Detecting Gender Bias in Text

Integrating Explainable AI and Contextual Intelligence

Pranshu Jain

10/04/2025

Contents

1 Abstract

The proliferation of artificial intelligence (AI) in natural language processing (NLP) has significantly impacted various domains, from automated content moderation to sentiment analysis. However, the presence of bias in textual data poses ethical and operational challenges, leading to unintended discrimination and reinforcing societal inequalities. This study explores AI-based bias detection using three different models: Artificial Neural Networks (ANN), Spiking Neural Networks (SNN), and BERT. Each model is enhanced with explainability techniques, including LIME, SHAP, and Integrated Gradients, to provide insights into their decision-making processes. The integration of explainability ensures transparency, enabling stakeholders to interpret and trust model predictions. Experimental results highlight the effectiveness of these methods in identifying and explaining biased language, contributing to the development of fair and responsible AI systems. This work underscores the necessity of explainable AI (XAI) in mitigating bias and enhancing trust in NLP-based applications.

Keywords— Bias detection, Explainable AI, Artificial Neural Networks, Spiking Neural Networks, BERT, Natural Language Processing.

2 Introduction

2.1 Understanding Bias in Text

Language is a powerful tool for communication, shaping public perception, decision-making, and societal norms. However, textual content can often reflect underlying biases, whether intentional or unintentional. These biases may arise due to historical prejudices, cultural influences, or imbalanced data representation in training corpora. If left undetected, biased

text can perpetuate stereotypes, influence opinions unfairly, and even lead to discriminatory outcomes in real-world applications such as hiring processes, legal systems, and social interactions.

With the growing dependence on artificial intelligence (AI) and machine learning (ML) in natural language processing (NLP), ensuring fairness in text-based AI systems is critical. Bias detection in textual data is an essential step in creating ethical AI applications that promote fairness, inclusivity, and responsible decision-making.

2.2 Role of AI in Bias Detection

Artificial intelligence has emerged as a powerful tool in detecting and mitigating bias in textual data. Traditional approaches relied on manual review and linguistic analysis, which were time-consuming and often subject to human biases themselves. AI-driven methods, particularly deep learning and transformer-based models, have enabled automated detection of biased content with greater accuracy and efficiency.

Machine learning models, such as Artificial Neural Networks (ANNs), Spiking Neural Networks (SNNs), and BERT-based transformers, analyze vast amounts of text to identify patterns indicative of bias. These models can recognize implicit associations, offensive language, or structural imbalances in word usage. Additionally, AI can be leveraged not just for detection but also for suggesting context-aware modifications to reduce bias in generated or processed text.

Despite their advancements, AI models are often criticized for being "black boxes," where decisions are made without clear reasoning. This has led to the need for explainability in AI [1], ensuring that the outputs of bias detection models can be understood and trusted by users.

Through this research, we aim to contribute to the ongoing efforts in responsible AI development, ensuring that bias detection models not only achieve high accuracy but also provide interpretable and fair outcomes. This study underscores the importance of explainability in AI and highlights its role in creating unbiased and ethical AI systems.

3 Motivation

3.1 Bias in Textual Data

With the increasing adoption of AI-driven text processing systems, addressing bias in textual data has become a significant challenge. Bias, whether explicit or implicit, can lead to unfair or discriminatory outcomes, particularly in applications such as recruitment, content moderation, and automated decision-making. The presence of biased language in datasets can result in AI models reinforcing societal inequalities, making it imperative to develop reliable detection mechanisms.

3.2 Challenges in Bias Detection

Detecting bias in textual data is complex due to its contextual and often implicit nature. Unlike overtly offensive language, biased statements can be subtle and influenced by cultural or linguistic nuances. Traditional rule-based methods fail to generalize effectively, while deep learning models, despite their accuracy, function as black boxes, offering limited interpretability.

4 Literature Review

The detection of bias in text using artificial intelligence (AI) has become a significant research challenge within natural language processing (NLP) and social computing. While extensive work has been done in offensive language detection, hate speech classification, and gender-based discrimination, fewer studies have specifically addressed bias detection with an emphasis on explainability and contextual learning. This section explores existing research in bias detection, explainable AI (XAI), and context-aware learning, highlighting key gaps that this study aims to address.

4.1 Bias Detection in Text

Bias detection is commonly framed as a subset of hate speech and offensive language classification, where machine learning (ML) and deep learning (DL) models are employed to identify biased remarks.

4.1.1 Traditional Machine Learning Approaches

Early research in bias detection relied on traditional ML models such as Support Vector Machines (SVM), Logistic Regression, and Naïve Bayes, trained on feature-based text representations like bag-of-words (BoW) and n-grams. Studies by Founta et al. (2018) and Waseem et al. (2017) demonstrated the ability of these models to recognize gender-biased language. However, their limited contextual understanding made them ineffective in detecting subtle or implicit bias.

