Detecting Hateful and Offensive Speech in Arabic Social Media Using Transfer Learning

Zakaria Boulouard 1, Mariya Ouaissa 2, Mariyam Ouaissa 2, Moez Krichen 3,4, Mutiq Almutiq 5,*, and Karim Gasmi 6

Abstract: The democratization of access to internet and social media has given an opportunity for every individual to openly express his or her ideas and feelings. Unfortunately, this has also created room for extremist, racist, misogynist, and offensive opinions expressed either as articles, posts, or comments. While controlling offensive speech in English-, Spanish-, and French-speaking social media communities and websites has reached a mature level, it is much less the case for their counterparts in Arabic-speaking countries. This paper presents a transfer learning solution to detect hateful and offensive speech on Arabic websites and social media platforms. This paper will compare the performance of different BERT-based models trained to classify comments as either abusive or neutral. The models were evaluated based on five metrics: Accuracy, Precision, Recall, F1-Score, and Confusion Matrix.

Keywords: deep learning; hate speech detection; natural language processing; social media analytics; text mining

1. Introduction

Access to the internet and social media has seen an important democratization over the past few years. According to Kemp [1], the total number of active internet users has reached 4.95 billion people worldwide as of January 2022. Among them, 4.6 billion users are active on social media (thus, a 10% increase when compared to 2021). In the Middle East and North Africa (MENA) regions, countries such as Egypt, Saudi Arabia, Algeria, and Morocco were ranked among the top 30 countries with the largest numbers of Facebook users in 2022.

Social media platforms nowadays represent the essence of free speech since they offer their users a space where they can express their opinions with no control or censorship. This is, usually, a healthy situation, but not when some individuals tend to use expressions, statements, and allegations that may offend other counterparts with different beliefs, backgrounds, genders, or even races.

Even though they are protected by the CDA 230 in its latest version [2], a great number of social media platforms are putting the effort to provide the best protection for their users against hateful and offensive content. However, the main problem of the approaches they offer is that they rely heavily on human monitoring and user reports.
The last couple of years have seen a serious increase in hate speech on social media platforms worldwide, especially during the COVID-19 lockdown. For instance, the COVID-19 lockdown was behind a worrying 20% rise in online hate speech in the United Kingdom [3].

This problem has caught the attention of several researchers worldwide and has urged them to seek solutions in order to detect and prohibit such behavior. The recent literature provides different strategies for automatic hate speech detection. However, most of these works were dedicated to the most spoken languages worldwide, such as English, Spanish, and other languages.

Surprisingly, despite being among these languages, works related to Arabic, as will be covered in Section 2, have been rare, and mostly dialect-focused. This comes from the fact that social media users in Arabic speaking countries tend to use their own dialects instead of the standard Arabic. Another reason is the choice of letters used when writing. In Middle Eastern countries, social media users prefer Arabic letters, while in the North African countries, the Romanized Arabic, also known as “Arabizi”, is preferred.

In this work, as will be described in Sections 3 and 4, we focus on the Middle Eastern region as it has a wider variety of dialects, and also because of their use of the Arabic letters on social media platforms. We propose a selection of transfer learning models based on BERT [4] for hateful and offensive speech detection that can cover statements (social media comments) written in the standard Arabic, as well as three of most spoken Arabic dialects in the region (Egyptian, Iraqi, and Gulf). Section 5 will conclude this paper and suggest possible future directions.

2. Related Works

Having a proper definition of hateful and offensive speech has always caused controversy since these definitions tend to have religious, cultural, or ethnical backgrounds. The United Nations [5] defined hate speech as any expression that attacks a person on a group based on any of their identity factors, would it be ethnic, religious, racial, etc. As stated earlier, hateful and offensive speech has grown significantly on social media in the last couple of years, especially in the aftermath of the COVID-19 lockdown. With the rise of Artificial Intelligence (AI) methods and their great performance in many fields [6–10], several researchers from around the globe have also presented several AI-based solutions to automate the detection of hateful and offensive speech in their online communities.

