Computing, Telecommunication and Control, 2025, Vol. 18, No. 3, Pp. 23–35.
Информатика, телекоммуникации и управление. 2025. Том 18, № 3. С. 23–35.

Research article
DOI: https://doi.org/10.18721/JCSTCS.18302
UDC 004.8

APPLICATION OF MACHINE LEARNING ALGORITHMS AND NEURAL NETWORKS FOR ANALYZING THE INFLUENCE OF DATA TYPE IN HATE SPEECH DETECTION

L.P. Mbele Ossiyi ✉, *P.D. Drobintsev, S.M. Ustinov*

Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

✉ lucprucell@gmail.com

Abstract. At present, communication has reached an unprecedented level of activity thanks to online social platforms that have overcome geographical and linguistic barriers. However, the shift to online communication is accompanied by the spread of hate speech, which negatively affects the social environment of these platforms. In the field of natural language processing, research is being conducted to develop models for detecting and classifying hate speech, aimed at improving the safety and quality of the online environment. However, many of these studies are based on commonly used datasets that turn out to be unbalanced and insufficiently adapted to the new grammatical features of hate speech. This article presents a comparative study of the effectiveness of machine and deep learning algorithms in detecting hate speech based on a synthetic dataset. Three separate experiments were conducted using original and synthetically perturbated data. The findings indicate that employing a synthetic dataset enhances the representation of extremely negative or infrequently encountered communication scenarios, contributing to their more effective detection. Deep learning algorithms demonstrated superior performance in all experiments. The top-performing models in the first and second experiments, both using zero-shot learning, yielded accuracies of 52.04% and 62.13%, respectively. The last experiment revealed that the BiGRU + fastText architecture outperformed other models, achieving an accuracy of 72.68%.

Keywords: sentiment analysis, emotion recognition in text, attention mechanism, embedding, CNN, LSTM, GRU

Citation: Mbele Ossiyi L.P., Drobintsev P.D., Ustinov S.M. Application of machine learning algorithms and neural networks for analyzing the influence of data type in hate speech detection. Computing, Telecommunications and Control, 2025, Vol. 18, No. 3, Pp. 23–35. DOI: 10.18721/JCSTCS.18302

Научная статья
DOI: https://doi.org/10.18721/JCSTCS.18302
УДК 004.8

ПРИМЕНЕНИЕ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ И НЕЙРОННЫХ СЕТЕЙ ДЛЯ АНАЛИЗА ВЛИЯНИЯ ТИПА ДАННЫХ ПРИ ВЫЯВЛЕНИИ НЕНАВИСТНИЧЕСКИХ ВЫСКАЗЫВАНИЙ

Л.П. Мбеле Оссийи ✉, *П.Д. Дробинцев, С.М. Устинов*

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация

✉ lucprucell@gmail.com

Аннотация. В настоящее время общение достигло беспрецедентного уровня активности благодаря онлайн-социальным платформам, которые преодолели географические и языковые барьеры. Однако этот переход сопровождается распространением ненавистнических высказываний, которые негативно влияют на социальную среду этих платформ. В области обработки естественного языка ведутся исследования по разработке моделей для выявления и классификации ненавистнических высказываний, направленные на улучшение безопасности и качества онлайн-среды. Однако многие из этих исследований основаны на наборах данных, которые часто используются и оказываются несбалансированными и недостаточно адаптированными к новым грамматическим особенностям ненавистнических высказываний. В этой статье представлено сравнительное исследование эффективности алгоритмов машинного и глубокого обучения в выявлении ненавистнических высказываний на основе синтетического набора данных. Три отдельных эксперимента были проведены с использованием оригинальных и искусственно искаженных данных. Результаты показывают, что использование синтетического набора данных позволяет лучше представить крайне негативные или нечасто встречающиеся сценарии коммуникации, что способствует их более эффективному выявлению. Алгоритмы глубокого обучения продемонстрировали превосходную производительность во всех экспериментах. Лучшие модели в первом и втором экспериментах, основанные на «обучении без примеров», показали точность 52,04% и 62,13% соответственно. Последний эксперимент показал, что архитектура BiGRU + fastText превзошла другие модели, достигнув точности 72,68%.

Ключевые слова: анализ тональности текста, распознавание эмоций в тексте, механизм внимания, эмбеддинги, CNN, LSTM, GRU

Для цитирования: Mbele Ossiyi L.P., Drobintsev P.D., Ustinov S.M. Application of machine learning algorithms and neural networks for analyzing the influence of data type in hate speech detection // Computing, Telecommunications and Control. 2025. Т. 18, № 3. С. 23–35. DOI: 10.18721/JCSTCS.18302

Introduction

One of the challenges in modern online communication environments, such as forums, blogs and social media, is hate speech. Directed at individuals or groups of people, it is often based on characteristics such as skin color, religion, gender, nationality and others. The level of toxicity on the internet, measured by the amount of hate speech, has increased since the beginning of the COVID-19 pandemic in 2020 [1, 2], when a significant portion of social interactions shifted to online platforms. A number of international organizations, such as UNESCO, reported an increase in hate speech and conspiracy theories against specific communities on social media. According to a UNESCO/Ipsos report conducted in 2023 in 16 countries, 67% of internet users have encountered toxic messages and comments.

To create safe digital spaces where hate speech will be automatically detected, extensive research has been conducted [3–6]. An analysis of these studies suggests that the use of machine learning algorithms and neural networks for hate speech detection is becoming critical in modern conditions. For instance, multilayer neural network architectures enable the learning of hierarchical data representations, which is highly valuable for understanding the context and nuances of human language. Hate speech detection relies on two main approaches: supervised learning and unsupervised learning. In the context of this work and the available dataset, a supervised learning approach will be employed, where models are trained on labeled datasets containing examples of both hate speech and ordinary statements.

The work [7] explores a research direction that has not yet been widely covered in scientific literature – namely, the use of synthetic data as a non-traditional approach to overcoming the difficulties associated with collecting and annotating real data. Synthetic datasets enable the generation of a wide range of scenarios and hate speech instances that may be underrepresented in real datasets. In [8], the developed a method to maintain baseline model performance in case of future perturbations, instead of training and retraining the model on data with introduced perturbations as a mitigation method. However, this method is effective only for perturbations that preserve text semantics and exclude those that alter semantics, which are prevalent in [7]. Furthermore, this approach is suitable only for large language models with numerous parameters and high training costs. Experiments in [9] demonstrated that within fine-tuning, the performance of large language models improved by 7–19% partly due to the use of a specific synthetic dataset from [7]. Similarly, the work [10], also based on [7], focused on the automatic detection of dehumanizing statements and achieved promising results. However, it relied exclusively on large language models with extensive parameters. While the studies aim to enhance classifier performance using synthetic data with introduced perturbations, none of them investigate the impact of data type on the performance and robustness of machine learning and deep learning classifiers that do not require a large number of parameters.

Our work continues the line of research initiated in [7]. The central idea is to evaluate the influence of data type (original and synthetically perturbated) on classifier performance in binary classification, where the input is one-dimensional textual data.

The key contributions of this work are as follows:

- Utilization of a synthetic dataset.
- Application of binary classification through the training and testing of various classifiers, including traditional machine learning models (Linear Support Vector Classifier, Logistic Regression, Stochastic Gradient Descent, XGBoost) and deep learning models (CNN, LSTM, GRU, BiGRU, BiGRU + CNN) for hate speech detection.
- Investigation of the impact of static context-independent embeddings models (fastText and GloVe) on classifier performance.
- Examination of how original and synthetically perturbated data influence classifier performance, as this issue has not yet been sufficiently addressed in scientific literature.

