-the-art detectors is resampled.

Tables 4.2 and 4.3 report the drift prediction accuracy, categorized by severity, along with the H_{DD} score for all experimental drift detectors, classifiers, and use cases. Table 4.2 employs larger data windows compared to Table 4.3 due to the datasets containing more samples. For each use case, drift detector, and window size, the following values are provided: (i) the mean accuracy by drift percentage, and (ii) the overall harmonic drift detection mean H_{DD}.

For the larger dataset (Table 4.2), DRIFTLENS consistently achieves superior drift prediction performance across all experimental use cases and window sizes. Notably, it obtains a $H_{DD} \geq 0.93$ for all use cases and window sizes, except for use case 1.3 with a window of 500 samples where $H_{DD} \geq 0.87$. All the detectors effectively distinguish between normal ($D\% = 0\%$) and drifted windows ($D\% \geq 0\%$). However, DRIFTLENS is the only detector able to detect drifted windows with low severity ($D\% = 5\%$), while the other detectors start predicting drift when present with higher severity. The average H_{DD} achieved by DRIFTLENS on these use cases is 0.97. In contrast, the other detectors are less effective in distinguishing between windows without and with drift. The average H_{DD} is 0.79 for MMD, 0.81 for KS, 0.80 for LSDD, and 0.82 for CVM. Thus, on this evaluation, DRIFTLENS outperforms the best state-of-the-art detector by 0.15 in the H_{DD} score.

For smaller data volume (Table 4.3), it still proves to be the most effective technique overall. It shows an H_{DD} score greater than the other detectors for all the use cases and window sizes, except for use case 2.3 with a window of 500 samples. For these use cases, the results of state-of-the-art detectors are often unreliable. Detectors such as KS and CVM tend to predict always the presence of drift. On average, DRIFTLENS achieves an H_{DD} of 0.83. Instead, the average H_{DD} is 0.58 for MMD, 0.09 for KS, 0.47 for LSDD, and 0.07 for CVM. For these use cases, DRIFTLENS outperforms the best detector by 0.25 in the H_{DD} score.

Summary. DRIFTLENS proves to be highly effective in detecting drift. It stands out as the only detector achieving strong performance across all experimental use cases, making it the most reliable and generally applicable drift detection solution across different NLP models and datasets.

4.5.3 Efficiency evaluation

This evaluation aims to determine the effectiveness of DRIFTLENS in performing near real-time drift detection. To achieve this, we compare the running times of different drift detectors while varying the sizes of the reference window, the data stream window, and the embedding dimensionality. Given the high dimensionalities tested, these experiments are conducted on synthetically generated embedding vectors.

Evaluation metrics. We measure the mean execution time in seconds required to detect the presence of drift (i.e., the drift label) for each data window, given the

Table 4.2 **Drift detection performance evaluation for *larger* data volume**. For each drift detector and window size $m_w \in \{500, 1000, 2000\}$ are reported (i) the accuracy separately per drift percentage $D\% \in \{0\%, 5\%, 10\%, 15\%, 20\%\}$, and (ii) the harmonic drift detection mean H_{DD}. Accuracy is computed over 100 windows and averaged by repeating 5 runs. The best-performing detector for each use case based on the H_{DD} overall is highlighted, and separately per window size is in bold.

Use Case	Drift Detector	Data Stream Window Size m_w = 500						Data Stream Window Size m_w = 1000						Data Stream Window Size m_w = 2000					
		Drift Percentage $D\%$						Drift Percentage $D\%$						Drift Percentage $D\%$					
		0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}
1.1 Ag News BERT	MMD	1.00	0.00	0.10	0.96	1.00	*0.68*	1.00	0.00	0.83	1.00	1.00	*0.83*	1.00	0.06	1.00	1.00	1.00	*0.87*
	KS	1.00	0.01	0.21	0.98	1.00	*0.71*	0.99	0.03	0.95	1.00	1.00	*0.85*	1.00	0.38	1.00	1.00	1.00	*0.91*
	LSDD	1.00	0.00	0.13	0.93	1.00	*0.68*	1.00	0.00	0.83	1.00	1.00	*0.83*	1.00	0.12	1.00	1.00	1.00	*0.88*
	CVM	1.00	0.01	0.22	0.99	1.00	*0.71*	0.99	0.02	0.99	1.00	1.00	*0.86*	1.00	0.44	1.00	1.00	1.00	*0.92*
	DRIFTLENS	0.99	0.83	1.00	1.00	1.00	***0.97***	1.00	0.98	1.00	1.00	1.00	***1.00***	0.97	1.00	1.00	1.00	1.00	***0.98***
1.2 Ag News DistilBERT	MMD	1.00	0.00	0.05	0.91	1.00	*0.66*	1.00	0.00	0.78	1.00	1.00	*0.82*	1.00	0.03	1.00	1.00	1.00	*0.86*
	KS	1.00	0.01	0.14	0.93	1.00	*0.68*	1.00	0.03	0.87	1.00	1.00	*0.84*	1.00	0.25	1.00	1.00	1.00	*0.89*
	LSDD	1.00	0.00	0.06	0.90	1.00	*0.66*	1.00	0.00	0.80	1.00	1.00	*0.82*	1.00	0.03	0.99	1.00	1.00	*0.86*
	CVM	1.00	0.01	0.15	0.97	1.00	*0.69*	1.00	0.04	0.93	1.00	1.00	*0.85*	0.99	0.31	1.00	1.00	1.00	*0.90*
	DRIFTLENS	0.99	0.71	1.00	1.00	1.00	***0.96***	0.99	0.98	1.00	1.00	1.00	***0.99***	0.99	1.00	1.00	1.00	1.00	***0.99***
1.3 Ag News RoBERTa	MMD	1.00	0.00	0.18	0.98	1.00	*0.70*	1.00	0.01	0.94	1.00	1.00	*0.85*	1.00	0.09	1.00	1.00	1.00	*0.87*
	KS	1.00	0.01	0.16	0.92	1.00	*0.69*	1.00	0.02	0.85	1.00	1.00	*0.84*	1.00	0.18	1.00	1.00	1.00	*0.89*
	LSDD	1.00	0.00	0.19	0.99	1.00	*0.71*	1.00	0.01	0.98	1.00	1.00	*0.86*	1.00	0.26	1.00	1.00	1.00	*0.90*
	CVM	1.00	0.00	0.15	0.96	1.00	*0.69*	1.00	0.02	0.90	1.00	1.00	*0.84*	1.00	0.16	1.00	1.00	1.00	*0.88*
	DRIFTLENS	1.00	0.09	0.98	1.00	1.00	***0.87***	1.00	0.47	1.00	1.00	1.00	***0.93***	1.00	0.96	1.00	1.00	1.00	***1.00***

Table 4.3 **Drift detection performance evaluation for *smaller* data volume**. For each drift detector and window size $m_w \in \{250, 500, 1000\}$ are reported (i) the accuracy separately per drift percentage $D\% \in \{0\%, 5\%, 10\%, 15\%, 20\%\}$, and (ii) the harmonic drift detection mean H_{DD}. Accuracy is computed over 100 windows and averaged by repeating 5 runs. The best-performing detector for each use case based on the H_{DD} overall is highlighted, and separately per window size is in bold.