Whereas English is backed by most of the mature solutions, other top languages such as French and Spanish have also presented interesting approaches. Vanetik and Mimoun [11] have built and annotated a dataset of 2856 French tweets where 927 were labeled as “racist” whereas the rest was “not racist”. They applied TF-IDF, N-gram, and BERT sentence embedding for text representation, then went on to compare different binary classification models such as “Random Forest” (RF), “Logistic Regression” (LR), and “Extreme Gradient Boosting” (XGBoost). The best performance for that dataset was provided by LR when it was backed by BERT word embedding, with 79% accuracy. Following similar steps, Arcila-Calderón et al. [12] have built a dataset of 10855 Spanish tweets where 2773 were labeled as “hateful” and the rest as “non-hateful”. They used this dataset to train machine learning and deep learning models able to detect gender and sex orientation based hateful online statements. They used “Bag of Words” (BoW) as a text representation for the machine learning algorithms, and word embedding for the RNN deep learning model. The latter combination provided the best performance with 84% accuracy, exceeding by far the second best, which was BoW with LR (76%). Another attempt to detect hate speech in Spanish social media was presented by Plaza-Del-Arco et al. [13] who selected tweets from two different datasets. From the first dataset, they selected 6000 tweets and labeled 1567 of them as “hateful”, and the remaining 4433 as “not hateful”. From the second dataset, they collected 6600 tweets and labeled 2739 of them as “hateful”, and the remaining 3861 as “not hateful”. When comparing the performances of different pre-trained models, BETO, a
monolingual transformer model based focused on the Spanish language, outperformed BERT and XLM.

The literature also presents tentative hate speech detection models in other languages such as Urdu [14], Turkish [15], or Chinese [16].

Another interesting approach that several researchers suggested is to propose multilingual hate speech detection models. Chiril et al. [17] combined two datasets. The first one, in English, contains tweets with hateful content against women and immigrants, whereas the second one, in both English and French, contains sexist tweets. The combined dataset contains a total of 16,156 tweets, of which, 6171 were labeled as “hate”, and the remaining 9985 were labeled as “non-hate”. They used FastText [18] and GloVe [19] for multilingual embedding as backing for the Bidirectional Long-Short Term Memory (BiLSTM) algorithm. The best performance was provided by the combination FastText-BiLSTM with a 79% accuracy. Corazza et al. [20] combined three datasets of tweets. The first one, in English, contains 16,000 tweets, from which, 5006 were labeled as “positive” for racism and sexism, whereas the remaining 10,884 tweets were labeled as “negative”. The second dataset, in Italian, contains 4000 tweets, from which, 1296 were labeled as “hateful”, whereas the remaining 2704 were labeled as “not hateful”. The third dataset, in German, contains 5009 tweets, of which, 1688 were labeled as “offensive”, whereas the remaining 3321 were labeled as “other”. Using these numbers, the authors made sure to keep a 1 to 3 ratio between hateful and non-hateful tweets in their datasets. The best performances were provided by the FastText backed Long Short Term Memory (LSTM) algorithm with an F1-Score of 82% for English, word embedding of Italian tweets along with LSTM for the Italian language with an 80% F1-Score, and FastText embedding with Gated Recurrent Unit (GRU) for German with a 75.8% F1-Score. Ranasinghe and Zampieri [21] have trained and compared the performance of mBERT [4] and XLM-R [22] models on a multilingual dataset containing tweets and Facebook comments in Bengali (4000 FB comments), Hindi (8000 tweets), and Spanish (6600 tweets), which were labeled as “hateful” or “non-hateful”. The overall best performance for this dataset was provided by XLM-R with F1-scores of 84% for Bengali, 85% for Hindi, and 75% for Spanish.

When compared to other languages, works dedicated to hateful and offensive language detection in Arabic are scarcer. To the best of our knowledge, most of the earliest works started in 2017 when Abozinadah and Jones [23] suggested a statistical learning approach to back the Support Vector Machines (SVM) algorithm to classify tweets in Arabic as either “abusive” or not. Mubarak et al. [24] are also considered pioneers of Arabic obscene language detection. They set up a list of common obscene words in Arabic social media, and based on it, they extracted and annotated 32000 tweets that contain these words. Among these tweets, 79% were considered “offensive”, 2% were considered “obscene”, and the remaining 19% were considered “normal”.

