ISSN 2007-9737

Comparing Pre-Trained Language Model for Arabic Hate Speech Detection

Kheir Eddine Daouadi[1,*], Yaakoub Boualleg[1], Oussama Guehairia[2]

[1] Echahid Cheikh Larbi Tebessi University,
Laboratory of Vision and Artificial Intelligence,
Algeria

[2] Mohamed Khider University of Biskra,
Faculty of Sciences and Technology,
Algeria

{kheireddine.daouadi, yaakoub.boualleg}@univ-tebessa.dz, oussama.guehairia@univ-biskra.dz

Abstract. Today, the classification of hate speech in Arabic tweets has garnered significant attention from scholars worldwide. Although numerous classification approaches proposed in response to this interest, two primary challenges persist are reliance on handcrafted features and limited performance rates. This paper addresses the task of identifying Arabic hate speech on Twitter, aiming to deepen insights into the efficacy of novel machine-learning techniques. Specifically, we compare the performance of traditional machine learning-based approaches with state-of-the-art pre-trained language models based on Transfer Learning, as well as deep learning models. Our experiments, conducted on a benchmark dataset using a standard evaluation scenario, reveal several key findings. Firstly, multidialectal pre-trained language models demonstrate superior performance compared to monolingual and multilingual variants. Secondly, fine-tuning the pre-trained large language models significantly enhances the accuracy of hate speech classification in Arabic tweets. Our primary contribution lies in achieving promising results for the corresponding task through the application of multidialectal pre-trained language models trained on Twitter data.

Keywords. Arabic hate speech detection, fine-tuning, transfer learning, AraBERT.

1. Introduction

Nowadays, hate speech has garnered significant attention from scholars worldwide. Originally, this form of content was shared via conventional media outlets.

However, the global availability of the Internet, facilitated by social media like Twitter, YouTube, and Facebook, has led to an exponential increase in users expressing their opinions and sharing posts. Regrettably, these posts can occasionally exert adverse psychological impacts on social media users, with extreme cases even resulting in instances of suicide [2]. The proliferation of unregulated text on social media represents a concerning phenomenon, particularly when such content contains hate speech. The European Union has adopted a legislative approach to address this issue.

Specifically, the Commission of the European Union has exerted pressure on numerous social media platforms to adopt a hate speech code. As part of this code, platforms have committed to reviewing the 'notifications for elimination of hate speech' within a 24-hour and facilitating direct notification to law enforcement agencies.

However, fulfilling this pledge proves challenging owing to the missed of clarity regarding the precise scope of hate speech, stemming from inadequate data collection and systematic reporting mechanisms.

Consequently, platforms often rely on their user communities to identify and report instances of hateful speech.

This task poses significant complexity for social media platforms. Given the vast volume of data shared daily, coupled with the absence of efficient automated systems, the community of natural

language processing is motivated to undertake research into hate speech detection.

Additionally, there are a significant demand for study focused on language than English [3]. Today, researchers are leveraging Twitter data to propose various approaches for Arabic hate speech classification.

However, two primary challenges persist reliance on handcrafted features and limited performance rates. Automatic classification of Arabic hate speech using conventional learning algorithms like Support Vector Machine (SVM), and Naïve Bayes (NB) has demonstrated acceptable results.

Nevertheless, they rely on handcrafted features derived using pre-defined methods like Term Frequency-Inverse Document Frequency (TF-IDF), Bag of Word (BoW), and Term Frequency (TF). Recently, Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and Convolution Neural Network (CNN) have already shown promising results. However, they depend on some pre-defined word embedding models like AraBERT, Mazajk, and AraVec.

This paper offers a comparative examination of various machine-learning methodologies for the classification of Arabic hate speech on Twitter. We evaluate the classification models using a benchmark dataset that contains tweets annotated for hate speech classification.

The major contributions of this work are briefly noted as follows:

- We evaluate three suggested DL-based approaches (Bi-LSTM, LSTM, and CNN) along with traditional machine learning models (SVM and NB).
- We compare the accuracy results of the recent pre-trained language model utilizing transformer mechanisms. Including multi-lingual ones (XLM and BERT), a mono-lingual model (AraBERT), and a multi-dialectal model (AraBERT-Twitter).
- We compare the performance of the transformers-based model with our baseline.

The rest of this manuscript is structured as follows. Section 2 presents the related works. In Section 3, we discuss the data and methodology. In Section 4, we focus on the experiments and evaluation results. In Section 5, we discuss our main contribution. In Section 6, we conclude the paper.

Table 1. Overview of our interested dataset

Parameters	Value
Tweet counts	11634
Words Counts	138.3 K
Unique words	37.9 K
Average words per tweet	11.9

2. Related Works

The emergence of the Twitter platform has encouraged a multitude of research avenues including topic detection [4], organization detection [5, 6], and bot detection [6, 7]. Thanks to their importance, Arabic hate speech classification has garnered significant attention from scholars worldwide.

Numerous methods and systems have been suggested to tackle this challenging classification task. They follow two major approaches: a traditional based approach and a deep learning-based approach.

2.1 Traditional Approaches

In this scenario, conventional classification methods depend on feature engineering, wherein texts are transformed into feature vectors before classification using standard algorithms like SVM and NB. Examples of conventional approaches are outlined briefly. The authors in [8] underscore the significance of utilizing datasets from multiple platforms to enhance the generalizability of the classifier in detecting offensive language.