Use Case	Drift Detector	Data Stream Window Size m_w = 250						Data Stream Window Size m_w = 500						Data Stream Window Size m_w = 1000					
		Drift Percentage $D\%$						Drift Percentage $D\%$						Drift Percentage $D\%$					
		0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}	0%	5%	10%	15%	20%	H_{DD}
2.1 20News BERT	MMD	0.83	0.33	0.74	0.98	1.00	*0.80*	0.06	1.00	1.00	1.00	1.00	*0.11*	0.00	1.00	1.00	1.00	1.00	*0.00*
	KS	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	1.00	0.01	0.04	0.19	0.48	*0.31*	0.98	0.14	0.41	0.76	0.96	*0.72*	0.61	0.65	0.93	1.00	1.00	*0.72*
	CVM	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	0.92	0.28	0.68	0.96	1.00	***0.81***	0.89	0.42	0.92	1.00	1.00	***0.86***	0.84	0.78	1.00	1.00	1.00	***0.89***
2.2 20News DistilBERT	MMD	1.00	0.01	0.08	0.61	0.98	*0.59*	0.95	0.12	0.63	1.00	1.00	*0.80*	0.56	0.72	1.00	1.00	1.00	*0.70*
	KS	0.42	0.68	0.89	1.00	1.00	*0.57*	0.01	0.99	1.00	1.00	1.00	*0.02*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	1.00	0.00	0.01	0.05	0.25	*0.14*	1.00	0.01	0.06	0.35	0.83	*0.48*	0.98	0.03	0.33	0.92	0.99	*0.72*
	CVM	0.31	0.74	0.90	1.00	1.00	*0.46*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	1.00	0.15	0.72	0.99	1.00	***0.83***	1.00	0.23	0.95	1.00	1.00	***0.89***	1.00	0.40	1.00	1.00	1.00	***0.92***
2.3 20News RoBERTa	MMD	1.00	0.00	0.06	0.40	0.91	*0.51*	0.96	0.22	0.78	1.00	1.00	***0.84***	0.37	0.95	1.00	1.00	1.00	*0.54*
	KS	0.33	0.84	0.06	0.40	0.91	*0.41*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	1.00	0.00	0.00	0.03	0.09	*0.06*	1.00	0.00	0.03	0.17	0.51	*0.30*	0.97	0.09	0.36	0.86	1.00	*0.72*
	CVM	0.26	0.87	0.98	1.00	1.00	*0.41*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	0.98	0.08	0.26	0.57	0.88	***0.61***	0.98	0.07	0.34	0.82	0.99	*0.71*	0.99	0.07	0.53	0.98	1.00	***0.78***
3 20News BERT	MMD	1.00	0.00	0.04	0.59	0.98	*0.57*	0.99	0.08	0.76	1.00	1.00	*0.83*	0.48	0.89	1.00	1.00	1.00	*0.61*
	KS	0.01	1.00	1.00	1.00	1.00	*0.02*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	LSDD	1.00	0.00	0.00	0.03	0.35	*0.17*	1.00	0.00	0.02	0.55	0.99	*0.56*	1.00	0.01	0.41	1.00	1.00	*0.75*
	CVM	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*	0.00	1.00	1.00	1.00	1.00	*0.00*
	DRIFTLENS	0.97	0.21	0.65	0.96	1.00	***0.82***	0.98	0.35	0.94	1.00	1.00	***0.89***	0.98	0.63	1.00	1.00	1.00	***0.94***

embedding vectors already extracted. These experiments are conducted on an Apple M1 MacBook Pro 13 (2020) equipped with 16GB of RAM.

Results. Figure 4.7 shows the execution time in seconds, depicted on a logarithmic scale, for each detector in predicting the drift label while varying: (a) the reference window size m_b, (b) the data stream window size m_w, and (c) the embedding dimensionality d. In computing the execution time, one dimension is varied at a time while the others are held constant at the following values: window size $m_w = 1,000$, embedding dimensionality $d = 1,000$, and reference window size $m_b = 5,000$.

These charts show that DRIFTLENS consistently outperforms all evaluated techniques in terms of running time, operating at least 5 times faster. Additionally, its execution time experiences nearly negligible growth as the number of analyzed variables increases, unlike other techniques which are significantly impacted by changes in window sizes and embedding dimensionality.

Figure 4.8 shows the running time of DRIFTLENS only when processing large volumes of data. This time, the reference window m_b is increased up to $500,000$ samples, and the data stream window size m_w is up to $10,000$. Other drift detectors typically cannot handle such dimensionalities. When varying the reference window size (m_b), the data stream window size remains fixed at $m_w = 5,000$. Conversely, when varying the data stream window size (m_w), the reference window size is held constant at $m_b = 500,000$.

Figure 4.8-(a) shows DRIFTLENS is nearly unaffected by the reference window's size, as only distributions are loaded during the online phase (mean vector and covariance matrix with dimensionality d'). Therefore, it can effectively use training datasets with a huge number of samples as the reference set. This feature is particularly useful because, typically, the amount of annotated data can increase during the life cycle of an application. DRIFTLENS can exploit all the annotated data to model the absence of drift without increasing the complexity.

Figure 4.8-(b) demonstrates that the running time experiences minimal growth as the window size increases. Hence, it can be effectively employed for real-time drift detection, even on data streams with high throughput. Remarkably, the running time consistently remains below 0.2 seconds. This feature is particularly useful in real-world applications, such as drift detection for content moderation or topic detection models where the amount of produced and processed texts is huge.

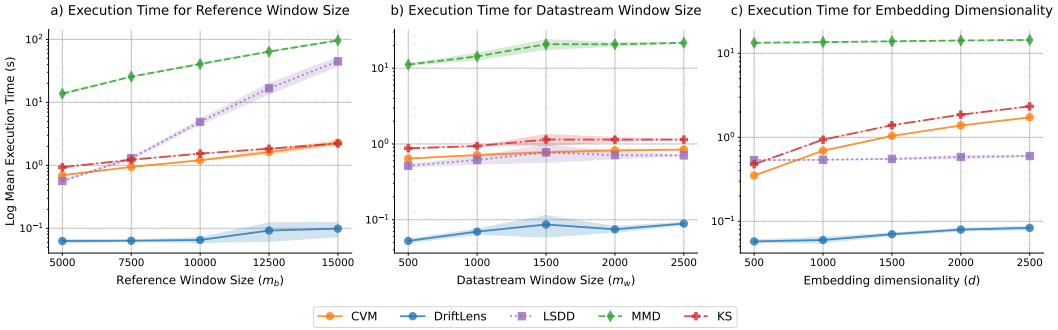


Fig. 4.7 **Running time comparison on a logarithmic scale.** For each drift detector, the mean and standard deviation of the running time in seconds to process a new window are reported, with variations in a) the reference window size, b) the data stream window size, and c) the embedding dimensionality, while keeping the other parameters fixed. The fixed values are: reference window size $m_b = 5000$, window size $m_w = 1000$, and embedding dimensionality $d = 1000$. The mean and standard deviation are computed over 5 runs.

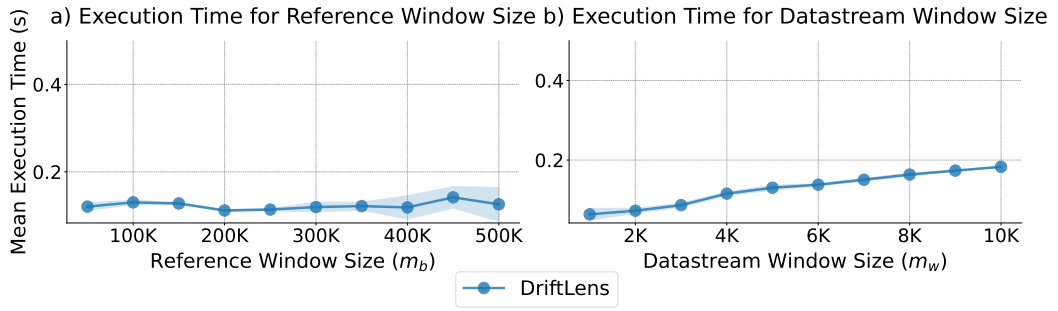


Fig. 4.8 DRIFTLENS mean running time in seconds by varying (a) the reference m_b and (b) the data stream m_w (b) window sizes.

Summary. DRIFTLENS proved to be the fastest drift detector in terms of running time, capable of near real-time drift detection regardless of the volume of data in both the reference set and the data stream.

4.5.4 Drift curve evaluation

This evaluation aims to assess DRIFTLENS' capability to accurately represent and characterize the drift curve.

Evaluation metrics. We measure the Spearman correlation coefficient [207] to assess the correlation between the *per-batch* (FDD) distances over time and the injected drift in the data stream. The Spearman correlation evaluates the monotonic

Table 4.4 **Drift patterns evaluation.** Spearman correlation between the amount of drift and the *per-batch* distribution distance (FDD) for different drift patterns and use cases.

	Drift Pattern		
	Sudden drift	*Incremental drift*	*Recurrent drift*
Spearman Corr.	$0.875 \pm .02$	$0.993 \pm .01$	$0.849 \pm .00$

relationship of two curves, making it suitable for the non-linearities observed in the assessed patterns (e.g., sudden and periodic). The coefficient ranges from -1 to +1, where +1 (-1) indicates a perfect positive (negative) monotonic relationship, and 0 indicates no monotonic relationship. The injected drift curve consists of 0 in windows without drift and the percentage of drift ($D\%$) in windows containing drift. A high correlation coefficient implies a high capability to correctly represent the drift curve over time. Additionally, we conduct a qualitative assessment of the *per-batch* and *per-label* drift curves for three drift patterns.

Data streams generation. The data streams are generated by randomly sampling 100 windows, each containing $1,000$ samples. In the sudden pattern, drift occurs after 50 windows and remains constant at a percentage of $D\% = 40\%$. In the incremental pattern, drift initiates after 50 windows with a percentage of $D\% = 20\%$ and increases by $\Delta D\% = 1\%$ after each window. For the periodic pattern, there are 20 windows without drift followed by 20 windows containing $D\% = 40\%$ of recurring drift.