Different approaches have been proposed since then. Albadi et al. [25] presented an approach dedicated to religious hate speech detection. They constructed a dataset of 6000 tweets, where each 1000 is specific to one of the six most common religions or sects in the Middle East. They extracted the features from the tweets using AraVec [26] and then, proceeded to a binary classification using LR, SVM, and GRU algorithms. The latter provided the best performance with a 79% accuracy. Anezi [27] collected a dataset of 4203 comments from different social media platforms. Each of these comments was manually annotated by a group of Arab native speaker to one of seven classes (“Against Religion”, “Racist”, “Against Gender Equality”, “Insulting or bullying”, “Violent or offensive”, “Normal Positive”, and “Normal Negative”). For the classification, they opted for Recurrent Neural Networks (RNN) for an accuracy rate of 84.14%. Shannaq et al. [28] built a dataset of 4505 tweets from four different domains where there is a high possibility of finding offensive speech (“Celebrities”, “Gaming”, “News”, and “Sports”). They fine-tuned AraVec and GloVe models and fed the word embeddings to two classifiers (SVM and XGBoost) that had their hyperparameters optimized by the Genetic Algorithm (GA). The best performance was provided by the AraVec-backed GA-SVM combination with an
88.2% accuracy. Alsafari et al. [29] built a dataset of 5631 tweets which were annotated by two female and one male Gulf native speakers. The dataset had six labels ("Clean", "Offensive", "Religious Hate", "Gender Hate", "Nationality Hate", and "Ethnicity Hate"). They used different word embedding techniques such as AraVec, FastText, and mBERT, and then, fed the data to three deep learning algorithms: LSTM, GRU, and Convolutional Neural Networks (CNN). The mBERT-CNN combination provided the best results with an F-Macro of 75.51%, which they considered encouraging given the training limitations.

3. Materials and Methods

Most of the abovementioned methods were based either on standard Machine Learning/Deep Learning approaches, or on Word2Vec/AraVec approaches. This statement was also confirmed by Anezi [27] in his literature review. The main issue with these methods is that, when they try to represent a word, they tend to miss on its context. For example, a word like "المغرب" will be represented in the same way every time regardless of its context. The thing here is that the same word, depending on context, can either mean "Morocco", "west", or "sunset".

As it will be further developed in this section, we tried to build our hate speech detection model while taking context into consideration. This prompted us to explore the possibilities offered by BERT [4]-based models. These models can provide more insight on context by scouting a particular word’s successors and predecessors, rather than relying just on its successors, as it was the case with the previous methods.

Based on the usual classification performance indicators, i.e., Accuracy, Precision, Recall, F1-Score, and Confusion Matrix, we will compare different BERT-based approaches and see which can provide the best results. The approaches we will compare are the following:

- Translating the comments into English and using the classical BERT [4];
- Using the Multilingual BERT (mBERT) [4] on the translated comments;
- Using mBERT on the original comments in Arabic;
- Using AraBERT [30], a BERT-based model trained on Arabic text, on the original comments in Arabic.

3.1. BERT Based Models

3.1.1. BERT

BERT, or Bidirectional Encoder Representations from Transformers, is pre-trained to provide word representations based on context learnt from both sides. It is based on the Transformer architecture [31], from which, it only takes the encoder side.

According to Alammar’s description [32], BERT is presented in two versions:

- **BERT\_BASE**: with 12 encoder layers, 768 hidden unit feedforward layers, and 12 attention heads;
- **BERT\_LARGE**: with 24 encoder layers, 1024 hidden unit feedforward layers, and 16 attention heads.

Both of these versions are larger than the default version suggested in the original paper [31], which consists of 6 encoder layers, 512 hidden unit feedforward layers, and 8 attention heads.

Figure 1 displays the encoder stacks in both BERT\_BASE and BERT\_LARGE. Each encoder can be broken down into two sublayers.

The first layer is the self-attention layer whereas the second layer is the feed forward neural network layer. The self-attention layer allows the encoder to investigate other words from a sentence while encoding a specific word. This can be done in different steps.

The first step is to multiply the embedding of the word $x_i$ by three weight matrices $W^Q, W^K,$ and $W^V$, thus extracting three vectors: the query vector $q_i$, the key vector $k_i$, and the value vector $v_i$, respectively.

