


Deep Random Forest and AraBert for Hate Speech Detection from Arabic Tweets


Kheir Eddine Daouadi

(Laboratory of Vision and Artificial Intelligence (LAVIA), Echahid Cheikh Larbi Tebessi University, Tebessa, Algeria

 <https://orcid.org/0000-0003-2348-7192>, kheireddine.daouadi@univ-tebessa.dz)


Yaakoub Boualleg

(Laboratory of Vision and Artificial Intelligence (LAVIA), Echahid Cheikh Larbi Tebessi University, Tebessa, Algeria

 <https://orcid.org/0000-0001-9665-5594>, yaakoub.boualleg@univ-tebessa.dz)

Oussama Guehairia

(Laboratory of LESIA, Mohamed Khider University of Biskra, Biskra, Algeria

 <https://orcid.org/0000-0002-6755-4329>, oussama.guehairia@univ-biskra.dz)

Abstract: Nowadays, hate speech detection from Arabic tweets attracts the attention of many researchers. Numerous systems and techniques have been proposed to address this classification challenge. Nonetheless, three major limits persist: the use of deep learning models with an excess of hyperparameters, the reliance on hand-crafted features, and the requirement for a huge amount of training data to achieve satisfactory performance. In this study, we propose Contextual Deep Random Forest (CDRF), a hate speech detection approach that combines contextual embedding and Deep Random Forest. From the experimental findings, the Arabic contextual embedding model proves to be highly effective in hate speech detection, outperforming the static embedding models. Additionally, we prove that the proposed CDRF significantly enhances the performance of Arabic hate speech classification.

Keywords: Twitter, Hate Speech Detection, Arabic Tweet Classification, Contextual Deep Random Forest, Fine-tuning, Pre-trained Contextual Embedding

Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

DOI: 10.3897/jucs.112604

1 Introduction

Today, social media like Twitter have become a major part of our day-to-day life. This growing phenomenon has been very widespread in Arabic communities. Arabic Twitter users generate over 27 million tweets per day. The coming of Twitter has encouraged numerous axis research directions such as hate speech detection [Al-Hassan and Al-Dossari 2022, Duwairi et al. 2021, Mubarak et al. 2020], sentiment classification [Chamlertwat et al. 2012, Kalampokis et al. 2016, Baker et al. 2020, Abirami and Askarunisa 2016], bot detection [Daouadi et al. 2019b, Daouadi et al. 2020], organization detection [Daouadi et al. 2018a, Daouadi et al. 2019a, Daouadi et al. 2018b] and many more [Choi et al. 2013, Kalampokis et al. 2017] etc.

Recently, researchers have proposed many systems and techniques for Arabic hate speech detection. However, three of the major limits confronted in this context owing to the leverage of deep learning models based on many parameters, the leverage of

handcrafted features, and their accuracy performance are limited. Automatic hate speech detection from Arabic tweets using traditional machine learning algorithms such as Support Vector Machine (SVM) [Husain 2020b] and Naïve Bayes (NB) [Mulki et al. 2019] have shown good results. Nevertheless, they focused mainly on hand-crafted features like Term Frequency (TF), Term Frequency-Inverse Document Frequency (TF-IDF), and Bag of Word (BoW). On the other hand, the latest developments in the field of deep learning classifiers like Convolution Neural Network (CNN) [Alsafari et al. 2020a] and Bidirectional Encoder Representation for Transformer (BERT) [Mubarak et al. 2020] have already demonstrated a remarkable performance for hate speech detection from Arabic tweets.

In this paper, we present a thorough investigation of DRF and contextual embedding for hate speech detection from Arabic tweets. Contextual embedding aims to convert tweet words into numerical values by using the available embedding models. The DRF consists of two major parts: Multi-Grained Scanning (MGS) and Cascade Forest (CF). The MGS aims to handle the different sliding windows and inputs them to the decision tree algorithms to create a feature vector for different sliding window sizes. The CF enables feature processing in a layer-by-layer fashion, based on an ensemble of random forests. Each cascade layer receives data produced by its predecessor. The output of the CF is the predicted tweet class. Therefore, we illustrate the leverage of our proposed approach by classifying a recent benchmark dataset of Arabic hate speech into 5 distinct classes: general hate, religious, sexism, racial, or none. Moreover, we conducted an extensive parameter sensitivity to tune CDRF for our Arabic hate speech classification task. A comparison between CDRF and existing hate speech classification approaches is presented. Experimental results demonstrated that CDRF improves the accuracy of hate speech detection from Arabic tweets, and outperforms existing works with a considerable margin. Our proposition may offer an important advantage by using ensemble learning that achieves better performance results than one single classifier, while CDRF is simple and easy to train based only on a few parameters. The contributions of this paper are presented as follows:

- We proposed a novel and efficient deep learning architecture combining contextual embedding and Deep Forest. The suggested architecture improves the accuracy of hate speech detection from Arabic tweets and outperforms the latest state-of-the-art results.
- We compare the performance of the static embedding models (AraVec, ArWord2vec, Mazajak, and Fastext) as well as contextual embedding models like (AraBERT, AraElectra, Multilingual BERT, and XLM-Roberta). Experiments show that the Arabic contextual embedding models trained on Twitter data improve the accuracy results of hate speech detection from Arabic tweets.
- We present an empirical study to select the hyperparameters for the CDRF architecture for hate speech detection from Arabic tweets.

The remainder of this paper is structured as follows. Section 2 discusses state-of-the-art approaches. Section 3 details our proposed approach. Section 4 presents the experimental results, and Section 5 summarizes the manuscript.

2 Related Works

Today, hate speech detection from Arabic tweets has attracted the attention of many researchers in the world. Different systems have been proposed to face this social chal-

lenge. They leverage two main approaches: a deep learning approach and a traditional approach.

2.1 Traditional Approaches

Traditional classification approaches rely on feature engineering, which converts texts to feature vectors and classifies them with machine learning algorithms like SVM and NB. The majority of the features were lexical-based features such as TF-IDF, N-grams, and BOW. The most relevant traditional approaches are described as follows.

Authors in [Husain 2020b] investigate the impact of the preprocessing step on hate speech and offensive language classification. They showed that an intensive preprocessing technique demonstrates its significant impact on the detection rate. The best accuracy results are obtained using BoW and SVM, which yielded an F1 score result of 89% and 95% for offensive language and hate speech.

In a similar, in [Husain 2020a] the authors classify tweets as offensive or not based on BoW and TF-IDF features. They showed that the ensemble learning classifier (Bagging) outperformed the single learner classifier, which yielded an F1 score of 88%.