Experimental Framework

Dataset

Hate speech detection typically involves the use of various benchmark datasets (e.g., *Wikipedia Detox*, 2016; *Jigsaw Toxic Comment Classification*, 2018; *SemEval-2019 Task 5*) for heuristic studies. However, it should be noted that most of these datasets, although some are relatively large and of high quality, gradually become outdated and lose relevance over time. Therefore, in this work we employ the *Dynamically Generated Hate Speech Dataset* from Vidgen et al. (2021), which has not yet been widely utilized or extensively discussed in scientific literature.

Dataset description

The *Dynamically Generated Hate Speech Dataset* comprises approximately 40000 entries (~10000 per round), generated and annotated by trained annotators across four rounds of dynamic data creation using a human-in-the-loop process. The dataset is balanced, with hate speech instances constituting 54%. All entries are labeled as either *hateful* or *non-hateful*. For entries labeled as *hateful*, secondary annotations are provided, specifying the *type* and *target* of hate speech.

The dataset contains both original and synthetically perturbated data (~15000 complex perturbations). The original data consists of unmodified instances (e.g., without altered annotations). The synthetically perturbated data, available in the version used in our work, comprise statements that were initially considered *non-hateful* but, after modification (typically syntactic), were re-annotated as *hateful*.

As noted in [7, 11], perturbations are generally described as sufficient manipulations of the original text to alter the label (e.g., from *hateful* to *non-hateful*). Such perturbations can significantly change the meaning of a sentence and, consequently, the model's predictions. According to [11], perturbation-based methods applied to text remain in their early stages. Nevertheless, recent studies [8, 12–15] have proposed several semantic-preserving and semantic-altering perturbation techniques. When applied to the text, these techniques allow models for developing robustness against future or adversarial perturbations that might otherwise cause misclassification.

Dataset analysis

The dataset consists of 12 columns, including *label*, *type* and *target*.

The *label* column takes two values: *hateful* or *non-hateful*, indicating whether a given utterance constitutes hate speech.

The *type* column provides an additional annotation for hateful utterances. If an utterance is labeled *hateful*, the *type* column can take one of five values: *animosity*, *derogation*, *dehumanization*, *threatening* or *support*.

The *target* column specifies the group subjected to hate speech. Examples include *wom* (women), *bla* (black people) and *mus* (muslims). The *target* column contains more than 400 unique values. For example, the phrase "*There are so many black women at my workplace, it really annoys me*" in the dataset is labeled as *hateful*, with *type* = *animosity* and *target* = *bla.wom* (referring to black women).

The distribution of entries across the *label* and *type* columns is illustrated in Fig. 1. The presence of the "*not given*" category in relation to *type* is explained by the absence of hate-type annotation in round 1. Among the *type* categories, *derogation* (utterances that explicitly attack, demonize, humiliate or insult a group) is the most frequent, while *support* (utterances that praise or endorse events, organizations, actions that propagate hate) is the least frequent.

Fig. 2 presents the distribution of words and characters across the *label* column. The maximum utterance length does not exceed 600 characters or 150 words. Both *hateful* and *non-hateful* labels show a similar distribution in terms of word and character length. However, there is a notable difference: approximately 28% of *non-hateful* utterances contain words with lengths between 1 and 25 characters, compared to ~35% of *hateful* utterances.

Data preprocessing

The data preprocessing procedure was designed to reduce vocabulary size without removing essential content. A smaller vocabulary not only decreases the memory required for analysis, but also enhances the reliability of estimated word parameters. In this work, standard preprocessing operations were applied, albeit with some modifications. As noted in [16, 17], these operations included lowercasing, tokenization, punctuation handling, stop-word removal, part-of-speech (POS) tagging (to improve semantic understanding of text and facilitate more accurate lemmatization) and lemmatization.

However, to provide classifiers with a more favorable learning environment, we followed the approach of [18] and replaced contracted negative forms with their full equivalents. In addition, emojis were substituted with their corresponding semantic meanings. Furthermore, as part of the preprocessing pipeline, the maximum length of individual posts was limited to 100 words and 500 characters, respectively, for subsequent operations.

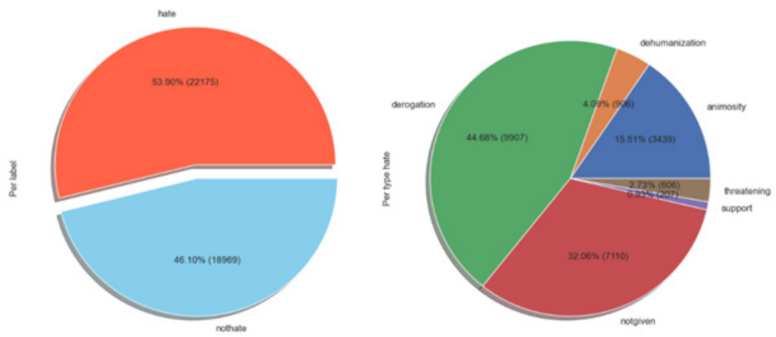

Fig. 1. Distribution of the dataset across *labels* and hate *types*

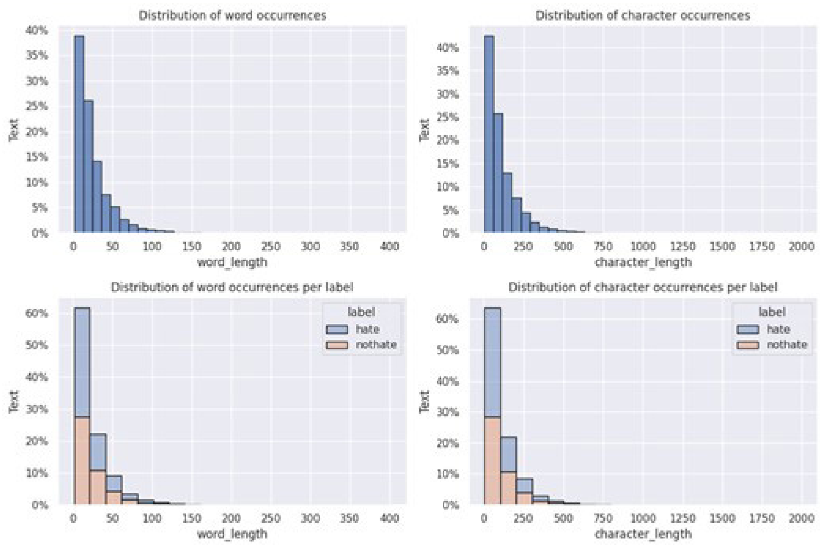

Fig. 2. Distribution of words and characters by *label*

Models

The primary focus of this work was on deep neural network architectures. For comparative analysis, several traditional machine learning methods were employed as baseline models, including Linear Support Vector Classifier (Linear SVC), Stochastic Gradient Descent (SGD), Logistic Regression (LR) and Extreme Gradient Boosting (XGBoost).

The rationale for selecting these algorithms is as follows: Linear SVC, a specific case of Support Vector Machines, assumes a linear decision boundary (effective when classes are well-separated in feature space, as in our case), handles high-dimensional spaces efficiently and thereby mitigates overfitting. SGD serves as an optimization algorithm that updates model parameters incrementally, allowing for faster convergence compared to batch gradient descent. LR is effective in tasks where the relationship between features and class labels can be approximated linearly, as demonstrated in our experiments. Finally, XGBoost excels at handling missing values, prevents overfitting and can capture complex feature interactions and non-linear relationships.

Neural networks

In this work, we employed five neural network architectures: Convolutional Neural Networks (CNNs) [19, 20], Long Short-Term Memory networks (LSTM) [19], Gated Recurrent Units (GRU), Bidirectional GRU (BiGRU) and a hybrid CNN + BiGRU model. All these models were implemented with word embedding methods (fastText and GloVe) [19]. To enhance the effectiveness of the neural networks, we hypothesized that performance could be improved using attention mechanisms and pooling operations within the network architecture. The attention mechanism assigns different weights to sequence elements, allowing models to focus on specific parts of the input data, thereby improving their ability to generate accurate and contextually relevant predictions. Pooling, in turn, reduces computational complexity and facilitates the handling of long sequences.

