



Article

A Deep Learning Approach for Automatic Hate Speech Detection in the Saudi Twittersphere

Raghad Alshalan *  and Hend Al-Khalifa * 

Department of Information Technology, College of Computer and Information Sciences, King Saud University, Riyadh P.O Box 12371, Saudi Arabia

* Correspondence: rsalshalan@gmail.com (R.A.); hendk@ksu.edu.sa (H.A.-K.)

Received: 24 October 2020; Accepted: 23 November 2020; Published: 1 December 2020



Abstract: With the rise of hate speech phenomena in the Twittersphere, significant research efforts have been undertaken in order to provide automatic solutions for detecting hate speech, varying from simple machine learning models to more complex deep neural network models. Despite this, research works investigating hate speech problem in Arabic are still limited. This paper, therefore, aimed to investigate several neural network models based on convolutional neural network (CNN) and recurrent neural network (RNN) to detect hate speech in Arabic tweets. It also evaluated the recent language representation model bidirectional encoder representations from transformers (BERT) on the task of Arabic hate speech detection. To conduct our experiments, we firstly built a new hate speech dataset that contained 9316 annotated tweets. Then, we conducted a set of experiments on two datasets to evaluate four models: CNN, gated recurrent units (GRU), CNN + GRU, and BERT. Our experimental results in our dataset and an out-domain dataset showed that the CNN model gave the best performance, with an F1-score of 0.79 and area under the receiver operating characteristic curve (AUROC) of 0.89.

Keywords: abusive language; Arabic; Arabic tweets; hate speech detection

1. Introduction

In the Arab region, Twitter is considered to be one of the most popular platforms used by Arabic-speaking users [1], and it has indeed revolutionized the way people communicate and share opinions, ideas, and information. However, due to the dynamic, democratic, and unrestricted nature of the Twitter platform, it has been increasingly exploited for the dissemination of aggressive and hateful content. While there is no formal definition of hate speech, there is a general agreement among scholars and service providers to define it as any language that attacks a person or a group on the basis of a characteristic such as race, color, ethnicity, gender, or religion [2].

Despite the legal efforts and the established regulations to counter illegal hate speech, the massive scale of social media platforms still requires an automatic solution for detecting such a toxic phenomenon. The pressing need for effective automatic solutions for hate speech detection has attracted significant research efforts recently. Several studies have relied on traditional machine learning approaches and leveraged surface features such as bag of words, and word and character n-grams, which have been shown to be very effective for hate speech detection [3–6]. Recently, there has been a clear trend towards the adoption of deep learning methods, which indeed showed better performance over classic methods in hate speech detection tasks [7–11]. However, there is a very limited work that employed deep learning approaches to address the problem in Arabic language [1]. To the best of our knowledge, Albadi et al.'s [12] work is the only Arabic study published in the area, wherein it examined gated recurrent units (GRU) architecture, focusing mainly on religious hate speech.

While different deep learning models have been investigated in this area, we are more interested to explore these methods for the Arabic language. We believe that the special nature of the Arabic language and its richness and complexity at morphological, lexical, and orthographical levels [13] pose some unique challenges that could complicate the task of detecting hate speech. Moreover, the high variety of dialectal Arabic used by Arabic users on social media makes the problem even more complex. In fact, dialects of Arabic are manifold, varying not only from country to country but also within the same country, resulting in many similarly spelled words having different meanings across different dialects and regions.

Recently, Devlin et al. [14] introduced BERT (bidirectional encoder representations from transformers), a new language representation model that has been successfully applied to numerous NLP tasks, achieving state-of-the-art results for 11 NLP tasks, including sentiment analysis, question-answering, and textual entailment. However, works examining BERT effectiveness for hate speech detection in Arabic specifically are very limited.

To this end, this paper presents a set of experiments to investigate the merit of using different neural networks including recurrent neural networks (RNNs) and convolutional neural networks (CNNs) and their variants for hate speech detection in Arabic. We also evaluated BERT on our downstream classification task by using the pre-trained BERT model, adding a simple task-specific layer and then fine-tuning the entire model on the hate speech detection task. Given the fact that Arabic is considered as a low-resourced language, this paper also described a new approach for creating a hate speech dataset for Arabic, covering racist, religious, and ideological hate speech. Following this approach, we created a new dataset containing a total of 9316 tweets labeled as normal, abusive, or hateful, as well as a new annotation guideline.

The main contributions of this work are threefold:

1. Constructing a public dataset of 9316 tweets labelled as hateful, abusive, and normal.
2. Comparing the performances of three neural network models—CNN, GRU, CNN + GRU—for the Arabic hate speech detection task.
3. Evaluating the recent language representation model BERT for the Arabic hate speech detection task.

The remainder of the paper is organized as follows. Section 2 reviews works related to hate speech detection. Section 3 describes the data construction process. Section 4 describes the experiment setup, including a detailed description of the investigated models' architectures and the preprocessing steps. Section 5 details the experimental setup, and the results are discussed in Section 6. Finally, Section 7 concludes our work and discusses future directions.

2. Related Work

Abusive language in social media is a complex phenomenon with a wide spectrum of overlapping forms and targets. Hate speech, offensive language, and cyberbullying are examples of abusiveness, and several works in the literature have been conducted to detect and locate such types of language.

Recently, researchers have shown an increased interest in automatic hate speech detection in social media. Most of the existing studies in the literature have primarily modeled the problem as a supervised classification task, whether using classical machine learning approaches or deep neural network approaches [2,6].

Machine learning approaches: As for classical machine learning approaches, there are several studies that leveraged simple surface and linguistic features such as bag of words (BOW), n-grams, and POS(part-of-speech) as fundamental features. Schmidt and Wiegand [2] presented a review of the various features that were used for the task in the literature. One of the earliest studies that addressed the problem of hate speech detection was presented by [15], which focused on detecting anti-Semitic language as the specific type of hate speech. Kwok and Y. Wang [16] proposed a supervised machine learning model to detect racist hate speech against black people in Twittersphere and they

suggested improving the model by considering other features such as sentiment features and bigrams. Waseem and Hovey [3] presented a study to detect hate speech in the Twitter platform, specifically targeting two types of hate: racism and sexism. They investigated several features including character n-gram along with demographic, lexical, and geographic users' features. The results showed that using character n-grams of lengths up to four provided the best results, and incorporating gender as an additional feature could yield slight improvement. They also built and published a dataset of 16,000 annotated tweets that has become a well-known benchmarking dataset for following studies. Word embeddings as features were also investigated by Djuric et al. [17]. They used paragraph2vec to learn low-dimensional text embeddings to classify users' comments as abusive or clean. Compared with BOW representations, the experimental results on Yahoo! Finance user comments showed that paragraph2vec representation performed better in term of area under the curve (AUC). Following this study, Nobata et al. [18] incorporated various features such as n-grams, linguistic and syntactic features, and distributional semantics in order to detect abusive language in online user comments. The results on Yahoo! Finance user comments showed that combining all features led to outperformance of the prior art in terms of AUC. Davidson et al. [4] trained a multi-class classifier to distinguish between offensive and hate language in tweets. They also built and published a dataset of tweets labelled for three categories: hate speech, offensive language, or neither, resulting into 24,802 labelled tweets.