Besides, authors in [Chowdhury et al. 2020] highlight the importance of multiple platform datasets for the generalization of classifier performance of offensive language detection. Their experiment with TF-IDF and SVM yielded an F1 score result of 84%.

Likewise, authors in [Aljarah et al. 2021] use BoW, TF-IDF, TF, Profile, and emotion features to classify tweets as being hate speech or not. The best accuracy results are obtained using Random Forest, which yielded an F1 score result of 91.3%.

Furthermore, authors in [Mulki et al. 2019] reported the classification rate using SVM and NB with unigrams, bigrams, and trigrams. The best experimental results were obtained using NB, which yielded an F1 score of 74.4% for (Non-Hate vs. hate vs. abusive) and 89.6% for (Abusive and Hate vs. Non-Hate).

In a different strategy, in [Abozinadah and Jones Jr 2017] the authors leverage profile-based, Social Graph feature, and tweet-based features to distinguish abusive Twitter accounts from those non-abusive. They reported that the NB algorithm yielded the best accuracy results of 85%.

2.2 Deep Learning Approaches

Deep Learning (DL) approaches rely on neural networks, which automatically learn the representation of input texts with different levels of abstraction and use the acquired knowledge for the classification task. The majority of used embedding models are AraVec, Mazajk, and AraBERT. The most popular deep learning architectures adopted in the field of hate speech detection from Arabic tweets are CNN, BERT, Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU). Some examples of DL approaches are described as follows.

Authors in [Faris et al. 2020] use the AraVec model based on N-grams and Skip-gram to classify tweets into those hateful and Non-hate. The Hybrid CNN-LSTM is used for classification, which yielded an F1 score result of 71.69%.

In similar, authors in [Al-Hassan and Al-Dossari 2022] classify tweets into five classes: general hate, sexism, racial, religious, or none. They randomly initialized the word embedding and learn the embedding of each word using the training dataset. The hybrid CNN-LSTM achieved the highest accuracy results, which yielded an average F1 score result of 73%.

Besides, authors in [Alsafari et al. 2020a] investigate the impact of both neural network architectures and word embedding models on the accuracy rate. They train different embedding models based on an Arabic text corpus. Additionally, they compared different neural networks for each classification task. The highest accuracy results are achieved using the Skip-gram embedding model and CNN, which yielded F1 score results of 87.22%, 75.16%, and 70.80% for (Non-hate, Offensive, or Hate), (Non-Hate, Offensive, or Hate) and (Non-Hate, Offensive, Religion Hate, Gender Hate, ethnicity hate or Nationality hate) classification task, respectively.

In a similar study [Alghanmi et al. 2020], the authors use both contextual (AraBERT) and static (AraVec) embedding to classify tweets as Non-Hate, hateful, or abusive. The CNN achieved the best accuracy results and yielded an average F1 score of 72.1%.

Likewise, authors in [Alsafari et al. 2020c] use contextual Multilingual BERT embedding model with CNN, which yielded F1 score results of 87.03%, 78.99%, and 75.51% for (Non-hate vs. Offensive and Hate), (Non-Hate, Offensive or Hate) and (Non-Hate, Offensive, Religion Hate, Gender Hate, ethnicity hate or Nationality hate) classification task, respectively.

Furthermore, authors in [Haddad et al. 2020] use bidirectional GRU augmented with attention layer and AraVec embedding model to detect hate speech and offensive language tweets. Additionally, they investigate the impact of various pre-processing and oversampling techniques to increase the performance rate. They achieved F1 score results of 85.9% and 75% F1 score for offensive language and hate speech detection, respectively.

In different strategies, the authors in [Mubarak et al. 2020, Shapiro et al. 2022, Khezzar et al. 2023, Boulouard et al. 2022] use transfer learning based on different Arabic Bert models, the proposed approaches achieved promising results for classifying Arabic hate speech.

2.3 Gaps and Contributions

To date, no study has proposed a DL technique based on non-neural networks. In this paper, we investigated the use of non-neural networks for hate speech detection from Arabic tweets.

Our choice goes somewhat in contradiction to that of many DL and traditional learning techniques, which are based on huge amounts of hyperparameters, while their performance is limited. Besides, their computational efficiency is suboptimal, and their interpretability is challenging.

3 Methodology

To overcome the previously mentioned challenges, we proposed a special DL architecture known as Contextual Deep Random Forest (CDRF). As highlighted in Figure 1, our proposition consists of 4 major steps. Specifically, Pre-processing, Contextual Embedding, Multi-Grained Scanning, and Cascade Forest are described as follows.

3.1 Pre-processing

The inputs of our proposition are the textual content of tweets, which consists of raw tweet content as highlighted in Figure 1. This step is dedicated to cleaning the textual content of tweets to produce more consistent and standard tweets. We performed some pre-processing tasks as follows [Al-Hassan and Al-Dossari 2022]:

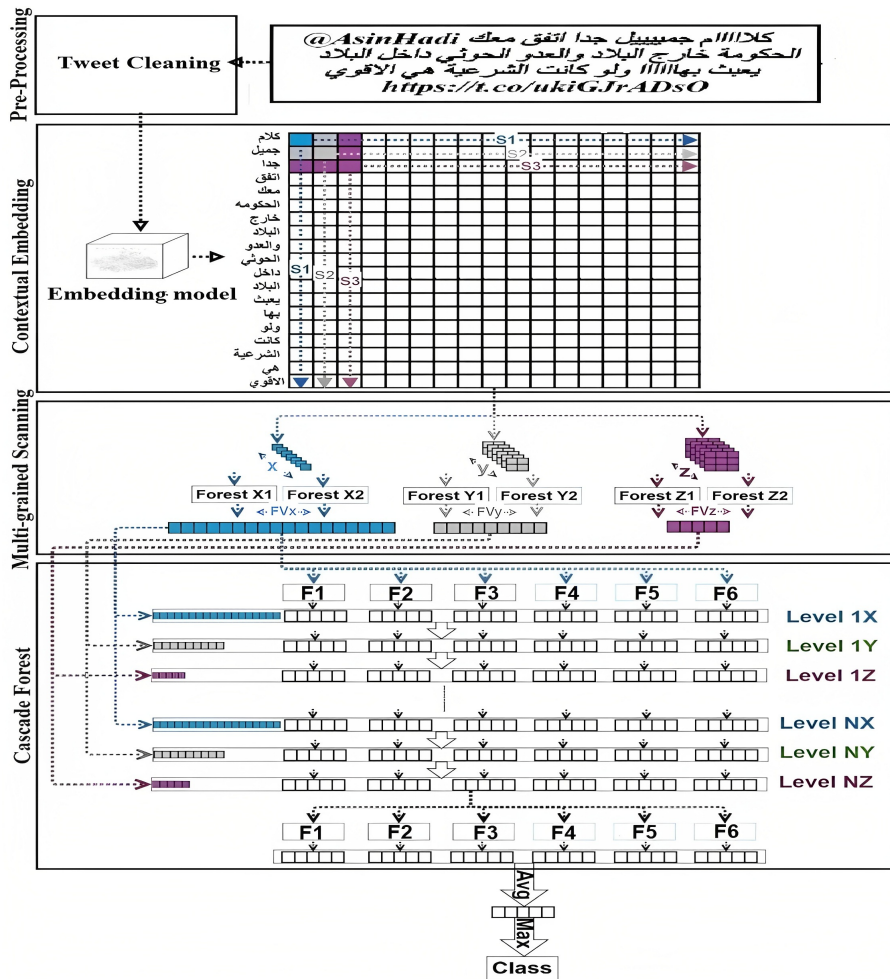


Figure 1: Our proposed CDRF approach.