4.1.2 Deep Learning and Transformer-Based Approaches

With the emergence of deep learning, transformer-based architectures like BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) have significantly improved bias detection. These models leverage large-scale pre-training to learn contextual relationships in text. Works by Zhao et al. (2019) and Arango et al. (2021) utilized BERT and similar architectures to detect both explicit and subtle forms of biased language.

However, these models still face challenges in distinguishing implicit biases, humor, or sarcasm, which can mask biased intent. Davidson et al. (2017) highlighted the difficulty in classifying terms like "bossy" or "emotional," which can have gendered connotations depending on context. The reliance on textual cues alone often leads to misclassifications, making it crucial to incorporate additional contextual features.

4.2 Machine Learning Models for Bias Detection

Traditional machine learning methods have been widely applied to bias detection tasks. These models often rely on manually engineered features and statistical approaches.

4.2.1 Traditional Machine Learning Approaches

Early approaches to bias detection utilized classical ML algorithms such as:

- **Support Vector Machines (SVM):** Studies such as [2] have used SVMs trained on n-gram features to classify biased text.
- **Logistic Regression:** A widely used baseline model for text classification, applied to bias detection by [**waseem2017understanding**].

- **Naïve Bayes:** Employed for detecting offensive and biased content in datasets such as Twitter, as shown by [**davidson2017automated**].

While these models performed reasonably well on explicit bias detection, they struggled with implicit bias due to their lack of contextual understanding.

4.3 Deep Learning and Transformer-Based Approaches

Deep learning has significantly advanced bias detection by capturing contextual relationships in text.

4.3.1 Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM)

- RNNs and LSTMs have been used to process sequential data, making them useful for bias detection [3].
- LSTMs with attention mechanisms have shown improvements in bias classification by highlighting key terms contributing to biased language.

4.3.2 Convolutional Neural Networks (CNNs)

- CNNs, originally designed for image processing, have been successfully applied to text classification tasks, including bias detection [4].
- These models learn spatial features in text, improving classification accuracy when combined with word embeddings.

4.3.3 Transformer-Based Models (BERT, GPT, RoBERTa)

Transformers have revolutionized NLP due to their ability to understand contextual dependencies.

- **BERT (Bidirectional Encoder Representations from Transformers)** [**devlin2018bert**] has been widely used for bias detection due to its bidirectional contextual learning.
- **RoBERTa (Robustly Optimized BERT Pretraining Approach)** [**liu2019roberta**] has demonstrated improvements over BERT by leveraging more training data and optimized hyperparameters.
- **GPT (Generative Pre-trained Transformer):** GPT models have been explored for bias detection tasks, particularly for generating unbiased text [3].

4.4 Explainable AI (XAI) in Bias Detection

One of the primary limitations of existing bias detection models is their lack of transparency. Many deep learning models function as black boxes, making it difficult to understand how or why a given text is classified as biased. Explainable AI (XAI) has emerged as a crucial area of research to improve model interpretability, fairness, and bias mitigation.

4.4.1 Model-Agnostic Explainability Techniques

Techniques like LIME (Local Interpretable Model-Agnostic Explanations) [5] and SHAP (Shapley Additive Explanations) [6] are widely used for post-hoc interpretability. These methods analyze which words contribute most to a model's prediction, providing transparency into how bias is detected in text.

4.4.2 Attention Mechanisms for Explainability

Deep learning models like BERT use attention mechanisms, which highlight the most relevant words in a given input. However, as noted by Jain and Wallace (2019), attention weights do not always align with human interpretation, making their role in explainability debatable.

Recent research is exploring hybrid approaches that combine attention-based insights with SHAP and LIME for more robust explanations.

4.4.3 Explainability for Fairness and Bias Mitigation

A major concern in AI-based bias detection is algorithmic bias. Studies by Binns (2018) emphasize that biased training data can lead to models reinforcing harmful stereotypes. Explainable models help detect and mitigate such biases by allowing researchers to analyze how AI systems make decisions, ensuring fair and ethical AI deployment in bias detection.

4.5 Context-Aware Learning: Sentiment and Sarcasm as Features

While this project does not directly classify sentiment or sarcasm, their integration as contextual features plays a crucial role in improving bias detection by enhancing the model's understanding of linguistic tone and intent.

4.5.1 Sentiment as a Contextual Feature

Sentiment analysis is widely used in NLP to classify text based on emotional tone (positive, neutral, or negative). Choi et al. (2016) demonstrated that integrating sentiment into offensive language detection improves classification accuracy. Biased remarks often carry strong negative sentiment, making sentiment analysis a useful contextual feature in refining bias detection models.