They experimented with SVM and TF-IDF, and achieved F1 score of 84%. Besides, authors in [9] explore the influence of preprocessing steps on offensive language and hate speech classification. They demonstrate that thorough preprocessing techniques have notable effects on detection rates. The best experimental outcomes were achieved using SVM and BoW, attaining F1 scores of 95%

Table 2. Optimized values of hyperparameters explored in DL models

Parameters	CNN	LSTM	BiLSTM
Size	100	75	75
Dropout	0.25	0.25	0.5
Activation	tanh	relu	relu
Optimizer	Adam	Adam	Adam
Batch size	8	32	64
Learning rate	0.01	0.002	0.002

and 89% for hate speech and offensive language, respectively.

Likewise, authors in [10] use the Arabert embedding with Deep Forest, the best experimental results showed acceptable macro-averaged and weighted-average F1-score results of 63% and 80%, respectively.

Furthermore, authors in [11] employ BoW and TF-IDF to categorize tweets as offensive or normal. Their findings indicate that ensemble classifier (Bagging) outperforms single classifier, achieving F1 score of 0.88. In similar work, the authors in [12] categorize tweets into those of normal, hate, and abusive.

They utilize NB and SVM classifiers with trigrams, bigrams, and unigrams. The best experimental outcomes are achieved with NB, resulting F1 score of 0.896 and 0.744% for (abusive and hate vs. normal) and (normal vs. hate vs. abusive) tasks, respectively.

In a distinct approach, researchers in [13] use Social Graph, tweet-based and profile features to differentiate non-abusive Twitter accounts from abusive ones. The best-achieved F1 score was 85% using the NB classifier.

2.2 Deep Learning Approaches

In this context, these methodologies utilize a neural network capable of automatically learning representations of input tweet texts by varying the level of the abstraction. These learned representations are then leveraged to execute the classification task. Commonly employed embedding models include Mazajk and AraVec.

The prevailing DL architectures utilized for Arabic hate speech classification encompass BERT, LSTM, and GRU. Below, we briefly outline some examples of DL approaches. Authors in [14] categorize tweets into religious, general-hate, racial, sexism, or normal.

They employ an embedding layer randomly initialized to learn the word embedding from the training data. The best experimental outcomes are achieved using the Hybrid CNN-LSTM model, resulting in an F1 score of 73%. In a similar study, the authors in [15] evaluate two AraVec models to categorize tweets into normal and hateful. The experiment with Hybrid CNN-LSTM achieved F1 score results of 71.68%.

Likewise, authors in [16] explore the influence of word embedding and neural networks on the performance rates across various classification tasks. They train multiple embedding models and subsequently employ these models to train several neural networks for different classification task.

The best experimental outcomes are observed with Skip-gram and CNN, resulting in F1 scores of 70.80%, 75.16%, and 87.22% for the 6-class, 3-class, and 2-class classifications, respectively. Besides, the authors in [18] use ensemble CNN and Bidirectional LSTM (BiLSTM) classifiers based on the AraBERT. The best outcome is obtained using the average-based ensemble approach, yielded F1 score of 80.23%(BiLSTMs), 84.01% (CNNs), and 91.12% (CNNs) for 6 class, 3 class, and binary classification tasks, respectively.

In a similar, authors in [17] use AraVec and AraBERT to categorize tweets as being normal, abusive, or hateful. The best performance was achieved using CNN, yielding F1 score of 0.721. In a similar, the authors in [19] use CNN with Multilingual BERT embedding model, yielding F1 score of 75.51%, 78.9%, and 87.03% for 6 class, 3 class, and binary classification tasks, respectively.

Furthermore, authors in [20] utilize a bidirectional GRU enhanced by an attention layer alongside the AraVec to identify offensive language and hate speech. Moreover, they examine the effect of different oversampling techniques and pre-processing techniques on the performance results.

The best outcomes consist of F1 score results of 0.859% and 0.75 for offensive and hate speech, respectively.

In a distinct strategy, researchers in [21] fine-tune the pre-trained AraBERT [47] for classifying

tweets being offensive, vulgar, hate speech, or clean. They achieved an F1 score result of 83.2%.

2.3 Gaps and Contributions

After reviewing the current studies, we can realize that some current pre-trained language models have not yet been evaluated for hate speech classification from Arabic tweets. Moreover, there is no existing study where Monolingual, Multilanguage, and Multidialectal pre-trained language models based on transformers are compared to demonstrate their validity for classifying Arabic hate speech. In this work, we rely on transfer learning models due to their major advantages:

They can capture long-term dependency in language, while it does not need a large dataset. Additionally, we conduct a comparative analysis between deep learning approaches and conventional approaches as our baselines.

3. Data and Methodology

2.4 Dataset Description

The dataset used in this paper was published in [14]. The basic characteristics of the corresponding dataset are presented in Table. 1. The tweets were collected using a curated list of hashtags known to elicit hateful content on Twitter. Subsequently, the retrieved tweets were manually annotated. The racial hate speech class constitutes a minor subset of the tweets, whereas the majority belong to the non-hate class.

2.5 Features Representation

The efficacy of a classification system depends on how it represents the text. Specifically, for tasks such as tweet classification, it is essential to convert the tweet's textual content into an appropriate representation for learning a classifier.

Hence, in this work, we adopt three distinct representations, which are outlined briefly below. The Bag of Words (BoW) [22] method stands as one of the foremost techniques employed for information retrieval. BoW centers on counting the occurrences of words within a given text corpus.