- Removing Tweet features: the RT word, user mentions '@', hashtag symbol '#', URLs, punctuation, numerical characters, and special characters.
- non-Arabic letters, new lines as well as diacritics.
- Removing repeated letters: such as (مرحبًا) which means “Hellooooo”, to be (مرحبًا), which means “Hello”.
- Arabic letters normalization: in Arabic language, they are different variations for some letters’ representation which are:
 - Letter (Taa Marbouta) (ة) which can be mistaken and written as (ه), we normalized it to (ه).
 - Letter (Alef) (أ) that has the forms (ا, آ, إ, ا), all these letters are standardized into (ا).

- letter Dash that is leveraged to expand the word (مرحبا) to (مر—حبا) has been removed.
- Letter (Alef Maqsoora) (ي) has been normalized to (ى).

3.2 Contextual Embedding

The output of the previous steps is the normalized tweet content. Pre-trained contextualized word embedding plays an important role in constructing contextual tweet representation. The contextual relationships extracted by the embedding model are useful for detecting relevant contextual similarities. Contextual embeddings are calculated through a process involving :

- Tokenization: The input text (like a tweet) is divided into tokens (individual words or subwords). Each token is given an index, and special tokens like [CLS] and [SEP] might be added to the structure.
- Word Embedding Lookup: Each token's index is used to retrieve its pre-trained word embedding from a matrix. This matrix contains vectors representing a wide range of words.
- Positional Embeddings: Since the models don't naturally understand word order, positional embeddings are added to each token's embedding. These embeddings indicate where the token is located in the text.
- Attention Mechanism: Contextual embeddings use attention, allowing each token to consider information from all other tokens. Attention scores determine how important each token's connection is with others.
- Multiple Layers and Attention Heads: Contextual embeddings are produced through multiple layers and attention heads. Each layer refines embeddings by incorporating various contextual levels. Attention heads capture different relationships between words.
- Transformer Architecture: Models like BERT and GPT use a transformer architecture, composed of encoder and decoder layers. BERT uses only the encoder to understand input context. BERT is bidirectional, considering both the left and right context. These capture complex context relationships.
- Contextualized Embeddings: Final embeddings are contextualized, representing a token's meaning within its context. Words with multiple meanings can have different embeddings based on context.

To train our CDRF, all tweet instances are normalized to the longest tweet in our data by applying zero embedding. The contextualized embedding is used as input for the following step.

3.3 Multi-Grained Scanning

The major step of DRF is Multi-Grained Scanning (MGS), whose goal is to preserve the relationships between the word embedding vectors. The Sliding Window (SW) is the core building block MGS, which aims to handle the spatial relationships between the contextual Word Embedding vector. This helps to analyze the presence of such patterns or features in the tweet text. In this step, a sliding $S_j \in R^{k \times k}$ is applied to the contextual tweet representation Emb_{Tweet} . As illustrated in Figure 1, where three different SW sizes are highlighted. The Stride (s) can shift the SW across s words for each step. The

Sliding operation produces N matrices instances (i.e. x , y , and z (cf Figure 1)) for each SW as follows:

$$N_i = E((ML - S_i)/s + 1) * E((dim - S_i)/s + 1) \quad (1)$$

where ML is the maximum length of a tweet in our data, dim represents the dimension of WE, S_i represents the i^{th} sliding window size, s represent the stride, and $E(z)$ represents the integer part of z .

Let MI_i be the Matrices Instance outputted by the i^{th} sliding window operation and NF represent the number of forests used in MGS. Each MGS forest (i.e. Forest X1, Forest X2, Forest Y1, Forest Y2, Forest Z1, and Forest Z2 (cf Figure 1)) outputs class probabilities as follows:

$$MGSForest_{nf}(MI_{ni}) = CP_{ni}^{nf} \quad (2)$$

Where $nf \in [1, NF]$, $ni \in [1, Number\ of\ instances]$, and $|CP_{ni}^{nf}|$ represent the number of classes. The output of MGS are the features vectors FV_{nsw} (i.e. FV_x , FV_y , and FV_z (cf Figure 1)), which is the concatenation of all $(CP_{ni}^{nf})_{nsw}$ of the same SW S_{nsw} as described in the following equation:

$$FV_{nsw} = (CP_1^1)_{nsw} + \dots + (CP_{ni}^{nf})_{nsw} \quad (3)$$

where $nsw \in [1, Number\ of\ Sliding\ Window]$ and :

$$|FV_{nsw}| = N_{nsw} * NF * |CP_{ni}^{nf}|_{nsw} \quad (4)$$

3.4 Cascade Forest

As presented in Figure 1, the Cascade Forest (CF) has their own structure that performs layer-by-layer feature processing [Boualleg et al. 2019, Guehairia et al. 2020, Daouadi et al. 2021]. The output feature vectors from MGS are inputted through the cascade layers from the first to the last one. Each layer of the CF is an ensemble of random forests that receives data from the previous layer. Let NF' is the number of forests in CF (i.e. F1, F2, F3, F4, F5, and F6 (cf Figure 1)), for each layer (i.e. Level 1X ... Level NZ (cf Figure 1)) a Class Distribution CD vector is produced as follows:

$$|CD_{nsw}^{nf'}| = NF' * number_of_classes \quad (5)$$

The final layer gets the vectors of class distribution probabilities from the previous cascade layer to calculate the tweet class as follows:

$$Probability_classes = Avg(CD_{nsw}^{nf'}) \quad (6)$$

Finally, we used the majority voting technique to calculate the final predicted class label as follows:

$$Y = Argmax(Probability_classes) \quad (7)$$

where $Probability_classes$ is the vector of the probability distribution of each class.

