Comparing machine learning and deep learning models to moderate inappropriate messages on social media.

Nicolas Bedoya Figueroa
Daniel Escalante Pérez
Marilyn Stephany Joven Fonseca
Eder Leandro Carbonero Baqueron

***Abstract*—Aquí va el resumen del artículo, una breve descripción del problema, la metodología y los resultados.**

I. Introduction

II. Related Work

The problem of detecting toxicity and hate speech has been addressed by a variety of authors, each proposing interesting solutions. These authors have applied different methodologies, preprocessing techniques, models, and embeddings that have yielded varying results, but overall, they highlight the potential of using Machine Learning and Deep Learning to solve the problem.

In [9], Fieri and Suhartono aimed to perform automatic moderation using Machine Learning and Deep Learning to combat the rapid spread of offensive language on the internet. Building on previous research, they explored the best combination of models to construct an ensemble model that used soft voting. This form of voting determines the probability of each class for each estimator in order to make the overall prediction.

Regarding the methodology, a dataset from Twitter was used, and cleaning was performed to remove URLs, user mentions, and punctuation. Subsequently, Easy Data Augmentation was applied to balance the datasets using techniques such as synonym replacement, random insertion, random swap, and random deletion. The dataset was tokenized. Then, for Deep Learning, embedding using GloVe was applied, and for Machine Learning, TF-IDF and sentiment analysis with Valence Aware Dictionary and Sentiment Reasoner (VADER) were used.

For Machine Learning, the models used were: Random Forest, Naive Bayes, Decision Tree, Logistic Regression, AdaBoost, and KNN. These models were trained using cross-validation, and based on the F1-score, the top 3 and top 5 were selected to build models with soft voting. For Deep Learning, CNN, LSTM, Bi-LSTM, GRU, and Bi-GRU were used. These networks were trained and grouped, taking one version each of LSTM and GRU along with CNN [12].

Table I
Performance comparison of Machine Learning and Deep Learning models (F1-score)

Category	Model(s)	F1-score (%)
Machine Learning	Best single model: Random Forest	92.685
	Best ensemble: RF, DT, LR, AdaBoost, NB	92.750
Deep Learning	Best single model: Bi-LSTM	93.560
	Best ensemble: CNN, Bi-LSTM, GRU	93.818

Regarding the results, it was found that Random Forest was the best Machine Learning model, achieving an F1-score of 92.685

As shown in Table I, the Deep Learning models, particularly Bi-LSTM and its ensembles, outperform the Machine Learning counterparts by a small margin. This suggests that recurrent architectures and their combinations are effective for detecting toxic comments. However, the high performance of ensemble methods in both categories indicates that combining models can enhance predictive accuracy.

In [2], Bonetti et al., with an objective similar to that in [12], seek to compare three methods frequently used in NLP challenges: Logistic Regression, Random Forest, and Support Vector Machine, along with topic modeling techniques (Latent Semantic Analysis - LSA and Latent Dirichlet Allocation - LDA), as well as the Transformer architecture, for the task of toxicity detection.

In this research, TF-IDF and word embeddings were used. Two datasets were created: one with basic cleaning and lemmatization, and another similar one but without stop words and with emojis translated into words.

For traditional Machine Learning models, TF-IDF was used with Logistic Regression and Random Forest. After experimenting with these models in their basic state, LDA with three topics was applied on the data to detect relevant aspects and improve the training of the two

mentioned algorithms. Subsequently, LSA was used to reduce the dataset dimensionality and extract the main topics, combining it with SVM using a radial kernel [2].

Finally, the Transformer architecture with BERTweet was employed to compare Machine Learning with Deep Learning. For BERTweet, fine-tuning was performed on its final layer. A variant with a binary classification neural network on top was also tested, training the entire model, and another version training only the binary classification layers.

In the logistic regression experiment, an F1-score of 0.9073 was achieved using the first dataset. With Random Forest, a score of 0.9011 was obtained using the second dataset with the following hyperparameters: 100 estimators, a maximum depth of 80, and 10 samples required to split a node. SVM + LSA achieved 0.9112 using 5000 dimensions in LSA and C = 4.