Results. Table 4.1 presents the mean and standard deviation of the Spearman correlation coefficient computed across all experimental use cases (Table 2.9). Experiments are repeated 5 times and averaged. The *per-batch* (FDD) distance exhibits a high correlation with the generated drift curve. The Spearman correlation coefficient consistently exceeds 0.85 for all three considered drift patterns. Remarkably, for the incremental pattern, it is nearly 1. These results quantitatively demonstrate the ability of DRIFTLENS to accurately characterize the drift trend over time.

Figures 4.9 and 4.10 qualitatively show two examples of DRIFTLENS monitors obtained by generating a data stream (a) *without* drift, and with a *sudden* (b), *incremental* (c), and *periodic* (d) drift patterns for use cases 1.1 and 2.1, respectively. Each drift pattern has been generated with the previously described settings. These figures qualitatively show that the *per-batch* and *per-label* curves of distances (FDD)

modeled by DRIFTLENS are highly coherent with the generated drift patterns for both use cases. This trend is confirmed for all experimental use cases.

For the use case 1.1 (Figure 4.9), the *per-label* curves indicate that the *world* (red) and *business* (green) labels exhibit the most significant impact from drift. This is likely because most examples of the newly injected class (i.e., *Science/Technology*) are classified with these labels. A possible reason is that the *World* label is the most general class, and the *Business* label is more related to *Science/Technology* than the *Sport* one.

Similarly, for the use case 2.1 (Figure 4.10), labels *Politics* and *Science* are severely, and *Sale-Ads* moderately impacted by drift. Instead, *Technology* and *Religion* are impacted almost negligibly.

Summary. The trend of the drift curve (i.e., *FDD* distances) produced by DRIFTLENS closely aligns with the observed drift severity levels and patterns. Additionally, the *per-label* distances offer valuable insights for characterizing the drift.

4.5.5 Parameters sensitivity evaluation

This evaluation aims to assess the robustness and sensitivity of DRIFTLENS to its parameters. To achieve this, we evaluate its performance in drift prediction by varying the values of the following parameters: (i) the number of randomly sampled windows n_{th} used to estimate the threshold value, (ii) the threshold sensitivity parameter value T_α, and (iii) the number of principal components used for reducing the embedding dimensionality d'. We additionally measure the variation in performance by decreasing the amount of reference data m_b to assess how the performance would change, even if this is not a real parameter of the framework (DRIFTLENS always exploits the full reference set).

Evaluation metrics. We compute the accuracy in predicting drift across various severity levels and the H_{DD} metric by varying the parameter values.

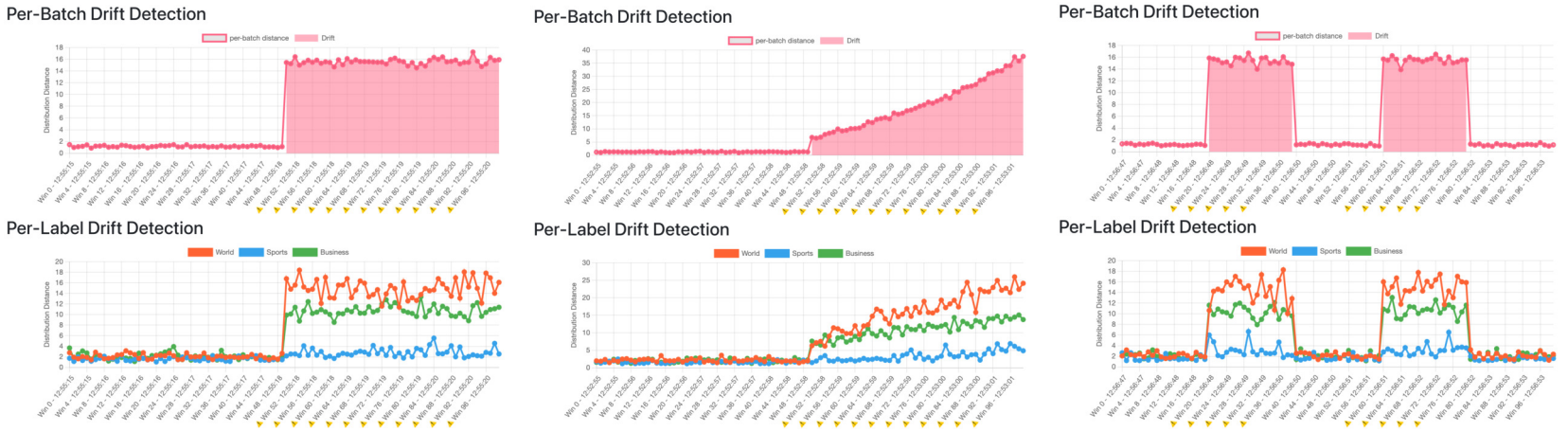
Results. Table 4.5 shows the accuracy and H_{DD} metrics, varying one parameter at a time while maintaining the others values fixed. The fixed parameter values are as follows: $n_{th} = 10{,}000$, $T_\alpha = 0.01$, and $d' = 150$. The experiments are repeated 5 times and averaged for the use case 1.1. The default parameter values are underlined.

Table 4.5 Parameters sensitivity.

Use Case	Parameter	Drift Percentage $D\%$					
		0%	5%	10%	15%	20%	H_{DD}
Number of sampled windows for threshold estimation n_{th}							
1.1	$n_{th} = 1k$	1.00	1.00	1.00	1.00	1.00	***1.00***
	$n_{th} = 5k$	0.99	0.99	1.00	1.00	1.00	*0.99*
	$n_{th} \in \{100, \underline{10k}, 25k\}$	0.99	1.00	1.00	1.00	1.00	*0.99*
Threshold sensitivity T_α							
1.1	$T_\alpha = 0.00$	1.00	0.82	1.00	1.00	1.00	*0.97*
	$T_\alpha = \underline{0.01}$	0.99	1.00	1.00	1.00	1.00	***0.99***
	$T_\alpha = 0.05$	0.95	1.00	1.00	1.00	1.00	*0.97*
	$T_\alpha = 0.10$	0.90	1.00	1.00	1.00	1.00	*0.95*
	$T_\alpha = 0.25$	0.75	1.00	1.00	1.00	1.00	*0.86*
Number of principal components d'							
1.1	$d' = 50$	0.97	1.00	1.00	1.00	1.00	*0.98*
	$d' = 100$	0.99	0.99	1.00	1.00	1.00	***0.99***
	$d' \in \{\underline{150}, 200, 250\}$	0.99	1.00	1.00	1.00	1.00	***0.99***
Reference window size m_b							
1.1	$w_b\% \in \{20\%, 40\%, 60\%\}$	0.97	1.00	1.00	1.00	1.00	*0.98*
	$w_b\% \in \{80\%, \underline{100\%}\}$	0.99	1.00	1.00	1.00	1.00	***0.99***

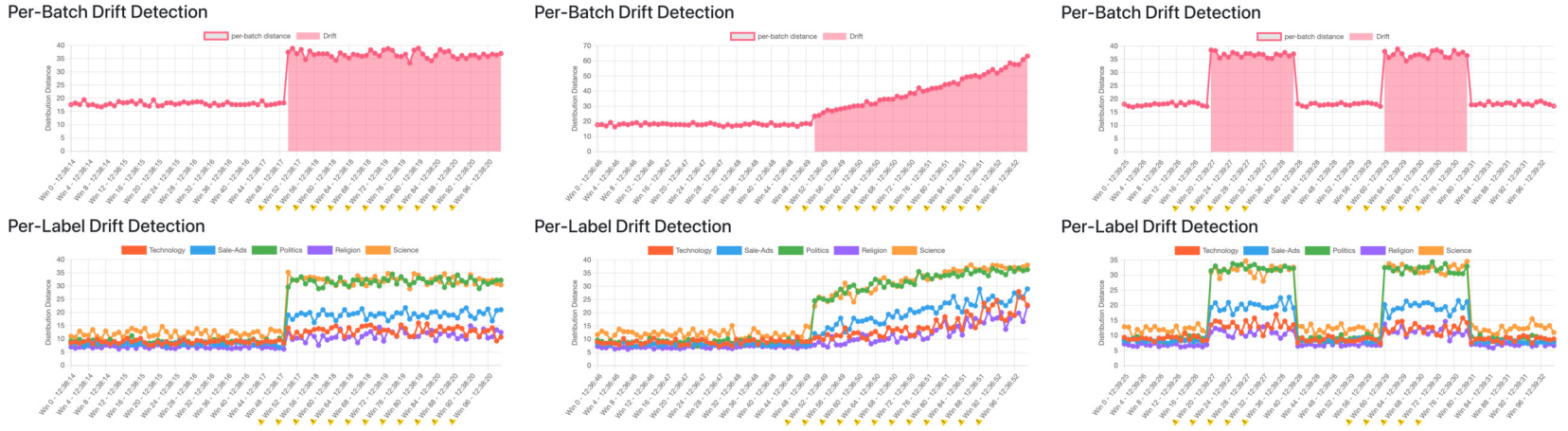
The results indicate that altering the parameter values does not significantly impact performance, resulting in a maximum performance reduction of 0.03. The only exception is the threshold sensitivity T_α. By increasing the value of this parameter, DRIFTLENS decreases the estimated threshold values. Therefore, the number of false positives– normal windows predicted as drift– increases. Indeed, it became less accurate when the drift was not present ($D\% = 0\%$).