The choice of deep neural network architectures is motivated by the fact that traditional machine learning methods largely rely on manual feature engineering, whereas deep learning models are capable of learning abstract data representations and automatically extracting features [18, 21].

CNN

CNNs were originally developed for computer vision tasks and are highly effective in image classification [22, 23]. However, CNNs have also demonstrated strong applicability in natural language processing (NLP), particularly for text classification tasks [24, 25]. While CNNs are primarily designed for processing data represented as matrices rather than sequences, they can outperform recurrent neural networks (RNNs) [22], especially in their ability to capture higher-level features. The role of a CNN layer is to extract meaningful substructures that are useful for solving the overall prediction task. In this work, we implemented a CNN with a global max pooling mechanism to reduce computational complexity and the number of outputs [26].

LSTM and GRU

LSTM and GRU networks are types of recurrent neural networks [24, 26]. In text classification tasks, each LSTM or GRU block processes both the embedding vector of the current word and the output of the previous block, recursively accumulating information from all other words in the text. Unlike traditional RNNs, LSTM and GRU networks are specifically designed to overcome the problems of long-term dependencies and the issues of exploding and vanishing gradients [18, 26].

These models employ more advanced mechanisms for computing hidden states at each step to mitigate gradient-related problems [27]. Both LSTM and GRU incorporate gating mechanisms that enable selective retention or forgetting of information from previous inputs. LSTMs feature a more complex structure consisting of four components: input gate, forget gate, cell state and output gate. In contrast, GRUs represent a generalized approach, with LSTMs being a special case [27]. GRUs typically require fewer filters and fewer computational operations than LSTMs [26, 27].

BiGRU and BiGRU + CNN

The concept of bidirectionality was applied in cases where the meaning of certain words depends on subsequent words in the sentence. This is particularly relevant for synthetically perturbated data, where adding a word at the end of a sentence may alter its entire meaning. In addressing this issue, a choice was made between BiGRU and BiLSTM. Ultimately, BiGRU was selected, primarily due to the simpler architecture and faster training of GRUs, as well as their ability to be effectively trained to preserve information over long sequences without loss of temporal dependencies [3]. To further improve key aspects of our work – such as addressing CNN limitations in capturing inter-word semantics, enhancing prediction accuracy, modeling complex relationships, extracting features and patterns, managing long-range dependencies and ensuring robustness to noise and outliers – we adopted a hybrid approach [23, 28] that combines CNN and BiGRU. We hypothesized that this hybrid architecture leverages the strengths of both models while compensating for their respective weaknesses.

Experimental Setup

In [16], it was demonstrated that word embedding methods (such as fastText and GloVe), which are most used in deep learning models, can also yield strong results when applied within machine learning frameworks. Following this line of reasoning, we adopted the same approach in our baseline machine learning experiments. Alongside word embedding methods, we also employed the TF-IDF bag-of-words model [19, 29] to extract features from textual sequences. The same word embedding techniques were consistently applied across all deep learning models.

The performance of classifiers was evaluated using several metrics, including accuracy, macro-precision, macro-recall and macro-F1 score. Additionally, as in [18], to better handle the influence of true negatives – which are of limited utility in detecting hate speech – we incorporated the area under the precision-recall curve (AUC-PRC), in addition to the area under the receiver operating characteristic curve (AUC-ROC).

In this work, we focused on binary classification (hate / non-hate) and trained and evaluated the models across three experimental settings:

- Models were trained exclusively on synthetically perturbated data, but developed and tested on original data.
- Models were trained solely on original data, but developed and tested on synthetically perturbated data.
- Models were trained, developed and tested on a combination of both synthetically perturbated and original data.

The corpus was split into training, cross-validation and test sets in accordance with the nature of the data (original and synthetically perturbated).

For experimentation, we employed the Google Colab environment, which supports TensorFlow (version 2.15.0) and provides access to fast, high-performance computing resources such as GPU and TPU. The programming language used was Python 3.10, and computations were run on an MSI Katana 17 (i7-12650H, 16 GB RAM). The source code is publicly available[1].

The table below presents the configuration parameters for all classifiers. All parameters were obtained through hyperparameter tuning.

Table 1

Hyperparameter settings of the baseline models

Models	**Parameters**
Linear Support Vector Classification	*C = 0.1, max_iter = 1000*
Logistic Regression	*C = 1, penalty = 'l2', solver = 'liblinear', max_iter = 10000*
Stochastic Gradient Descent	*loss:'hinge', alpha: 0.0001, penalty: 'l2'*
Extreme GBOOST	*learning_rate = 0.1, n_estimators = 100, max_depth = 5, min_child_weight = 1, gamma = 0, subsample = 0.8, colsample_bytree = 0.8, objective = 'binary: logistic', nthread = 4, scale_pos_weight = 1, seed = 27*
CNN	*filters = 512, kernel_size = 6, dropout_rate = 0.5, dense_units = 512, emb_dim = = 300, optimizer = 'Adagrad', learning_rate = 0.00001*
LSTM	*lstm_units = 64, dense_units = 512, k_regularizer = 0.001, dropout_rate = 0.3, recurrent_dropout = 0.0, emb_dim = 300, optimizer = 'Adam', learning_rate = 0.001*
GRU	*gru_units = 64, dropout_rate = 0.5, k_regularizer = 0.00001, recurrent_dropout = = 0.0, emb_dim = 300, optimizer = 'Adam', learning_rate = 0.001*
BiGRU	*gru_units = 256, k_regularizer = 0.00001, dropout_rate = 0.5, recurrent_dropout = = 0.0, emb_dim = 300, optimizer = 'Adam', learning_rate = 0.001*
BiGRU + CNN	*filters = 16, kernel_size=6, dropout_rate= 0.5, dense_units= 64, gru_units = 256, k_regularizer = 0.00001, recurrent_dropout= 0.0, emb_dim = 300, optimizer = = 'Adam', learning_rate= 0.003*

[1] GitHub – LucasMbele/Hate-speech-synthetic-dataset: In this repository, we train and test some classifiers on original and perturbated data from a synthetical dataset for hate classification tasks in binary and multiclass case. Available: https://github.com/LucasMbele/Hate-speech-synthetic-dataset (Accessed 12.09.2025)

Results

Experiment 1: Training on original data, development and testing on synthetically perturbated data

The data were split as follows: 25813 for the training set, 10332 for the development set and 4429 for the test set.

Based on the results presented in Fig. 3, among all machine learning methods, the logistic regression algorithm combined with fastText generally outperforms and achieves the best results in terms of accuracy (**45.3%**), F1-score (**44.2%**) and AUC-ROC (**45.2%**). It is worth noting that the differences between the results of other machine learning algorithms and those of logistic regression are negligible.

It is evident that neural networks outperform machine learning algorithms, as expected. BiGRU + CNN + GloVe (**52.04%** accuracy, **51.09%** F1-score) achieves better performance than other models; however, its loss function value is considerably high. Measuring the difference between the model's predictions and the actual values, the loss function plays a crucial role in the efficiency of neural networks.