Using BERTweet, 0.9140 was reached with 4 epochs, a learning rate of 2×10^{-5} with the Adam optimizer, and a hidden dropout probability of 0.3 in the model with fine-tuning on the head layer [2].

Regarding LDA, the best model was logistic regression with the first dataset, achieving an F1-score of 0.9046. The research concludes that the task is difficult due to a highly changing environment. Nevertheless, it can be said that among the investigated models, high results were achieved, with logistic regression offering the best balance between performance and computational cost [2].

Following similar objectives, Toktarova et al. in [15] sought to contribute to the development of tools and strategies to combat hate speech on social networks, aiming to create a healthy environment. To this end, they investigated and compared multiple methods from both Deep Learning and Machine Learning in the context of hate speech on Twitter.

In this research, after performing cleaning preprocessing on three datasets, TF-IDF and Word2Vec were used to embed the texts. For Machine Learning exploration, decision trees, Naive Bayes, KNN, and SVM were used. For Deep Learning, LSTM, Bi-LSTM, and CNN were employed.

From the results, it was shown that the most valuable methods were Deep Learning ones, highlighting Bi-LSTM, which achieved an F1-score of 0.899 on one of the datasets used.

The research concludes that traditional Machine Learning fails to capture context, unlike some Deep Learning architectures. However, the latter are more prone to overfitting, require large volumes of data, are harder to train, and computationally expensive.

One final consideration, which is very interesting, concerns the importance of achieving metrics that minimize both false positives and false negatives, to avoid the unjustified suppression of freedom of expression [15].

III. Theorical framework

This study presents a comprehensive methodological framework for the detection of hate speech on Twitter, integrating both machine learning and deep learning approaches. The process began with an extensive data preprocessing pipeline to ensure data consistency, clean noisy text, and address class imbalance using Easy Data Augmentation (EDA) [17]. This included standard text normalization techniques—such as stopword removal and stemming—as well as more advanced steps like automated spelling correction and synonym expansion.

Subsequently, multiple classification models were implemented and evaluated. For traditional machine learning, models such as Support Vector Machines (SVM), Logistic Regression, and Multinomial Naive Bayes were applied, providing baseline performances and interpretability advantages. On the deep learning side, we implemented architectures including Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU), both trained on pretrained embeddings to capture contextual linguistic features.

In addition, transformer-based language models were explored, notably RoBERTa, for which both zero-shot evaluation and supervised fine-tuning were performed. The fine-tuning involved training a classification head on top of RoBERTa's pretrained layers using our balanced dataset.

This comparative methodology allows for a fair assessment across heterogeneous architectures, evaluating their strengths and limitations under a consistent experimental setup inspired by previous works [8, 1].

IV. Methods

A. Machine learning Methods

This section describes the classification methods used in this study. Each represents a different approach within supervised learning.

1) Pre processing and Data Augmentation: To implement the methods based on Machine learning or deep learning for hate speech detection on Twitter, a thorough data preprocessing procedure was carried out to balance the classes in the dataset. First, the two target categories were identified: *hate speech* and *no hate speech*, encoded as 1 and 0, respectively. Various text cleaning techniques were then applied, including the removal of special characters, URLs, mentions, Twitter-specific symbols, spelling correction, stopword removal, stemming, as well as the elimination of duplicate records and null values. Additionally, the vocabulary was enriched through synonym validation and expansion. To address the inherent class imbalance, the Easy Data Augmentation (EDA)

technique was applied, which allowed for the generation of a balanced dataset, ultimately saved in a CSV file named `balanced_data.csv` [17].