Table 3. Optimized hyperparameter for the pre-trained language models (E=Epochs, BS=Batch Size, LR=Learning Rate)

Model	E	BS	LR
xlm-roberta-base	10	16	$3e^{-5}$
xlm-roberta-large	5	64	$1e^{-5}$
bert-base-arabic	3	8	$4e^{-5}$
bert-large-arabic	2	16	$1e^{-5}$
bert-base-arabert	4	8	$2e^{-5}$
bert-large-arabert	5	8	$1e^{-5}$
base-multilingual-cased	5	8	$1e^{-5}$
multi-dialect-bert-base-arabic	4	8	$3e^{-5}$
albert-base-arabic	3	8	$2e^{-5}$
albert-large-arabic	3	8	$1e^{-5}$
base-arabertv02-twitter	4	16	$1e^{-5}$
large-arabertv02-twitter	2	16	$1e^{-5}$

This approach generates a vocabulary comprising unique words found across all tweets and utilizes these as feature vectors to indicate the absence or presence of such words within the vocabulary. Term Frequency Inverse Document Frequency (TF–IDF) weighting scheme that combines Inverse Document Frequency (IDF) with Term Frequency (TF).

This technique is commonly used for Text Mining and Information Retrieval, which converts the tweet to a matrix of integer producing sparse matrices of the counts [23]. Word Embedding (WE) [24, 25] stands as an effective technique that has seen considerable success in recent years. A feature vectors space consists of unsupervised word embedding vectors.

These vectors represent the semantic spaces of each word in a real-valued space. Word embedding vectors offer a dense representation of word meaning, where the word is characterized as a real-valued features vector. Word embedding models can be produced using static pre-trained models like word2vec [25], GloVe [26], and fastText [27], or by employing contextual pre-trained embedding models like BERT [28].

Table 4. F1 score results of both traditional and DL classifiers

Model	Macro	Weighted
NB-BoW	43.47	70.98
SVM-BoW	48.77	73.07
NB-TF_IDF	28.68	65.02
SVM-TF_IDF	48.29	72.88
NB-AraVec	22.97	61.52
SVM- AraVec	38.78	63.29
CNN-AraVec	50.49	73.57
LSTM-AraVec	49.72	73.26
BiLSTM-AraVec	51.54	74.13

2.6 Model Description

This subsection presents the classification models. We outline the conventional learning systems, the deep learning architectures, and the transfer learning models we have used.

Traditional machine learning. We evaluate two models most commonly used the Multinomial Naïve Bayes and Support Vector Machine, which predict classes based on a combination of features.

Support Vector Machine (SVM) [29] most well-known classifiers since it is highly accurate and effective in text classification. This classifier offers the advantage of typically performing well even when trained with a limited amount of data [30]. For the hyperparameter optimization, we experiment with various values: 'C' = [1, 0.01, 10, 0.1], 'class_weight' = [balanced, None], 'penalty' = [l2, l1]. Following the optimization, we utilize the linear SVM classifier with its default configuration.

Multinomial Naïve Bayes (MNB) [31] is one of the most well-known classifiers since it is highly accurate and effective in text classification.

It operates by considering the frequency of such word to generate in a multinomial fashion the data distribution. For the hyper-parameter optimization, we evaluate the values of: 'Alpha' = [0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1] and 'fit_prior' = [True, False]. After optimization, the default parameters of the classifier were used. To derive feature vectors for inclusion for such classifiers, we utilize the three types of tweet representation previously discussed (i.e. Word Embedding, TF-IDF, and BoW).

Deep Learning. The deep learning methods explored in this study are briefly described in the following. Convolution Neural Network (CNN) stands as the most effective neural network model, offering an alternative approach to traditional feedforward neural networks. In CNN architecture, different layers are sparsely connected, linking a local region of an input layer with neurons in the subsequent layer.

The work by [32] was the first that applied CNN to text classification, wherein words are transformed into numerical values via word embedding. A 2-dimensional matrix is formed from the tweet text, where each row is a word vector in that tweet. The typical CNN architecture encompasses several stages, including a fully connected layer, a pooling layer, and a convolutional layer.

Recurrent Neural Network (RNN) [33] is another class of neural network to address the challenge of sequential learning faced by the conventional neural network. The connections among nodes construct the directed graph over the temporal sequences, enabling the model to highlight the dynamic temporal behaviors.

Long Short-Term Memory (LSTM) represents the most widely recognized variant, as introduced by [34], and trained via backpropagation through time. LSTM networks are equipped with memory blocks, enabling them to learn the temporal sequence and their long-terms dependency effectively. On the other hand, Bi-directional Long Short-Terms Memory (BiLSTMs) facilitate the two-way information flow. This architecture involves training two LSTM network simultaneously, one for the forward and one for the reverse direction [35].

In this scenario, we use word embedding as a feature representation. We particularly use Aravec [36], which consists of 300-dimensional vectors for each word.

Transfer Learning. In this scenario, the process is to adjust a pre-trained language model to a new dataset through the transfer of the learned features. In other words, a technique to improve learning of a new task by transferring knowledge from the learned task [37].

The transformer operates as an attention mechanism, enabling the learning of contextual relationships between words within a text. It includes two primary components: the encoder, which processes the textual input; and the decoder generates estimates for the corresponding task [38].