There are also several works that tackle the problem of hate speech in other languages using classical machine learning methods. Jaki and De Smedt [5] targeted the hate speech occurring in right-wing German tweets. Bohra et al. [19] experimented with several features to detect hate speech in Hindi-English code-mixed tweets. De Smedt et al. [20] presented a study that tackles the problem of online Jihadist hate speech written in many languages in Twitter.

Deep learning approaches: In recent years, attention has shifted toward deep learning approaches to tackle the problem of hate speech detection. Most of the works have employed various architectures of convolutional neural network (CNN) and recurrent neural networks (RNN). Park and Fung [7] proposed a two-step classification approach by combining two classifiers: one to classify the text to abusive or not, and another to classify the text into a specific type of abusive language (sexist or racist) given that the text is abusive. The evaluation was conducted using a benchmark dataset provided in [3], and the two-step approach compared to the one-step approach showed a promising result. On the same benchmark dataset, Gambäck and Sikdar [8] proposed the use of a CNN classifier with word2vec word vectors, and their results outperformed the prior art. Pitsilis et al. [10] built an ensemble of LSTM-based classifiers to detect hate speech content in Twitter. They experimented the proposed method with the existing dataset provided by Waseem and Hovey [3]. The results showed that the proposed approach achieved better results compared to the state-of-the-art approaches. Zhang et al. [11] used a CNN + GRU neural network model to detect hate speech, building their own dataset from Twitter by searching tweets related to religion and refugees in general. For evaluation, they used a total of seven public datasets including their dataset (2435 annotated tweets) and compared the performance of the proposed CNN + GRU against several state-of-the-art baselines. The results showed that the CNN + GRU model achieved the highest F1 on all datasets. Badjatiya et al. [9] investigated three different neural models for hate speech detection. The proposed models were experimented on the Waseem and Hovey dataset [3], and the results showed that embeddings learned from the LSTM model when combined with GBDT (gradient boosting decision tree) led to the best accuracy result.

Cross-domain abusive language detection has also been a topic of interest recently, in which knowledge can be transferred between different abusive domains (hate speech, offensive, aggression, etc.). In [21], authors used nine publicly available datasets that focused on different abusive language types such as aggressive behavior, hate speech, and toxic comments. These datasets have different labels, however, the authors binarized the labels on all datasets into positive (abusive language) and negative (non-abusive language). To assess the model generalization, the authors trained a support vector machine (SVM) classifier using a training split of one dataset and then tested it on the testing splits of other datasets. Results showed, expectedly, that the performance on out-domain data declined

by more than 50% of F1 score in most cases. Pamungkas et al. [22] also presented cross-domain experiments using two models: linear support vector classifier (LSVC) with unigram representation and LSTM neural model, and they concatenated a hate lexicon (HurtLex). Using four English datasets, results showed that the performance on out-domain datasets was always lower. The results also showed that LSTM performed better than LSVC, and that using HurtLex helped in boosting the recall in most cases.

Arabic studies that address the problem of hate speech detection are still limited; however, there are several studies that address some related problems such as detecting ISIS (Islamic State of Iraq and Syria) support messages [23] and identifying vulgar, obscene, offensive, or inflammatory language [24,25]. A recent dataset for Levantine hate speech and abusive detection was published by Mulki et al. [26]. This dataset contained 5846 tweets, labeled as “hate”, “abusive”, or “normal”, collected from Twitter using terms of potential entities that are usually targeted by abusive/hate speech. To the best of our knowledge, there is only one study that tackled the problem of automatic Arabic hate speech detection and it was presented by Albadi et al. [12]. In this study, the authors proposed an RNN architecture with gated recurrent units (GRU) and pre-trained word embeddings to detect religious hate speech in Arabic tweets. For evaluation, they created a dataset containing 6000 Arabic tweets, 1000 for each of the six religious’ groups (Muslims, Jews, Christians, atheists, Sunnis, and Shia). The tweets were annotated as hate or non-hate tweets. The results showed that the GRU-based RNN model outperformed other traditional classifiers with respect to all evaluations. They also published their dataset as the first and the only available hate speech dataset for the Arabic language.

3. Dataset Construction

In this work, we built a new dataset that covers different types of hate speech, e.g., racist, religious, and ideological hate speech. We used the standard Twitter search/streaming API with *tweepy* [27] python library to collect our data. The data was collected in span of 6 months, from March 2018 to August 2018, using a keyword-based and thread-based search. In the keyword-based approach, we included in the search query a total of 164 impartial unique terms (excluding the various morphological variations of the same term) that refer to groups or to people belonging to groups that are likely to be targeted by hate speech. For example, we used terms that refer to people who belong to different tribes (such as “Otaibi”, “عتيبي”) and different regions (“Hijaz”, “حجازي”) to collect tweets related to tribal and regional hate speech, respectively. For the thread-based search, we included in the search query different hashtags that discusses controversial topics that are considered to be strong indicators of hate speech. We monitored twitter trends during the period of the data collection, and we ended up with a total of 10 hashtags used for data retrieving. Examples of the used terms and hashtags are illustrated in Table 1. Since hateful tweets are generally less common than natural tweets, we further boosted our dataset and improved the representation of the hate class by using the top-scored terms in the lexicon of religious hate terms released by Albadi et al. [12]. For all queries, we collected only original and Arabic tweets, excluding retweets and non-Arabic tweets from the search. In total, we retrieved 54 million tweets, from which we sampled 10,000 tweets for annotation. To obtain the annotations for the dataset, Figure Eight (now Appen) [28] crowdsourcing platform was utilized to recruit both crowd and internal workers (volunteers and freelancers). Before submitting the task to the annotators, we generated an annotation guideline for hate speech annotation with the help of experts in interreligious dialogue, Islamic jurisprudence, and media studies. The guideline is available online on GitHub [29]. The entire annotation process was completed in a course of 3 weeks, starting from 1 to 23 September 2019. The dataset was submitted to the annotators in different batches. The first batch (1000 tweets) was annotated by crowd workers. The second batch (4000 tweets) was annotated by 15 different Saudi annotators. The third and final batch (5000 tweets) was annotated by three freelancers who are familiar with the Saudi dialect.

Table 1. Examples of terms used for data retrieving.