Summary. DRIFTLENS's performance remains largely unaffected by parameter selection, showing that it is robust the its parameters. The only parameter that must be carefully selected is the threshold sensitivity. However, the default value of $T_\alpha = 0.01$ proved to be largely effective in all the experimental use cases.



(a) Sudden drift. (b) Incremental drift. (c) Periodic drift.

Fig. 4.9 Qualitative evaluation of the DRIFTLENS' drift curves for the sudden (a), incremental (b), and periodic (c) drift patterns (use case 1.1).



(a) Sudden drift. (b) Incremental drift. (c) Periodic drift.

Fig. 4.10 Qualitative evaluation of the DRIFTLENS' drift curves for the sudden (a), incremental (b), and periodic (c) drift patterns (use case 2.1).

4.6 Discussion

To address **Challenge 3** introduced in Chapter 1, this chapter proposes DRIFTLENS, an unsupervised concept drift detection framework designed for deep learning classifiers on unstructured data, including NLP. DRIFTLENS is able to detect concept and data drift affecting the classifiers' robustness over time for application scenarios characterized by the unavailability of true labels for newly processed data and high throughput. We demonstrate its superior effectiveness and efficiency to previous drift detection detectors in NLP classification tasks. However, the methodology is independent of the data type and can be applied to any classifier working on unstructured data (e.g., also images and speech classifiers), showing similar superior performance [26]. We also show that the drift curves modeled by DRIFTLENS are highly correlated with the amount of drift present and that the per-label drift analysis provides useful insights into understanding what labels are the most impacted by the drift. Finally, we show the robustness of the framework to its parameters.

4.6.1 Implications

This work significantly contributes to the research community by focusing on unsupervised drift detection, particularly for deep learning classifiers working on unstructured data like those used in NLP. For instance, DRIFTLENS empowers researchers and practitioners to monitor and identify concept and data drift in NLP models without relying on labeled data. This capability is crucial in ensuring model reliability over time, particularly in dynamic environments where data distribution changes are frequent and impactful. DRIFTLENS can have an impact on diverse domains, including NLP applications like topic detection and content moderation on online platforms, where vast amounts of data are produced and processed. DRIFTLENS operates in real-time even with massive data volumes, ensuring timely detection of concept drifts in dynamic environments. Its ability to handle high data throughput effectively positions it as a valuable tool for ensuring model robustness and reliability over time. Specifically, this work has three main implications:

(1) DRIFTLENS Framework. We release an open-source framework[12] that can be easily integrated into any deep learning classifier to monitor concept and data drift.

[12] https://github.com/grecosalvatore/drift-lens

We also release an application tool[13] employing the proposed methodology that contains a graphical user interface to facilitate the use of DRIFTLENS in performing concept and data drift experiments in new models and data [25].

(2) Ensuring classifiers robustness. DRIFTLENS can help in monitoring classifiers' robustness over time. It can be integrated into monitoring dashboards to check *whether* and *when* drift occurs, as well as identify which labels are most impacted in NLP applications that process large volumes of data, where true labels are absent. By offering early warnings to detect drifting distributions that could negatively impact model performance, this approach enhances the robustness and reliability of NLP production models. Machine learning developers can leverage this tool to continuously monitor their models and take corrective actions to restore the original performance of predictive models in real-world applications.

(3) Scalable concept drift detection. DRIFTLENS is highly effective in emerging scenarios characterized by high-dimensional data such as texts, images, and audio, where substantial volumes of processed data originate from social media and online platforms. Its efficiency and real-time processing capabilities empower accurate detection of concept drift without requiring GPU acceleration. This capability allows deployment on edge devices with low power consumption, facilitating monitoring in new applications such as autonomous driving. Here, the ability to promptly and reliably adapt models is essential for ensuring safe and efficient operations.

4.6.2 Limitations

DRIFTLENS uses the Fréchet distance to measure distributions' distances (i.e., FDD), and consequently, it potentially inherits some of its limitations:

(1) Noise and selection bias with small datasets. The Fréchet distance, and thus the FDD, may be influenced by noise and selection bias in distance calculations when computed on small datasets. However, empirical evidence demonstrates that DRIFTLENS performs well even with smaller reference data and window sizes.

(2) Limited number of statistics. The FDD score utilizes a limited number of statistics in its distance calculation, specifically the mean and covariance. Consequently,

[13] https://github.com/grecosalvatore/DriftLensDemo

it may not fully capture all aspects of the distributions. While it accounts for the first two moments of the distributions, it overlooks others, such as skewness and kurtosis.

(3) Distance score range. The *FDD* score computes distances within the range of $[0, \infty]$, but for better interpretability, it would be advantageous to normalize it to the range of $[0, 1]$. This can be achieved by representing the distance as a relative value compared to the estimated thresholds.

4.6.3 Future research directions

A Future research direction includes the integration of explainability techniques, based on embedding representations [22, 33] or on concepts [89], to develop an explainable concept drift detection tool. To this end, the framework can be extended to not only characterize drift by identifying the labels the most impacted but also explain to humans how they are changing, using explainability techniques.

Another important future research direction consists of addressing the problem of drift adaptation in unsupervised or supervised settings with limited labels, to retrain the classifier on the new concepts once drift is detected. This requires a first step of drift localization–identifying the drifting data points, and then appropriate methodology to retrain the classifier by using only the new concepts or a combination of the old and new ones. For the drift localization, the *per-label* drift analysis performed by DRIFTLENS can potentially simplify the task by focusing on the most impacted labels, thereby reducing the research space.

Finally, another future research direction includes the extension of the methodology to other deep learning models suitable for various tasks beyond classification, including LLMs based on transformer-decoder architectures. This expansion could facilitate drift detection and localization across a wider range of applications, such as text generation, machine translation, and question answering, providing deeper insights into performance degradation in more complex and dynamic NLP systems.

Chapter 5

Conclusion

This dissertation proposes innovative solutions to inspect deep learning-based NLP models, aiming to enhance their trustworthiness, reliability, and robustness in real-world applications. Specifically, it addressed three main challenges:

- **Challenge 1:** *"Explaining the Predictions of NLP Classifiers"*.
- **Challenge 2:** *"Mitigating the use of Protected Attributes by NLP Classifiers"*.
- **Challenge 3:** *"Detecting Concept and Data Drift in NLP Classifiers"*.

To address **Challenge 1**, in Chapter 2, we proposed T-EBANO [22]: a novel explanation framework designed for deep learning NLP classifiers. It provides explanations in two forms: (1) *local explanations*, which identify the most important predictive words used to classify individual input texts, and (2) *global explanations*, which assess the overall importance of each word for each class label. We demonstrated its effectiveness and broad applicability across various NLP models and classification tasks. By leveraging the internal knowledge of the models, it generates explanations more efficiently and with greater precision compared to previous model-agnostic techniques, which do not utilize internal model knowledge. Additionally, we showed that the proposed explanations align well with human judgment.

To address **Challenge 2**, in Chapter 3, we proposed NLPGUARD [23]: a novel framework designed to mitigate the reliance on protected attributes in NLP classifiers. The framework ensures "*fairness through unawareness*" by reducing the model's dependence on protected attributes while maintaining predictive performance. We

assessed the framework's sensitivity to its components, its effectiveness, and its general applicability in reducing the reliance of NLP classifiers on protected attributes while maintaining predictive performance across various NLP classification tasks. We also demonstrate its superior effectiveness in achieving this objective compared to previous bias mitigation techniques. Moreover, we demonstrated that LLM-based annotation of protected attributes outperforms human annotators, thus enabling the automation of the framework.