Unlike directional models that sequentially process textual input (e.g., right-to-left or left-to-right), the encoder of the transformer simultaneously processes the entire sequence.

This approach enables model to capture the word context based on their surrounding context as a whole. The authors in [38] achieved an enhancement in the translation task with the use of the attention mechanism avoiding relying on RNN, paving the way for additional transformer architectures.

Bidirectional Encoder Representations from Transformers (BERT) is the first transformer-based language model introduced by Google. The model is pre-trained on large unsupervised text data based on two self-supervision tasks:

Masked Language Modeling and Next Sentence Prediction. In the first task, approximately 0.15 of the words in such sentences were masked at random, and the model forecast the masked words.

The second task involves the classification of two sentences, wherein the model was tasked with discerning the original orders between the two sentences, thereby enhancing document-levels understanding.

Alternatively, the authors in [39] proposed a cross-lingual language model refer as XLM, improving BERT while attaining remarkable achievements across different machine translation and cross-lingual classification tasks.

Unlike BERT, which is not adjusted for multi-lingual tasks due to limited shared vocabulary across languages, XLM tackles this challenge by processing all languages using a shared vocabulary generated based on a preprocessing method called Byte Pair Encoding [40, 41]. Additionally, XLM uses the dual-language training mechanism alongside BERT so as to learn inter-language word relationship effectively.

Table 5. F1 score results of pre-trained language models before fine-tuning

Model	Macro	Weighted
xlm-roberta-base	31.64	67.13
xlm-roberta-large	29.03	65.99
bert-base-arabic	51.90	74.96
bert-large-arabic	52.54	74.82
bert-base-arabert	54.06	75.74
bert-large-arabert	53.98	75.22
base-multilingual-cased	47.12	72.57
multi-dialect-bert-base-arabic	**63.38**	**80.85**
albert-base-arabic	57.75	78.10
albert-large-arabic	59.27	78.59
base-arabertv02-twitter	57.23	77.84
large-arabertv02-twitter	64.07	80.82

3 Experiment and Evaluation

This section outlines the experimental procedures and the evaluation conducted to assess the efficacy of the pre-trained language model. By conducting experiments on a recently established benchmark Twitter dataset, aiming to address these research questions:

- **RQ1:** Can a multi-dialectal pre-trained language model, based on Twitter data, improve hate speech detection accuracy in Arabic tweets?
- **RQ2:** Does fine-tuning a pre-trained language model enhance hate speech detection accuracy in Arabic tweets?

First, we present the pre-processing step we have applied to the chosen datasets as well as the hyperparameters used for DL architectures and transfer learning models. Then, we discuss the evaluation metrics and finally present the achieved performance results.

3.1 Tweet Preprocessing

The preprocessing stage plays a pivotal role in natural language processing systems, particularly

in text classification [42]. They contain elongated words, hashtags, user mentions, and expressions that make tokenization difficult. To mitigate these challenges, we implement the following steps:

- Removing tweet features: This involves eliminating user URLs, mentions '@', hashtag symbols '#', punctuation, the word "RT", special characters (emoticons), and numerical characters.
- Removing non-Arabic letters, Arabic stop words, diacritics and new lines.
- Eliminating repeated characters: like (مرحباااااا) which means "Hellooooooo", to be (مرحبا), which is "Hello".
- Arabic letters standardization:
  - The letter (Taa Marbouta) (ة), which can be mistaken and written as (ه), we standardize it to (ه).
  - The Letter (Alef) (أ), which has the following forms (ا-آ-إ-أ), all the four letters were standardized into (ا).
  - The Arabic dash that is used for expanding words like in (مرحبـــا) to be (مرحبا).
  - The Letter (Alef Maqsora) (ى) has been standardized to (ي).

3.2 Hyperparameter Optimization for DL Models

The DL architectures evaluated in this work contain numerous hyperparameter, which necessitate estimation to achieve optimal results. To achieve this, we used the performance of a validation dataset to select the most suitable hyperparameter for the test dataset. For the hyperparameter optimization, we conduct 10-folds cross-validation using the corresponding dataset.

We employ the test data to make predictions while evaluating the predictions based on the optimized hyperparameters.

Table 2 illustrates the optimal hyperparameter for the corresponding model (LSTM, CNN, and BiLSTM). To avoid over-fitting during the supervised training of a neural network, we utilize early stopping by ending the training procedure before the converging of the weights.

Table 6. F1 score results of pre-trained language models after fine-tuning

Model	Macro	Weighted
xlm-roberta-base	52.57	74.57
xlm-roberta-large	48.98	72.74
bert-base-arabic	56.43	76.71
bert-large-arabic	55.30	75.68
bert-base-arabert	56.59	76.89
bert-large-arabert	57.05	76.65
base-multilingual-cased	48.91	73.26
multi-dialect-bert-base-arabic	65.95	81.86
albert-base-arabic	61.26	79.20
albert-large-arabic	62.21	79.74
base-arabertv02-twitter	**70.76**	**84.69**
large-arabertv02-twitter	69.71	84.45

3.3 Transfer Learning Fine-Tuning

The transformers-based models used in this work are pre-trained trained based on formal general corpora (Arabert, XLM-RoBERTa, and Multilanguage-BERT) and based on informal corpora (i.e. AraBERT-Twitter).