To reduce the risk of overfitting, we performed K-fold cross-validation to calculate the class probability of each random forest. Notably, each tweet instance in the training data was leveraged K-1 times, which resulted in K-1 classes, which are then averaged to

calculate the class probabilities as improved feature vectors for the next cascade level. The performance metrics of a whole level have been evaluated on the test sets after the expansion of a novel level. The training procedure ends where there is no significant gain in performance. In contrast, to the majority of DL models, whose complexity is fixed, CDRF chooses its complexity as terminating training procedure when adequate.

4 Experiments and Evaluation

In this section, we present the experimental and evaluation results we carried out to show the effectiveness of CDRF. Through the experiments on the benchmark Twitter dataset, we try to find answers to the following research questions:

- Can a contextual pre-trained embedding model improve the accuracy of hate speech detection from Arabic tweets?
- Can a Deep Learning architecture based on a non-neural network classify hate speech from Arabic tweets accurately?

To answer these questions, we first introduce the datasets used in our experiments. Subsequently, we provide an analysis of the performance results, followed by a comprehensive discussion of the findings.

	NH	GH	ReH	RaH	S	Total
Number of tweets	8332	1397	722	526	657	11634
Word count	96.9 K	17.2 K	9.2 K	6.6 K	8.3 K	138.3 K
Unique words	29 K	8.7 K	4.9 K	4 K	4.4 K	37.9 K
Avg words per tweet	11.6	12.3	12.7	12.7	12.6	11.9

Table 1: Overview of the ground truth after pre-processing (*K* denotes a thousand, *NH* = Non-Hate Speech, *GH* = General Hate Speech, *ReH* = Religious Hate Speech, *RaH* = Racial Hate Speech, *S* = Sexism).

4.1 Datasets

To evaluate our proposition, we refer to the dataset published in [Al-Hassan and Al-Dossari 2022]. An Overview of this dataset after pre-processing is shown in Table 1. For collecting the tweets, the authors used the Tweepy library with a list of hashtags that trigger and attract Twitter hateful content. Balancing the number of non-hate tweets and hate tweets does not reflect the real situation and is not realistic. The authors identified a list of hashtags that contains surely both non-hateful and hateful content to preserve the realistic and natural scenario. Then, the retrieved tweets were manually annotated by 2 annotators. The annotators were provided with a guide to distinguish the tweets classes, which yielded 11634 labeled tweets out of 37K collected tweets. As presented in Table 1, the minor set of tweets is the racial hate speech class, while the major set is the non-hate class.

In addition, we evaluate the proposed approach using train/test datasets published in [Alsafari et al. 2020c]. The dataset comprises 5340 tweets (3738 for training and 1602 for testing) manually classified as Clean, Offensive, Religious hate speech, Gender hate speech, Ethnicity hate speech, and Nationality hate speech. The annotation process was conducted by three native speakers from the Gulf region, including two females and one male, all possessing a higher level of education. These annotators were initially provided with task descriptions, guidelines, and three illustrative examples for each hate category.

4.2 Experimental Setup

To evaluate our proposition, we used the 10-fold cross-validation to estimate the performance metrics. The data was divided into ten equally sized segments while preserving the balance of each class in the original dataset. One of the segments was used as testing data and the other ones were used as training data. This is repeated ten times, the average F1-score metrics were obtained using the ten iterations. In our experiments, we used Scikit-Learn [Pedregosa et al. 2011], Ktrain [Maiya 2022], and Keras ¹ which uses Tensorflow ² as back-end. All the experiments were executed on a machine equipped with an Intel Core i7-7700 16GB RAM. The initial parameters of CDRF are SW=1*1, NT=50, NF=5, NT'=2, and NF'=100.

4.3 Evaluation Metrics

To evaluate the performance of our proposition, we used different evaluation metrics that can judge the model. Since the task of hate-speech detection is an imbalanced classification problem, we will pay big attention to the Macro-averaged and Weighted-averaged to calculate the overall performance measures described as follows.

Precision (P) (also called positive predictive value) represents the fraction of correctly classified positive observations over the total observations classified as positive. For instance, the Precision of the Non-Hate class is calculated as follows:

$$P_{Non-Hate} = CC_{Non-Hate} / TC_{Non-Hate} \quad (8)$$

where $CC_{Non-Hate}$ is the number of tweets correctly classified as Non-Hate and $TC_{Non-Hate}$ is the total number of tweets classified as Non-Hate.

Recall (R) is the fraction of correctly classified positive observations over the total positive observations. For instance, the Recall of the Non-Hate class is calculated as follows:

$$R_{Non-Hate} = CC_{Non-Hate} / TN_{Non-Hate} \quad (9)$$

where $CC_{Non-Hate}$ is the number of tweets correctly classified as Non-Hate and $TN_{Non-Hate}$ is the total number of Non-Hate tweets.

F1-score (F1) represents the harmonic mean between Recall and Precision. For instance, the $F1_N$ of the Non-Hate class is calculated as follows:

$$F1_{NH} = 2(P_{NH} * R_{NH}) / (P_{NH} + R_{NH}) \quad (10)$$

¹ Keras is an open-source neural network library written in Python which is designed for fast experimentation with deep neural networks. <https://keras.io>

² TensorFlow is an open-source software library for data flow programming across a range of tasks, which is used for different machine learning applications, such as neural networks. <https://www.tensorflow.org>

4.4 Results

We performed extensive experiments on the proposed CDRF for hate speech detection from Arabic tweets. We compared the accuracy results of both static and contextual embedding models, namely: ArWord2vec CBOW-5 (W2V_C_5) and ArWord2vec SG-5 (W2V_S_5) [Fouad et al. 2020]; Mazajak 100M-SG (M_100_S) and Mazajak 100M-SG (M_250_S) [Farha and Magdy 2019]; AraVec Tweet-CBOW (AV_Twt_C) and AraVec Tweet-SG (AV_Twt_S) [Soliman et al. 2017]; Fasttext [Joulin et al. 2016]; AraBERT and AraBERTv02-twitter (AraB_v02_T) [Antoun et al. 2020a]; AraElectra [Antoun et al. 2020c]; Multilingual BERT (MBERT) [Devlin et al. 2018], XLM-Roberta (XLM-R) [Conneau et al. 2019]; BERT Arabic (BERTA) [Safaya et al. 2020], Multi-dialectal Arabic BERT (MdABERT) [Talafha et al. 2020]; and MARBERT [Abdul-Mageed et al. 2020].

The first experiment aims to select the word embedding model that attains the highest F1 score results. Table 2 shows a comparison of both static and contextual embedding models. The results show that the Arabic contextual embedding model trained on Twitter data outperforms the other ones with a considerable, yielded Macro-averaged F1-score result of 0.55 and a Weighted-averaged F1-score result of 0.77. The primary factors are the datasets used to train the embedding model and the effectiveness of contextual embeddings in capturing the meaning of words by analyzing the words that appear around them in sentences or texts.