Hate Speech Type	Example of Terms/Hashtags
Racist	عرب الشمال, الحجاز_هوية Northern Arab, Hijaz identity
Regional	نجدي, قيصيمي, حجازي Najdi, Qassimi, Hijazi
Tribal	عتيبي, زهراني, قحطاني Otaibi, Zhrani, Qahtani
Inter-religious	الوهابية, الشيعة, الأشاعرة Wahhabism, Shia Asharies
Ideological	ليبرالية, يسارية, نسوية Liberalism, leftism, feminism

4. Experiment Setup

In this section, we present the main components of our proposed approach to detect Arabic hate speech. As a first step, the tweets pass through various preprocessing steps to clean and prepare the data for the training phase (see Section A). Then, several classification models for detecting hate speech in Arabic texts are investigated. In this work, we evaluated three neural network architectures: CNN-based classifier, GRU-based classifier, and finally a classifier that combines both CNN and GRU. Moreover, we evaluated BERT [14], a recent language representation model, on the task of hate speech detection.

4.1. Data Preprocessing

Before feeding the tweets as an input to the classification models, we firstly applied several pre-processing steps on both testing and training datasets. Pre-processing is a vital step when dealing with data that has a noisy and colloquial nature such as Twitter. This becomes even more important for the Arabic language, given its inherited ambiguity and the high variety of the used dialectal Arabic in Twittersphere. We applied several preprocessing steps to our data before feeding them as input to our experimented models. Steps include:

Hashtag removal: This step was taken to avoid any bias toward certain classes; we dropped any hashtag used as a keyword search while collecting the data from the tweets.

Spam filtering: In this step, we implemented the rule-based algorithm proposed in [30] to filter spam. We used the same spam lexicon created by the authors and extended it with new terms and hashtags that we found to be highly associated with spam tweets.

Stop words removal: In this step, we used a list of 356 stop words that were made publicly available by [12] on GitHub [31].

Emoji description: A recent work on irony detection and sentiment analysis of tweets published by Abhishek et al. [32] showed that replacing emojis with their textual description is more effective for tweet classification than using other strategies such as using pretrained emoji embeddings or treating emojis as regular tokens. Motivated by this work, we used demoji [33], a python library that provides utilities to replace emojis with their description. This library utilized readily available lists of emojis and their textual descriptions in English [34]. For our task, we translated the list into Arabic by Google translator API [35]. Then, the produced translations were manually revised and updated. The translated list was used with demoji to replace emojis in tweets with their Arabic textual description.

Cleaning: In this step, punctuation, additional whitespaces, diacritics, and non-Arabic characters are removed.

Normalization: The goal of this step is to reduce orthographical variations observed in tweets as well as normalizing Twitter-specific tokens. The normalization steps are as follows:

- Different forms of “ل” (“لّ”, “لُ”, and “لٌ”) were replaced by “ل”.
- “ى” is replaced by “ي”.
- “ة” is replaced by “ه”.
- Links and mentions are replaced by “mention” and “URL”, respectively.

Lemmatization: We applied this step using the Farasa stemmer [36]. We selected Farasa on the basis of a study by El Mahdaouy et al. [37] that showed that Farasa outperformed other tools such as MADAMIRA.

4.2. Neural Network Models

In this section, we describe the architectures of our evaluated neural network models (CNN, GRU, and CNN + GRU). In all models, the features are represented as word embeddings through Word2Vec [38]; the feature representation is described in detail in the next subsection. Next, these embeddings are passed into the neural network model (CNN, GRU, or CNN + GRU). We described each architecture model in the following subsections.

4.2.1. Feature Representation

The first layer of all proposed architectures is an embedding layer that maps each tweet, represented as a sequence of integer indexes, to a 300-dimension vector space using a pretrained word2vec model [38]. We trained our own word2vec model using the Continuous Bag of Words (CBOW) training algorithm. We used a subset of our collected data as training collection. The collection had a total of 17.6 million tweets and 536,000 tokens. Before training, all tweets were preprocessed following the same preprocessing steps described earlier. As for the model’s hyperparameters, we selected a window of size 3, since the length of tweets is generally small. We set the vector dimensions to 300 and the other hyper-parameters to their defaults. To prepare the tweets for the embedding layer, we started by creating a vocabulary index based on word frequency where every unique word in our dataset was assigned a unique integer value (0 was reserved for padding). Then, we converted each tweet to a sequence of integer indexes and padded them to ensure that all sequences had the length of the longest tweet in the dataset. Then the tweets’ sequences were fed into an embedding layer that mapped word indexes to a pretrained word embeddings. The output of the embedding layer was an ($m \times 300$) tweet matrix, where m is the length of the longest tweet in the dataset.

4.2.2. CNN Architecture

The first model we evaluated was a CNN model inspired from Kim’s work [39] for text classification. Our CNN architecture, as illustrated in Figure 1, contains five layers: an input layer (embedding layer); a convolution layer, which basically consists of 3 parallel convolution layers with different kernel sizes (2, 3, and 4); a pooling layer; a hidden dense layer; and finally the output layer. As we mentioned before, all tweets are mapped into 300-dimensional real valued vectors by the embedding layer. Let m be length of longest tweet, the embedding layer then passes an input feature matrix with a shape of $m \times 300$ to a dropout layer with a rate of 0.2. The main purpose of dropout layer is to reduce the problem of overfitting. Then, the output is received by the three parallel convolution layers. Each layer has 100 filters with kernel sizes of 2, 3, and 4 and a rectified linear unit function for activation. As part of convolution operation, each layer applies the 100 filters and produces 100 new feature maps with a size of $(m - (\text{kernel size} - 1) \times 100)$, passed into a dropout layer with a rate of 0.2. After that, these convolution features are fed as input to a max pooling layer (global) for down sampling, producing 3 (1×100) vector outputs. These 3 vectors are then concatenated and passed as input to

a fully connected dense layer containing 50 neurons followed by a dropout layer with a rate of 0.2. The output is then fed to the output layer with sigmoid activation to produce the final predictions.

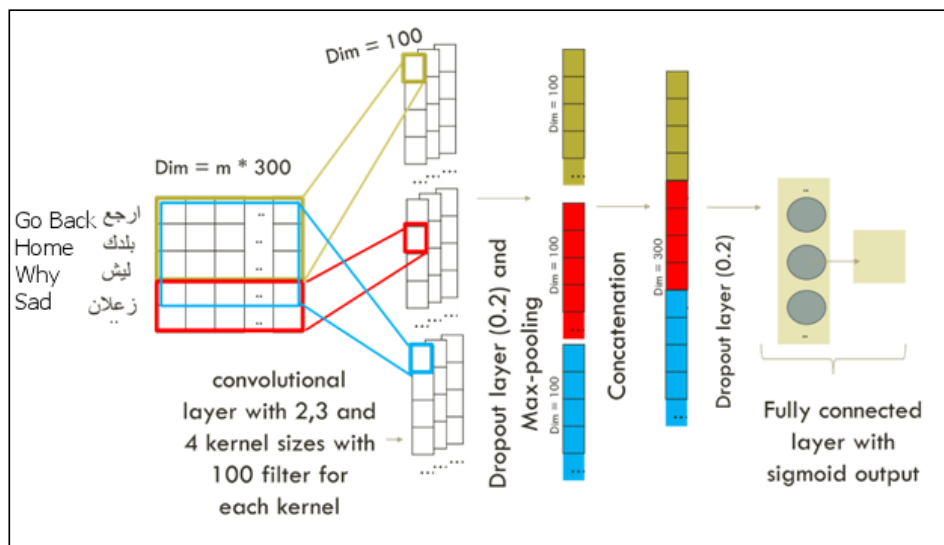


Figure 1. Convolutional neural network (CNN) architecture.