These vectors will be used, in the second step, to calculate the attention score of the processed word. This score evaluates the focus that needs to be placed on the other
words of the sentence while processing the current word. This score is the dot product of
the processed word’s query vector $q_i$ and the key vector $k_j$ of each of the other words
of the sentence.

![Figure 1. Encoder stacks in both BERTBASE and BERTLARGE [32].](image)

The third step is the division by the square root of the key vector’s dimension $\sqrt{d_k}$ to
keep the gradients stable.

The scores will be normalized in the fourth step using a softmax function.

The fifth step is to multiply the softmax score by the value vector $v_i$ so we can keep
the needed words while discarding the irrelevant ones.

These weighted value scores will be summarized in the sixth step to produce the
output of the self-attention layer for the processed word.

For better performance, the abovementioned steps are applied on matrices instead of
individual vectors. For instance, the embedded sentences will be collected in a matrix
that will be multiplied by the weight matrices $W^Q$, $W^K$, and $W^V$ to produce the query $Q$,
key $K$, and value $V$ matrices.

The output of the self-attention layer $Z$ will be calculated according to the
following equation:

$$ Z = \text{SOFTMAX}(\frac{QK^T}{\sqrt{d_k}})V $$

Having multiple attention heads (12 for BERT\textsubscript{BASE}, and 24 for BERT\textsubscript{LARGE}) allows for
the model to focus on different positions and have multiple representation subspaces with
different $Z$ matrices. As the feedforward layer expects only one $Z$ matrix, the different
matrices from the attention heads are concatenated into one, which will be multiplied by
another weight matrix $W^O$.

Each encoder sublayer, as described in Figure 2, applies a Layer Norm operation [33] on
its output before feeding it to the next sublayer (or layer). This output can be expressed as:

$$ Z = \text{LayerNorm}(X + Z) $$

BERT’s pre-training was based on two tasks going together. In the first task, “Next
Sentence Prediction” (NSP), two sentences are fed to the model with an embedding “A” for
the first sentence and “B” for the second sentence. The sentence “B” is provided as the real
next sentence in half of the cases, whereas a random sentence is provided in the other half
of the cases. In the second task, “Masked Language Model” (MLM), 15% of the tokens are
replaced with a [MASK] token in 80% of the cases, and a random token for 10% of the cases,
while they remain as they were for the last 10% of the cases. This task is applied right after
the tokenization process.
Figure 2. Detailed architecture of the first encoder in a BERT stack [34].

For BERT’s pre-training, the authors used the configuration described in Table 1. The datasets used in the pre-training process are the “BooksCorpus” by Zhu et al. [35] with 800 million words, and the English Wikipedia with 2.5 billion words.

Table 1. BERT pre-training configuration.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>0.9</td>
</tr>
<tr>
<td>$\beta_2$</td>
<td>0.999</td>
</tr>
<tr>
<td>L2 Weight Decay</td>
<td>0.01</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.1</td>
</tr>
<tr>
<td>Activation</td>
<td>Gelu [36]</td>
</tr>
</tbody>
</table>

3.1.2. Multilingual BERT (mBERT)

Multilingual BERT (mBERT) is a version of BERT created by Devlin et al. [4] to narrow the language gap by training it on corpora of multiple languages. Latin-based languages which have comparable structure and vocabulary can benefit from common representations [22]. Other languages, such as Arabic, vary in syntactic and morphological structure and share very little with the multitude of Latin-based languages. Hence, mBERT still lags behind single-language based models due to a lack of data representation and a restricted language-specific vocabulary.

3.1.3. AraBERT

Developed by Antoun et al. [30], AraBERT is a transfer learning model that was pre-trained for three specific tasks:

- Sentiment Analysis (SA);
- Named Entity Recognition (NER);
- Question Answering (QA).
It is based on BERT [4] and is designed to tackle the issue of non-contextual word representations in the Arabic language that Word2Vec-based models such as AraVec employ [26].