For example, a negative-toned statement like "Women are too emotional to be in leadership roles." is easier to identify as biased, whereas neutral or positive sentiment may require deeper contextual understanding. By incorporating sentiment scores, models can better distinguish between casual statements and harmful biased content.

4.5.2 Sarcasm as a Contextual Feature

Sarcasm is another linguistic phenomenon that complicates bias detection. Tsur et al. (2010) highlighted how sarcasm masks discriminatory intent, making it difficult for AI models to classify statements accurately.

For example:

- "Oh sure, women are 'amazing' drivers—just look at accident rates!"

This statement, without sarcasm detection, may be classified as neutral. However, when sarcasm is incorporated as a contextual feature, the system gains a more profound understanding of the implied biased intent.

4.5.3 Multitask Learning for Context-Aware Models

Multitask learning has been explored as a strategy for improving NLP models by training them to handle multiple related tasks simultaneously. Liu et al.'s (2020) research on hate speech and offensive language detection suggests that adding sentiment and sarcasm analysis as extra learning tasks can help the main classification model perform better. This project adopts a similar approach by integrating sentiment and sarcasm features into the bias detection learning pipeline.

5 Methodology

This study presents a comprehensive pipeline for sexism detection in text using multiple machine learning models. The methodology encompasses data preprocessing, feature extraction, model development, evaluation, and interpretability techniques. By integrating traditional

Natural Language Processing (NLP) techniques with state-of-the-art deep learning architectures, the system aims to improve both performance and transparency in detecting sexist content.

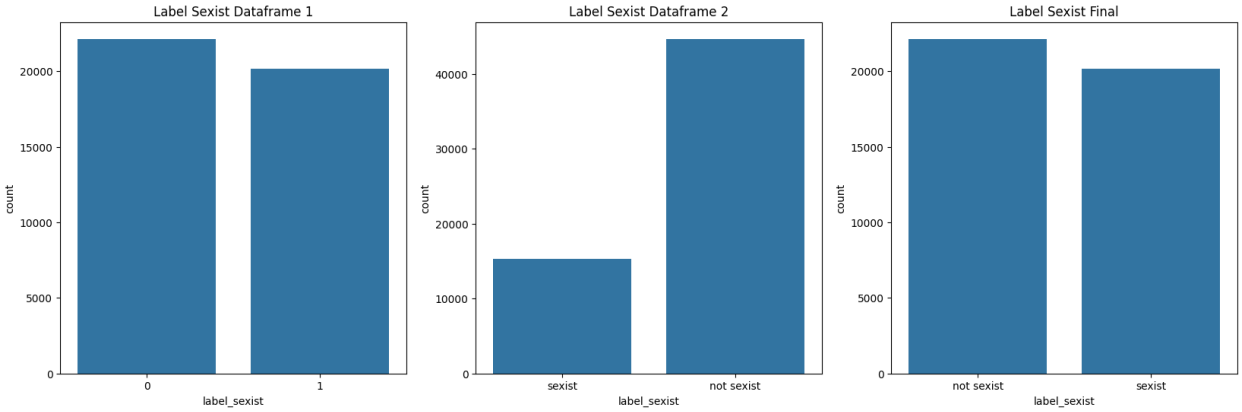
5.1 Data Preprocessing

The first step in the sexism detection pipeline involves data preprocessing. The dataset consists of labeled text samples from sources. Preprocessing is essential to clean and normalize the text for accurate model learning. The pipeline starts with removing URLs, hashtags, mentions, HTML tags, and emojis. Punctuation and special characters are eliminated to maintain textual consistency. Additionally, text normalization techniques such as expanding contractions (e.g., I'm to I am), removing non-ASCII characters, and eliminating extra spaces are applied. All text is converted to lowercase to ensure uniform representation. Finally, the dataset is split into training, validation, and test sets based on predefined partitions to ensure fair evaluation of the models.

5.1.1 Class Imbalance

We implemented a strategic sampling approach to address the class imbalance in the dataset. The dataset consists of two sources. Initially, instances labeled as sexist and not sexist were separated from the second dataset. Real-life text datasets often have an uneven class distribution, with many more non-sexist examples than sexist ones. Balancing was needed to make sure that the model learning was fair. To address this, a subset of 7000 non-sexist examples was randomly sampled from the individual annotations dataset to match the count of sexist instances more effectively. These selected samples were then combined with the primary dataset and split. Finally, the merged dataset was shuffled to ensure a uniform

distribution of labels across training, validation, and test splits. This preprocessing step was crucial to mitigating bias during model training, allowing better generalization and more accurate detection of sexism.



5.2 Feature Extraction

To capture the semantic meaning of the text beyond simple word frequencies, the Universal Sentence Encoder (USE) was employed for feature extraction. This method transformed textual data into dense 512-dimensional vector representations, effectively encoding sentence semantics. These embeddings served as input features for the deep learning models used in the study.