4.2.3. GRU Architecture

Currently, GRU is the state-of-the-art model for Arabic hate speech detection. In this experiment, we employed a GRU-based network similar to the one applied in [12]. Our network architecture, as illustrated in Figure 2, contains 4 layers: an input layer (embedding layer), a GRU layer, a hidden dense layer, and finally the output layer. Similar to CNN architecture, the inputs (tweets) are fed into the embedding layer, which maps tweets’ tokens into a 300-dimensional real valued vector. Letting m be the length of the longest tweet, the embedding layer produces an output matrix with a shape of $(m \times 300)$ passed into a dropout layer with a rate of 0.2 to reduce the problem of overfitting. Then, the output of the dropout layer is passed into a GRU layer with 250 units to model long-range contextual information, producing an output with a shape of (1×250) . This output is then passed to a dense layer with 50 neurons and rectified linear unit function for activation. After that, a dropout layer with a rate of 0.2 is used for regularization. The final layer is a sigmoid layer that produces the final predictions.

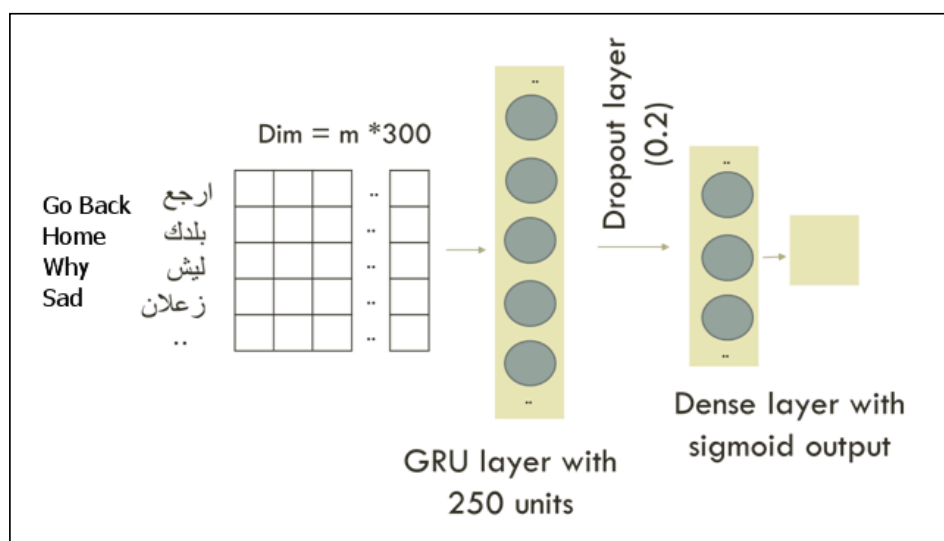


Figure 2. Gated recurrent unit (GRU) architecture.

4.2.4. CNN + GRU Architecture

The third experimented architecture is a combination of both CNN and GRU, as illustrated in Figure 3. In this architecture, CNN is used as a feature extractor that passes the “extracted feature” to a GRU layer, which treats the generated feature dimension as timesteps. The CNN + GRU architecture contains 6 layers: an input layer (embedding layer), a convolution layer with 100 filters and a kernel size of 4, a max pooling layer, a GRU layer, another max pooling layer, and finally the output layer. The embedding layer starts by mapping the tweets into a 300-dimensional vector space, producing an $(m \times 300)$ tweets matrix, with m being the length of longest tweet. This output matrix is then passed to a dropout layer with a rate of 0.2 to avoid the overfitting problem. Then, the output of the dropout layer is fed into the convolution layer, which has 100 filters with kernel sizes of 4. The rectified linear unit function is used for activation. The filters perform convolution on the tweets matrix and generates $(n \times 100)$ feature maps, where $n = m - (\text{kernel size} - 1)$. The feature maps are then passed into a dropout layer with a rate of 0.2 and a max pooling layer with a pool size of 4. This produces vector output, which can be considered as “extracted features”. These extracted features are then passed to the GRU layer with 200 units, followed by a dropout layer with a rate of 0.2. Finally, the output of the dropout layer is then fed into the output layer with sigmoid activation to produce the final predictions.

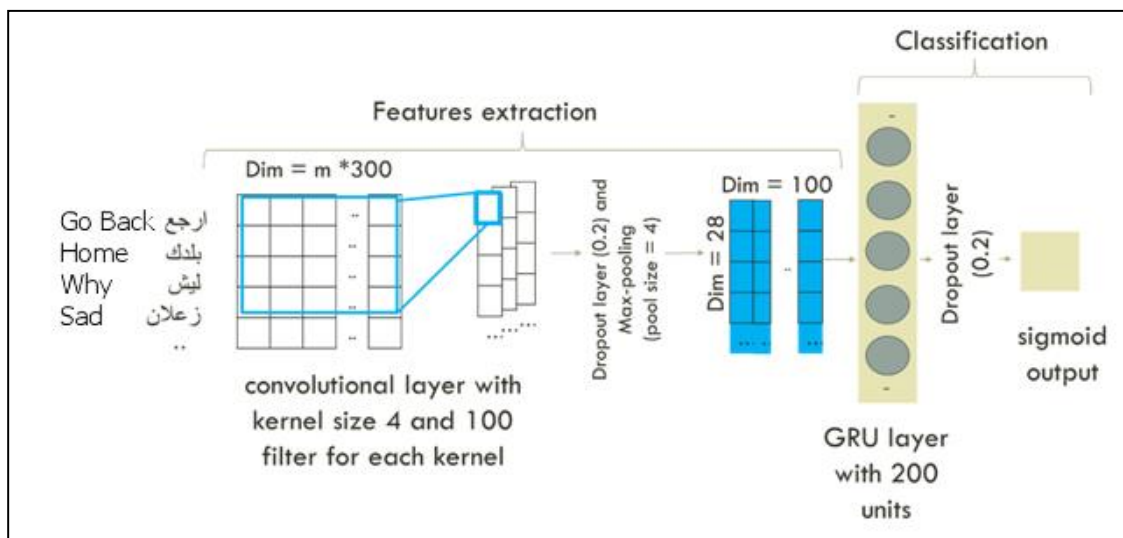


Figure 3. CNN + GRU architecture.