The developers of AraBERT [30] followed the same NSP-MLM pre-training based procedure as the one adopted by the original BERT team [4]. However, due to the small size of the Arabic Wikipedia dumps, when compared to their English counterparts, Antoun et al. [30] needed to scrap articles from different news websites and add the data they collected to two publicly available datasets: the “Open Source International Arabic News Corpus” by Zeroual et al. [37], and the “1.5 billion words Arabic corpus” made available by El-Khair [38], Thus making AraBERT’s pre-training dataset reach an approximate total size of 24GB of text containing around 70 million non-duplicate sentences. In their later version (AraBERT v2), which we will use in our hate speech detection and classification, the size of the pre-training dataset had risen to 77GB.

### 3.2. Fine-Tuning BERT-Based Models for Text Classification

Chi Sun et al. [39] investigated various BERT finetuning approaches for text classification and maintained that:

1. BERT’s top layer is helpful for text classification;
2. BERT can solve the catastrophic forgetting problem with an adequate layer-wise decreasing learning rate;
3. Further within-task and within-domain pre-training can considerably improve its performance;
4. A prior multi-task fine-tuning is similarly beneficial to single-task fine-tuning, although its advantage is less than that of additional pre-training;
5. BERT can provide good performance with small-sized datasets.

For fine-tuning BERT and mBERT, Devlin et al. [4] found that optimal hyperparameters are mostly task-specific, but the range of possible values described in Table 2 usually provides satisfactory results whatever the task is. The same setup was also confirmed by Antoun et al. [30] for fine-tuning AraBERT.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Range of Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch Size</td>
<td>16, 32</td>
</tr>
<tr>
<td>Learning Rate (Adam)</td>
<td>$5 \times 10^{-5}, 3 \times 10^{-5}, 2 \times 10^{-5}$</td>
</tr>
<tr>
<td>Number of epochs</td>
<td>2, 3, 4</td>
</tr>
</tbody>
</table>

### 3.3. Dataset

Unlike other languages, Arabic content on social media (tweets, posts, comments, etc.) are mostly written in local dialects instead the standard Arabic, considered to be the official language in all Arab countries. This comes from the fact that standard Arabic is used for formal, administrative, educational, and religious purposes. Arabs tend to use their local dialects as a means of communication when attending their usual day-to-day tasks, and thus, in social media. For this reason, any efficient attempt to detect offensive and hateful speech in Arabic social media needs to be able to work with Arabic dialects. This urged many researchers, as we have seen in the previous section, to manually collect and annotate their own datasets, serving specific needs.

However, there are some publicly available datasets in the literature, though rare, that can provide good performance with proper training. Mulki et al. [40] created a dataset with 5846 tweets in Levantine dialects that were annotated to either (“Hate”, “Abusive”, or “Normal”). Another interesting dataset was presented by Alakrot et al. [41] who collected 15050 YouTube comments in different dialects, namely Gulf, Egyptian, and Iraqi. These comments were labeled as either “offensive” or “inoffensive”. To the best of our knowledge,
this dataset may be among the largest and most diverse one available so far in the literature, and as such, we will use it for our approach.

In their annotation process, Alakrot et al. used the help of three annotators, two of them were from Iraq and Egypt, two countries that were highly represented in the dataset, whereas the third was from Libya, a country that was almost absent from the dataset. For their final annotation, Alakrot et al. provided two datasets with two different scenarios based on which they would consider if a comment was considered hateful or not. In the first scenario, the annotation is based on a unanimous vote, whereas in the second scenario, it is based on a majority vote.

For training our BERT-based models, we chose the dataset of the second scenario, which has a better proportion of the two classes. Since BERT is trained in English, and the comments our dataset are in Arabic, we used Google API to translate them into English.

3.4. Preprocessing

The first step of preprocessing the dataset is to remove the missing values as they represent a small minority when compared to the useful data. This will leave a total of 11,268 YouTube comments from which 4748 comments were considered as hateful and labeled as “1”, against 6520 comments that were considered as non-hateful and labeled as “0”. In terms of percentage, it would show approximately 42% of the total as hateful comments against 58% non-hateful.

After a quick look at the data, as displayed in Figure 3, you realize that it still needs to be further processed as to remove the emojis, punctuation, stop words, as well as the extra letters used for emphasis. Some words also need to be stemmed and lemmatized to keep their root.

![Figure 3. Examples of comments from the dataset.](image-url)

Preprocessing the Arabic comments was done using the Farasa library by Abdelali et al. [42], which covers most of the abovementioned issues. This can give us the possibility to go from words such as “العاهرات” to segmented words such as “العاهرات”, and thus, keeping the root “عاهر”, which will be used in the further steps.