The second set of experiments aims to select the optimal SW sizes. The accuracy rates of multiple and single sizes of sliding windows are presented in Table 3. The results show that the multiple sizes of SW increase the F1-score as they extract more N-gram features, yielding 0.58 and 0.78 for the Macro-averaged F1-score and the Weighted-averaged

Model	NH	S	ReH	GH	RaH	M	W
Random	75.81	41.54	15.75	39.12	21.66	38.78	63.29
M_100_S	87.04	45.75	56.89	31.25	32.96	50.78	73.69
M_250_S	86.78	43.73	61.82	29.03	26.16	49.51	73.12
W2V_C_5	86.58	45.68	58.13	34.37	27.68	50.49	73.57
W2V_S_5	86.22	42.20	58.68	33.16	25.49	49.15	72.90
AV_Twt_S	87.30	44.00	60.99	31.82	33.58	51.54	74.13
AV_Twt_C	87.17	43.19	61.43	31.40	32.05	51.05	73.85
Fasttext	86.45	44.07	57.28	34.06	26.75	49.72	73.26
MdABERT	87.46	48.28	51.00	42.86	25.83	51.09	74.84
BERTA	87.33	47.28	52.34	41.83	25.59	50.87	74.64
AraElectra	87.58	47.83	62.71	38.55	26.50	52.63	75.16
AraBERT	88.05	47.39	57.41	42.52	28.84	52.84	75.70
MARBERT	88.40	50.00	61.40	41.62	30.74	54.43	76.33
AraB_v02_T	88.59	51.49	64.61	41.32	31.26	55.45	76.73
MBERT	86.51	38.46	44.72	39.01	24.58	46.65	72.69
XLM-R	85.22	23.47	41.65	38.02	22.55	42.18	70.52

Table 2: Comparison of different embedding models based on F1-score (%) (M=Macro-Averaged, W=Weighted-Averaged).

F1-score, respectively

The third experiment aims to select the optimal number of NF. The effect of NF on the F1-score is presented in Table 4. The F1-score results increased when NF increased from 5 to 10. Thus, it decreased when NF increased further.

The fourth experiment aims to select the optimal number of NT. The F1-score results remain fixed when NT increased from 50 to 200 As shown in Table 5. Thus, it decreased when NT increased further.

The fifth experiment aims to find the optimal number of NF'. As highlighted in Table 6, the F1-score increased when NF' increased from 2 to 4. Thus, it decreased when NF' increased further.

The sixth experiment aims to select the optimal number of NT'. The F1-score increased when NT' increased from 100 to 300 as highlighted in Table 7. Thus kept stable when NT' increased further.

The best parameters of CDRF for our hate speech detection task are presented in Table 8 where NF (F)= 10, NT (T)= 200, SW = {1, 3, 5, 7, 9}, NF'(F') = 4 and NT'(T') = 300. The number of the level of Cascade Layer (C) is 2 when using the optimized parameters, while (N') is the count of nodes in the trees fixed to 10 nodes, and the total

SWS	NH	S	ReH	GH	RaH	M	W
1	88.59	51.49	64.61	41.32	31.26	55.45	76.73
3	88.71	46.42	61.57	45.14	31.63	54.69	76.82
5	88.64	52.09	63.58	41.93	31.63	55.57	76.83
7	88.76	49.40	58.74	45.66	32.15	54.94	76.93
9	88.72	48.62	59.70	45.94	32.52	55.10	76.97
1,3	88.68	46.59	69.56	44.70	34.44	56.79	77.38
3,5	88.56	45.81	69.62	44.25	33.33	56.32	77.15
5,7	88.72	47.02	70.20	44.76	35.14	57.17	77.51
7,9	88.70	46.60	70.07	44.65	34.64	56.93	77.43
1,3,5	88.77	47.69	69.26	45.35	35.69	57.35	77.62
3,5,7	88.84	47.88	69.57	45.24	36.27	57.56	77.71
5,7,9	88.86	48.24	70.87	44.83	35.79	57.72	77.76
1,3,5,7	88.94	49.70	69.22	45.90	36.00	57.95	77.94
3,5,7,9	88.71	46.95	70.06	44.78	34.81	57.06	77.48
1,3,5,7,9	89.01	53.83	64.92	46.29	36.22	58.05	78.01

Table 3: The effect of multiple sliding windows on F1-score (%).

NF	NH	S	ReH	GH	RaH	M	W
5	89.01	53.83	64.92	46.29	36.22	58.05	78.01
10	89.43	58.51	64.23	46.45	37.15	59.15	78.59
15	89.11	54.04	65.36	46.39	36.67	58.31	78.15
20	88.97	49.28	69.28	45.83	36.40	57.95	77.94
25	88.75	47.56	69.34	45.34	35.39	57.27	77.59

Table 4: The effect of NF on F1-score (%).

NT	NH	S	ReH	GH	RaH	M	W
50	89.43	58.51	64.23	46.45	37.15	59.15	78.59
100	89.42	58.17	66.14	47.98	39.16	60.17	78.96
150	89.47	58.88	66.71	48.02	39.76	60.57	79.10
200	89.58	60.27	69.16	47.26	40.16	61.29	79.34
250	89.45	58.27	65.05	46.47	37.35	59.32	78.66

Table 5: The effect of NT on F1-score (%).

NF'	NH	S	ReH	GH	RaH	M	W
2	89.58	60.27	69.16	47.26	40.16	61.29	79.34
4	89.62	61.77	72.90	46.09	41.13	62.30	79.59
6	89.54	61.43	73.76	45.43	40.64	62.16	79.46
8	89.42	58.69	66.54	47.86	39.27	60.36	79.01
10	89.52	60.02	67.70	47.65	39.72	60.92	79.22

Table 6: The effect of NF' on F1 score F1-score (%).

NT'	NH	S	ReH	GH	RaH	M	W
100	89.62	61.77	72.90	46.09	41.13	62.30	79.59
200	89.60	57.58	74.51	48.04	42.22	62.39	79.72
300	89.84	58.98	75.31	48.98	43.63	63.35	80.20
400	89.47	58.88	66.71	48.02	39.76	60.57	79.10
500	89.54	60.03	74.32	45.69	40.40	62.00	79.44

Table 7: The effect of NT' on F1-score (%).