To address **Challenge 3**, in Chapter 4, we proposed DRIFTLENS [24–26]: a novel drift detection framework for deep learning classifiers working on unstructured data, including NLP. DRIFTLENS is an unsupervised drift detection technique that does not require true labels in newly processed data. It is based on the distribution distances of the embedding representations produced by deep learning models. Thanks to its low complexity, it can run in real-time, even in applications characterized by high-dimensional and large amounts of processed data. We demonstrated its superior performance and efficiency compared to previous detectors across several NLP classification tasks and models. Additionally, we showed its ability to correctly model and characterize drift trends over time.

5.1 Towards a unified NLP inspection tool

In the previous chapters, we presented the three frameworks as stand-alone solutions since each addresses a significant pillar of the research literature. However, they all belong to the broader domains of *Trustworthy* and *Responsible AI* [18–20], aiming to solve critical issues, including transparency, fairness, accountability, robustness, and ethical considerations in the development and deployment of AI systems, in this case, based on NLP. Nevertheless, we believe these frameworks can be integrated into a unified NLP inspection tool addressing all the issues together, given their closely related nature.

To this end, we envision a unified framework comprising the three proposed sub-modules for inspecting NLP classifier deployed in production applications to (1) Explain the predictions of newly processed texts using local and global explanations; (2) Identify and monitor the fairness of the classifier by assessing the use of protected attributes in the predictions; and (3) Monitor and detect drift in the distributions.

Specifically, both T-EBANO and DRIFTLENS work on the embedding representations. Therefore, these representations can be extracted at once along with the predicted labels for a new window of processed data. Then, T-EBANO generates the local and the global explanations, while DRIFTLENS computes the distribution distances for detecting possible drift.

The global explanations produced by T-EBANO can be used by NLPGUARD for identifying the use of protected attributes in new texts, utilizing T-EBANO as the *Explainer* component. When drift is detected, the *per-label* analysis of DRIFTLENS, and the global explanations produced by T-EBANO over time can provide useful insights into which labels are affected by drift while providing explanations about the reasons behind it.

These insights enable developers and researchers to take appropriate actions to restore the classifier's predictive performance when drift is detected. For instance, explanations may reveal emerging concepts in specific classes that require inclusion in model training or the emergence of entirely new classes. Finally, before re-training the classifier, the new data are processed by NLPGUARD to ensure fairness while restoring the predictive performance.

By integrating these capabilities into a unified framework, we aim to establish a comprehensive monitoring system that ensures explainability, fairness, reliability, and robustness in AI systems deployed for real-world NLP applications. This framework enhances trust through detailed predictive explanations, evaluates fairness by analyzing the influence of protected attributes, and identifies distribution drift to monitor model performance. Prioritizing these aspects promotes the adoption of trustworthy and ethical AI practices, fostering trust and equitable outcomes in AI-based systems over time.

5.2 Future research directions

The proposed frameworks are suitable for the NLP domain. However, in the future, they can be extended to address the same challenges for classifiers working with multiple modalities simultaneously (multimodal models) [208]. The explainability framework T-EBANO has already been redesigned and implemented for explaining image classifiers based on convolutional neural networks (CNNs) [33]. The

DRIFTLENS framework already works with image and speech classifiers and demonstrated similar performance as showed in NLP [26]. In contrast, NLPGUARD must be completely redesigned to mitigate the use of pixels' features corresponding to protected attributes in image classification.

Future work must enhance and adapt these frameworks to effectively address the challenges posed by multimodal models. Multimodal models integrate diverse data modalities, such as text and images, where the interactions between these modalities significantly influence model behavior and decision-making processes. One of the primary challenges lies in providing nuanced explanations for model decisions that consider the complex interplay between different data types [209]. For instance, explaining how textual and visual information jointly contribute to predictions requires powerful methods that comprehensively capture and present these interactions. Moreover, detecting and mitigating biases in multimodal models is another crucial aspect [210]. Biases can manifest differently across modalities and may interact in complex ways, impacting the fairness and reliability of model outcomes. Enhancing the frameworks to effectively identify and address biases arising from multimodal interactions will be pivotal in ensuring equitable and accurate model performance across diverse application domains. Finally, the same challenge of interaction between modalities also applies to detecting concepts and data drift. Indeed, concept drift can manifest in each modality independently but also in the combination of the modalities together.

Future research can evolve these frameworks to address multimodal complexities, enhancing their capability to provide effective explanations of model decisions and improving their ability to detect and mitigate biases while ensuring robustness over time. This evolution is essential for advancing the interpretability, fairness, and robustness of AI systems operating in multimodal environments.

References

[1] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[3] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[4] Paula Fortuna, Juan Soler, and Leo Wanner. Toxic, hateful, offensive or abusive? what are we really classifying? an empirical analysis of hate speech datasets. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 6786–6794, Marseille, France, May 2020. European Language Resources Association.

[5] Anna Schmidt and Michael Wiegand. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain, April 2017. Association for Computational Linguistics.

[6] Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Comput. Surv.*, 51(4), jul 2018.

[7] Vedant Bhatia, Prateek Rawat, Ajit Kumar, and Rajiv Ratn Shah. End-to-end resume parsing and finding candidates for a job description using bert. *arXiv preprint arXiv:1910.03089*, 2019.

[8] Arvind Kumar Sinha, Md Amir Khusru Akhtar, and Ashwani Kumar. Resume screening using natural language processing and machine learning: A systematic review. *Machine Learning And Information Processing: Proceedings Of ICMLIP 2020*, pages 207–214, 2021.

[9] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[10] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), aug 2018.

[11] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021.

[12] Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings, 2016.

[13] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2019.

[14] Xuezhi Wang, Haohan Wang, and Diyi Yang. Measure and improve robustness in nlp models: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4569–4586, 2022.

[15] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2018.

[16] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[17] Nina Grgic-Hlaca, Muhammad Bilal Zafar, Krishna P. Gummadi, and Adrian Weller. The case for process fairness in learning: Feature selection for fair decision making. In *Proceedings of the NIPS Symposium on Machine Learning and the Law*, volume 1, page 2, 2016.

[18] Davinder Kaur, Suleyman Uslu, Kaley J. Rittichier, and Arjan Durresi. Trustworthy artificial intelligence: A review. *ACM Comput. Surv.*, 55(2), jan 2022.

[19] Bo Li, Peng Qi, Bo Liu, Shuai Di, Jingen Liu, Jiquan Pei, Jinfeng Yi, and Bowen Zhou. Trustworthy ai: From principles to practices. *ACM Comput. Surv.*, 55(9), jan 2023.

[20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[21] Brent Daniel Mittelstadt, Patrick Allo, Mariarosaria Taddeo, Sandra Wachter, and Luciano Floridi. The ethics of algorithms: Mapping the debate. *Big Data & Society*, 3(2):2053951716679679, 2016.

[22] Francesco Ventura, Salvatore Greco, Daniele Apiletti, and Tania Cerquitelli. Trusting deep learning natural-language models via local and global explanations. *Knowledge and Information Systems*, 2022.

[23] Salvatore Greco, Ke Zhou, Licia Capra, Tania Cerquitelli, and Daniele Quercia. NLPGuard: A framework for mitigating the use of protected attributes by nlp classifiers. *Proc. ACM Hum.-Comput. Interact.*, 8(CSCW2):1–25, nov 2024.

[24] Salvatore Greco and Tania Cerquitelli. Drift lens: Real-time unsupervised concept drift detection by evaluating per-label embedding distributions. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 341–349, 2021.

[25] Salvatore Greco, Bartolomeo Vacchetti, Daniele Apiletti, and Tania Cerquitelli. Driftlens: A concept drift detection tool. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, pages 806–809. OpenProceedings.org, 2024.

[26] Salvatore Greco, Bartolomeo Vacchetti, Daniele Apiletti, and Tania Cerquitelli. Unsupervised concept drift detection from deep learning representations in real-time, 2024.

[27] Salvatore Greco, Ke Zhou, Licia Capra, Tania Cerquitelli, and Daniele Quercia. Nlpguard: A framework for mitigating the use of protected attributes by nlp classifiers, 2024.

[28] Giuseppe Attanasio, Salvatore Greco, Moreno La Quatra, Luca Cagliero, Michela Tonti, Tania Cerquitelli, and Rachele Raus. E-mimic: Empowering multilingual inclusive communication. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4227–4234, 2021.

[29] Raus, Rachele, Tonti, Michela, Cerquitelli, Tania, Cagliero, Luca, Attanasio, Giuseppe, La Quatra, Moreno, and Greco, Salvatore. L'analyse du discours et l'intelligence artificielle pour réaliser une écriture inclusive : le projet emimic. *SHS Web Conf.*, 138:01007, 2022.

[30] Moreno La Quatra, Salvatore Greco, Luca Cagliero, and Tania Cerquitelli. Inclusively: An ai-based assistant for inclusive writing. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track*, pages 361–365, Cham, 2023. Springer Nature Switzerland.