After the segmentation, comes the step of tokenization. In this step, every extracted word will be converted into a token and provided a token ID, which will be used in the training process. The tokens from each comment will be stored in a set of a maximum length of 512. However, and for further optimization, we have seen that most of the comments contain less than 150 tokens, as displayed in Figure 4. For this reason, we will adopt a maximum length of 150.

The next step is to shuffle the dataset before splitting it into a training and testing subsets. The hyperparameters we used for fine-tuning AraBERT to the dataset are the ones presented in Table 2.
Figure 4. Number of tokens in each comment from the dataset.

3.5. Candidate Algorithms

As explained in the beginning of this section, in this study, we will explore the potential of the four following BERT-based approaches:

- Translating the comments into English and using the classical BERT [4]. In the following, the approach will be referred to as “BERT\textsubscript{EN}”;
- Using the Multilingual BERT (mBERT) [4] on the translated comments. In the following, the approach will be referred to as “mBERT\textsubscript{EN}”;
- Using mBERT on the original comments in Arabic. In the following, the approach will be referred to as “mBERT\textsubscript{AR}”;
- Using AraBERT [30], a BERT-based model trained on Arabic text, on the original comments in Arabic. In the following, the approach will be referred to as “AraBERT”.

In a previous study [43], we compared the performance of the following Shallow and Deep Learning algorithms:

- Logistic Regression (LR);
- Naïve Bayes (NB);
- Random Forests (RF);
- Support Vector Machines (SVM);
- Long Short-Term Memory (LSTM).

In that study, we found that LSTM provided the best performance, so we will keep it as a candidate in this current study.

We will also take the Linear Support Vector Classification (LinearSVC) based on the approach presented by Alakrot et al. [44], in which they demonstrated that LinearSVC can reach a 90% accuracy, which is higher than our LSTM-based approach, reaching an accuracy of 82%.

To measure the performance of the abovementioned approaches, we will rely on the usual classification metrics, i.e., Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix.

3.6. Development Environment

The environment in which this study was conducted is described as follows:

- Programming Language: Python 3;
- IDE: Google Colab Pro;
- RAM: 32GB;
- GPU: Nvidia Tesla P100;
- CPU: Intel(R) Xeon(R) CPU @ 2.30GHz.
4. Results and Discussion

Based on the fine-tuning recommendations described in Table 2, we proceeded to train our candidate models on our dataset. Table 3 provides an overview on the performance of these models.

Table 3. Classification Performance Scores.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT\textsubscript{EN}</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>AraBERT</td>
<td>0.96</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
</tr>
<tr>
<td>LinearSVC [44]</td>
<td>0.9</td>
<td>0.89</td>
<td>0.76</td>
<td>0.81</td>
</tr>
<tr>
<td>mBERT\textsubscript{AR}</td>
<td>0.83</td>
<td>0.84</td>
<td>0.82</td>
<td>0.83</td>
</tr>
<tr>
<td>LSTM [43]</td>
<td>0.82</td>
<td>0.92</td>
<td>0.74</td>
<td>0.82</td>
</tr>
<tr>
<td>mBERT\textsubscript{EN}</td>
<td>0.81</td>
<td>0.82</td>
<td>0.80</td>
<td>0.81</td>
</tr>
</tbody>
</table>

BERT\textsubscript{EN} outperformed all the other candidate models with an Accuracy of 98%, a Precision of 98%, a Recall of 98%, and an F1-Score of 98%. AraBERT was a close runner-up with an Accuracy of 96%, a Precision of 95%, a Recall of 96%, and an F1-Score of 95%. mBERT\textsubscript{AR} achieved an Accuracy of 83%, a Precision of 84%, a Recall, of 82%, and an F1-Score of 83%. mBERT\textsubscript{EN} achieved an Accuracy of 81%, a Precision of 82%, a Recall of 80% and an F1-Score of 81%.

The results proved that the Precision and the Recall of the BERT-based models are almost alike. This means that these models are not as biased as the baseline models, such as LSTM [43] and LinearSVC [44], and perform equally well for both the positive and the negative comments.

mBERT is versatile and generic at the cost of performance. Moreover, mBERT\textsubscript{AR} performs better than the mBERT\textsubscript{EN}.