4.2.5. BERT Model

BERT [14] is a novel transformer approach that has achieved state-of-the-art results on various NLP tasks. Therefore, we aim in this work to evaluate how the BERT pre-trained model performed in the Arabic hate speech detection task. BERT was released in multilingual versions, pre-trained on monolingual corpora (Wikipedia) in 104 languages including Arabic. In this experiment, we fine-tuned the cased version of the multilingual pre-trained model by adding a simple classification layer that performs a binary classification to classify the tweets into hate or non-hate. To prepare our data for BERT, we examined 2 different preprocessing procedures. The first one followed the same steps described in Section 5. The second one is a lite version of the first one in which stop words, punctuation, and non-Arabic words are kept. The lite preprocessing showed a better result in terms of F1 score on fivefold cross validation on the training set. The next step was to prepare our pre-processed data to be in BERT format. To do so, we added a special token to mark the beginning (CLS) and separation/end (SEP) of each tweet. Next, we tokenized our inputs into tokens that corresponded to BERT’s vocabulary. This input was fed into BERT and then into the final classification layer, where all of the parameters were fine-tuned end-to-end using our labeled data.

5. Experiments

We conducted a set of experiments to evaluate the proposed models for Arabic hate speech detection tasks (CNN, GRU, CNN + GRU, and BERT). In all experiments, we performed a binary classification task in which tweets were classified to hate or non-hate classes. In this section, we describe the datasets used in our experiments and the experiment setup including the models' implementation, hyperparameter tuning, baselines, and evaluation metrics.

5.1. Datasets

In all experiments, we used our created dataset (as described earlier in this paper) to train the proposed models for hate speech detection. The dataset contains a total of 9316 tweets classified as hateful, abusive, or normal (not hateful or abusive). Since our task was to perform a binary classification to classify the tweet into hate or non-hate speech, we only kept tweets annotated as hateful or normal. This left us with 8964 tweets that were split 75–25% for training (6425 instances) and testing (2539 instances). We used the 75% for training and hyperparameter-tuning through cross-validation and then we tested the optimized models on the 25% held-out data. Hereinafter, we refer to this dataset as general hate speech dataset (GHSD).

To investigate to what extent the models trained on our dataset generalize to other datasets that focused on different/specific types of hate speech, we used Albadi et al.'s [12] religious hate speech dataset (referred as religious hate speech dataset (RHSD) hereinafter) to test our models. In particular, we trained our proposed models using the whole GHSD dataset and used the testing set of RHSD that contained 600 tweets to test the models. Statistics of the used datasets are shown in Table 2.

Table 2. Statistics of datasets used in the experiments.

	GHSD	RHSD (Testing)
# Tweets	8964	600
# Hate instances	2539	250
# Non-hate instances	6425	350

5.2. Experimental Settings

Implementation: We implemented all neural network models using Keras library with a TensorFlow as a backend [40], while we implemented BERT using huggingface Pytorch library [41]. We ran the experiments on Google Collaboratory [42], which provides a free Jupyter notebook environment with GPU accelerator.

Hyperparameter tuning: In all experimented models, we employed a grid search using fivefold cross-validation on the training set to find the optimal training-specific hyperparameters, such as the batch size, class weights, and learning rate, with F1-score as a scoring metric. We also used a grid search using fivefold cross-validation to find the best model-specific hyperparameters such as the number of hidden layers, the number of units, pooling layer size, and the dropout rate. However, due to the limitation of time and computational resources, some of our model-specific hyperparameters were selected on the basis of empirical findings reported previously, determined empirically through cross-validation over the training dataset (without grid search) or set to default values. All proposed network architectures were experimented with binary cross entropy as a loss function and “Adam” as an optimizer. We modified the loss function to incorporate class weights, since our dataset is relatively imbalanced, and we were particularly interested in increasing the recall of the positive class (hate class). Then, we evaluated all network architectures with and without incorporating class weights.

When fine-tuning the BERT model, all hyperparameters were kept the same as in pretraining except batch size, learning rate, and number of training epochs. To select the best performing values, we performed a grid-search through fivefold cross-validation on a training set with F1 as a scoring metric. For batch size and number of epochs, we searched according to the suggested ranges in [14].

For the learning rate, we found that the suggested range produced poor results, and thus we extended the searched range to be $\{0.2 \times 10^{-5}, 0.3 \times 10^{-5} \dots 3 \times 10^{-5}, 5 \times 10^{-5}\}$.

5.3. Baselines and Evaluation Metrics

For our baselines, we used SVM (support vector machine) and LR (logistic regression) classifiers because they have been shown to be effective in previous studies [3,4,43]. For features, studies showed that n-gram-based models performed very well in hate detection tasks (word and characters). Therefore, for both classifiers, we tested different n-gram features with 5-fold cross-validation on the training set. We found that character n-gram features ($n = 1-4$) yielded the highest F1 score. We used Python scikit-learn library [44] to implement both models. Optimal combination of the model hyperparameters was optimized using a grid search using fivefold cross-validation on the training set and with F1 scoring metric.

For evaluation, we calculated true negatives (TN), true positives (TP), false negatives (FN), and false positives (FP) by comparing predicted and true labels. Then, we used macro-averaged precision (P), recall (R), accuracy, and F1-measures as in the following equations:

$$P = TP / (TP + FP)$$

$$R = TP / (TP + FN)$$

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

$$F1 = 2 \times (P \times R) / (P + R)$$

We also used area under the receiver operating characteristic curve (AUROC), which can be used as an indicator of how good the model is for distinguishing the given classes. ROC is a curve with true positives rate (TPR, sensitivity) plotted on the vertical axis and false positives rate (FPR = 1 - specificity) plotted on the horizontal axis for various thresholds.

Moreover, since our hate speech data collection is relatively imbalanced and given the serious consequences of failing to detect a hateful content, we also reported our results using positive class recall.

6. Experiments Results

In this section, we discuss the results of two sets of experiments. The first one is the in-domain experiments, where all models were trained on the training set of our created dataset (GHSD) and tested on its testing set. We also performed a set of out-domain experiments to assess how well our models, when trained on one dataset, generalize to work on different datasets. For that purpose, we trained all the models on the whole GHSD dataset, then we used the Albadi et al. [12] test set (RHSD) for testing.

6.1. Results and Discussion

Table 3 summarizes the best results achieved by our evaluated models: CNN, GRU, CNN + GRU, and BERT in terms of precision, recall, F1-score, accuracy, hate class recall, and AUROC. The left part of table shows the results of the in-domain experiments. The right part of the table illustrates the out-domain experiment results.

Since neural networks are stochastic in nature, we expected some variations in the models' performance at every different run. Therefore, we performed 10 runs for each experiment and reported the average of all metrics (we limited the number of runs to 10 given the time constraint on the project).

Baselines performance: As we mentioned in the previous section, we performed a grid search over different possible features for SVM and LR classifiers (word and character n-grams with different lengths), and we found that character n-grams ($n = 1-4$) outperformed other features. Thus, in Table 3, we reported the results obtained by our baselines (SVM and LR) using character n-grams ($n = 1-4$) only. Results in all metrics show that SVM and LR classifiers achieved almost the same performance on

both datasets (GHDS and RHSD). We can also notice that both classifiers achieved considerably high AUROC scores, suggesting that they were highly capable of distinguishing between the two classes.