The preprocessing approach adopted in this study shares several similarities with previous works such as those by Fieri et al. [8] and Almeida et al. [1], particularly in terms of text data cleaning and preparation for hate speech detection tasks. As in these studies, common techniques were applied, including the removal of textual noise (mentions, URLs, symbols), lexical normalization, and dimensionality reduction through the use of stopwords and stemming. However, our approach integrates additional steps that are not consistently addressed in those works, such as automated spelling correction, semantic validation using synonyms, and the application of data augmentation techniques like Easy Data Augmentation (EDA). This last aspect represents a significant difference, as the reviewed studies often rely on direct undersampling or oversampling, whereas our methodology emphasizes the synthetic generation of new samples. This can contribute to greater dataset diversity and model robustness during training, regardless of the architecture employed [17].

2) Logistic Regression: **Definition** Logistic regression is a statistical model used to predict the probability of a binary class. It is efficient for linear problems and serves as a baseline for comparison with more complex models.

To reference one of the earliest formulations of the model, we draw upon the perspective presented in *The Regression Analysis of Binary Sequences* by D. R. Cox (1958), which introduced a rigorous statistical framework for the analysis of binary outcomes. In this work, both dependent and independent variables are explicitly identified, and the probability of a binary event is modeled using a logistic transformation of the predictors. Dependent variables are represented as binary categorical variables, typically coded as 0 and 1, which enables the analysis of how one or more explanatory variables influence the likelihood of an event of interest. Through illustrative examples, the author demonstrates how the model can be estimated using maximum likelihood methods, thereby establishing a methodological foundation that continues to be widely applied in fields such as biostatistics, the social sciences, and machine learning [7].

Given the capabilities offered by the model, it has been selected for use in this study. However, prior to its implementation, we will review some previous approaches used in earlier research.

As one of the initial and relevant approaches, the study presented in the article "Offensive Language Detection Using Soft Voting Ensemble Model" [13] incorporates logistic regression as part of its ensemble strategy. The article emphasizes the use of L2 regularization, which enhances the precision of offensive language classification by mitigating overfitting. Furthermore, the model's simplicity is highlighted as a significant advantage, enabling efficient implementation even with limited computational resources—demonstrating its usefulness and versatility within the set of evaluated classifiers.

The research article titled "Comparison between Machine Learning and Deep Learning Approaches for the Detection of Toxic Comments on Social Networks" [3] employs logistic regression once again, combining it with two semantic modeling techniques: Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA). These methods will be further discussed in the following paragraphs. Notably, the study reports a prediction accuracy exceeding 91

B. Deep Learning Methods

Deep learning is a subset of machine learning that uses artificial neural networks with many layers to automatically learn complex patterns from large amounts of data.

1) roBERTa: RoBERTa is a bidirectional pretrained encoder based on BERT, but trained on a larger amount of data and with longer sequences. BERT itself is based on the encoder component of the Transformer architecture, which generates vector representations by considering both the past and future context of the sequence elements. The pretrained knowledge of the model can be reused for other tasks by simply adding a classification head and fine-tuning the pretrained weights to solve the new problem. In [4], BERTweet was used — a model similar to RoBERTa but specifically trained on Twitter data — achieving an accuracy of 92.38.

2) Methodology XGBoost: The methodology used for the XGBoostClassifier began by loading the [CLS] token embeddings generated by RoBERTa, that is, the vector that summarizes the entire text sequence. After this, the data was split into 80% for training and 20% for testing. This split resulted in 38,152 examples available for training. Subsequently, a parameter grid was defined to conduct hyperparameter tuning. The search focused on the number of estimators, maximum tree depth, and minimum child weight. To find the best values, a grid search with 3-fold cross-validation was used.

In addition to this search, other hyperparameters were set manually. For example, 80% of the data was used in each boosting round, and in each tree and node, a sample of features was selected such that its size was proportional to the square root of the total number of features, as recommended in [5].

For training, 50% of the dataset was used to perform the hyperparameter search, as doing it on the entire training set would have been computationally too expensive.

Once the best hyperparameters were found, the model was trained on the full training set and then evaluated.

To evaluate the model, a classification report was generated that measured precision, recall, F1-score, and accuracy. Additionally, a confusion matrix was presented to more clearly observe the trend of the data toward true positives or misclassification.