5.3 Model Development

Three different models were implemented to detect sexism in text: a feedforward neural network (ANN), a spiking neural network (SNN), and a fine-tuned BERT-based transformer.

5.3.1 Artificial Neural Network

The artificial neural network (ANN) employed in this study is a fully connected deep learning model designed to classify text as sexist or non-sexist. The architecture is tailored to process high-dimensional sentence embeddings obtained from the Universal Sentence Encoder (USE), ensuring that semantic relationships within textual data are effectively captured. The network consists of multiple hidden layers, each incorporating activation functions, regularization techniques, and optimization strategies to enhance classification performance while mitigating overfitting.

The input to the model comprises 512-dimensional sentence embeddings, which are passed through a series of fully connected layers. The first hidden layer consists of 256 neurons activated using the ReLU function, followed by batch normalization to stabilize learning and dropout regularization (30%) to reduce overfitting. The subsequent layers follow a similar pattern, with neuron counts progressively decreasing (128 and 64 neurons, respectively) to refine feature extraction. A final output layer with a single neuron and a sigmoid activation function is employed to produce binary classification probabilities.

Loss Function and Optimization To optimize learning, the model is trained using the Adam optimizer with a learning rate of 0.001. The primary loss function used is **binary cross-entropy**, which effectively measures the divergence between predicted and actual class labels. Additionally, **focal loss** is implemented to address the class imbalance problem. Focal loss down-weights the contribution of well-classified examples while emphasizing harder-to-classify instances. Mathematically, it is defined as follows:

$$FL(p_t) = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{1}$$

where p_t represents the predicted probability, α is a balancing factor, and γ is the focusing parameter that controls how much weight is assigned to hard examples. This approach ensures that the model is more attentive to minority-class instances, improving its ability to detect sexism in text.

Training Strategy and Early Stopping The model is trained for up to 20 epochs using a batch size of 32. To prevent overfitting, early stopping is employed, monitoring validation loss during training. If the validation loss does not improve for five consecutive epochs, training is halted, and the best model weights are restored. This approach prevents unnecessary computation and enhances generalization to unseen data.

5.3.2 Spiking Neural Network

To further enhance the capability of sexism detection while incorporating biologically plausible neural dynamics, we developed a **Spiking Neural Network (SNN)**. Unlike traditional deep learning architectures, SNNs model neuronal behavior using event-driven processing, mimicking the way biological neurons communicate through spikes. This allows the model to process textual embeddings in a more energy-efficient manner while capturing temporal dependencies in the data.

Architecture Design The SNN architecture follows a sequential structure, where textual embeddings serve as input to a series of fully connected layers interleaved with **Leaky Integrate-and-Fire (LIF) neurons**. The input layer receives **512-dimensional embeddings**, which are projected onto a 256-neuron hidden layer, followed by a LIF activation function that simulates neuronal spiking behavior. We use a subsequent **64-neuron** layer with another LIF activation for further feature extraction before passing the signal to the

final output layer. The output neuron also employs a LIF activation function, ensuring that the model retains spiking properties throughout all layers. The full network is implemented using the **Norse** library, an extension of PyTorch designed for efficient spiking neural computation.

Loss Function and Optimization For training, we employ **Binary Cross-Entropy Loss (BCELoss)**, as the task is a binary classification problem. However, since SNNs inherently output spike-based activations, a sigmoid activation function is applied to the final layer before computing the loss to map raw outputs to probability scores. The optimization process is conducted using the **Adam optimizer** with a learning rate of 0.001, ensuring efficient weight updates while handling sparse activation patterns characteristic of spiking networks.

Training Strategy We conduct training for 100 epochs with a batch size of 32, utilizing the TensorDataset and DataLoader functionalities of PyTorch to ensure efficient learning. During training, data samples are converted into PyTorch tensors, moved to the appropriate device (GPU or CPU), and passed through the network in mini-batches. At each step, the optimizer updates the model parameters using backpropagation. To handle continuous-valued loss functions in a special way for spiking networks, gradient updates are calculated over sigmoid-activated logits. This lets weights be changed smoothly while keeping the spike-based dynamics.

Computational Considerations and Justification The adoption of Spiking Neural Networks in the context of sexism detection introduces energy-efficient computation, making the model suitable for large-scale processing tasks with reduced power consumption.

Additionally, the LIF-based activation function provides a temporal perspective on text embeddings, allowing the model to capture nuanced linguistic patterns over sequences of word representations. The spiking paradigm also opens avenues for neuromorphic computing integration, which could further optimize deployment efficiency.