Models	Folds	Accuracy score			F1-score			ROC score		
		Training set	Dev	Test	Training set	Dev	Test	Training set	Dev	Test
Logistic Regression + TF-IDF	CV Fold 1	0,726			0,71			0,777		
	CV Fold 2	0,65			0,607			0,71		
	CV Fold 3	0,718			0,683			0,805		
	CV Average	**0,698**	**0,456**	**0,436**	**0,667**	**0,429**	**0,409**	**0,764**	**0,455**	**0,436**
Logistic Regression + FastText	CV Fold 1	0,619			0,576			0,684		
	CV Fold 2	0,634			0,622			0,655		
	CV Fold 3	0,689			0,667			0,676		
	CV Average	**0,647**	**0,47**	**0,453**	**0,622**	**0,46**	**0,442**	**0,672**	**0,469**	**0,452**
Logistic Regression + GloVE	CV Fold 1	0,603			0,54			0,658		
	CV Fold 2	0,629			0,601			0,633		
	CV Fold 3	0,686			0,645			0,644		
	CV Average	**0,639**	**0,453**	**0,448**	**0,595**	**0,433**	**0,425**	**0,645**	**0,452**	**0,447**
Linear SVM + TF-IDF	CV Fold 1	0,726			0,708			0,777		
	CV Fold 2	0,649			0,598			0,71		
	CV Fold 3	0,714			0,676			0,806		
	CV Average	**0,696**	**0,452**	**0,437**	**0,661**	**0,417**	**0,402**	**0,764**	**0,45**	**0,436**
Linear SVM + FastText	CV Fold 1	0,62			0,572			0,682		
	CV Fold 2	0,634			0,62			0,655		
	CV Fold 3	0,691			0,666			0,681		
	CV Average	**0,648**	**0,467**	**0,452**	**0,619**	**0,455**	**0,44**	**0,673**	**0,467**	**0,452**
Linear SVM + GloVE	CV Fold 1	0,599			0,53			0,654		
	CV Fold 2	0,627			0,597			0,632		
	CV Fold 3	0,686			0,639			0,647		
	CV Average	**0,637**	**0,453**	**0,447**	**0,589**	**0,431**	**0,419**	**0,644**	**0,452**	**0,446**
Stochastic Gradient Descent + TF-IDF	CV Fold 1	0,716			0,683			0,783		
	CV Fold 2	0,635			0,554			0,71		
	CV Fold 3	0,7			0,646			0,813		
	CV Average	**0,684**	**0,45**	**0,448**	**0,628**	**0,354**	**0,356**	**0,769**	**0,448**	**0,447**
Stochastic Gradient Descent + FastText	CV Fold 1	0,619			0,587			0,669		
	CV Fold 2	0,63			0,615			0,653		
	CV Fold 3	0,669			0,614			0,686		
	CV Average	**0,639**	**0,464**	**0,443**	**0,605**	**0,458**	**0,436**	**0,669**	**0,463**	**0,442**
Stochastic Gradient Descent + GloVE	CV Fold 1	0,603			0,511			0,643		
	CV Fold 2	0,613			0,552			0,632		
	CV Fold 3	0,693			0,504			0,663		
	CV Average	**0,636**	**0,452**	**0,442**	**0,522**	**0,401**	**0,386**	**0,646**	**0,45**	**0,441**
XGBOOST + TF-IDF	CV Fold 1	0,672			0,641			0,711		
	CV Fold 2	0,626			0,569			0,662		
	CV Fold 3	0,703			0,658			0,726		
	CV Average	**0,667**	**0,417**	**0,42**	**0,623**	**0,373**	**0,377**	**0,7**	**0,416**	**0,42**
XGBOOST + FastText	CV Fold 1	0,662			0,652			0,714		
	CV Fold 2	0,633			0,623			0,688		
	CV Fold 3	0,695			0,678			0,749		
	CV Average	**0,663**	**0,393**	**0,391**	**0,651**	**0,388**	**0,386**	**0,717**	**0,392**	**0,39**
XGBOOST + GloVE	CV Fold 1	0,658			0,648			0,714		
	CV Fold 2	0,629			0,617			0,675		
	CV Fold 3	0,697			0,678			0,746		
	CV Average	**0,661**	**0,391**	**0,406**	**0,648**	**0,387**	**0,401**	**0,712**	**0,391**	**0,406**

Models	Folds	Accuracy score				Precision score		Recall score		Test		
		CV	Loss	Test	Loss	CV	Test	CV	Test	AUPRC score	AUC score	F1-score
CNN + FastText	CV Fold 1	0,5096	0,7207			0,5083		0,3897				
	CV Fold 2	0,503	0,7286			0,4995		0,3057				
	CV Fold 3	0,5028	0,7311			0,4992		0,2919				
	CV Average/ Best Loss	***0,5051***	***0,7207***	***51,46%***	0,7205	**0,5023**	**51,62%**	**0,3291**	**51,52%**	**0,5142**	**0,5183**	**50,76%**
CNN + GloVE	CV Fold 1	0,5069	0,6988			0,5026		0,7048				
	CV Fold 2	0,4994	0,6977			0,4952		0,4014				
	CV Fold 3	0,5031	0,6996			0,4996		0,2627				
	CV Average/ Best Loss	**0,5031**	**0,6977**	**50,62%**	0,6993	**0,4991**	**50,60%**	**0,4563**	**50,51%**	**0.4952**	**0.5017**	**48,74%**
LSTM + FastText	CV Fold 1	0,4441	0,9449			0,4054		0,2555				
	CV Fold 2	0,5022	0,938			0,4991		0,582				
	CV Fold 3	0,4875	0,9916			0,4864		0,5688				
	CV Average/ Best Loss	**0,4779**	**0,938**	**49,97%**	0,9471	**0,4636**	**49,92%**	**0,46877**	**49,92%**	**0.4834**	**0.4861**	**49.58%**
LSTM + GloVE	CV Fold 1	0,4654	0,9853			0,4571		0,4072				
	CV Fold 2	0,5109	0,9413			0,51		0,3911				
	CV Fold 3	0,5063	0,9342			0,504		0,3843				
	CV Average/ Best Loss	**0,4942**	**0,9342**	**50,87%**	0,941	**0,4904**	**50,99%**	**0,3942**	**50,94%**	**0,5104**	**0,5074**	**50,21%**
GRU + FastText	CV Fold 1	0,4714	0,9326			0,459		0,3589				
	CV Fold 2	0,4808	0,9175			0,4772		0,4739				
	CV Fold 3	0,4867	0,992			0,4838		0,4971				
	CV Average/ Best Loss	**0,47963**	**0,9175**	**48,97%**	0,9878	**0,4733**	**48,97%**	**0,4433**	**48,97%**	**0,4938**	**0,4915**	**48,96%**
GRU + GloVE	CV Fold 1	0,4977	0,8744			0,4934		0,4215				
	CV Fold 2	0,4562	0,9203			0,4502		0,4289				
	CV Fold 3	0,4678	0,9602			0,4597		0,4078				
	CV Average/ Best Loss	**0,4739**	**0,8744**	**50,08%**	0,877	**0,4678**	**50,12%**	**0,4194**	**50,12%**	**0,4898**	**0,4866**	**49,82%**
BIGRU + FastText	CV Fold 1	0,4838	0,8684			0,4708		0,3161				
	CV Fold 2	0,4828	0,9147			0,4723		0,3525				
	CV Fold 3	0,494	0,946			0,4906		0,4875				
	CV Average/ Best Loss	**0,4869**	**0,8684**	**48,66%**	0,9556	**0,4779**	**48,65%**	**0,3854**	**48,65%**	**0,492**	**0,4803**	**48,65%**
BIGRU + GloVE	CV Fold 1	0,4991	0,9206			0,4944		0,373				
	CV Fold 2	0,479	0,959			0,4465		0,2042				
	CV Fold 3	0,4833	1,023			0,4783		0,4447				
	CV Average/ Best Loss	**0,48713**	**0,9206**	**50,96%**	0,913	**0,4731**	**51,08%**	**0,34063**	**51,03%**	**0,5011**	**0,5091**	**50,31%**
BIGRU + CNN + FastText	CV Fold 1	0,4383	0,9533			0,4101		0,2983				
	CV Fold 2	0,4691	1,087			0,4106		0,158				
	CV Fold 3	0,4666	1,1431			0,4174		0,1865				
	CV Average/ Best Loss	**0,458**	**0,9533**	**46,67%**	1,0986	**0,4127**	**45,10%**	**0,2143**	**46,84%**	**0,4407**	**0,4412**	**41,58%**
BIGRU + CNN + GloVE	CV Fold 1	0,4775	1,0201			0,4021		0,1064				
	CV Fold 2	0,5227	1,0387			0,527		0,3825				
	CV Fold 3	0,5099	1,1369			0,5091		0,3693				
	CV Average/ Best Loss	**0,5034**	**1,0201**	**52,04%**	1,0402	**0,4794**	**52,31%**	**0,2861**	**52,12%**	**0,5053**	**0,5159**	**51,09%**