Table 3. Evaluation results of the experimented models. The best result for each metric is boldfaced *.

Model	GHSD (In-Domain)						RHSD (Out-Domain)					
	P	R	F1	A	HCR	AUROC	P	R	F1	A	HCR	AUROC
SVM (char n-grams)	0.74	0.74	0.74	0.78	0.63	0.85	0.70	0.66	0.66	0.68	0.46	0.73
LR (char n-grams)	0.75	0.74	0.75	0.79	0.63	0.84	0.68	0.66	0.65	0.68	0.47	0.72
CNN	0.81	0.78	0.79	0.83	0.67	0.89	0.72	0.69	0.69	0.70	0.56	0.79
GRU	0.80	0.77	0.78	0.82	0.63	0.87	0.70	0.65	0.64	0.68	0.43	0.76
CNN + GRU	0.80	0.76	0.77	0.82	0.62	0.88	0.74	0.67	0.67	0.70	0.44	0.77
BERT	0.76	0.76	0.76	0.80	0.65	0.76	0.63	0.60	0.59	0.63	0.38	0.6

* We denoted precision as P, recall as R, accuracy as A, and hate class recall as HCR.

Neural model performance on GHDS: As Table 3 shows, we experimented with three neural models, namely, CNN, GRU, and CNN + GRU, and compared their performance against our baselines. Generally, the results showed that all models outperformed the baselines by approximately 3–5% in all metrics. The only exception was the hate class recall for both GRU and CNN + GRU, in which they achieved similar and 1% lower results than the baselines, respectively. It can be noticed that CNN model achieved the best results among the three neural network models. It showed a consistent improvement in all metrics with a high AUROC score. It also achieved the highest hate class recall compared to other models. The performance improvement achieved by CNN can be attributed to its ability at extracting local and position-invariant features such as surrounding words and word orders. These features can work very well on tasks such as hate speech, since hate speech, much similar to sentiment, can be determined by some terms and multi-word phrases. GRU also showed very similar results to those achieved by CNN, with a minor decrease (only 1%) in the performance with respect to all metrics except for the hate class recall. While not expected, stacking the GRU layer on the top of the CNN architecture led to a slight drop in CNN performance. We suspect that the reason of the performance decline is the complexity of the CNN + GRU architecture, which may not be able to work very well on relatively small datasets. As a final note, neural network models were shown to be more effective in the task of hate speech detection, with this being attributed to their abilities to capture more contextual information and long-term dependencies, and hence a deeper understanding of the texts. However, it is worth mentioning that our baselines were also able to achieve very competitive results, specifically in terms of AUROC and hate class recall. This confirms previous studies that show that character n-grams can be a highly predictive feature in hate speech detection.

BERT performance on GHDS: We fine-tuned the BERT model for the binary text classification task by adding a simple classification layer. From Table 3, we can notice that BERT offers a slight performance improvement over the baselines with respect to almost all metrics. However, it showed a large drop (approximately 10%) in terms of AUROC compared to the baselines. Compared against our neural network models, BERT failed to provide any improvement in the classification performance. This was not expected, since the BERT model has been proven to be very powerful, achieving state-of-the-art results in many NLP tasks such as sentiment analysis, question-answering, and textual entailment [14]. The poor performance of BERT can be attributed to the fact, as stated by Al-Twairesh et al. [45], that BERT was trained on a different dataset genre (Wikipedia). In Wikipedia, pages are written in MSA (Modern Standard Arabic), whereas on Twitter, tweets are mostly written in dialectal Arabic. The linguistic variations between MSA and dialectal text might cause BERT to not perform very well on our tweet datasets.

Out-domain experiments: The right part of Table 3 shows the results of the out-domain experiments. The main purpose of these experiments was to assess how well our models can generalize across different datasets. Not surprisingly, the performance of all models, including the baselines, was always lower on the out-domain dataset. While being always lower, results of the out-domain experiments were very consistent with the results obtained in the in-domain experiments. Specifically, CNN in both sets of experiments achieved the best results compared to the baselines and other models in almost all metrics. Moreover, GRU and CNN + GRU showed comparative results. However, in out-domain experiments, BERT failed to beat the baselines with respect to all metrics. There are many reasons that could cause the consistent drop in the classification performance of all experimented methods on the RHSD dataset. Firstly, RHSD focuses on a different type of hate speech, namely, religious hate speech. This type of hate speech was not directly considered while collecting the GHSD dataset (used for training), which focuses on a wider range of hate speech types including racist, ideological, and inter-religious hate speech. In fact, this could cause the two datasets to have different types of swear/abusive/offensive terms and also different target names. This can greatly impact the lexical distribution of the two datasets, hence impacting how/what the models learn as predictive features. To investigate this, we firstly generated two lexicons from our dataset (GHSD) following two corpus statistical-based approaches: chi-square (χ^2) and pointwise mutual information (PMI), which leverages the labeled corpus in order to learn a domain-specific lexicon and measures the association strength between a term and a class (hate or non-hate classes, in our case). More details on how χ^2 and PMI values are calculated can be found in [46]. Then, we examined the lexicons generated from RHSD by Albadi et al. [12] and from GHSD and analyzed the terms with high positive scores (strongly associated with hate class). We found that some of the highly scored terms in RHSD had a very low score (nearly zero) in GHSD. Examples of these words are the religious affiliations such as “Christian/نصراني” and “polytheist/مشرک”. Besides the lexicon differences, we notice that tweets in the RHSD testing set were mostly written in MSA, while tweets in the GHSD dataset were mostly written in the Saudi dialect. The differences of the linguistic characteristics between MSA and dialects could also affect the ability of our in-domain models to generalize well. In the next section, we attempt to analyze the error produced by our evaluated models to gain deeper understanding of the limitation and the challenges of the hate speech detection task.

6.2. Error Analysis

To understand the challenges of this task, we carried out further analysis of the errors made by our models, including the baselines. In the task of hate speech detection, we are more interested in “detecting” all hateful instances than being precise in our predictions. Given that, we tried to understand why our models failed at detecting hateful tweets. We identified the hateful tweets that all evaluated models on both datasets predicted incorrectly (misclassified as non-hate). We ended up with a total of 151 tweets for GHSD and 51 tweets for RHSD. We further qualitatively analyzed these tweets to understand the possible reasons that they were misclassified by the models. We found that nearly 90% of the misclassified tweets did not have offensive/abusive terms—this could confuse the classifiers and cause its performance to be degraded. We also found that many tweets would even be difficult for humans to classify as hate without being provided with the full context of the tweet. For example, some of the tweets contain comments on images or videos, which will be very tricky to interpret without seeing the associated media. The lack of the context problem also appeared with tweets that refer to external links or quoted tweets. An example is the following tweet that quoted another tweet with this comment: “This is serious, this means that those people will be back to work in secret so that no one will pay attention to them”. This tweet was written by an extremely patriotic user who uses “those” to refer to foreigners working in Saudi Arabia, indicating they have some agenda to call for separation of the Hijaz region from the rest of the country. This tweet is difficult to interpret correctly without knowing what it quoted exactly. Mentions also have the same problem—some mentions contain only few natural words, but it should be classified as hate when we take into account the original tweet that it replied

to. For example, this tweet “@user Qahtani freshman” replied to an offensive tweet that negatively stereotypes a particular tribe’s members, but the classifiers incorrectly classified it as non-hate. Finally, we found that some tweets were indirect, implicit, or sarcastic, which make it difficult for automatic solutions to detect correctly. For example, this sarcastic tweet “*And now we introduce to you, Mohammed bin Salman’s play ‘so free’ in the era of the great Saudi*” was misclassified as non-hate, but it was written by non-Saudi user against a Saudi user as a part of hateful conversation between them.