Thus, it is important to study the contextual information derived from the pre-trained layers while fine-tuning it for our interested downstream task. The fine-tuning consists of updating weights using the annotated dataset. BERT takes a sequence of 512 tokens as input and outputs 12 self-attention heads and a 768-dimensional vector.

For the optimization, we use the Adam optimizer [44,45] which performs well for natural language processing and the BERT model specifically. Additionaly, we evaluate other multilingual model, we chose the xlm-roberta-base and xlm-roberta-large checkpoints which include 100 languages.

For the purposes of fine-tuning, authors in [46] have recommended choosing from the values of the following parameters: number of epochs, batch size, learning rate, and maximum sequence. We fine-tuned the corresponding models by evaluating different parameters as presented in Table 3. We

Table 7. Comparison of F1 score results with the latest state-of-the-art classification approaches (N=Non-hate speech, S=Sexism, Re=Religious, Ra=Racial, M=Macro-averaged)

	N	S	Re	G	Ra	M
A	0.85	0.21	0.10	0.12	0.09	0.27
B	0.86	0.42	0.50	0.25	0.24	0.45
C	0.87	0.41	0.54	0.30	0.29	0.48
D	0.86	0.43	0.59	0.31	0.25	0.49
E	0.85	0.42	0.56	0.45	0.20	0.50
F	0.87	0.50	0.64	0.49	0.27	0.55
G	0.86	0.44	0.59	0.47	0.22	0.52
H	0.87	0.37	0.47	0.28	0.26	0.45
I	0.87	0.49	0.56	0.47	0.24	0.53
J	0.88	0.49	0.67	0.41	0.37	0.57
K	0.89	0.50	0.73	0.42	0.32	0.57
O	**0.92**	**0.70**	**0.80**	**0.61**	**0.50**	**0.71**

set the maximum sequence length as 64 for all the experiments.

3.4 Evaluation Metrics

To assess the effectiveness of the corresponding approaches, we will use various evaluation measures capable of accurately assessing the model's performance. Given that hate speech classification poses the imbalanced learning challenge, we will particularly emphasize the Macro-average and Weighted-average metrics to compute comprehensive performance metrics.

These metrics are presented as follow: Precision (P), referred to as positive predictive value, indicates the proportion of correctly classified positive instances out of all instances classified as positive. For example, the Precision of the Normal class is estimated as follows:

$$P_{Normal} = CC_{Normal} / TC_{Normal}, \quad (1)$$

where CC_{Normal} is the Correctly Classified as Normal and TC_{Normal} is the Total Classified as Normal. Recall (R) (also known as sensitivity), is the division of correctly predicted positive instances to the total positive instances. For example, the Recall of the Normal class is calculated as follows:

$$R_{Normal} = CC_{Normal} / TN_{Normal}, \quad (2)$$

where CC_{Normal} is the number of Correctly Classified as Normal, and TN_{Normal} is the Total number of Normal instances. F1 measure (F1) is the harmonic mean between the Precision and Recall. For example, the F1 of the Normal class is estimated as follows:

$$F1_{Normal} = 2(P_{Normal} \times R_{Normal}) / (P_{Normal} + R_{Normal}). \quad (3)$$

3.5 Results Analysis

In this subsection, we explore the achieved results of the approaches we evaluated. We employed the tenfold cross-validation method to evaluate performance metrics. This method involved splitting the dataset into 10 equally sized parts while maintaining a balanced representation of each class from the original dataset. One part was designated for testing, while the remaining parts were utilized for training.

This process was repeated 10 times, and the performance metric scores were averaged across the 10 iterations of cross-validation. Table 4 presents the prediction performances attained for both conventional machines leaning and DL classifiers. We utilized word embedding as an input feature vector in all models.

In the case of traditional machine learning approaches, also we tested also the statistical-features BoW and TF–IDF including SVM-BoW, SVM-TF-IDF, MNB-BoW, and MNB-TF–IDF. In most traditional classifiers, BoW achieves better performance results than the TF-IDF and word embedding features.

The baseline experiments (DL and traditional learning approaches) did not perform satisfactorily due to an insufficient number of training instances. The DL-based approaches achieve better performance results than conventional machine learning approaches.

These findings are in line with the majority of related works, where DL based approaches found to have comparable accuracy results to traditional learning algorithms on the corresponding task.

Table 8. The number of misclassified tweets by best pre-trained language model. (TE = Total Error, Q = All in Common)

	N	S	Re	G	Ra	TE
X	604	201	162	571	247	1785
Y	631	240	185	535	203	1794
W	890	183	166	536	176	1951
Z	713	240	160	545	243	1865
Q	334	45	30	160	57	726

As highlighted in Table 5, the pre-trained language model before fine-tuning substantially outperforms the baseline systems. It is important to note that multidialectal-based transformer models achieve better results than monolingual and multilingual-based models, yielding F1 score results of 63.38% and 80.85% for Macro averaged and Weighted averaged, respectively.

Table 6 shows the accuracy results of the pre-trained language model after fine-tuning. It is important to note that the fine-tuning of the pre-trained language model improves the accuracy results. Additionally, the multidialectal-based models outperform monolingual and multilingual ones, yielded F1 score results of 70.76% and 84.69% for Macro averaged and Weighted averaged, respectively.