Fig. 3. Performance of machine learning and deep learning models in Experiment 1

Models	Folds	Accuracy score			F1-score			ROC score		
		Training set	Dev	Test	Training set	Dev	Test	Training set	Dev	Test
Logistic Regression + TF-IDF	CV Fold 1	0,686			0,686			0,741		
	CV Fold 2	0,651			0,645			0,716		
	CV Fold 3	0,58			0,556			0,65		
	CV Average	**0,639**	**0,454**	**0,454**	**0,629**	**0,448**	**0,449**	**0,702**	**0,448**	**0,449**
Logistic Regression + FastText	CV Fold 1	0,627			0,627			0,661		
	CV Fold 2	0,647			0,647			0,697		
	CV Fold 3	0,579			0,576			0,609		
	CV Average	**0,618**	**0,465**	**0,46**	**0,617**	**0,461**	**0,456**	**0,656**	**0,462**	**0,456**
Logistic Regression + GloVE	CV Fold 1	0,637			0,637			0,692		
	CV Fold 2	0,627			0,626			0,667		
	CV Fold 3	0,558			0,555			0,577		
	CV Average	**0,607**	**0,472**	**0,477**	**0,606**	**0,465**	**0,469**	**0,645**	**0,465**	**0,469**
Linear SVM + TF-IDF	CV Fold 1	0,685			0,685			0,738		
	CV Fold 2	0,649			0,642			0,714		
	CV Fold 3	0,577			0,551			0,648		
	CV Average	**0,637**	**0,453**	**0,453**	**0,626**	**0,448**	**0,45**	**0,7**	**0,448**	**0,45**
Linear SVM + FastText	CV Fold 1	0,624			0,624			0,661		
	CV Fold 2	0,647			0,647			0,698		
	CV Fold 3	0,578			0,575			0,607		
	CV Average	**0,616**	**0,462**	**0,458**	**0,615**	**0,458**	**0,454**	**0,655**	**0,459**	**0,455**
Linear SVM + GloVE	CV Fold 1	0,638			0,638			0,691		
	CV Fold 2	0,627			0,626			0,666		
	CV Fold 3	0,554			0,551			0,574		
	CV Average	**0,606**	**0,475**	**0,479**	**0,605**	**0,468**	**0,47**	**0,644**	**0,468**	**0,47**
Stochastic Gradient Descent + TF-IDF	CV Fold 1	0,674			0,67			0,741		
	CV Fold 2	0,617			0,577			0,72		
	CV Fold 3	0,548			0,488			0,659		
	CV Average	**0,613**	**0,434**	**0,446**	**0,578**	**0,433**	**0,444**	**0,707**	**0,447**	**0,459**
Stochastic Gradient Descent + FastText	CV Fold 1	0,619			0,664			0,664		
	CV Fold 2	0,648			0,694			0,694		
	CV Fold 3	0,564			0,616			0,616		
	CV Average	**0,610**	**0,444**	**0,443**	**0,658**	**0,443**	**0,442**	**0,658**	**0,452**	**0,451**
Stochastic Gradient Descent + GloVE	CV Fold 1	0,648			0,503			0,699		
	CV Fold 2	0,627			0,59			0,664		
	CV Fold 3	0,549			0,557			0,589		
	CV Average	**0,608**	***0,481***	***0,485***	**0,55**	***0,47***	***0,473***	**0,651**	***0,47***	***0,473***
XGBOOST + TF-IDF	CV Fold 1	0,709			0,707			0,764		
	CV Fold 2	0,678			0,676			0,731		
	CV Fold 3	0,621			0,61			0,663		
	CV Average	***0,669***	**0,457**	**0,46**	***0,664***	**0,457**	**0,46**	***0,719***	**0,463**	**0,466**
XGBOOST + FastText	CV Fold 1	0,611			0,611			0,659		
	CV Fold 2	0,618			0,617			0,66		
	CV Fold 3	0,558			0,552			0,581		
	CV Average	**0,596**	**0,379**	**0,392**	**0,593**	**0,378**	**0,39**	**0,633**	**0,379**	**0,392**
XGBOOST + GloVE	CV Fold 1	0,636			0,684			0,684		
	CV Fold 2	0,617			0,658			0,658		
	CV Fold 3	0,569			0,59			0,59		
	CV Average	**0,607**	**0,421**	**0,427**	**0,644**	**0,419**	**0,424**	**0,644**	**0,42**	**0,424**

Models	Folds	Accuracy score				Precision score		Recall score		Test		
		CV	Loss	Test	Loss	CV	Test	CV	Test	AUPRC score	AUC score	F1-score
CNN + FastText	CV Fold 1	0,4996	0,728			0,449		0,5949				
	CV Fold 2	0,5024	0,7244			0,4482		0,5576				
	CV Fold 3	0,5035	0,7214			0,4452		0,5132				
	CV Average/ Best Loss	***0,5018***	***0,7214***	***50,72%***	**0,72**	**0,447**	**50,93%**	**0,555**	**50,95%**	**0,439**	**0,507**	**50,61%**
CNN + GloVE	CV Fold 1	0,503	0,6986			0,3934		0,2353				
	CV Fold 2	0,4887	0,6999			0,3976		0,3106				
	CV Fold 3	0,4826	0,7009			0,4027		0,3596				
	CV Average/ Best Loss	**0,4914**	**0,6986**	**50,05%**	**0,7**	**0,3979**	**46,91%**	**0,3018**	**47,11%**	**0,402**	**0,449**	**45,42%**
LSTM + FastText	CV Fold 1	0,5351	0,9364			0,4744		0,5066				
	CV Fold 2	0,5729	0,905			0,517		0,4747				
	CV Fold 3	0,5592	0,9327			0,5001		0,5289				
	CV Average/ Best Loss	**0,5557**	**0,905**	**57,39%**	**0,91**	**0,4972**	**56,52%**	**0,5034**	**56,45%**	**0,494**	**0,571**	**56,47%**
LSTM + GloVE	CV Fold 1	0,4849	0,8501			0,432		0,5352				
	CV Fold 2	0,5487	0,8494			0,4874		0,4595				
	CV Fold 3	0,5429	0,8675			0,4844		0,5733				
	CV Average/ Best Loss	**0,5255**	**0,8494**	**54,79%**	**0,85**	**0,4679**	**53,94%**	**0,5227**	**53,92%**	**0,474**	**0,545**	**53,92%**
GRU + FastText	CV Fold 1	0,5022	0,8773			0,4339		0,4241				
	CV Fold 2	0,5163	0,8892			0,4535		0,4746				
	CV Fold 3	0,5468	0,8438			0,486		0,4886				
	CV Average/ Best Loss	**0,5218**	**0,8438**	**55,58%**	**0,9**	**0,4578**	**54,47%**	**0,4624**	**54,35%**	**0,468**	**0,55**	**54,33%**
GRU + GloVE	CV Fold 1	0,5451	0,8036			0,478		0,3465				
	CV Fold 2	0,5233	0,4549			0,4549		0,4112				
	CV Fold 3	0,5629	0,5045			0,5045		0,4703				
	CV Average/ Best Loss	**0,5438**	**0,4549**	**55,73%**	**0,81**	**0,4791**	**54,88%**	**0,4093**	**54,84%**	**0,466**	**0,554**	**54,85%**
BIGRU + FastText	CV Fold 1	0,5374	0,9361			0,4756		0,4821				
	CV Fold 2	0,5035	0,9822			0,4435		0,4954				
	CV Fold 3	0,5341	1,0659			0,4639		0,3659				
	CV Average/ Best Loss	**0,5250**	**0,9361**	**53,62%**	**0,94**	**0,461**	**53,13%**	**0,4478**	**53,16%**	**0,46**	**0,528**	**53,11%**
BIGRU + GloVE	CV Fold 1	0,5286	0,8264			0,465		0,4611				
	CV Fold 2	0,5642	0,8131			0,5066		0,4355				
	CV Fold 3	0,5219	0,8792			0,4545		0,4218				
	CV Average/ Best Loss	**0,53823**	**0,8131**	**56,46%**	**0,81**	**0,4754**	**55,34%**	**0,43947**	**55,17%**	**0,476**	**0,557**	**55,15%**
BIGRU + CNN + FastText	CV Fold 1	0,55	1,0292			0,4836		0,4306				
	CV Fold 2	0,5867	1,0314			0,5318		0,4576				
	CV Fold 3	0,6197	0,9674			0,57		0,5298				
	CV Average/ Best Loss	**0,58547**	**0,9674**	**62,13%**	**0,98**	**0,5285**	**61,29%**	**0,4727**	**61,29%**	**0,5377**	**0,6054**	**61,33%**
BIGRU + CNN + GloVE	CV Fold 1	0,528	0,7952			0,4445		0,3185				
	CV Fold 2	0,5406	0,8138			0,4699		0,3948				
	CV Fold 3	0,4903	0,8455			0,4219		0,4476				
	CV Average/ Best Loss	**0,5196**	**0,7952**	**55,75%**	**0,98**	**0,4454**	**54,77%**	**0,3870**	**54,59%**	**0,47**	**0,539**	**54,48%**