Hyperparameters	Embedding Model	SW	NF	NT	NF'	NT'
Value	AraBERTv02-Twitter	{1,3, 5, 7, 9}	10	200	4	300

Table 8: Optimized parameters of CDRF.

of SW (N) used in CDRF is 5. The total Trainable Parameters in CDRF is estimated using the following equations:

$$TP_{CDRF} = TP_{MGS} + TP_{CF} \quad (11)$$

$$TP_{MGS} = (F * T * N) \quad (12)$$

$$TP_{CF} = (T' * F' * N * N' * C) + (T' * F' * N) \quad (13)$$

According to the aforementioned Equations, the number of TP in CDRF is equal to 0.152 million trainable parameters, unlike most of the state-of-the-art models that employ more than a million parameters.

Thereafter, the F1-score of CDRF is compared against four existing traditional machine learning approaches [Mulki et al. 2019, Husain 2020a, Chowdhury et al. 2020, Miller et al. 2017], six existing DL approaches, [Al-Hassan and Al-Dossari 2022, Alsafari

Approach	NH	S	ReH	GH	RaH	M	W
[Mulki et al. 2019]	0.85	0.21	0.10	0.12	0.09	0.27	0.64
[Husain 2020a]	0.86	0.42	0.50	0.25	0.24	0.45	0.71
[Chowdhury et al. 2020]	0.87	0.41	0.54	0.30	0.29	0.48	0.73
[Al-Hassan and Al-Dossari 2022]	0.86	0.43	0.59	0.31	0.25	0.49	0.73
[Alsafari et al. 2020c]	0.85	0.42	0.56	0.45	0.20	0.50	0.73
[Haddad et al. 2020]	0.87	0.50	0.64	0.49	0.27	0.55	0.76
[Alghanmi et al. 2020]	0.86	0.44	0.59	0.47	0.22	0.52	0.74
[Faris et al. 2020]	0.87	0.37	0.47	0.28	0.26	0.45	0.72
[Alsafari et al. 2020b]	0.87	0.49	0.56	0.47	0.24	0.53	0.75
[Mubarak et al. 2020]	0.88	0.49	0.67	0.41	0.37	0.57	0.77
[Khezzar et al. 2023]	0.89	0.55	0.65	0.52	0.42	0.61	0.79
Arabert + Random Forest	0.89	0.51	0.63	0.51	0.40	0.59	0.79
Arabert + [Miller et al. 2017]	0.89	0.47	0.61	0.55	0.37	0.58	0.79
Aragpt [Antoun et al. 2020b]	0.85	0.23	0.42	0.38	0.23	0.42	0.70
CDRF	0.90	0.59	0.75	0.49	0.44	0.63	0.80

Table 9: Comparison of CDRF and the latest baseline approaches based on F1-score.

et al. 2020c, Haddad et al. 2020, Alghanmi et al. 2020, Faris et al. 2020, Alsafari et al. 2020b], and three fine-tuning approaches [Mubarak et al. 2020, Khezzar et al. 2023, Antoun et al. 2020b]. While comparing CDRF with the baselines presented in Tables 9 and 10, the F1-score of CDRF is the highest. The main reasons are due to the contextual representation of the tweet. When compared with LSTM, BERT, and CNN, current DL hate speech classification models are built based on Neural Networks trained based on backpropagation. On the other hand, CDRF is a DL approach that leverages an ensemble of random forests and their training procedure doesn't depend on backpropagation. When compared with SVM, NB, and Bagging, CDRF has not extracted and designed hand-crafted features. In addition, CDRF deals with social media data, where the user uses many slangs, abbreviations, etc. (i.e. the language structure is not preserved). CDRF analyzes the sequence of words by correlating them with past and future words.

Even though quite successful, previous DL Arabic hate speech detection approaches are so expensive, as they are built upon Neural Networks trained based on backpropagation with so many hyperparameters. As an example, taking [Mubarak et al. 2020, Husain and Uzuner 2021] are recent DL models that used AraBERT (with over 100 million parameters). On the contrary, CDRF is a DL approach based on an ensemble of Random Forest, where the training procedure doesn't depend on backpropagation, while using fewer hyperparameters than the existing approaches. Furthermore, current traditional approaches used handcrafted features, which faced the curse of dimensionality and data sparseness. Conversely, CDRF can automatically detect features from textual content. Finally, the comparison between the accuracy rate of CDRF and 14 baselines is performed, and the obtained result highlights the efficiency of CDRF against existing approaches. Resulting promising weighted averaged F1 score result of (> 80%). In Table 2, we can notice that: the minor accuracy results go for the multilingual contextual embedding models; the medium results go for Arabic static embedding models and contextual em-

Approach	C	O	R	G	N	E	M	W
[Mulki et al. 2019]	0.89	0.49	0.47	0.52	0.56	0.69	0.61	0.77
[Husain 2020a]	0.89	0.54	0.48	0.55	0.57	0.70	0.62	0.78
[Chowdhury et al. 2020]	0.91	0.54	0.52	0.55	0.64	0.68	0.64	0.79
[Al-Hassan and Al-Dossari 2022]	0.87	0.57	0.71	0.64	0.68	0.69	0.69	0.80
[Alsafari et al. 2020c]	0.91	0.72	0.71	0.68	0.74	0.77	0.76	0.85
[Haddad et al. 2020]	0.86	0.56	0.70	0.63	0.67	0.70	0.69	0.79
[Alghanmi et al. 2020]	0.95	0.67	0.68	0.67	0.73	0.75	0.74	0.86
[Faris et al. 2020]	0.94	0.61	0.66	0.63	0.71	0.73	0.71	0.85
[Alsafari et al. 2020b]	0.96	0.74	0.73	0.76	0.76	0.79	0.79	0.89
[Mubarak et al. 2020]	0.95	0.77	0.71	0.80	0.79	0.80	0.80	0.89
[Khezzar et al. 2023]	0.96	0.80	0.75	0.74	0.80	0.79	0.81	0.90
Arabert + RF	0.92	0.84	0.77	0.73	0.84	0.80	0.82	0.88
Arabert + [Miller et al. 2017]	0.92	0.76	0.77	0.75	0.78	0.79	0.80	0.87
Aragpt [Antoun et al. 2020b]	0.88	0.47	0.45	0.48	0.56	0.67	0.58	0.76
CDRF	0.96	0.85	0.80	0.82	0.83	0.83	0.85	0.91

Table 10: Comparison of F1-score of CDRF and the latest baseline approaches based using train/test datasets published in [Alsafari et al. 2020c] (C = Clean, O=Offensive, R=Religious hate speech, G = Gender hate speech, N = Nationality hate speech, E = Ethnicity hate speech)

bedding trained on general data while the major results go for the Arabic contextual model trained on Twitter data. In Table 9, we can notice that the minor accuracy results go for the racial hate speech class, while the major results go for the non-hate class. This is due to the severe class label imbalance. Thus, future work should be harnessed to face the problem of imbalanced data.