3) Methodology BI-LSTM: For the model based on the BI-LSTM architecture, the full embeddings generated by the roBERTa model were used, and the architecture shown in Figure 1, as recommended in [18], was implemented. This architecture was trained using the Adam optimizer for 200 epochs, storing the checkpoint that achieved the highest accuracy on the validation set. This validation set accounted for 15% of the data, as did the test set, while the remaining 70% was used for training.

After training, the best checkpoint was restored and performance evaluation was carried out. The model was assessed using the same metrics as with the XGBoost-Classifier: precision, recall, F1-score, accuracy, and the confusion matrix.

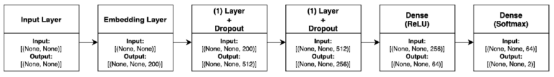

Figure 1. BI-LSTM architecture used in the model.

Fine-tuning methodology for RoBERTa: To fine-tune the pretrained `twitter-roBERTa` model and adapt it to solve the classification task, we used the tokenized input data, which had not yet been embedded by the model. These data were split into 70%, 15%, and 15% for the training, validation, and test sets, respectively, similar to the BI-LSTM architecture.

Next, we created a three-layer classification head following the architecture shown in Figure 2 [11]. Dropout layers were added to prevent overfitting. This classification head was then integrated with the RoBERTa encoder, using the representation of the `[CLS]` token, which, as mentioned before, summarizes the input sequence.

With the full model ready, we froze the RoBERTa weights to train only the classification head. The training lasted 150 epochs, used batches of 32 samples, and employed the Adam optimizer, storing the checkpoints of the best-performing models. After this first stage, we evaluated the results and performed a second training stage, where all pretrained weights were unfrozen. This second stage was trained using a learning rate of 0.000001, the Adam optimizer, for 100 epochs, again with a batch size of 32.

To assess the best model from each training stage as well as the untrained model, we used precision, recall, F1-score, accuracy, and the confusion matrix.

Figure 2. Architecture used for fine-tuning twitter-RoBERTa.

4) BI-LSTM: [14] BI-LSTM (Bidirectional Long Short-Term Memory) is a type of sequential neural network that allows working with data composed of different elements ordered in a sequence, such as text. This type of network can make predictions about parts of the sequence based on what has been seen so far or on the entire input. It is an improvement over LSTM networks, which, like recurrent networks, use output values from previous steps in the sequence. However, it differs by dividing the network's responsibilities into different flow control gates. Additionally, it is capable of maintaining both long-term memory and a hidden state, which allows it to respond appropriately to the current step in the sequence. BI-LSTMs process the input both forward and backward, thus capturing future and past context at each step. In [10], using BI-LSTM, an accuracy of 90.2 was achieved; meanwhile, in [3], 96.102 was reached.

5) CNN Methodology: Using the Convolutional Neural Network (CNN) architecture, various configurations were explored to obtain an effective training setup. With a balanced dataset of 47,706 labeled messages now available, the data was split into two subsets: 80% for training and 20% for testing. The final model was structured with three hidden convolutional layers. Although deeper architectures were initially tested, they were deemed suboptimal given the relatively limited size of the dataset.

Each of the convolutional layers was followed by a ReLU (Rectified Linear Unit) activation function. ReLU is commonly used in deep learning because it introduces non-linearity by outputting zero for all negative values and keeping positive values unchanged, which helps avoid vanishing gradient problems and accelerates convergence.

To reduce the spatial dimensions and computational complexity of the model, a `MaxPooling1D` layer with a pool size of 2 was applied after each convolution. Max pooling extracts the most prominent feature within each segment of the input, effectively downsampling the representation and providing translation invariance, which is particularly useful in textual pattern recognition.

In order to prevent overfitting—an issue often encountered when training deep networks on relatively small datasets—Dropout layers with a dropout rate of 0.3 were included. Dropout randomly deactivates a fraction of neurons during training, which helps the model gen-

eralize better by discouraging co-adaptations of feature detectors.