Fig. 4. Performance of machine learning and deep learning models in Experiment 2

The higher the loss function value, the harder it is for the model to make accurate predictions, thereby indicating the need for further improvements. In the experiment conducted, the loss function reached **1.0201**. CNN models (CNN + fastText and CNN + GloVe, as reported in [4]), particularly due to their robust feature extraction capabilities, show a clear improvement in the loss function (0.6993 with GloVe), comparable accuracy (**51.46%** with fastText) and superior results in terms of AUPRC-score (**51.42%** with fastText) and AUC-ROC score (**51.83%** with fastText).

The perturbations introduced into the validation and test datasets proved difficult for the models to learn, resulting in poor performance.

Experiment 2: Training on synthetically perturbated data, development and testing on original data

The data were split as follows: 14761 for the training set, 19359 for the development set and 6454 for the test set.

As shown in Fig. 4, neural networks significantly outperform machine learning models. Among machine learning algorithms, XGBoost and SGD stand out. XGBoost combined with TF-IDF achieved the best results on the training set, while SGD with GloVe obtained the best results on the test set (48.5%

accuracy, which is +3.2% higher than the accuracy of the best algorithm in Experiment 1; 47.3% F1-score, which is +3.1% higher than the F1-score of the best algorithm in Experiment 1; and 47.3% AUC-ROC score, which is +2.1% higher than the AUC-ROC score of the best algorithm in Experiment 1).

BiGRU + CNN + fastText, as in Experiment 1, outperformed all other algorithms, achieving **62.13**% accuracy, **61.29**% precision, **61.33**% recall, **61.33**% F1-score, **53.77**% AUPRC and **60.54**% AUC-ROC. Training models on synthetically perturbated data and testing them on original data substantially improved the performance of neural networks compared to Experiment 1. Overall, an improvement of +**10.9**% in accuracy and +**10.24**% in F1-score was observed when comparing the best model from Experiment 2 to the best model from Experiment 1.

Experiment 3: Training, development, and testing on both original and synthetically perturbated data

The data were split as follows: 19475 for the training set, 16230 for the development set and 4869 for the test set.

As shown in Fig. 5, among all machine learning algorithms, XGBoost (combined with TF-IDF), as in Experiment 2, achieved the best performance (**65.8**% F1-score, **65.9**% AUC-ROC). Logistic regression combined with TF-IDF achieved the highest accuracy (**66.2**%).

Models	Folds	Accuracy score			F1-score			ROC score		
		Training set	Dev	Test	Training set	Dev	Test	Training set	Dev	Test
Logistic Regression + TF-IDF	CV Fold 1	0,641			0,623			0,716		
	CV Fold 2	0,654			0,635			0,729		
	CV Fold 3	0,646			0,628			0,717		
	CV Average	**0,647**	**0,656**	***0,662***	**0,629**	**0,643**	**0,651**	***0,721***	**0,645**	**0,652**
Logistic Regression + FastText	CV Fold 1	0,608			0,587			0,641		
	CV Fold 2	0,612			0,591			0,653		
	CV Fold 3	0,612			0,594			0,648		
	CV Average	**0,611**	**0,614**	**0,627**	**0,591**	**0,595**	**0,609**	**0,647**	**0,6**	**0,614**
Logistic Regression + GloVE	CV Fold 1	0,588			0,55			0,63		
	CV Fold 2	0,583			0,537			0,641		
	CV Fold 3	0,599			0,556			0,639		
	CV Average	**0,59**	**0,595**	**0,607**	**0,548**	**0,557**	**0,574**	**0,637**	**0,575**	**0,589**
Linear SVM + TF-IDF	CV Fold 1	0,638			0,614			0,715		
	CV Fold 2	0,65			0,626			0,728		
	CV Fold 3	0,639			0,617			0,716		
	CV Average	**0,642**	**0,637**	**0,655**	**0,619**	**0,637**	**0,641**	**0,720**	**0,64**	**0,644**
Linear SVM + FastText	CV Fold 1	0,606			0,583			0,64		
	CV Fold 2	0,609			0,585			0,652		
	CV Fold 3	0,613			0,593			0,648		
	CV Average	**0,609**	**0,612**	**0,624**	**0,587**	**0,591**	**0,604**	**0,647**	**0,598**	**0,61**
Linear SVM + GloVE	CV Fold 1	0,583			0,54			0,629		
	CV Fold 2	0,578			0,526			0,639		
	CV Fold 3	0,595			0,547			0,638		
	CV Average	**0,585**	**0,591**	**0,6**	**0,538**	**0,549**	**0,561**	**0,635**	**0,57**	**0,58**
Stochastic Gradient Descent + TF-IDF	CV Fold 1	0,616			0,547			0,713		
	CV Fold 2	0,619			0,552			0,724		
	CV Fold 3	0,615			0,553			0,715		
	CV Average	**0,617**	**0,618**	**0,621**	**0,551**	**0,555**	**0,56**	**0,717**	**0,592**	**0,596**
Stochastic Gradient Descent + FastText	CV Fold 1	0,596			0,52			0,639		
	CV Fold 2	0,606			0,552			0,652		
	CV Fold 3	0,608			0,483			0,65		
	CV Average	**0,603**	**0,595**	**0,607**	**0,518**	**0,552**	**0,568**	**0,647**	**0,574**	**0,587**
Stochastic Gradient Descent + GloVE	CV Fold 1	0,564			0,406			0,627		
	CV Fold 2	0,55			0,475			0,64		
	CV Fold 3	0,558			0,411			0,637		
	CV Average	**0,557**	**0,546**	**0,546**	**0,431**	**0,396**	**0,393**	**0,635**	**0,509**	**0,51**
XGBOOST + TF-IDF	CV Fold 1	0,658			0,657			0,709		
	CV Fold 2	0,664			0,664			0,715		
	CV Fold 3	0,651			0,651			0,702		
	CV Average	***0,658***	***0,661***	**0,658**	***0,657***	***0,661***	***0,658***	**0,709**	***0,662***	***0,659***
XGBOOST + FastText	CV Fold 1	0,605			0,597			0,653		
	CV Fold 2	0,604			0,597			0,662		
	CV Fold 3	0,6			0,593			0,656		
	CV Average	**0,603**	**0,605**	**0,611**	**0,596**	**0,598**	**0,604**	**0,657**	**0,598**	**0,604**
XGBOOST + GloVE	CV Fold 1	0,616			0,609			0,667		
	CV Fold 2	0,607			0,6			0,663		
	CV Fold 3	0,663			0,603			0,664		
	CV Average	**0,629**	**0,607**	**0,627**	**0,604**	**0,6**	**0,62**	**0,665**	**0,6**	**0,62**