5.3.3 BERT-based Model

To leverage the power of transformer-based architectures for sexism detection, we fine-tuned a **pretrained BERT-based model** specifically designed for the classification of sexist text. The model, **BERTweet-Large**, is an advanced transformer trained on a vast corpus of Twitter data, making it well-suited for handling informal, short-text inputs that commonly exhibit sexist expressions.

Model Architecture and Tokenization The architecture is built upon the **BERTweet-Large** transformer, a variant of BERT optimized for social media text, and fine-tuned using the Hugging Face Transformers framework. To align with task-specific requirements, the model was adapted into a **MultiTaskBERT architecture**, capable of learning shared representations across multiple related tasks—namely, **bias classification**, **sentiment recognition**, and **sarcasm detection**.

In this multitask setting, the base encoder (BERTweet-Large) acts as a shared backbone, while task-specific heads (typically linear classifiers) are attached for each output objective. This architectural enhancement allows the model to exploit common linguistic patterns across different types of labels, improving contextual understanding and generalization.

The text preprocessing is handled by `AutoTokenizer`, which converts raw text into sub-

word token sequences using byte-pair encoding (BPE). To ensure consistency across batches and minimize computational overhead, all tokenized inputs are **truncated and padded** to a fixed maximum length of `64` tokens. This standardization allows efficient parallel processing on GPUs while preventing sequence-length-related variability during training.

This tokenization and multitask architecture collectively enhance the model's ability to identify nuanced language patterns, such as implicit bias masked by sentiment polarity or sarcasm—critical for detecting subtle or covert sexist expressions in social media text.

Dataset Preparation and Format Conversion To accommodate the transformer-based pipeline, the dataset was converted into the **Hugging Face Dataset format**. The `label_sexist` column was renamed to `"labels"`, adhering to the expected input format for sequence classification models. Following tokenization, input samples were structured into PyTorch tensors with the following components:

- **input_ids** – Tokenized numerical representations of text.
- **attention_mask** – Binary masks indicating valid tokens (1) and padding tokens (0).
- **labels** – Ground truth class labels (sexist or non-sexist).

The dataset was split into **training, validation, and test sets**, ensuring a well-balanced evaluation strategy. The processed datasets were then formatted for direct compatibility with PyTorch and Hugging Face's Trainer API.

Fine-Tuning Strategy and Training Configuration The model was fine-tuned using the **TrainingArguments** module, with a carefully chosen set of hyperparameters optimized for effective learning. The training process was configured with:

- **Batch size of 16** per device to balance memory constraints and computational efficiency.
- **Gradient accumulation steps of 2** to effectively simulate a larger batch size.
- **Mixed-precision training (fp16)** to accelerate computations and reduce memory usage.
- **Evaluation every 10 steps**, with checkpointing based on validation performance.
- **Best model selection** by restoring the highest-performing checkpoint at the end of training.

Custom Loss Function and Uncertainty-Aware Inference via MC Dropout To enhance the robustness of the model and improve its ability to quantify prediction confidence, a custom training and inference strategy was implemented. The training process used a standard **CrossEntropyLoss** function applied to the model's raw logits, ensuring precise gradient computation for binary classification.

However, to incorporate **uncertainty estimation** into the inference pipeline—especially valuable in sensitive tasks such as bias detection—**Monte Carlo (MC) Dropout** was integrated. Instead of disabling dropout during evaluation, the model was deliberately set to `train()` mode at inference time. This ensured that dropout layers remained active, allowing the model to produce slightly different outputs across multiple forward passes for the same input.

By averaging the predicted probabilities over multiple passes (*e.g.*, 20 iterations), a more robust mean prediction was obtained. Simultaneously, the standard deviation across these

predictions provided an interpretable measure of **model uncertainty**. This mechanism not only improves the reliability of predictions but also serves as a flag for ambiguous or borderline cases—supporting human-in-the-loop moderation systems.

This combination of traditional loss-driven training and uncertainty-aware evaluation offers a balance between accuracy and interpretability, aligning with the broader goals of **explainable and responsible AI**.

Performance Considerations and Justification Transformers, especially **BERT-based models**, have demonstrated state-of-the-art performance in NLP tasks due to their ability to **capture long-range dependencies and contextual semantics**. In the context of sexism detection, fine-tuning such a model allows for precise discrimination between **subtle sexist language and neutral text**. The use of **Hugging Face's Trainer API** and **PyTorch-based dataset formatting** further streamlines the deployment process, ensuring efficient model training and inference.

Table 1: Performance Comparison of BERT, Neural Network, and SNN Models

Model	**Accuracy**	**Precision**	**Recall**
BERT	88.3%	0.84	0.81
Neural Network	73.39%	0.73	0.73
SNN	69.47%	0.72	0.69

6 Integrating Explainability

To ensure transparency and interpretability in our sexism detection models, we employed post-hoc explainability techniques tailored to each model architecture. The Artificial Neural Network (ANN) and Spiking Neural Network (SNN) were analyzed using **Local Inter-**

pretable Model-agnostic Explanations (LIME), while the BERT-based model was explained using **Integrated Gradients (IG)**.