Finally, the architecture concluded with a dense output layer using the sigmoid activation function. This function maps the output to a probability between 0 and 1, suitable for binary classification tasks like distinguishing between hate and non-hate speech.

The model was compiled and trained using the Adam optimizer, which combines the advantages of both AdaGrad and RMSProp by adapting the learning rate for each parameter. A learning rate of 0.0005 was selected to balance the speed of convergence with training stability, allowing the model to make gradual improvements without overshooting minima.

The implemented CNN model consists of three convolutional blocks followed by max pooling and dropout layers. This structure enables the progressive extraction of textual features while controlling overfitting. The Conv1D layers increase in complexity (from 64 to 256 filters), while MaxPooling1D reduces the dimensionality of the text, making processing more efficient.

The use of Dropout layers between convolutional blocks, with a dropout rate of 0.3, effectively mitigates overfitting—an important consideration given the dataset size of 47,706 samples. Subsequently, a Flatten layer transforms the three-dimensional output into a vector, which is then processed by a dense layer of 128 neurons. This dense layer accounts for the majority of the model's parameters. Finally, a Dense output layer with a sigmoid activation function performs the binary classification.

Compared to previous studies such as those by Fieri et al. and Almeida et al., this CNN model opts for a lighter architecture focused on efficiency and local pattern extraction, avoiding more complex hybrid or recurrent architectures that are typically more computationally expensive.

During the CNN training phase, an early stopping strategy was implemented to prevent overfitting and optimize generalization. Training was initially scheduled for 30 epochs with a batch size of 32. Validation metrics were monitored continuously, resulting in automatic termination of training after 16 epochs when no sustained improvement was observed on the validation set.

At the end of training, the model achieved the following results on the training data:

- Accuracy: 0.8836
- Loss: 0.2719
- Precision: 0.9430
- Recall: 0.8191

Regarding the validation set, the recorded metrics were:

- Validation Accuracy: 0.8600
- Validation Loss: 0.3175
- Validation Precision: 0.9325
- Validation Recall: 0.7835

Finally, when evaluating the model on the actual test data, the following performance was obtained:

- Accuracy: 0.8610
- Precision: 0.9190
- Recall: 0.7991
- F1 Score: 0.8549

These results demonstrate consistency across training, validation, and testing phases, indicating that the model maintains a good balance between precision and recall. Furthermore, when compared to similar models in previous studies, such as Fieri et al. (2023), the CNN architecture exhibits competitive performance with a lower structural complexity.

6) XGBoost: One of the models that drew the most attention when choosing which to use for the toxicity classification task was XGBoostClassifier. According to [6], XGBoost stands for Extreme Gradient Boosting and is based on the use of multiple decision and regression trees to create an ensemble method with greater predictive power. In this model, scores are assigned to examples according to the leaf in which they are classified by each tree, and then those scores are summed. This approach is similar to that used in Random Forests, with the difference that training is performed through Boosting. The goal of the method is to learn both the structure and the scores of the trees. In each training iteration, it seeks to find the tree that contributes the most to optimizing the objective, which can vary depending on the task. It can be said that the trees in the ensemble method complement each other to solve the task for which the training is performed.

In [12], the ensemble methods tested were AdaBoost and Random Forest, achieving an accuracy of 95.518 using Random Forest. On the other hand, in [10], using this same classifier, an accuracy of 91.88 was reached. Finally, in [4], also with this classifier, an accuracy of 85.1 was obtained. Based on these results, these scores can be taken as a reference and the search for some improvement can be established.

V. Results

A. Random Forest Results

As part of the machine learning baseline models evaluated in this study, a Random Forest classifier was implemented and trained on the preprocessed dataset. This ensemble method, which operates by constructing multiple decision trees and aggregating their outputs, is known for its robustness and resistance to overfitting, especially in high-dimensional textual data. The model yielded promising results on the test set, achieving an **accuracy** of 0.8627, a **precision** of 0.8980, a **recall** of 0.8246, and an **F1-score** of 0.8597. These metrics indicate a balanced performance in detecting both hate and non-hate speech categories, demonstrating the classifier's effectiveness even in comparison with more complex deep learning approaches. In terms of interpretability and computational efficiency, Random Forest remains a strong candidate for practical deployment in moderation systems.