Models	Folds	Accuracy score				Precision score		Recall score		Test		
		CV	Loss	Test	Loss	CV	Test	CV	Test	AUPRC score	AUC score	F1-score
CNN + FastText	CV Fold 1	0,5473	0,7001			0,5086		0,4144				
	CV Fold 2	0,5492	0,6995			0,512		0,415				
	CV Fold 3	0,555	0,6986			0,513		0,416				
	CV Average/ Best Loss	***0,5505***	***0,6986***	***55,58%***	0,69	**0,5112**	**54,39%**	**0,4151**	**54,11%**	**0,506**	**0,5608**	**53,76%**
CNN + GloVE	CV Fold 1	0,5463	0,6876			0,5104		0,3254				
	CV Fold 2	0,5468	0,6874			0,5112		0,3246				
	CV Fold 3	0,5473	0,6874			0,5122		0,3208				
	CV Average/ Best Loss	**0,5468**	**0,6874**	**55,29%**	0,69	**0,5113**	**54,45%**	**0,3236**	**53,60%**	**0,509**	**0,552**	**52,07%**
LSTM + FastText	CV Fold 1	0,6972	0,7062			0,7295		0,5427				
	CV Fold 2	0,7004	0,6966			0,7007		0,608				
	CV Fold 3	0,7047	0,698			0,7014		0,623				
	CV Average/ Best Loss	**0,7008**	**0,6966**	**70,63%**	0,69	**0,7105**	**70,58%**	**0,59123**	**70,08%**	**0,758**	**0,779**	**70,16%**
LSTM + GloVE	CV Fold 1	0,6999	0,6568			0,7112		0,5847				
	CV Fold 2	0,7002	0,6519			0,7352		0,5437				
	CV Fold 3	0,7049	0,6552			0,7394		0,5532				
	CV Average/ Best Loss	**0,7017**	**0,6519**	**70,32%**	0,65	**0,7286**	**71,05%**	**0,561**	**69,28%**	**0,75**	**0,776**	**69,23%**
GRU + FastText	CV Fold 1	0,7131	0,5806			0,7165		0,6221				
	CV Fold 2	0,7087	0,5773			0,6669		0,7321				
	CV Fold 3	0,7166	0,5774			0,7069		0,6553				
	CV Average/ Best Loss	**0,713**	**0,5773**	**71,97%**	0,57	**0,6968**	**71,85%**	**0,670**	**71,56%**	**0,781**	**0,8**	**71,64%**
GRU + GloVE	CV Fold 1	0,7126	0,577			0,7288		0,5969				
	CV Fold 2	0,7116	0,5782			0,7195		0,6108				
	CV Fold 3	0,7136	0,5818			0,7117		0,6336				
	CV Average/ Best Loss	**0,7126**	**0,577**	**71,64%**	0,58	**0,7200**	**71,59%**	**0,614**	**71,12%**	**0,77**	**0,795**	**71,21%**
BIGRU + FastText	CV Fold 1	0,7215	0,6688			0,7221		0,641				
	CV Fold 2	0,7228	0,6784			0,72		0,6497				
	CV Fold 3	0,7205	0,6625			0,7244		0,6329				
	CV Average/ Best Loss	**0,7216**	**0,6625**	**72,68%**	0,66	**0,722**	**72,65%**	**0,6412**	**72,19%**	**0,79**	**0,81**	**72,29%**
BIGRU + GloVE	CV Fold 1	0,7145	0,5873			0,7077		0,6459				
	CV Fold 2	0,7042	0,6102			0,7662		0,5133				
	CV Fold 3	0,7087	0,6066			0,7181		0,6035				
	CV Average/ Best Loss	**0,709**	**0,5873**	**72,27%**	0,59	**0,7307**	**72,16%**	**0,588**	**71,87%**	**0,772**	**0,797**	**71,94%**
BIGRU + CNN + FastText	CV Fold 1	0,7091	0,7234			0,7319		0,5797				
	CV Fold 2	0,7123	0,7479			0,744		0,5705				
	CV Fold 3	0,71	0,7673			0,7269		0,5917				
	CV Average/ Best Loss	**0,7105**	**0,7234**	**71,04%**	0,74	**0,7343**	**71,73%**	**0,5806**	**70,05%**	**0,77**	**0,788**	**70,04%**
BIGRU + CNN + GloVE	CV Fold 1	0,6477	0,622			0,7713		0,3322				
	CV Fold 2	0,6779	0,5886			0,7321		0,4724				
	CV Fold 3	0,68477	0,5764			0,6798		0,5941				
	CV Average/ Best Loss	**0,6701**	**0,5764**	**68,72%**	0,57	**0,7277**	**68,61%**	**0,4662**	**68,16%**	**0,74**	**0,76**	**68,22%**

Fig. 5. Performance of machine learning and deep learning models in Experiment 3

We observe that the best results across all experiments combined were obtained by BiGRU + fastText, achieving **72.68%** accuracy, **72.65%** precision, **72.19%** recall, **72.29%** F1-score, **79%** AUPRC and **81%** AUC-ROC. By combining original and synthetically perturbated data, we achieved the highest performance across various models.

Conclusion

This work achieved several key objectives.

First, it demonstrated the relevance of using a synthetic dataset as a novel approach for hate speech classification, offering greater flexibility compared to traditional, outdated datasets commonly employed in literature. The experiments indicate that synthetic data circumvent limitations related to sensitive content and enable training on texts featuring highly negative or rarely occurring communication scenarios that are underrepresented in real-world datasets. Consequently, the resulting models exhibit improved effectiveness in detecting hate speech.

Second, the work investigated the impact of data type on model performance. The lowest accuracy (**52.04%**) was observed when models were trained on original data and evaluated on synthetically perturbated data. Training synthetically perturbated data and evaluating original data improved performance (**62.13%** accuracy). The highest performance (**72.68%** accuracy) was achieved when models were trained and evaluated on a combined dataset, regardless of the data's original or synthetically perturbated nature.

Neural networks consistently outperformed traditional machine learning algorithms. In particular, the **BiGRU + fastText** model achieved the best overall classification results, highlighting the effectiveness of bidirectional architectures, GRU units and fastText word embeddings. The first two experiments can be interpreted as involving zero-shot learning, suggesting that further performance improvements may require alternative architecture or larger datasets.

Finally, future work could focus on leveraging pre-trained large language models [30] on expanded synthetic datasets, as proposed in [7], to further enhance model performance.

REFERENCES

1. **Waseem Z., Hovy D.** Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. *Proceedings of the NAACL Student Research Workshop*, 2016, Pp. 88–93. DOI: 10.18653/v1/N16-2013

2. **Basina P.A., Gojko E.YU., Petrov E.Yu., Bakulin V.V.** Klassifikaciya publikacij soobshchestv "VKontakte" dlya ocenki kachestva zhizni naseleniya [Classification of publications of VKontakte communities for assessing the quality of life of the population]. *Komp'yuternaya lingvistika i intellektual'nye tekhnologii: po materialam ezhegodnoj mezhdunarodnoj konferencii "Dialog"* [*Computational linguistics and intelligent technologies: based on the materials of the annual international conference "Dialogue"*], 2022, Vol. 21 (C), Pp. 1001–1016.