5 Conclusion

The proposed CDRF approach combines contextual embedding with Deep Random Forest, yet achieves the highest F1 results (> 80%). Our proposed approach highlights a new research area for hate speech detection approaches. Experiments show that: (1) the Arabic contextual embedding models trained on Twitter data can be effectively leveraged for hate speech detection and outperform multilingual contextual embedding models, static embedding model, and Arabic contextual models trained on non-Twitter data; (2) the proposed CDRF improve the accuracy of hate speech detection of Arabic tweets and outperform the latest state-of-the-art results. In future works, we plan to pursue several directions. First, we want to focus on the contextual embedding model and try to adjust their vocabulary to support the hate speech detection task. Second, we plan to test and compare different data augmentation techniques to overcome the challenge of imbalanced learning. From a research perspective, we plan to use CDRF for analyzing Arabic Tweets in some contexts to study the extent of hate speech conversations with public discussion.

References

- [Abdul-Mageed et al. 2020] Abdul-Mageed, M., Elmadany, A., Nagoudi, E. M. B.: "ARBERT MARBERT: deep bidirectional transformers for Arabic". arXiv preprint arXiv:2101.01785., (2020).
- [Abirami and Askarunisa 2016] Abirami, A. M., Askarunisa, A.: "Feature Based Sentiment Analysis for Service Reviews"; J. Univers. Comput. Sci., 22(5), 650-670, (2016).
- [Abozinadah and Jones Jr 2017] Abozinadah, E. A., Jones Jr, J. H.: "A statistical learning approach to detect abusive twitter accounts". In Proceedings of the international conference on compute and data analysis, 6-13, (May 2017).
- [ALBayari et al. 2021] ALBayari, R., Abdullah, S., Salloum, S. A.: "Cyberbullying Classification Methods for Arabic: A Systematic Review"; In The International Conference on Artificial Intelligence and Computer Vision, 375-385, (May 2021).
- [Alghanmi et al. 2020] Alghanmi, I., Anke, L. E., Schockaert, S.: "Combining BERT with static word embeddings for categorizing social media"; In Proceedings of the sixth workshop on noisy user-generated text, 28-33, (Nov 2020).
- [Aljarah et al. 2021] Aljarah, I., Habib, M., Hijazi, N., Faris, H., Qaddoura, R., Hammo, B., Abushariah, M., Alfawareh, M.: "Intelligent detection of hate speech in Arabic social network: A machine learning approach"; Journal of Information Science, 47(4), 483-501, (2021).
- [Alsafari et al. 2020a] Alsafari, S., Sadaoui, S., Mouhoub, M.: "Effect of word embedding models on hate and offensive speech detection"; arXiv preprint arXiv:2012.07534., (2020).
- [Alsafari et al. 2020b] Alsafari, S., Sadaoui, S., Mouhoub, M.: "Deep Learning Ensembles for Hate Speech Detection"; In IEEE 32nd International Conference on Tools with Artificial Intelligence, 526-531, (2020).
- [Alsafari et al. 2020c] Alsafari, S., Sadaoui, S., Mouhoub, M.: "Hate and offensive speech detection on arabic social media"; Online Social Networks and Media, 19, 100096, (2020).
- [Al-Hassan and Al-Dossari 2022] Al-Hassan, A., Al-Dossari, H.: "Detection of hate speech in Arabic tweets using deep learning"; Multimedia systems, 28(6), 1963-1974, (2022).
- [Antoun et al. 2020a] Antoun, W., Baly, F., Hajj, H.: "Arabert: Transformer-based model for arabic language understanding"; arXiv preprint arXiv:2003.00104., (2020).
- [Antoun et al. 2020c] Antoun, W., Baly, F., Hajj, H.: "AraELECTRA: Pre-training text discriminators for Arabic language understanding"; arXiv preprint arXiv:2012.15516. (2020).
- [Antoun et al. 2020b] Antoun, W., Baly, F., Hajj, H.: "AraGPT2: Pre-trained transformer for Arabic language generation"; arXiv preprint arXiv:2012.15520., (2020).
- [Baker et al. 2020] Baker, Q. B., Shatnawi, F., Rawashdeh, S., Al-Smadi, M., Jararweh, Y.: "Detecting epidemic diseases using sentiment analysis of arabic tweets". J. Univers. Comput. Sci., 26(1), 50-70, (2020).
- [Boualleg et al. 2019] Boualleg, Y., Farah, M., Farah, I. R. (2019). Remote sensing scene classification using convolutional features and deep forest classifier. IEEE Geoscience and Remote Sensing Letters, 16(12), 1944-1948, (2019).
- [Boulouard et al. 2022] Boulouard, Z., Ouaisa, M., Ouaisa, M., Krichen, M., Almutiq, M., Gsmi, K.: "Detecting Hateful and Offensive Speech in Arabic Social Media Using Transfer Learning"; Applied Sciences, 12(24), 12823, (2022).
- [Chamlertwat et al. 2012] Chamlertwat, W., Bhattarakosol, P., Rungkasiri, T., Haruechaiyasak, C.: "Discovering Consumer Insight from Twitter via Sentiment Analysis"; J. Univers. Comput. Sci., 18(8), 973-992, (2012).
- [Choi et al. 2013] Choi, D., Kim, J., Piao, X., Kim, P.: "Text Analysis for Monitoring Personal Information Leakage on Twitter"; J. Univers. Comput. Sci., 19(16), 2472-2485, (2013).