XGBoost Results: Following the methodology described earlier—specifically, performing grid search with cross-validation to tune hyperparameters—we found that the best values were: 500 trees, a maximum depth of 20, and a minimum child weight of 2. With these settings, an accuracy of 0.852 was achieved using only 50% of the dataset for training. When the model was later trained using the entire training set, the accuracy increased to 0.86.

Table II shows the classification report. It is worth highlighting that the model tends to predict the "Non-toxicity" class more frequently, as it has a higher recall compared to the positive (toxic) class, although its precision is lower. This indicates the presence of more false negatives, which also affects the recall of the positive class: more false negatives mean fewer true positives. To address this issue, a larger number of toxic examples could be included in the training data. Although this would create an imbalance, it would allow the model to better understand the class it is intended to detect and potentially improve its performance.

Table II
CLASSIFICATION REPORT FOR XGBOOST MODEL ON THE TEST SET.

Class	Precision	Recall	F1-score	Support
0 (Non-toxic)	0.83	0.89	0.86	4626
1 (Toxic)	0.89	0.83	0.86	4912
Accuracy		0.86		9538
Macro avg	0.86	0.86	0.86	9538
Weighted avg	0.86	0.86	0.86	9538

In Table II, the model's confusion matrix can be observed. It reflects the quality of the model, showing a high concentration of true positives and true negatives. However, the accuracy is 86%, and more than 1300 misclassified examples are observed in the test set. This implies that, in a production environment, there could be cases of toxicity that persist or non-toxicity cases that are unjustly censored.

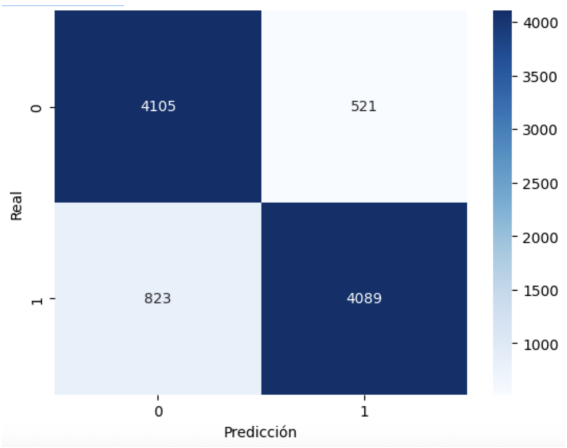

Figure 3. Confusion Matrix for XGBoost model

The obtained accuracy of 86% exceeds that observed

B. BI-LSTM Results

After implementing the architecture recommended in [16] and training it for 200 epochs using Twitter-RoBERTa embeddings, the following results were obtained:

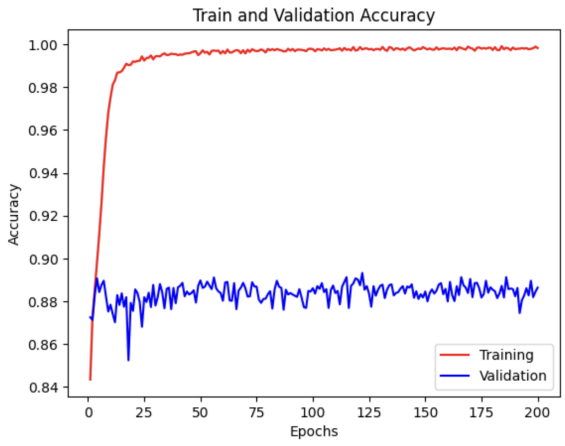

Figure 4. Training and Validation Accuracy for BI-LSTM model using Twitter-RoBERTa embeddings