3. **El Koshiry A.M., Eliwa E.H.I., El-Hafeez T.A., Omar A.** Arabic toxic tweet classification: Leveraging the AraBERT model. *Big Data and Cognitive Computing*, 2023, Vol. 4, No. 7, Art. no. 170. DOI: 10.3390/bdcc7040170

4. **Ribeiro A., Silva N.** INF-HatEval at SemEval-2019 Task 5: Convolutional neural networks for hate speech detection against women and immigrants on Twitter. *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, Pp. 420–425. DOI: 10.18653/v1/S19-2074

5. **Geet d'Sa A., Illina I., Fohr D.** Classification of hate speech using deep neural networks, *Revue d'Information Scientifique & Technique*, 2020, Vol. 25, No. 1, Art. no. hal-03101938.

6. **Smetanin S.I.** Toxic comments detection in Russian. *Computational Linguistics and Intellectual Technologies*, 2020, Vol. 26, No. 19, Pp. 1149–1159. DOI: 10.28995/2075-7182-2020-19-1149-1159

7. **Vidgen B., Thrush T., Waseem Z., Kiela D.** Learning from the worst: Dynamically generated datasets to improve online hate detection. *arXiv:2012.15761*, 2020. DOI: 10.48550/arXiv.2012.15761

8. **Bitton J., Pavlova M., Evtimov I.** Adversarial text normalization. *arXiv:2206.04137*, 2022. DOI: 10.48550/arXiv.2206.04137

9. **Hartvigsen T., Saadia G., Palangi H.** ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022, Vol. 1, Pp. 3309–3326. DOI: 10.18653/v1/2022.acl-long.234

10. **Saffari H., Shafiei M., Zhang H., Harris L., Moosavi N.S.** Beyond hate speech: NLP's challenges and opportunities in uncovering dehumanizing language, *arXiv:2402.13818*, 2024. DOI: 10.48550/arXiv.2402.13818

11. **Zhao X., Lu Z., Xu D., Yuan S.** Generating Textual adversaries with minimal perturbation. *arXiv:2211.06571*, 2022. DOI: 10.48550/arXiv.2211.06571

12. **Roth T., Gao Y., Abuadbba A., Nepal S., Liu W.** Token-modification adversarial attacks for natural language processing: A survey. *arXiv:2103.00676*, 2021. DOI: 10.48550/arXiv.2103.00676

13. **Wang B., Xu C., Liu X., Cheng Y., Li B.** SemAttack: Natural textual attacks via different semantic spaces. *arXiv:2205.01287*, 2022. DOI: 10.48550/arXiv.2205.01287

14. **Gutiérrez-Megías A., Jiménez-Zafra S.M., Ureña L.A., Martínez-Cámara E.** Smart lexical search for label flipping adversial attack. *Proceedings of the Fifth Workshop on Privacy in Natural Language Processing*, 2024, Pp. 97–106.

15. **Badri N., Kboubi F., Chaibi A.H.** Combining FastText and Glove word embedding for offensive and hate speech text detection. *Procedia Computer Science*, 2022, Vol. 207, Pp. 769–778. DOI: 10.1016/j.procs.2022.09.132

16. **Kosykh N.E., Molodkin I.A., Khomonenko A.D.** Features of text preprocessing for performing sentiment analysis. *Intellectual Technologies on Transport*, 2022, No. 3, Pp. 68–73. DOI: 10.24412/2413-2527-2022-331-68-73

17. **Zinovyeva E., Härdle W.K., Lessmann S.** Antisocial online behavior detection using deep learning. *Decision Support Systems*, 2019, Vol. 138, Art. no. 113362. DOI: 10.1016/j.dss.2020.113362

18. **Minaee S., Kalchbrenner N., Cambria E., Nikzad N., Chenaghlu M., Gao J.** Deep learning-based text classification: A comprehensive review. *ACM Computing Surveys* (*CSUR*), 2022, Vol. 54, No. 3, Art. no. 62. DOI: 10.1145/3439726

19. **Widiastuti N.I.** Convolution neural network for text mining and natural language processing. *IOP Conference Series: Materials Science and Engineering*, 2019, Vol. 662, No. 5, Art. no. 052010. DOI: 10.1088/1757-899X/662/5/052010

20. **Alkomah F., Ma X.** A literature review of textual hate speech detection methods and datasets. *Information*, 2022, Vol. 13, No. 6, Art. no. 273. DOI: 10.3390/info13060273

21. **Soni S., Chouhan S.S., Rathore S.S.** *TextConvoNet*: a convolutional neural network-based architecture for text classification. *Applied Intelligence*, 2023, Vol. 53, Pp.14249–14268. DOI: 10.1007/s10489-022-04221-9

22. **Georgakopoulos S.V., Tasoulis S.K., Vrahatis A.G., Plagianakos V.P.** Convolutional neural networks for toxic comment classification. *SETN '18: Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, 2018, Art. no. 35. DOI: 10.1145/3200947.320806

23. **Maslej-Krešňáková V., Sarnovský M., Butka P., Machová K.** Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification. *Applied Sciences*, 2020, Vol. 10, No. 23, Art. no. 8631. DOI: 10.3390/app10238631

24. **Budyl'skij D.V.** GRU i LSTM: sovremennye rekurrentnye nejronnye seti [GRU and LSTM: Modern Recurrent Neural Networks]. *Molodoj uchenyj* [*Young Scientist*], 2015, Vol. 95, No. 15, Pp. 51–54.

25. **Beniwal R., Maurya A.** Toxic comment classification using hybrid deep learning model. *Sustainable Communication Networks and Application*, 2021, Vol. 55, Pp. 461–473. DOI: 10.1007/978-981-15-8677-4_38

26. **Mironenko V.V., Saveleva A.A., Sodikov S.A.** Overview of methods for analysis of natural language based on machine learning models. *Aktual'nye problemy aviacii i kosmonavtiki* [*Current issues in aviation and astronautics*], 2021, Vol. 2, Pp. 169–170.

27. **Tan K.L., Lee C.P., Lim K.M.** RoBERTa-GRU: A hybrid deep learning model for enhanced sentiment analysis. *Applied Sciences*, 2023, Vol. 13, No. 6, Art. no. 3915. DOI: 10.3390/app13063915

28. **Ivanovs M., Kadikis R., Ozols K.** Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognition Letters*, 2021, Vol. 150, Pp. 228–234. DOI: 10.1016/j.patrec.2021.06.030

29. **Platonov E.N., Rudenko V.Y.** Identification and Classification of Toxic Statements by Machine Learning Methods. *Modelling and Data Analysis*, 2022, Vol. 12, No. 1, Pp. 27–48. DOI: 10.17759/mda.2022120103

30. **Kureichik V.V., Rodzin S.I., Bova V.V.** Deep learning methods for natural language text processing. *IzvestiyaSFedU. Engineering Sciences*, 2022, Vol. 226, No. 2, Pp. 189–199. DOI: 10.18522/2311-3103-2022-2-189-199

INFORMATION ABOUT AUTHORS / СВЕДЕНИЯ ОБ АВТОРАХ

Luc Prucell Mbele Ossiyi
Мбеле Оссийи Люк Прюсель
E-mail: lucprucell@gmail.com
ORCID: https://orcid.org/0009-0002-3090-2809

Pavel D. Drobintsev
Дробинцев Павел Дмитриевич
E-mail: drobintsev_pd@spbstu.ru

Sergey M. Ustinov
Устинов Сергей Михайлович
E-mail: usm50@yandex.ru
ORCID: https://orcid.org/0000-0003-4088-4798

Submitted: 15.01.2025; Approved: 11.04.2025; Accepted: 25.04.2025.
Поступила: 15.01.2025; Одобрена: 11.04.2025; Принята: 25.04.2025.