We also performed further analysis to understand the cases in which the classifiers misclassified non-hate tweets as hateful. We ended up with 33 misclassified non-hate tweets in GHSD and 20 tweets in RHSD. Mostly all of these tweets included one or more religious, national, or ideological affiliations terms that are highly associated with hate class (according to our generated lexicons). Examples: “*Hahaha... Fear Allah brother, first of all Khurasan scholars are not related to the Persians Magians! Second Arab was and still the head of knowledge*” and “*Hahaha... you blended with our Hadarms ladies so that’s why they consider you one of them; that’s the way our parents raised us even if you’re Saudi*”. These tweets may cause some confusion that leads the classifiers to classify them as hateful. We also found that some of the misclassified tweets are debatable and that it could be difficult in terms of deciding whether to consider them hate or non-hate, even by human annotators, such as the following tweet: “*Most of them moved from one radicalism to another; and what we got here in this world are people called the new Arab liberals and you can see what’s on their shoulders (different ideas that contradict the simplest principles of liberalism)*”.

All these factors led us to conclude that hate speech is still a difficult phenomenon. It is highly dependent on the context and, in fact, can be conveyed in a very polite and genuine form (without using any offensive words or being aggressive).

7. Conclusions and Future Direction

Hate speech in the Arabic Twittersphere has become a notable problem, resulting in a pressing need for effective automatic solution for hate speech detection. In this work, we constructed a public dataset of 9316 tweets labeled as hateful, abusive, and normal. We evaluated and compared four different models: CNN, GRU, CNN + GRU, and BERT. The obtained results from our experiments were promising, showing the effectiveness of the proposed models in the detection task. The results showed that CNN successfully outperformed other models, with an F1-score of 0.79 and AUROC of 0.89. Our results also showed that BERT failed to improve over the baselines and the other evaluated models. This could be attributed to the fact that BERT was trained on different dataset genre (Wikipedia).

We believe that there are several ways to extend and improve this study in the future. The dataset can be extended to capture more writing styles, patterns, and topics. The dataset can also be annotated with multi-labels to enhance the results of the detection task beyond the binary classification. For example, annotating the target of the hate speech and the aggressive level (weak/strong) could introduce a new direction for future works and applications.

We also think that the conducted experiments could be enhanced in different ways. First, we aim to extend our out-domain experiments and evaluate the proposed models on other abusive/offensive language datasets such as those found in [24,25]. This could help us to understand how to distinguish between hate speech and abusive/offensive language. Moreover, different embedding methods could be investigated besides Word2Vec, such as ELMo [47], BERT, FastText [48], and the Universal Sentence Encoder (USE) [49]. Other features can be also incorporated with word embeddings such as the user’s gender, age, and location. Moreover, to alleviate the lack of the context problem, we found that when we analyzed the errors made by our models we aimed to incorporate the tweets’ context (e.g., the original tweet of the replies, quoted tweets, text from the external links) into the feature space and investigate if this could boost the performance of the classifiers.

Author Contributions: Conceptualization, R.A. and H.A.-K.; methodology, R.A. and H.A.-K.; validation, R.A.; formal analysis, R.A. and H.A.-K.; investigation, R.A.; resources, R.A.; data curation, R.A.; writing—original draft preparation, R.A.; writing—review and editing, H.A.-K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Researchers Supporting Project number (RSP-2020/276), King Saud University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Al-Hassan, A.; Al-Dossari, H. Detection of Hate Speech in Social Networks: A Survey on Multilingual Corpus. In *Proceedings of the Computer Science & Information Technology (CS & IT), Dubai, UAE, 23–24 February 2019*; AIRCC Publishing Corporation: Tamil Nadu, India, 2019; pp. 83–100.
2. Schmidt, A.; Wiegand, M. A Survey on Hate Speech Detection using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, Valencia, Spain, 3 April 2017*; Association for Computational Linguistics: Valencia, Spain, 2017; pp. 1–10.
3. Waseem, Z.; Hovy, D. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop, San Diego, CA, USA, 12–17 June 2016*; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 88–93.
4. Davidson, T.; Warmusley, D.; Macy, M.; Weber, I. Automated Hate Speech Detection and the Problem of Offensive Language. *arXiv* **2017**, arXiv:1703.04009.
5. Jaki, S.; De Smedt, T. Right-wing German hate speech on Twitter: Analysis and automatic detection. *airXiv* **2018**, arXiv:1910.07518, Manuscript Submitted.
6. Fortuna, P.; Nunes, S. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.* **2018**, *51*, 1–30. [[CrossRef](#)]
7. Park, J.H.; Fung, P. One-step and Two-step Classification for Abusive Language Detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017*; Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 41–45.
8. Gambäck, B.; Sikdar, U.K. Using Convolutional Neural Networks to Classify Hate-Speech. In *Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017*; Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 85–90.
9. Badjatiya, P.; Gupta, S.; Gupta, M.; Varma, V. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, 3 April 2017*; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017; pp. 759–760.
10. Pitsilis, G.K.; Ramampiaro, H.; Langseth, H. Effective hate-speech detection in Twitter data using recurrent neural networks. *Appl. Intell.* **2018**, *48*, 4730–4742. [[CrossRef](#)]
11. Zhang, Z.; Robinson, D.; Tepper, J. Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network. In *Proceedings of the Semantic Web; Gangemi, A., Navigli, R., Vidal, M.-E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., Alam, M., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 745–760.*
12. Albadi, N.; Kurdi, M.; Mishra, S. Are they Our Brothers? Analysis and Detection of Religious Hate Speech in the Arabic Twittersphere. In *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, 28–31 August 2018*; IEEE: Barcelona, Spain, 2018; pp. 69–76.
13. Darwish, K.; Magdy, W.; Mourad, A. Language processing for Arabic microblog retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, 18–25 May 2012*; ACM: New York, NY, USA, 2012; pp. 2427–2430.
14. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2019**, arXiv:1810.04805.
15. Warner, W.; Hirschberg, J. Detecting Hate Speech on the World Wide Web. In *Proceedings of the Second Workshop on Language in Social Media, Montreal, QC, Canada, 7 June 2012*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2012; pp. 19–26.
16. Kwok, I.; Wang, Y. Locate the hate: Detecting tweets against blacks. In *Proceedings of the Twenty-seventh AAAI Conference on Artificial Intelligence, Washington, DC, USA, 14–18 July 2013*.
17. Djuric, N.; Zhou, J.; Morris, R.; Grbovic, M.; Radosavljevic, V.; Bhamidipati, N. Hate Speech Detection with Comment Embeddings. In *Proceedings of the 24th International Conference on World Wide Web—WWW '15 Companion, Florence Italy, 18–22 May 2015*; ACM Press: Florence, Italy, 2015; pp. 29–30.