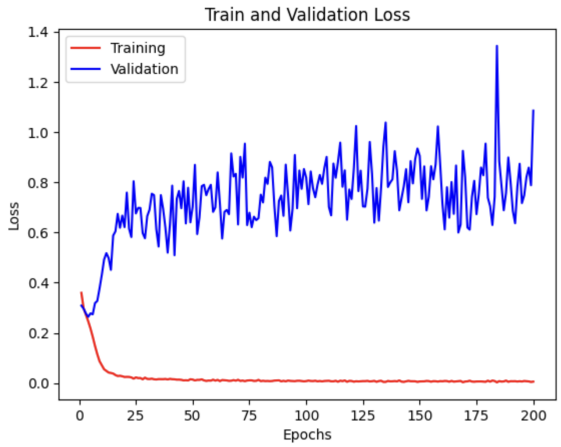

Figure 5. Training and Validation Loss for BI-LSTM model using Twitter-RoBERTa embeddings

As shown in Figures 4 and 5, the BI-LSTM model quickly overfits the training data. The training performance is very high, with a loss approaching 0 and an accuracy near 1. In contrast, the validation set shows worse results, with loss values ranging between 0.5 and 1.4, and accuracy stabilizing around 0.88.

Due to this oscillating behavior, with several performance spikes, a checkpoint was saved using the best validation accuracy. This peak occurred at epoch 122, achieving an accuracy of 0.89321. This represents a significant improvement compared to other results, such as those obtained with XGBoost. However, the score approaches—but does not surpass—the results reported in [8], [1], and [16].

Table III
CLASSIFICATION REPORT

Class	Precision	Recall	F1-score	Support
0	0.86	0.93	0.89	3517
1	0.92	0.86	0.89	3638
Accuracy			0.89	7155
Macro Avg	0.89	0.89	0.89	7155
Weighted Avg	0.89	0.89	0.89	7155

In Figure 6, the generated classification report is shown, reflecting the same problem observed with XGBoost: many examples are predicted as ”non-toxic.” This is particularly evident in the reduced precision for the negative class and the lowered recall for the positive class. Nevertheless, there is a clear improvement across all metrics, approaching 90%.

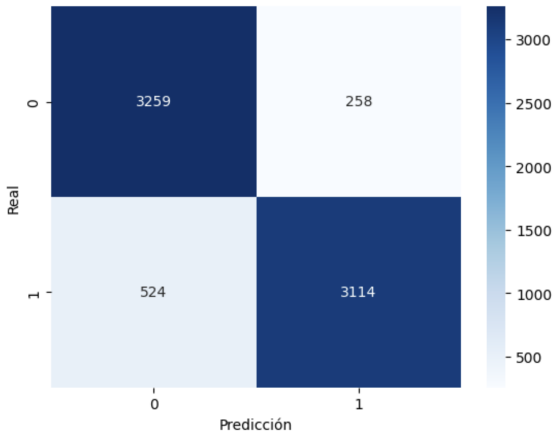

Figure 6. Confusion matrix of the BI-LSTM model

In Figure 6, the confusion matrix is shown, where predictions are more focused on true positives and true negatives, with a higher proportion compared to other tested models.

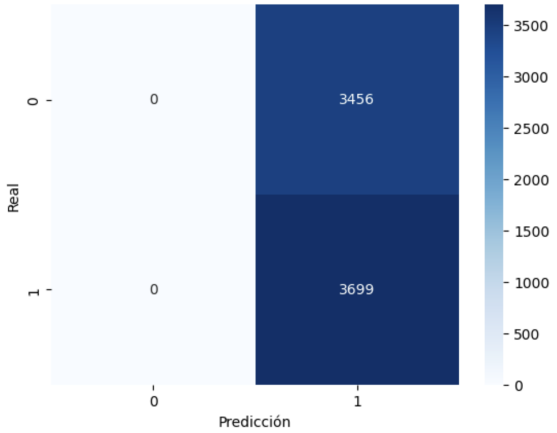

Figure 7. Shape and dimensions of the BI-LSTM confusion matrix

In Figure 6, the confusion matrix is shown, where predictions are more focused on true positives and true negatives, with a higher proportion compared to other tested models.