Subsequently, we compare the performance of fine-tuned multidialectal-based model base-arabertv02-twitter (O) with three existing conventional machine learning approaches:

a. [12], which leveraged N-grams with NB.
b. [11], which combine TF–IDF and BoW with Bagging.
c. [8], which use TF-IDF with SVM, and seven existing DL-based approaches, namely.
d. [14], that learned word Embedding using the training data and use the hybrid CNN-LSTM for classification.
e. [19], Multilingual BERT embedding model with CNN.
f. [20], which use AraVec and bidirectional GRU augmented with attention layer.
g. [17], Which used AraBERT and AraVec embedding with CNN.
h. [15], which used AraVec and hybrid CNN-LSTM.
i. [18], which used AraBERT embedding and ensemble CNNs.
j. [21], which fine-tuned the pre-trained AraBERT language model.
k. [47], which use base-arabertv02-twitter without fine tuning. When comparing our approach with other state-of-the-art classifiers presented in Table 7, our model exhibits the highest Accuracy.

Unlike CNN and LSTM, our method does not necessitate a substantial quantity of labeled dataset to achieve promising performance result; this is a common requirement in many DL approaches. Moreover, in contrast to NB and SVM, our model eliminates the need to extract and design handcrafted features.

Given the nature of social media data, characterized by frequent usage of slang, abbreviations, and informal language, our method effectively processes the input words while considering their contextual surroundings.

Experimental results demonstrated that our proposition outperforms existing approaches by a difference between [8% and 21%] and [14% and 44 %] in weighted averaged and macro averaged F1 scores respectively. Comparing these results, we highlight the significance of multidialectal-based models trained on Twitter data since those models achieve the best results.

Furthermore, we highlight the significance of parameter tuning to discover the optimal hyperparameter values. Subsequently, we conduct an error analysis on the best pre-trained language models. For each model, we scrutinize tweets that were misclassified.

Furthermore, we examine tweets that were misclassified by all four models. Table 8 showed the number of tweets misclassified by such pre-trained language model.

The four best accurate models (bert-base-arabertv02-twitter (X), bert-large-arabertv02-twitter (Y), bert-large-arabic (W), and bert-base-arabert (Z)) predicted the same wrong labels 726 times out of 5217.Regarding the best system bert-base-

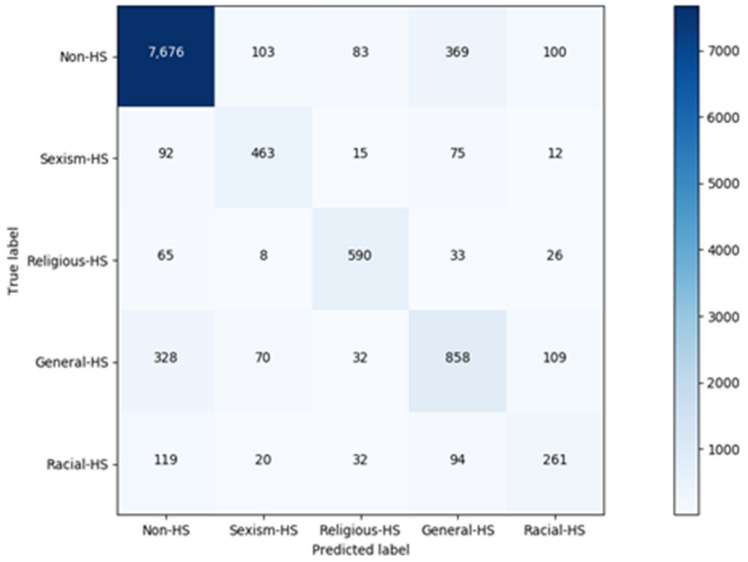

Fig. 1. Confusion matrix of our proposition

arabertv02-twitter, the Non-HS instances mislabeled are biased towards General-HS; the Sexism-HS, Religious-HS, Racial-HS, and General-HS instances mislabeled are biased towards Non-HS. Fig. 1 shows the confusion matrix of the optimized system.

4 Discussion

Although quite effective, current Arabic hate speech classification approaches are costly, as they need a huge number of labeled tweets to attain promising accuracy results. The tweets labeling process is very expensive and labor-intensive while hindering the deployment of artificial intelligence systems in the industry.

In contrast, our proposition does not require a huge number of labeled datasets. Furthermore, the machine-learning approaches use hand-crafted features, which have confronted data sparseness and the curse of dimensionality. Conversely, ours automatically learn features from the textual data.

As can be shown in Table 4, the minor performance results are obtained using the traditional learning classifiers followed by the deep learning classifiers, while the major accuracy results go for transfer learning-based classifiers. In Tables 5 and 6, we can notice that the fine-tuning of pre-trained language models improves the accuracy of results.

Furthermore, the multidialectal pre-trained language models based on Twitter data outperform monolingual and multilingual ones. In Table 7, we can notice that our proposition outperformed the latest state-of-the-art. The lower performance results are observed for the racial class, and the higher performance are obtained for the normal class. This disparity can be attributed to the significant class label imbalance present in the dataset.

6 Conclusion

Today, the detection of hate speech from Arabic tweets has garnered significant attention from scholars worldwide.

In this paper, we evaluate Arabic hate speech classification by utilizing transfer learning based on a pre-trained language model. We conducted extensive experiments following three approaches: two conventional machine-learning approaches, three DL approaches, and twelve transfer-learning approaches.

The results achieved by the transfer learning approaches outperform traditional and deep learning models utilized in this work. The major

contribution of this work is the evaluation of the recent pre-trained language models for Arabic hate speech classification. Specifically, we differentiate the performance of multilingual models with monolingual and multidialectal ones.