[31] Moreno La Quatra, Salvatore Greco, et al. Building foundations for inclusiveness through expert-annotated data. In *EDBT/ICDT Workshops*, 2024.

[32] Giuseppe Gallipoli, Moreno La Quatra, Daniele Rege Cambrin, Salvatore Greco, and Luca Cagliero. DANTE at geolingit: Dialect-aware multi-granularity pre-training for locating tweets within italy. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023), Parma, Italy, September*

7th-8th, 2023, volume 3473 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023.

[33] Francesco Ventura, Salvatore Greco, Daniele Apiletti, and Tania Cerquitelli. Explaining deep convolutional models by measuring the influence of interpretable features in image classification. *Data Mining and Knowledge Discovery*, pages 1–58, 2023.

[34] Simone Monaco, Salvatore Greco, Alessandro Farasin, Luca Colomba, Daniele Apiletti, Paolo Garza, Tania Cerquitelli, and Elena Baralis. Attention to fires: Multi-channel deep learning models for wildfire severity prediction. *Applied Sciences*, 11(22):11060, 2021.

[35] Luca Colomba, Alessandro Farasin, Simone Monaco, Salvatore Greco, Paolo Garza, Daniele Apiletti, Elena Baralis, and Tania Cerquitelli. A dataset for burned area delineation and severity estimation from satellite imagery. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 3893–3897, New York, NY, USA, 2022. Association for Computing Machinery.

[36] Paolo Bethaz, Sara Cavaglion, Sofia Cricelli, Elena Liore, Emanuele Manfredi, Stefano Salio, Andrea Regalia, Fabrizio Conicella, Salvatore Greco, and Tania Cerquitelli. Empowering commercial vehicles through data-driven methodologies. *Electronics*, 10(19), 2021.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[39] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.

[42] Usman Naseem, Imran Razzak, Katarzyna Musial, and Muhammad Imran. Transformer based deep intelligent contextual embedding for twitter sentiment analysis. *Future Generation Computer Systems*, 113:58 – 69, 2020.

[43] Mohammad Ehsan Basiri, Shahla Nemati, Moloud Abdar, Erik Cambria, and U. Rajendra Acharya. Abcdm: An attention-based bidirectional cnn-rnn deep model for sentiment analysis. *Future Generation Computer Systems*, 115:279 – 294, 2021.

[44] Bruno Lepri, Jacopo Staiano, David Sangokoya, Emmanuel Letouzé, and Nuria Oliver. *The Tyranny of Data? The Bright and Dark Sides of Data-Driven Decision-Making for Social Good*, pages 3–24. Springer International Publishing, Cham, 2017.

[45] Ashley Deeks. The judicial demand for explainable artificial intelligence. *Columbia Law Review*, 119(7):1829–1850, 2019.

[46] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Hansen, and Klaus-Robert Muller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. 01 2019.

[47] A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.

[48] A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 598–617, May 2016.

[49] Eliana Pastor and Elena Baralis. Explaining black box models by means of local rules. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, pages 510–517, New York, NY, USA, 2019. ACM.

[50] Manomita Chakraborty, Saroj Kumar Biswas, and Biswajit Purkayastha. Rule extraction from neural network trained using deep belief network and back propagation. *Knowledge and Information Systems*, 62(9):3753–3781, Sep 2020.

[51] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, Oct 2019.

[52] Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[53] Francesco Ventura, Tania Cerquitelli, and Francesco Giacalone. Black-box model explained through an assessment of its interpretable features. In *New Trends in Databases and Information Systems - ADBIS 2018 Short Papers and Workshops, AI*QA, BIGPMED, CSACDB, M2U, BigDataMAPS, ISTREND, DC, Budapest, Hungary, September, 2-5, 2018, Proceedings*, pages 138–149, 2018.

[54] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery.

[55] Yunzhe Jia, James Bailey, Kotagiri Ramamohanarao, Christopher Leckie, and Xingjun Ma. Exploiting patterns to explain individual predictions. *Knowledge and Information Systems*, 62(3):927–950, Mar 2020.

[56] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

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

[58] Sherin Mary Mathews. Explainable artificial intelligence applications in nlp, biomedical, and malware classification: A literature review. In Kohei Arai, Rahul Bhatia, and Supriya Kapoor, editors, *Intelligent Computing*, pages 1269–1292, Cham, 2019. Springer International Publishing.

[59] David Alvarez-Melis and Tommi S Jaakkola. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*, 2017.

[60] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[61] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure, 2016.

[62] W. James Murdoch and Arthur Szlam. Automatic rule extraction from long short term memory networks, 2017.

[63] Maksims Ivanovs, Roberts Kadikis, and Kaspars Ozols. Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognition Letters*, 150:228–234, 2021.

[64] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[65] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions, 2016.

[66] Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. Learning credible dnns via incorporating prior knowledge and model local explanation. *Knowledge and Information Systems*, Oct 2020.

[67] Jianbo Chen and Michael Jordan. Ls-tree: Model interpretation when the data are linguistic. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3454–3461, Apr. 2020.

[68] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 06–11 Aug 2017.

[69] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. ICML'17, page 3319–3328. JMLR.org, 2017.

[70] Piyawat Lertvittayakumjorn and Francesca Toni. Human-grounded evaluations of explanation methods for text classification. *ArXiv*, abs/1908.11355, 2019.

[71] Qifeng Zhou, Xiang Liu, and Qing Wang. Interpretable duplicate question detection models based on attention mechanism. *Information Sciences*, 2020.

[72] Xiaolin Zheng, Menghan Wang, Chaochao Chen, Yan Wang, and Zhehao Cheng. Explore: Explainable item-tag co-recommendation. *Information Sciences*, 474:170 – 186, 2019.

[73] Elham Khodabandehloo, Daniele Riboni, and Abbas Alimohammadi. Healthxai: Collaborative and explainable ai for supporting early diagnosis of cognitive decline. *Future Generation Computer Systems*, 2020.

[74] Valentin Trifonov, Octavian-Eugen Ganea, Anna Potapenko, and Thomas Hofmann. Learning and evaluating sparse interpretable sentence embeddings. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 200–210, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[75] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *ArXiv*, abs/1909.00512, 2019.

[76] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings.

[77] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[78] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[79] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[80] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[81] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings, 2019.

[82] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 491–500, New York, NY, USA, 2019. Association for Computing Machinery.

[83] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[84] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[85] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[86] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[87] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[88] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 2022.

[89] Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey, 2023.

[90] Alexandra Chouldechova and Aaron Roth. A snapshot of the frontiers of fairness in machine learning. *Communications of the ACM*, 63(5):82–89, 2020.

[91] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct. 2017.

[92] Luciano Floridi, Matthias Holweg, Mariarosaria Taddeo, Javier Amaya Silva, Jakob Mökander, and Yuni Wen. capAI - a procedure for conducting conformity assessment of AI systems in line with the EU artificial intelligence act. *SSRN Electronic Journal*, 2022.

[93] European Union. General data protection regulation., 2018. Accessed: June 2023.

[94] European Union. The ai act, 2023. Accessed: June 2023.

[95] European Union Law. Proposal for a regulation laying down harmonised rules on artificial intelligence and amending certain union legislative acts., 2023. Accessed: June 2023.

[96] Cecilia Panigutti, Ronan Hamon, Isabelle Hupont, David Fernandez Llorca, Delia Fano Yela, Henrik Junklewitz, Salvatore Scalzo, Gabriele Mazzini, Ignacio Sanchez, Josep Soler Garrido, and Emilia Gomez. The role of explainable ai in the context of the ai act. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '23, page 1139–1150, New York, NY, USA, 2023. Association for Computing Machinery.

[97] United Kingdom. Equality act 2010: guidance., 2010. Accessed: June 2023.

[98] Equal Employment Opportunity Commission. Prohibited employment policies/practices., 1977. Accessed: June 2023.

[99] White House. Blue print for an ai bill of rights., 2023. Accessed: June 2023.

[100] Anna Jobin, Marcello Ienca, and Effy Vayena. The global landscape of AI ethics guidelines. *Nature Machine Intelligence*, 1(9):389–399, September 2019.

[101] Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. *arXiv preprint arXiv:1908.06083*, 2019.

[102] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '18, page 67–73, New York, NY, USA, 2018. Association for Computing Machinery.

[103] Gabriel Stanovsky, Noah A. Smith, and Luke Zettlemoyer. Evaluating gender bias in machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1679–1684, Florence, Italy, July 2019. Association for Computational Linguistics.