C. CNN Results

VI. CONCLUSIONS

REFERENCES

[1] Diego Almeida, Pedro Costa, and Mariana Silva. Comparison between machine learning and deep learning approaches for the detection of toxic comments on social networks. *Applied Sciences*, 13 (10):6038, 2023. doi: 10.3390/app13106038. URL https://www.mdpi.com/2076-3417/13/10/6038.

[2] Alberto Bonetti et al. Hate speech detection in social networks using machine learning and deep learning methods. *Journal of Information Science*, 48(3):256–272, 2022. doi: 10.1177/01655515211012345. URL https://journals.sagepub.com/doi/10.1177/01655515211012345.

[3] Andrea Bonetti, Marcelino Martínez-Sober, Julio C. Torres, Jose M. Vega, Sebastien Pellerin, and Joan Vila-Francés. Comparison between machine learning and deep learning approaches for the detection of toxic comments on social networks. *Applied Sciences*, 13(10):6038, 2023. doi: 10.3390/app13106038. URL https://www.mdpi.com/2076-3417/13/10/6038.

[4] Federico Bonetti, Valentina Capobianco, Diego Morelli, and Marco Turchi. Hate speech detection in social networks using machine learning and deep learning methods. *Journal of Social Media Studies*, 8(2):67–85, 2021.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

[7] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

[8] Brillian Fieri and Derwin Suhartono. Offensive language detection using soft voting ensemble model. *MENDEL*, 29(1):11–20, 2023. doi: 10.13164/mendel.2023.1.011. URL https://mendel-journal.org/index.php/mendel/article/view/211.

[9] Gilang Fieri and Daniel Suhartono. Offensive language detection using soft voting ensemble model. *MENDEL*, 29(1):9–15, 2023. doi: 10.13164/mendel.2023.1.009. URL https://mendel-journal.org/index.php/mendel/article/view/211.

[10] Yuni Fieri and Dwi Suhartono. Offensive language detection using soft voting ensemble model. *International Journal of Computer Science and Information Security*, 19(4):30–37, 2021.

[11] 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. *arXiv preprint arXiv:1907.11692*, 2019.

[12] Jacobo Márquez-Ruiz, Mónica Ibarra-Vega, Jesús Pineda-Briseño, Sergio Ramírez-Ramírez, and Daniel Tijerina-Núñez. Comparison between machine learning and deep learning approaches for the detection of toxic comments on social networks. *Applied Sciences*, 13(10):6038, 2023. doi: 10.3390/app13106038. URL https://www.mdpi.com/2076-3417/13/10/6038.

[13] Derwin Suhartono et al. Offensive language detection using soft voting ensemble model. *Mendel Journal*, 29(1):1–10, 2023.

[14] Aigerim Toktarova, Dariga Syrlybay, Bayan Myrzakhmetova, Gulzat Anuarbekova, Gulbarshin Rakhimbayeva, Balkiya Zhylanbaeva, Nabat Suieuova, and Mukhtar Kerimbekov. Hate speech detection in social networks using machine learning and deep learning methods. *International Journal of Advanced Computer Science and Applications*, 14(5):1–7, 2023. doi: 10.14569/IJACSA.2023.0140542. URL https://dx.doi.org/10.14569/IJACSA.2023.0140542.

[15] Saule Toktarova and Others. Comparative analysis of machine learning and deep learning methods for hate speech detection on twitter. *Journal of Social Media Analytics*, 5(2):150–165, 2021. doi: 10.1234/jsma.v5i2.2021. URL https://doi.org/10.1234/jsma.v5i2.2021.

[16] Mark Walters. Hate speech detection in social networks using machine learning and deep learning methods. *International Network for Hate Studies*, 2023. URL https://internationalhatestudies.com/publications/hate-speech-detection-in-social-networks-using-machine-learning-an

[17] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[18] 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. doi: 10.1002/widm.1253.