Experimental results show that the multidialectal models trained on Twitter data outperformed monolingual and multilingual models trained on general data. In our future work, we intend to pursue various avenues. Primarily, we aim to refine the contextual embedding model, with a focus on adapting its vocabulary for the hate speech classification task. A costlier technique could be to consider training a novel AraBERT model that is customized for Arabic hate speech classification.

Additionally, we intend to evaluate various data augmentation approaches to overcome the challenges of imbalanced data. From a research standpoint, we will utilize our proposed systems to examine Arabic Twitter discussions on various subjects to determine the extent of hate speech conversations with public discourse and to understand how their capabilities and sophistication evolve.

References

1. **Council of Europe (2016).** ECRI general policy recommendation on combating hate speech. European Commission against Racism and Intolerance General Policy Recommendation, No. 15. hudoc.ecri.coe.int/eng?i=REC-15-2016-015-ENG.
2. **Hinduja, S., Patchin, J. W. (2010).** Bullying, cyberbullying, and suicide. Archives of Suicide Research, Vol. 14, No. 3, pp. 206–221. DOI: 10.1080/13811118.2010.494133.
3. **Fortuna, P., Nunes, S. (2018).** A survey on automatic detection of hate speech in text. ACM Computing Surveys, Vol. 51, No. 4, pp. 1–30. DOI: 10.1145/3232676.
4. **Daouadi, K. E., Rebaï, R. Z., Amous, I. (2021).** Optimizing semantic deep forest for tweet topic classification. Information Systems, Vol. 101, pp. 101801. DOI: 10.1016/j.is.2021.101801.
5. **Daouadi, K. E., Rebai, R. Z., Amous, I. (2018).** Organization vs. individual: Twitter user classification. Conference on Language Processing and Knowledge Management, pp. 1–8.
6. **Daouadi, K. E., Rebaï, R. Z., Amous, I. (2019).** Organization, bot, or human: towards an efficient twitter user classification. Computación y Sistemas, Vol. 23, No. 2, pp. 273–279. DOI: 10.13053/cys-23-2-3192.
7. **Daouadi, K. E., Rebaï, R. Z., Amous, I. (2019).** Bot detection on online social networks using deep forest. Artificial Intelligence Methods in Intelligent Algorithms, Vol. 985, pp. 307–315. DOI: 10.1007/978-3-030-19810-7_30.
8. **Chowdhury, S. A., Mubarak, H., Abdelali, A., Jung, S., Jansen, B. J., Salminen, J. (2020).** A multi-platform arabic news comment dataset for offensive language detection. Proceedings of the 12th Language Resources and Evaluation Conference, pp. 6203–6212.
9. **Husain, F. (2020).** OSACT4 shared task on offensive language detection: Intensive preprocessing-based approach. Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pp. 53–60.
10. **Daouadi, K. E., Boualleg, Y., Guehairia, O. (2023).** Deep random forest and araBert for hate speech detection from Arabic tweets. JUCS - Journal of Universal Computer Science, Vol. 29, No. 11, pp. 1319–1335. DOI: 10.3897/jucs.112604.
11. **Husain, F. (2020).** Arabic offensive language detection using machine learning and ensemble machine learning approaches. DOI: 10.48550/ARXIV.2005.08946.
12. **Mulki, H., Haddad, H., Ali, C. B., Alshabani, H. (2019).** L-HSAB: A Levantine twitter dataset for hate speech and abusive language. Proceedings of the Third Workshop on abusive Language Online. DOI: 10.18653/v1/w19-3512.
13. **Abozinadah, E. A., Jones, J. H. (2017).** A statistical learning approach to detect abusive twitter accounts. Proceedings of the

International Conference on Compute and Data Analysis, pp. 6–13. DOI: 10.1145/3093241.3093281.