[104] Ismael Garrido-Muñoz , Arturo Montejo-Ráez , Fernando Martínez-Santiago , and L. Alfonso Ureña-López . A survey on bias in deep nlp. *Applied Sciences*, 11(7), 2021.

[105] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. Mitigating gender bias in natural language processing: Literature review. *arXiv preprint arXiv:1906.08976*, 2019.

[106] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *CoRR*, abs/1607.06520, 2016.

[107] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[108] Hila Gonen and Yoav Goldberg. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. *CoRR*, abs/1903.03862, 2019.

[109] Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online, July 2020. Association for Computational Linguistics.

[110] Kristin M Kostick-Quenet, I Glenn Cohen, Sara Gerke, Bernard Lo, James Antaki, Faezah Movahedi, Hasna Njah, Lauren Schoen, Jerry E Estep, and JS Blumenthal-Barby. Mitigating racial bias in machine learning. *Journal of Law, Medicine & Ethics*, 50(1):92–100, 2022.

[111] Matan Halevy, Camille Harris, Amy Bruckman, Diyi Yang, and Ayanna Howard. Mitigating racial biases in toxic language detection with an equity-based ensemble framework. In *Equity and Access in Algorithms, Mechanisms, and Optimization*, EAAMO '21, New York, NY, USA, 2021. Association for Computing Machinery.

[112] Madeleine Waller, Odinaldo Rodrigues, and Oana Cocarascu. *Recommendations for Bias Mitigation Methods: Applicability and Legality*. CEUR Workshop Proceedings, July 2023.

[113] Nikita Kozodoi, Johannes Jacob, and Stefan Lessmann. Fairness in credit scoring: Assessment, implementation and profit implications. *European Journal of Operational Research*, 297(3):1083–1094, 2022.

[114] Pinkesh Badjatiya, Manish Gupta, and Vasudeva Varma. Stereotypical bias removal for hate speech detection task using knowledge-based generalizations. In *The World Wide Web Conference*, WWW '19, page 49–59, New York, NY, USA, 2019. Association for Computing Machinery.

[115] Ji Ho Park, Jamin Shin, and Pascale Fung. Reducing gender bias in abusive language detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2799–2804, 2018.

[116] Guanhua Zhang, Bing Bai, Junqi Zhang, Kun Bai, Conghui Zhu, and Tiejun Zhao. Demographics should not be the reason of toxicity: Mitigating discrimination in text classifications with instance weighting. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4134–4145, 2020.

[117] Giuseppe Attanasio, Debora Nozza, Dirk Hovy, and Elena Baralis. Entropy-based attention regularization frees unintended bias mitigation from lists. pages 1105–1119, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[118] Indira Sen, Mattia Samory, Claudia Wagner, and Isabelle Augenstein. Counterfactually augmented data and unintended bias: The case of sexism and hate speech detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4716–4726, Seattle, United States, July 2022. Association for Computational Linguistics.

[119] Francesco Ventura, Salvatore Greco, Daniele Apiletti, and Tania Cerquitelli. Trusting deep learning natural-language models via local and global explanations. *Knowledge and Information Systems*, 2022.

[120] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[121] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany, August 2016. Association for Computational Linguistics.

[122] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. ICML'17, page 3319–3328. JMLR.org, 2017.

[123] Ilse van der Linden, Hinda Haned, and Evangelos Kanoulas. Global aggregations of local explanations for black box models, 2019.

[124] Joel Ross, Lilly Irani, M. Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers? shifting demographics in mechanical turk. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, page 2863–2872, New York, NY, USA, 2010. Association for Computing Machinery.

[125] Yuling Sun, Xiaojuan Ma, Kai Ye, and Liang He. Investigating crowdworkers' identify, perception and practices in micro-task crowdsourcing. *Proc. ACM Hum.-Comput. Interact.*, 6(GROUP), jan 2022.

[126] Kevin Crowston. Amazon mechanical turk: A research tool for organizations and information systems scholars. In Anol Bhattacherjee and Brian Fitzgerald, editors, *Shaping the Future of ICT Research. Methods and Approaches*, pages 210–221, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[127] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. Is chatgpt a general-purpose natural language processing task solver?, 2023.

[128] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd-workers for text-annotation tasks. *arXiv preprint arXiv:2303.15056*, 2023.

[129] Yongqin Xian, Bernt Schiele, and Zeynep Akata. Zero-shot learning - the good, the bad and the ugly. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[130] Badr Youbi Idrissi, Martin Arjovsky, Mohammad Pezeshki, and David Lopez-Paz. Simple data balancing achieves competitive worst-group-accuracy. In Bernhard Schölkopf, Caroline Uhler, and Kun Zhang, editors, *Proceedings of the First Conference on Causal Learning and Reasoning*, volume 177 of *Proceedings of Machine Learning Research*, pages 336–351. PMLR, 11–13 Apr 2022.

[131] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[132] Deepak Kumar, Patrick Gage Kelley, Sunny Consolvo, Joshua Mason, Elie Bursztein, Zakir Durumeric, Kurt Thomas, and Michael Bailey. Designing toxic content classification for a diversity of perspectives. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, pages 299–318, 2021.

[133] Yan Xia, Haiyi Zhu, Tun Lu, Peng Zhang, and Ning Gu. Exploring antecedents and consequences of toxicity in online discussions: A case study on reddit. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW2), oct 2020.

[134] Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. The risk of racial bias in hate speech detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence, Italy, July 2019. Association for Computational Linguistics.

[135] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. *CoRR*, abs/1703.04009, 2017.

[136] Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. Racial bias in hate speech and abusive language detection datasets. *CoRR*, abs/1905.12516, 2019.

[137] Camille Harris, Matan Halevy, Ayanna Howard, Amy Bruckman, and Diyi Yang. Exploring the role of grammar and word choice in bias toward african american english (aae) in hate speech classification. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 789–798, New York, NY, USA, 2022. Association for Computing Machinery.

[138] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable AI for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, Suzhou, China, December 2020. Association for Computational Linguistics.

[139] Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, November 2020. Association for Computational Linguistics.

[140] Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. ferret: a framework for benchmarking explainers on transformers. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, May 2023.

[141] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[142] J Richard Landis and Gary G Koch. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174, 1977.

[143] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, nov 1995.

[144] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[145] Maria De-Arteaga, Alexey Romanov, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. Bias in bios: A case study of semantic representation bias in a high-stakes setting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 120–128, New York, NY, USA, 2019. Association for Computing Machinery.

[146] Andi Peng, Besmira Nushi, Emre Kıcıman, Kori Inkpen, Siddharth Suri, and Ece Kamar. What you see is what you get? the impact of representation criteria on human bias in hiring. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 7:125–134, Oct. 2019.

[147] Brent Mittelstadt, Sandra Wachter, and Chris Russell. The unfairness of fair machine learning: Levelling down and strict egalitarianism by default, 2023.

[148] Andi Peng, Besmira Nushi, Emre Kiciman, Kori Inkpen, and Ece Kamar. Investigations of performance and bias in human-ai teamwork in hiring. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):12089–12097, Jun. 2022.

[149] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023.

[150] Sanchit Sinha, Hanjie Chen, Arshdeep Sekhon, Yangfeng Ji, and Yanjun Qi. Perturbing inputs for fragile interpretations in deep natural language processing. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 420–434, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[151] Brent Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in ai. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 279–288, New York, NY, USA, 2019. Association for Computing Machinery.

[152] EQUINET European Network of Equality Bodies. Expanding the list of protected grounds within anti-discrimination law in the eu: An equinet report., 2022. Accessed: January 2024.

[153] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Why fairness cannot be automated: Bridging the gap between eu non-discrimination law and ai. *Computer Law Security Review*, 41:105567, 2021.

[154] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Know.-Based Syst.*, 245(C), jun 2022.

[155] Shreya Shankar and Aditya G. Parameswaran. Towards observability for production machine learning pipelines. *Proc. VLDB Endow.*, 15(13):4015–4022, sep 2022.

[156] Lennart Justen, Kilian Müller, Marco Niemann, and Jörg Becker. No time like the present: Effects of language change on automated comment moderation. In *2022 IEEE 24th Conference on Business Informatics (CBI)*, volume 01, pages 40–49, 2022.

[157] Sankha Subhra Mullick, Mohan Bhambhani, Suhit Sinha, Akshat Mathur, Somya Gupta, and Jidnya Shah. Content moderation for evolving policies using binary question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 561–573, Toronto, Canada, July 2023. Association for Computational Linguistics.

[158] Fabian Hinder, Valerie Vaquet, and Barbara Hammer. One or two things we know about concept drift – a survey on monitoring evolving environments, 2023.