- [Chowdhury et al. 2020] Chowdhury, S. A., Mubarak, H., Abdelali, A., Jung, S. G., Jansen, B. J., Salminen, J.: "A multi-platform Arabic news comment dataset for offensive language detection"; In Proceedings of the 12th Language Resources and Evaluation Conference, 6203-6212, (2020).
- [Conneau et al. 2019] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Edouard Grave, Ott, E. M. Zettlemoyer, M. Stoyanov, V.: "Unsupervised cross-lingual representation learning at scale"; arXiv preprint arXiv:1911.02116., (2019).
- [Daouadi et al. 2018a] Daouadi, K. E., Zghal Rebaï, R., Amous, I.: "Towards a Statistical Approach for User Classification in Twitter"; In Machine Learning for Networking: First International Conference, 33-43, (Nov 2018).
- [Daouadi et al. 2018b] Daouadi, K. E., Rebaï, R. Z., Amous, I.: "Organization vs. Individual: Twitter User classification"; In International Workshop on Language Processing and Knowledge Management, (2018).
- [Daouadi et al. 2019a] Daouadi, K. E., Rebaï, R. Z., Amous, I.: "Organization, Bot, or Human: Towards an Efficient Twitter User Classification"; *Computación y Sistemas*, 23(2), 273-280, (2019).
- [Daouadi et al. 2019b] Daouadi, K. E., Rebaï, R. Z., Amous, I.: "Bot Detection on Online Social Networks Using Deep Forest"; In Proc. Int. Conf on Computer Science On-line Conference, 307-315, (Apr 2019).
- [Daouadi et al. 2020] Daouadi, K. E., Rebaï, R. Z., Amous, I.: "Real-Time Bot Detection from Twitter Using the Twitterbot+ Framework"; *J. Univers. Comput. Sci.*, 26(4), 496-507, (2020).
- [Daouadi et al. 2021] Daouadi, K. E., Rebaï, R. Z., Amous, I.: "Optimizing semantic deep forest for tweet topic classification"; *Information Systems*, 101, 101801, (2021).
- [Devlin et al. 2018] Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: "Bert: Pre-training of deep bidirectional transformers for language understanding"; arXiv preprint arXiv:1810.04805., (2018).
- [Duwairi et al. 2021] Duwairi, R., Hayajneh, A., Quwaider, M.: "A deep learning framework for automatic detection of hate speech embedded in Arabic tweets"; *Arabian Journal for Science and Engineering*, 46, 4001-4014, (2021).
- [Farha and Magdy 2019] Farha, I. A., Magdy, W.: "Mazajak: An online Arabic sentiment analyser"; In Proceedings of the fourth arabic natural language processing workshop, 192-198, (Aug 2019).
- [Faris et al. 2020] Faris, H., Aljarah, I., Habib, M., Castillo, P. A.: "Hate Speech Detection using Word Embedding and Deep Learning in the Arabic Language Context"; In Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods, 453-460, (Feb 2020).
- [Fouad et al. 2020] Fouad, M. M., Mahany, A., Aljohani, N., Abbasi, R. A., Hassan, S. U.: "Ar-WordVec: efficient word embedding models for Arabic tweets"; *Soft Computing*, 24, 8061-8068, (2020).
- [Guehairia et al. 2020] Guehairia, O., Ouamane, A., Dornaika, F., Taleb-Ahmed, A. (2020). Feature fusion via Deep Random Forest for facial age estimation. *Neural Networks*, 130, 238-252.
- [Haddad et al. 2020] Haddad, B., Orabe, Z., Al-Abood, A., Ghneim, N.: "Arabic offensive language detection with attention-based deep neural networks"; In Proceedings of the 4th workshop on open-source Arabic corpora and processing tools, with a shared task on offensive language detection, 76-81, (May 2020).
- [Husain 2020a] Husain, F.: "Arabic offensive language detection using machine learning and ensemble machine learning approaches"; arXiv preprint arXiv:2005.08946., (2020).
- [Husain 2020b] Husain, F.: "OSACT4 Shared Task on Offensive Language Detection: Intensive Preprocessing-Based Approach"; arXiv preprint arXiv:2005.07297, (2020).

- [Husain and Uzuner 2021] Husain, F., Uzuner, O.: "Transfer Learning Approach for Arabic Offensive Language Detection System—BERT-Based Model". arXiv preprint arXiv:2102.05708, (2021).
- [Joulin et al. 2016] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: "Bag of tricks for efficient text classification"; arXiv preprint arXiv:1607.01759. (2016).
- [Kalampokis et al. 2016] Kalampokis, E., Karamanou, A., Tambouris, E., Tarabanis, K. A.: "Applying Brand Equity Theory to Understand Consumer Opinion in Social Media". J. Univers. Comput. Sci., 22(5), 709-734, (2016).
- [Kalampokis et al. 2017] Kalampokis, E., Karamanou, A., Tambouris, E., Tarabanis, K.: "On Predicting Election Results using Twitter and Linked Open Data: The Case of the UK 2010 Election"; Journal of Universal Computer Science, 23(3), 280-303, (2017).
- [Khezzar et al. 2023] Khezzar, R., Moursi, A., Al Aghbari, Z.: "arHateDetector: detection of hate speech from standard and dialectal Arabic Tweets"; Discover Internet of Things, 3(1), 1, (2023).
- [Maiya 2022] Maiya, A. S.: "ktrain: A low-code library for augmented machine learning"; The Journal of Machine Learning Research, 23(1), 7070-7075, (2022).
- [Miller et al. 2017] Miller, K., Hettinger, C., Humpherys, J., Jarvis, T., Kartchner, D.: "Forward thinking: Building deep random forests"; arXiv preprint arXiv:1705.07366., (2017).
- [Mubarak et al. 2020] Mubarak, H., Rashed, A., Darwish, K., Samih, Y., Abdelali, A.: "Arabic offensive language on twitter: Analysis and experiments"; arXiv preprint arXiv:2004.02192., (2020).
- [Mubarak et al. 2020] Mubarak, H., Darwish, K., Magdy, W., Elsayed, T., Al-Khalifa, H.: "Overview of OSACT4 Arabic offensive language detection shared task"; In Proceedings of the 4th Workshop on open-source arabic corpora and processing tools, with a shared task on offensive language detection, 48-52, (May 2020).
- [Mulki et al. 2019] Mulki, H., Haddad, H., Ali, C. B., Alshabani, H.: "L-hsab: A levantine twitter dataset for hate speech and abusive language"; In Proceedings of the third workshop on abusive language online, 111-118, (Aug 2019).
- [Pedregosa et al. 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: "Scikit-learn: Machine learning in Python"; the Journal of machine Learning research, 12, 2825-2830, (2011).
- [Safaya et al. 2020] Safaya, A., Abdullatif, M., Yuret, D.: "KUISAIL at SemEval-2020 Task 12: BERT-CNN for Offensive Speech Identification in Social Media"; In Proceedings of the Fourteenth Workshop on Semantic Evaluation, (Dec 2020), 2054-2059.
- [Shapiro et al. 2022] Shapiro, A., Khalafallah, A., Torki, M.: "Alexu-aic at arabic hate speech 2022: Contrast to classify"; arXiv preprint arXiv:2207.08557, (2022).
- [Soliman et al. 2017] Soliman, A. B., Eissa, K., El-Beltagy, S. R.: "Aravec: A set of arabic word embedding models for use in arabic nlp"; Procedia Computer Science, 117, 256-265, (2017).
- [Talafha et al. 2020] Talafha, B., Ali, M., Za'ter, M. E., Seelawi, H., Tuffaha, I., Samir, M., Farhan, W., Al-Natsheh, H. T.: "Multi-dialect arabic bert for country-level dialect identification"; arXiv preprint arXiv:2007.05612., (2020).