14. **Al-Hassan, A., Al-Dossari, H. (2021).** Detection of hate speech in Arabic tweets using deep learning. Multimedia Systems, Vol. 28, No. 6, pp. 1963–1974. DOI: 10.1007/s00530-020-00742-w.
15. **Faris, H., Aljarah, I., Habib, M., Castillo, P. (2020).** Hate speech detection using word embedding and deep learning in the Arabic language context. Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods, pp. 453–460. DOI: 10.5220/0008954004530460.
16. **Alsafari, S., Sadaoui, S., Mouhoub, M. (2020).** Effect of word embedding models on hate and offensive speech detection. arXiv. DOI: 10.48550/ARXIV.2012.07534.
17. **Alghanmi, I., Anke, L. E., Schockaert, S. (2020).** Combining BERT with static word embeddings for categorizing social media. Proceedings of the 6th Workshop on Noisy User-generated Text. DOI: 10.18653/v1/2020.wnut-1.5.
18. **Alsafari, S., Sadaoui, S., Mouhoub, M. (2020).** Deep learning ensembles for hate speech detection. IEEE 32nd International Conference on Tools with Artificial Intelligence, pp. 526–531. DOI: 10.1109/ictai50040.2020.00087.
19. **Alsafari, S., Sadaoui, S., Mouhoub, M. (2020).** Hate and offensive speech detection on Arabic social media. Online Social Networks and Media, Vol. 19, pp. 100096. DOI: 10.1016 /j.osnem.2020.100096.
20. **Haddad, B., Orabe, Z., Al-Abood, A., Ghneim, N. (2020).** Arabic offensive language detection with attention-based deep neural networks. Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pp. 76–81.
21. **Mubarak, H., Rashed, A., Darwish, K., Samih, Y., Abdelali, A. (2021).** Arabic offensive language on twitter: analysis and experiments. Proceedings of the 6th Arabic Natural Language Processing Workshop, pp. 126–135.
22. **De-Sousa-Pereira, A. B., Firmino-Alves, A. L., De-Oliveira, M. G., Baptista, C. D. (2018).** Using supervised classification to detect political tweets with political content. Proceedings of the 24th Brazilian Symposium on Multimedia and the Web, pp. 245–252. DOI: 10.1145/3243082.3243113.
23. **Garreta, R., Moncecchi, G. (2013).** Learning scikit-learn: Machine learning in python. Packt Publishing Ltd.
24. **Yang, X., Macdonald, C., Ounis, I. (2017).** Using word embeddings in twitter election classification. Information Retrieval Journal, Vol. 21, No. 2-3, pp. 183–207. DOI: 10.1007/s10791-017-9319-5.
25. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013).** Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, Vol. 26, pp. 3111–3119.
26. **Pennington, J., Socher, R., Manning, C. (2014).** Glove: global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.
27. **Joulin, A., Grave, E., Bojanowski, P., Mikolov, T. (2017).** Bag of tricks for efficient text classification. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Vol. 2, pp. 427–431.
28. **Devlin, J., Chang, M., Lee, K., Toutanova, K. (2019).** BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.
29. **Cortes, C., Vapnik, V. (1995).** Support-vector networks. Machine Learning, Vol. 20, No. 3, pp. 273–297. DOI: 10.1007/bf00994018.
30. **Ruiz, A. M., Cornet, A., Shimanoe, K., Morante, J. R., Yamazoe, N. (2005).** Effects

of various metal additives on the gas sensing performances of tio2 nanocrystals obtained from hydrothermal treatments. Sensors and Actuators B: Chemical, Vol. 108, No. 1-2, pp. 34–40. DOI: 10.1016/j.snb.2004.09.045.

31. **Mccallum, A., Nigam, K. (1998).** A comparison of event models for Naive Bayes text classification. AAAI Conference on Artificial Intelligence, pp. 41–48.
32. **Kim, Y. (2014).** Convolutional neural networks for sentence classification. Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751. DOI: 10.3115/v1/D14-1181.
33. **Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S. (2010).** Recurrent neural network based language model. Proceedings of the 11th Interspeech, pp. 1045–1048. DOI: 10.21437/Interspeech. 2010-343.
34. **Hochreiter, S., Schmidhuber, J. (1997).** Long short-term memory. Neural Computation, Vol. 9, No. 8, pp. 1735–1780. DOI: 10.1162/ neco.1997.9.8.1735.
35. **Schuster, M., Paliwal, K. (1997).** Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, Vol. 45, No. 11, pp. 2673–2681. DOI: 10.1109/78.650093.
36. **Soliman, A. B., Eissa, K., El-Beltagy, S. R. (2017).** AraVec: A set of Arabic word embedding models for use in Arabic NLP. Procedia Computer Science, Vol. 117, pp. 256–265. DOI: 10.1016/j.procs.2017.10.117.
37. **Torrey, L., Shavlik, J. (2010).** Transfer learning. Handbook of Research on Machine Learning Applications and Trends, pp. 242–264. DOI: 10.4018/978-1-60566-766-9.ch011.
38. **Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017).** Attention is all you need. Advances in Neural Information Processing Systems, Vol. 30, pp. 5998–6008.
39. **Conneau, A., Lample, G. (2019).** Cross-lingual language model pretraining. Advances in Neural Information Processing Systems, Vol. 32, pp. 7057–7067.
40. **Gage, P. (1994).** A new algorithm for data compression. C Users Journal, Vol. 12, No. 2, pp. 23–38.
41. **Sennrich, R., Haddow, B., Birch, A. (2016).** Neural machine translation of rare words with subword units. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/ P16-1162.
42. **Uysal, A. K., Gunal, S. (2014).** The impact of preprocessing on text classification. Information Processing and Management, Vol. 50, No. 1, pp. 104–112. DOI: 10.1016/j.ipm.20 13.08.006.
43. **Hutto, C., Gilbert, E. (2014).** Vader: a parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the International AAAI Conference on Web and Social Media, Vol. 8, No. 1, pp. 216–225. DOI: 10.1609/icwsm.v8i1.14550.
44. **Kingma, D. P., Ba, J. (2014).** Adam: A method for stochastic optimization. 3rd International Conference for Learning Representations, pp. 1–15. DOI: 10.48550/ARXIV.1412.6980.
45. **Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S. J., Kumar, S., Sra, S. (2020).** Why ADAM beats SGD for attention models. Proceedings of the International Conference on Learning Representations, pp. 1–18.
46. **Sun, C., Qiu, X., Xu, Y., Huang, X. (2019).** How to fine-tune Bert for text classification? Chinese Computational Linguistics, pp. 194–206. DOI: 10.1007/978-3-030-32381-3_16.
47. **Antoun, W., Baly, F., Hajj, H. (2020).** AraBERT: Transformer-based model for Arabic language understanding. Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pp. 9– 15.

Article received on 26/01/2022; accepted on 24/04/2024.
**Corresponding author is Kheir Eddine Daouadi.*