[159] Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda dos Santos. An overview of unsupervised drift detection methods. *WIREs Data Mining and Knowledge Discovery*, 10(6):e1381, 2020.

[160] Pei Shen, Yongjie Ming, Hongpeng Li, Jingyu Gao, and Wanpeng Zhang. Unsupervised concept drift detectors: A survey. In *Advances in Natural Computation, Fuzzy Systems and Knowledge Discovery*, pages 1117–1124, Cham, 2023. Springer International Publishing.

[161] Hanqing Hu, Mehmed Kantardzic, and Tegjyot S. Sethi. No free lunch theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining and Knowledge Discovery*, 10(2):e1327, 2020.

[162] Pingfan Wang, Hang Yu, Nanlin Jin, Duncan Davies, and Wai Lok Woo. QuadCDD: A quadruple-based approach for understanding concept drift in data streams. *Expert Systems with Applications*, 238:122114, 2024.

[163] Jan Niklas Adams, Cameron Pitsch, Tobias Brockhoff, and Wil M. P. van der Aalst. An experimental evaluation of process concept drift detection. *Proc. VLDB Endow.*, 16(8):1856–1869, apr 2023.

[164] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence – SBIA 2004*, pages 286–295, Berlin, Heidelberg, 2004.

[165] M Baena-Garcıa, J Del Campo-Ávila, R Fidalgo, A Bifet, R Gavalda, and R Morales-Bueno. Early drift detection method. *Fourth international workshop on knowledge discovery from data streams*, 6:77–86, 2006.

[166] João Gama and Gladys Castillo. Learning with local drift detection. In Xue Li, Osmar R. Zaïane, and Zhanhuai Li, editors, *Advanced Data Mining and Applications*, pages 42–55. Springer Berlin Heidelberg, 2006.

[167] Isvani Frías-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Rafael Morales-Bueno, Agustín Ortiz-Díaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2015.

[168] Anjin Liu, Guangquan Zhang, and Jie Lu. Fuzzy time windowing for gradual concept drift adaptation. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2017.

[169] Shuliang Xu and Junhong Wang. Dynamic extreme learning machine for data stream classification. *Neurocomputing*, 238:433–449, 2017.

[170] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[171] Shruti Arora, Rinkle Rani, and Nitin Saxena. SETL: a transfer learning based dynamic ensemble classifier for concept drift detection in streaming data. *Cluster Computing*, October 2023.

[172] P. Vorburger and A. Bernstein. Entropy-based concept shift detection. In *Sixth International Conference on Data Mining (ICDM'06)*, 2006.

[173] Philipp Grulich, René Saitenmacher, Jonas Traub, Sebastian Breß, Tilmann Rabl, and Volker Markl. Scalable detection of concept drifts on data streams with parallel adaptive windowing. 2018.

[174] Nick Sun, Ke Tang, Zexuan Zhu, and Xin Yao. Concept drift adaptation by exploiting historical knowledge. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[175] Mansour Zoubeirou A Mayaki and Michel Riveill. Autoregressive based drift detection method. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[176] Young In Kim and Cheong Hee Park. Concept drift detection on streaming data under limited labeling. In *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pages 273–280, 2016.

[177] Ahsanul Haque, Latifur Khan, and Michael Baron. Sand: Semi-supervised adaptive novel class detection and classification over data stream. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1652–1658. AAAI Press, 2016.

[178] Aditee Jadhav and Leena Deshpande. An efficient approach to detect concept drifts in data streams. In *2017 IEEE 7th International Advance Computing Conference (IACC)*, pages 28–32, 2017.

[179] Felipe Pinagé, Eulanda M. dos Santos, and João Gama. A drift detection method based on dynamic classifier selection. *Data Min. Knowl. Discov.*, 34(1):50–74, jan 2020.

[180] Elias Werner, Nishant Kumar, Matthias Lieber, Sunna Torge, Stefan Gumhold, and Wolfgang E. Nagel. Examining computational performance of unsupervised concept drift detection: A survey and beyond, 2023.

[181] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

[182] Denis Moreira dos Reis, Peter Flach, Stan Matwin, and Gustavo Batista. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1545–1554, New York, NY, USA, 2016. Association for Computing Machinery.

[183] Li Bu, Cesare Alippi, and Dongbin Zhao. A pdf-free change detection test based on density difference estimation. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2):324–334, 2018.

[184] Tania Cerquitelli, Stefano Proto, Francesco Ventura, Daniele Apiletti, and Elena Baralis. Towards a real-time unsupervised estimation of predictive model degradation. In *Proceedings of Real-Time Business Intelligence and Analytics*, BIRTE 2019. Association for Computing Machinery, 2019.

[185] Francesco Ventura, Stefano Proto, Daniele Apiletti, Tania Cerquitelli, Simone Panicucci, Elena Baralis, Enrico Macii, and Alberto Macii. A new unsupervised predictive-model self-assessment approach that scales. In *2019 IEEE International Congress on Big Data, Milan, Italy, July 8-13, 2019*, pages 144–148. IEEE, 2019.

[186] Sulaimon A. Bashir, Andrei Petrovski, and Daniel Doolan. Udetect: Unsupervised concept change detection for mobile activity recognition. In *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media*, MoMM '16, page 20–27, New York, NY, USA, 2016. Association for Computing Machinery.

[187] Rodrigo F. de Mello, Yule Vaz, Carlos H. Grossi, and Albert Bifet. On learning guarantees to unsupervised concept drift detection on data streams. *Expert Systems with Applications*, 117:90–102, 2019.

[188] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, page 180–191. VLDB Endowment, 2004.

[189] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In Vincent Corruble, Masayuki Takeda, and Einoshin Suzuki, editors, *Discovery Science*, pages 264–269. Springer Berlin Heidelberg, 2007.

[190] Stephan Rabanser, Stephan Günnemann, and Zachary Chase Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. In *Neural Information Processing Systems*, 2018.

[191] Edwin Lughofer, Eva Weigl, Wolfgang Heidl, Christian Eitzinger, and Thomas Radauer. Drift detection in data stream classification without fully labelled instances. In *2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–8, 2015.

[192] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. Odin: automated drift detection and recovery in video analytics. *Proc. VLDB Endow.*, 13(12):2453–2465, jul 2020.

[193] Mikhail Hushchyn and Andrey Ustyuzhanin. Generalization of change-point detection in time series data based on direct density ratio estimation. *CoRR*, abs/2001.06386, 2020.

[194] Kenji Yamanishi and Jun-ichi Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 676–681, New York, NY, USA, 2002. Association for Computing Machinery.

[195] Fabian Hinder, Valerie Vaquet, Johannes Brinkrolf, and Barbara Hammer. On the hardness and necessity of supervised concept drift detection. pages 164–175, 01 2023.

[196] Ömer Gözüaçık, Alican Büyükçakır, Hamed Bonab, and Fazli Can. Unsupervised concept drift detection with a discriminative classifier. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 2365–2368, New York, NY, USA, 2019. Association for Computing Machinery.

[197] Anjin Liu, Yiliao Song, Guangquan Zhang, and Jie Lu. Regional concept drift detection and density synchronized drift adaptation. pages 2280–2286, 08 2017.

[198] Shohei Hido, Tsuyoshi Idé, Hisashi Kashima, Harunobu Kubo, and Hirofumi Matsuzawa. Unsupervised change analysis using supervised learning. In Takashi Washio, Einoshin Suzuki, Kai Ming Ting, and Akihiro Inokuchi, editors, *Advances in Knowledge Discovery and Data Mining*, pages 148–159, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[199] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4), 2014.

[200] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[201] D.C Dowson and B.V Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982.

[202] L. N. Wasserstein. *Markov processes over denumerable products of spaces describing large systems of automata*. Probl. Inform, 1969.

[203] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[204] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[205] Tom Mitchell. Twenty Newsgroups. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C5C323.

[206] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, Ashley Scillitoe, Robert Samoilescu, and Alex Athorne. Alibi detect: Algorithms for outlier, adversarial and drift detection, 2019.

[207] *Spearman Rank Correlation Coefficient*, pages 502–505. Springer New York, New York, NY, 2008.

[208] Summaira Jabeen, Xi Li, Muhammad Shoib Amin, Omar Bourahla, Songyuan Li, and Abdul Jabbar. A review on methods and applications in multimodal deep learning. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19(2s):1–41, 2023.

[209] Gargi Joshi, Rahee Walambe, and Ketan Kotecha. A review on explainability in multimodal deep neural nets. *IEEE Access*, 9:59800–59821, 2021.

[210] Jialu Wang, Yang Liu, and Xin Wang. Assessing multilingual fairness in pre-trained multimodal representations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2681–2695, 2022.