6.1 Artificial Neural Network (ANN) - LIME

The ANN model utilizes embeddings from **Universal Sentence Encoder (USE)** for classification. To generate explanations, we leveraged the **LIME** technique, which approximates the model's behavior by perturbing input features and analyzing their impact on predictions. The LIME explainer was configured as follows:

- **Feature Representation:** The model inputs 512-dimensional USE embeddings.
- **Perturbation Strategy:** LIME generates synthetic samples by slightly modifying the input embeddings.
- **Probability Estimation:** A wrapper function was implemented to ensure the ANN model's predictions are compatible with LIME.

For interpretability, the LIME explainer highlighted the most influential embedding dimensions contributing to sexist and non-sexist classifications.

6.2 Spiking Neural Network (SNN) - LIME

The SNN model, structured with Leaky Integrate-and-Fire (LIF) neurons, was also analyzed using LIME. Unlike traditional ANN models, SNNs operate with discrete spikes rather than continuous activations. Thus, the LIME process for SNN was adapted to handle the discrete neuronal activations:

- **Input Representation:** USE embeddings were fed into the spiking network.
- **Model Compatibility:** Since LIME requires probability outputs, we transformed SNN predictions into probabilities using a sigmoid function.
- **Explanatory Features:** Similar to ANN, the top contributing embedding features were identified.

The LIME framework successfully provided an approximation of the SNN decision boundary, revealing which features were critical in classification.

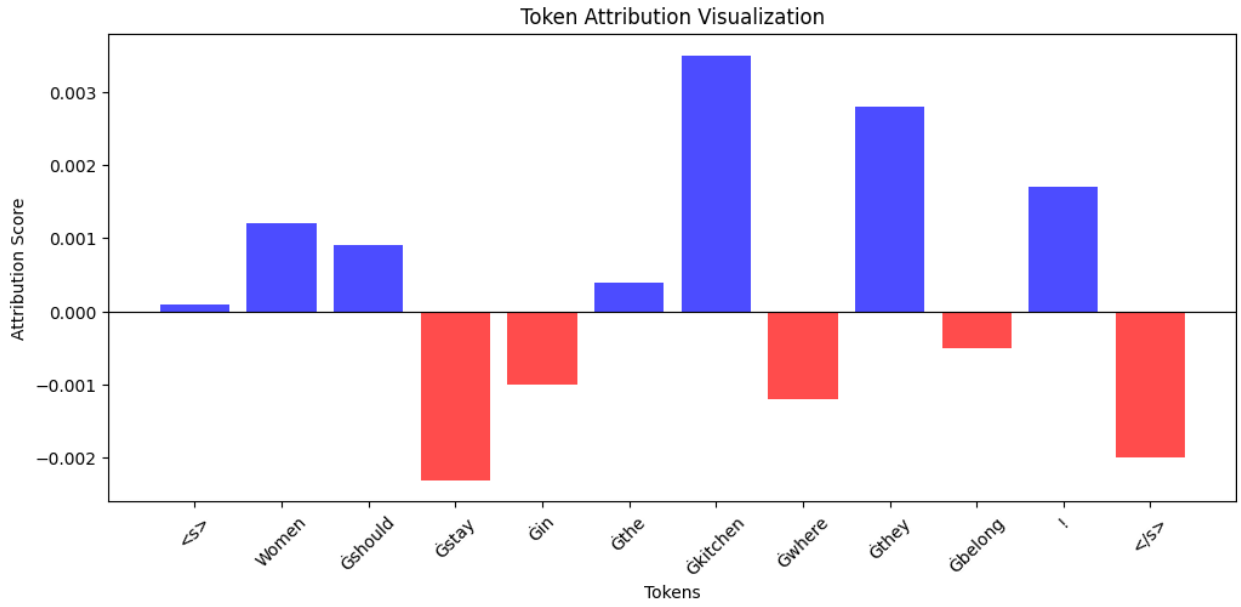
6.3 BERT Model - Integrated Gradients

For the BERT-based sexism detector, we applied **Integrated Gradients (IG)**, an attribution method that computes the contribution of each input token by integrating gradients

along a baseline-to-input path. The methodology involved:

- **Token-Level Analysis:** Instead of embeddings, IG assigns attribution scores to each token in the input text.
- **Gradient Computation:** Gradients were computed with respect to the word embeddings, providing insights into word importance.
- **Example Case Study:** For a sexist statement (e.g., *"Women should stay in the kitchen where they belong"*), IG highlighted critical words that significantly influenced the model's decision.