18. Nobata, C.; Tetreault, J.; Thomas, A.; Mehdad, Y.; Chang, Y. Abusive Language Detection in Online User Content. In *Proceedings of the 25th International Conference on World Wide Web—WWW '16, Montreal, QC, Canada, 11–15 May 2016*; ACM Press: Montreal, QC, Canada, 2016; pp. 145–153.
19. Bohra, A.; Vijay, D.; Singh, V.; Akhtar, S.S.; Shrivastava, M. A Dataset of Hindi-English Code-Mixed Social Media Text for Hate Speech Detection. In *Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, New Orleans, LA, USA, 6 June 2018*; Association for Computational Linguistics: New Orleans, LA, USA, 2018; pp. 36–41.
20. De Smedt, T.; De Pauw, G.; Van Ostaeyen, P. Automatic detection of online jihadist hate speech. *arXiv* **2018**, arXiv:1803.04596. preprint.
21. Karan, M.; Šnajder, J. Cross-Domain Detection of Abusive Language Online. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Brussels, Belgium, 31 October–1 November 2018*; Association for Computational Linguistics: Brussels, Belgium, 2018; pp. 132–137.
22. Pamungkas, E.W.; Patti, V. Cross-domain and Cross-lingual Abusive Language Detection: A Hybrid Approach with Deep Learning and a Multilingual Lexicon. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, Florence, Italy, 29 July–1 August 2019*; Association for Computational Linguistics: Florence, Italy, 2019; pp. 363–370.
23. Magdy, W.; Darwish, K.; Weber, I. #FailedRevolutions: Using Twitter to Study the Antecedents of ISIS Support. *arXiv* **2015**, arXiv:1503.02401.
24. Mubarak, H.; Darwish, K.; Magdy, W. Abusive Language Detection on Arabic Social Media. In *Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017*; Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 52–56.
25. Alakrot, A.; Murray, L.; Nikolov, N.S. Towards Accurate Detection of Offensive Language in Online Communication in Arabic. *Procedia Comput. Sci.* **2018**, *142*, 315–320. [CrossRef]
26. Mulki, H.; Haddad, H.; Bechikh Ali, C.; Alshabani, H. L-HSAB: A Levantine Twitter Dataset for Hate Speech and Abusive Language. In *Proceedings of the Third Workshop on Abusive Language Online, Florence, Italy, 1 August 2019*; Association for Computational Linguistics: Florence, Italy, 2019; pp. 111–118.
27. Tweepy. Available online: <https://www.tweepy.org/> (accessed on 18 November 2020).
28. Confidence to Deploy AI with World-Class Training Data. Available online: <https://appen.com/> (accessed on 18 November 2020).
29. rshalan *raghadsh/Arabic-Hate-speech*. Available online: <https://github.com/raghadsh/Arabic-Hate-speech> (accessed on 23 November 2020).
30. Al-Humoud, S.; Al-Twairish, N.; Altuwaijri, M.; Almoammar, A. Arabic Spam Detection in Twitter. In *Proceedings of the 2nd Workshop on Arabic Corpora and Processing Tools 2016 Theme: Social Media, Portorož, Slovenia, 27 May 2016*.
31. Albadi, N. *nuhaalbadi/Arabic_hatespeech*. Available online: https://github.com/nuhaalbadi/Arabic_hatespeech (accessed on 23 November 2020).
32. Singh, A.; Blanco, E.; Jin, W. Incorporating Emoji Descriptions Improves Tweet Classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; Volume 1 (Long and Short Papers), pp. 2096–2101.
33. Demoji PyPI. Available online: <https://pypi.org/project/demoji/> (accessed on 18 November 2020).
34. Emoji Keyboard/Display Test Data for UTS #51. Available online: <http://unicode.org/Public/emoji/12.0/emoji-test.txt> (accessed on 18 November 2020).
35. Cloud Translation Documentation|Google Cloud. Available online: <https://cloud.google.com/translate/docs/> (accessed on 18 November 2020).
36. Abdelali, A.; Darwish, K.; Durrani, N.; Mubarak, H. Farasa: A Fast and Furious Segmenter for Arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, San Diego, CA, USA, 2–4 June 2016*; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 11–16.
37. El Mahdaouy, A.; El Alaoui, S.O.; Gaussier, E. Word-embedding-based pseudo-relevance feedback for Arabic information retrieval. *J. Inf. Sci.* **2018**, *45*, 429–442. [CrossRef]
38. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *arXiv* **2013**, arXiv:1301.3781.

39. Kim, Y. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014*; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1746–1751.
40. Keras: The Python Deep Learning API. Available online: <https://keras.io/> (accessed on 18 November 2020).
41. Huggingface/Transformers: Transformers: State-of-the-Art Natural Language Processing for Pytorch and TensorFlow 2.0. Available online: <https://github.com/huggingface/transformers> (accessed on 18 November 2020).
42. Google Colaboratory. Available online: <https://colab.research.google.com/notebooks/intro.ipynb#recent=true> (accessed on 18 November 2020).
43. Chen, H.; McKeever, S.; Delany, S.J. A Comparison of Classical Versus Deep Learning Techniques for Abusive Content Detection on Social Media Sites. In *Proceedings of the Social Informatics, St. Petersburg, Russia, 25–28 September 2018*; Staab, S., Koltsova, O., Ignatov, D.I., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 117–133.
44. Scikit-Learn: Machine Learning in Python—Scikit-Learn 0.23.2 Documentation. Available online: <https://scikit-learn.org/stable/> (accessed on 18 November 2020).
45. Al-Twairish, N.; Al-Negheimish, H. Surface and Deep Features Ensemble for Sentiment Analysis of Arabic Tweets. *IEEE Access* **2019**, *7*, 84122–84131. [[CrossRef](#)]
46. Kaji, N.; Kitsuregawa, M. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, 28–30 June 2007.
47. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365, preprint.
48. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of Tricks for Efficient Text Classification. *arXiv* **2016**, arXiv:1607.01759.
49. Cer, D.; Yang, Y.; Kong, S.; Hua, N.; Limtiaco, N.; John, R.S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; et al. Universal Sentence Encoder. *arXiv* **2018**, arXiv:1803.11175.

Publisher’s Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).