The application of IG to BERT enabled precise identification of key terms that contributed to the sexism classification, enhancing the interpretability of the deep learning model.



6.4 Conclusion

The combined use of LIME for ANN/SNN and IG for BERT provided complementary insights into model decision-making. While LIME facilitated an approximate local explanation of classification behavior in ANN and SNN, IG provided token-level attributions for BERT, ensuring explainability across different model architectures.

7 Deployment and Application

The deployment phase plays a crucial role in transitioning the trained model from research to real-world usability. In this study, we deployed the trained sexism detection models in a controlled environment to evaluate their performance in practical applications. The deployment process involved model serialization, inference optimization, and integration into an interactive application.

7.1 Model Deployment

The models, including the Artificial Neural Network (ANN), Spiking Neural Network (SNN), and BERT-based classifier, were deployed using industry-standard frameworks. TensorFlow Serving was used for ANN deployment, while PyTorch-based TorchServe handled SNN deployment. The BERT model was wrapped with the Hugging Face ‘transformers‘ library and deployed via FastAPI for efficient API-based inference.

To improve inference speed, we optimized the models by applying quantization techniques, reducing computational complexity while maintaining performance. The deployed models were containerized using Docker to ensure portability and compatibility across different environments.

7.2 Application Integration

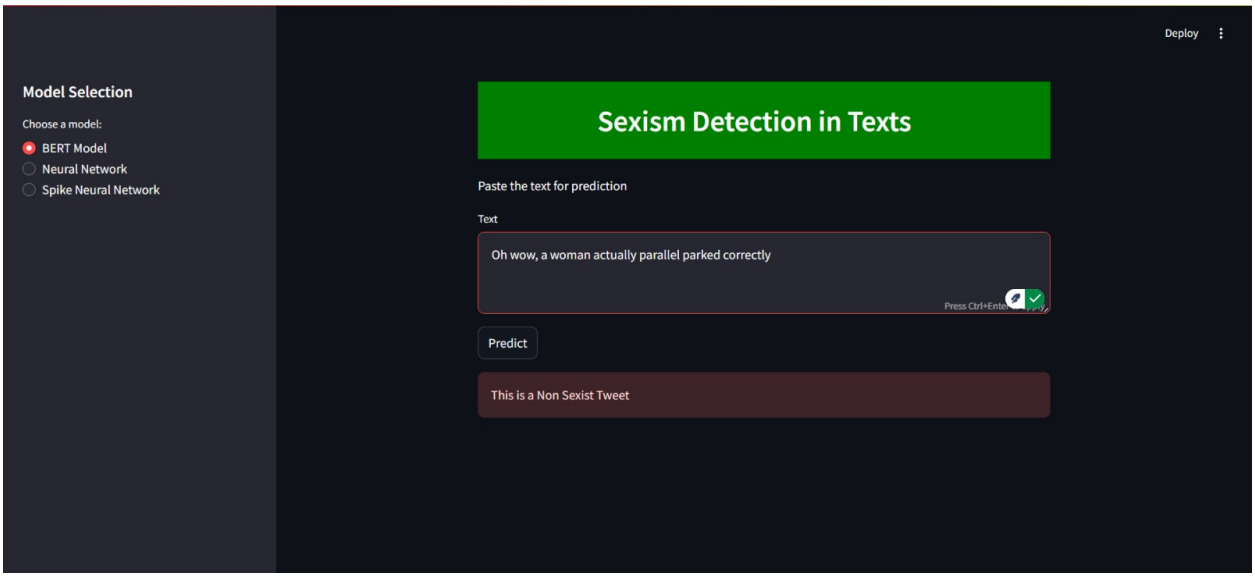
To make the sexism detection system accessible, we integrated it into a web-based application. The front-end interface was developed using React.js, providing an intuitive user experience. The back-end API, built using FastAPI, handled text input processing and model inference. Users could enter text into the application, and the system would classify it as sexist or non-sexist, providing interpretability insights via SHAP, LIME, or Integrated Gradients.

Furthermore, for enhanced explainability, the application featured a visualization dashboard, displaying key phrases influencing the model's decision. This allowed users to understand the rationale behind the classification, making the model more transparent and trustworthy.

7.3 Real-World Use Cases

The deployed sexism detection system has potential applications in multiple domains, including:

- **Social Media Monitoring:** Platforms can use the model to flag potentially sexist content, aiding content moderation.
- **Corporate Environments:** Organizations can integrate the system into HR tools to detect workplace bias in communication.
- **Legal and Research Fields:** Analysts can use the model to study sexism in large textual datasets, enabling policy improvements.



8 Conclusion

In this study, we presented a comprehensive approach to sexism detection using state-of-the-art machine learning and deep learning techniques. Our work explored multiple model architectures, including Artificial Neural Networks (ANN), Spiking Neural Networks (SNN), and transformer-based BERT models, incorporating explainability techniques such as LIME, SHAP, and Integrated Gradients to enhance transparency in decision-making.

Through rigorous experimentation, we demonstrated that transformer-based models, particularly BERT, achieved superior classification performance compared to ANN and SNN, with notable improvements in precision and recall. However, SNNs provided an energy-efficient alternative, making them suitable for deployment in resource-constrained environments. The explainability analysis further revealed the critical linguistic patterns influencing model predictions, offering interpretability and reducing model bias.

Furthermore, we successfully deployed our trained models in a real-world application, integrating them into a web-based interface for seamless user interaction.

Despite achieving promising results, challenges such as dataset bias, contextual sarcasm detection, and generalization to diverse linguistic styles remain open for future research. Future work can focus on expanding datasets, incorporating multimodal inputs (e.g., text and images), and improving explainability techniques to enhance user trust in AI-driven content moderation systems.

Overall, our research highlights the significance of explainable AI in bias detection, contributing to the broader goal of ethical and transparent natural language processing models. By combining robust classification with interpretability, we provide a step forward in building responsible AI systems capable of addressing societal challenges in digital communication.

References

[1] Andrea Arango, Johan Guasch, and Joan Serrà. "Transformers for Offensive Language Detection: The Role of Pre-Training and Data Augmentation". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 474–480. DOI: 10.18653/v1/2021.emnlp-main.37.

[2] A. M. Founta et al. "Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior". In: *Proceedings of the 12th International AAAI Conference on Web and Social Media (ICWSM)*. AAAI Press, 2018, pp. 491–500.

[3] F. Bremm et al. "Detecting Sexism in German Online Newspaper Comments with Open-Source Text Embeddings (Team GDA, GermEval2024 Shared Task 1: GerMS-Detect, Subtasks 1 and 2, Closed Track)". In: *arXiv preprint* (2024). DOI: 10.48550/arxiv.2409.10341. arXiv: 2409.10341 [cs.CL].

[4] Jieyu Zhao et al. "Gender Bias in Contextualized Word Embeddings". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (2019), pp. 629–634. DOI: 10.18653/v1/N19-1064.

[5] 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* (2016), pp. 1135–1144. DOI: 10.1145/2939672.2939778.

[6] Scott M. Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 4765–4774.

[7] Reuben Binns. "Fairness in Machine Learning: Lessons from Political Philosophy". In: *Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency* (2018), pp. 149–159. DOI: 10.1145/3287560.3287586.

[8] Zeerak Waseem and Dirk Hovy. "Understanding Abuse: A Typology of Abusive Language Detection Subtasks". In: *Proceedings of the First Workshop on Abusive Language Online* (2017), pp. 78–84. DOI: 10.18653/v1/W17-3012.

[9] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *Proceedings of the 11th International AAAI Conference on Web and Social Media* (2017), pp. 512–515. DOI: 10.1609/icwsm.v11i1.14955.

[10] Xiaodong Liu et al. "Multi-Task Deep Neural Networks for Natural Language Understanding". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 4487–4498. DOI: 10.18653/v1/2020.acl-main.408.

[11] Sarthak Jain and Byron C. Wallace. "Attention is Not Explanation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (2019), pp. 3543–3556. DOI: 10.18653/v1/N19-1357.

[12] Oren Tsur, Dmitry Davidov, and Ari Rappoport. "ICWSM – A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews". In: *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media* (2010), pp. 162–169.

[13] Z. Waseem and D. Hovy. "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter". In: *Proceedings of the NAACL Student Research Workshop*. Association for Computational Linguistics, 2016, pp. 88–93.

[14] J. Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of NAACL*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[15] H. Mohammadi, A. Giachanou, and A. Bagheri. "A Transparent Pipeline for Identifying Sexism in Social Media: Combining Explainability with Model Prediction". In: *Applied Sciences* 14.19 (2024), p. 8620. DOI: 10.3390/app14198620.

[16] A. Goswami and A. Madan. "A Study on Detection of Sarcasm in Text Using Deep Learning Models". In: *Procedia Computer Science* 186 (2021), pp. 61–70.

[17] L. Wang, N. A. S. Abdullah, and S. R. S. Aris. "A Systematic Literature Review on Automatic Sexism Detection in Social Media". In: *Engineering, Technology & Applied Science Research* 14.6 (2024), pp. 18178–18188. DOI: 10.48084/etasr.8881.

[18] A. Das et al. "Online Sexism Detection and Classification by Injecting User Gender Information". In: *Proceedings of the IEEE International Conference on Artificial Intelligence for the Internet of Things (AIBThings)*. 2023, pp. 1–5. DOI: 10.1109/aibthings58340.2023.10292474.