AraBERT performs well in this case as the source data is quite similar to the target data. As BERT was trained on a relatively larger language corpus when compared to that of AraBERT, the dominance of BERT\textsubscript{EN} validates that the volume of the language corpora for pretraining plays a vital role in a model’s performance.

As the BERT\textsubscript{EN} and AraBERT were clear winners, with very close values in their confusion matrices as depicted in Figure 5, we conducted a more granular analysis on these two models to understand what really made the difference between these two models.

![Figure 5](image-url) 
(a) (b)

Figure 5. The confusion matrices of BERT\textsubscript{EN} (a) and AraBERT (b).

A closer look at some of the comments that were wrongly predicted by AraBERT and correctly by BERT\textsubscript{EN} made us realize that these comments had sarcasm in them. This assumption was verified after annotating all the comments of the dataset as to having sarcasm or not.
Again, the difference in the size of the pre-training datasets of BERT and AraBERT has demonstrated their important roles in performance, especially in detecting the eventual presence of sarcasm.

Taking a deeper look at why BERT-based models outperformed other language models such as LSTM [43] and LinearSVC [44], in the past, conventional language models could only interpret text input sequentially — either from right to left or from left to right — but not simultaneously. BERT is unique since it can simultaneously read in both directions. Bidirectionality is the name for this capacity, which the invention of Transformers [31] made possible. BERT is pre-trained on two distinct but related NLP tasks—Masked Language Modeling and Next Sentence Prediction—using this bidirectional capacity. The Masked Language Model (MLM) training’s goal is to conceal a word in a phrase and then have the computer anticipate the hidden word based on the context of the concealed word. The goal of Next Sentence Prediction training is to make the algorithm determine if two provided phrases relate logically and sequentially or whether their relationship is just random [4]. This is the core reason why BERT-based models are very powerful when compared to other conventional unidirectional language models.

Another reason behind the power of BERT based models when compared to other conventional models, is again, their huge pre-training corpus. This makes these models turnkey and ready to be used in every use case scenario (provided a prior fine-tuning), whereas in the latter case, conventional models need to be trained from scratch.

5. Conclusions

In this paper, we explored the possibilities offered by BERT-based models to detect hateful and offensive speech in social media comments that are written as either in standard Arabic, or in any of the top three most spoken dialects in the Middle East (Gulf, Egyptian, and Iraqi). We trained different BERT-based models such as multilingual BERT (mBERT AR) and AraBERT on a dataset made available online by Alakrot et al. [41]. This dataset contains hateful YouTube comments written in the aforementioned dialects. We also trained BERT in its base (BERT EN) and multilingual (mBERT EN) forms on the English translations of the comments in Alakrot et al.’s [41] dataset.

BERT EN provided the best results with 98% accuracy, closely followed by AraBERT with 96% accuracy. BERT EN had the advantage of having larger training corpora than AraBERT, allowing the former to better detect subtleties such as sarcasm. A further optimization of AraBERT in this direction would be necessary. The multilingual BERT performed poorly in both the Arabic and English datasets because of its built-in versatility.

In further optimization of the performance of BERT-based models in detecting hateful and offensive speech, and for them to provide efficient results in the entirety of Arabic social media, we plan on training them on more datasets in Levantine and North-African dialects. The latter will be the main issue to tackle since in North Africa, social media users tend to use both Arabic letters and “Arabizi” (Arabic with Romanized letters) in their comments.

Author Contributions: Methodology, Z.B. and M.O. (Mariyam Ouaissa); Software, Z.B. and M.O. (Mariyam Ouaissa); Formal analysis, Z.B., M.O. (Mariyam Ouaissa) and K.G.; Data curation, M.O. (Mariyam Ouaissa), M.K. and M.A.; Writing—original draft, Z.B., M.O. (Mariyam Ouaissa), M.O. (Mariyam Ouaissa), M.K., M.A. and K.G.; Project administration, Z.B., M.O. (Mariyam Ouaissa), M.O. (Mariyam Ouaissa), M.K. and M.A.; Funding acquisition, M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no additional funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The researchers would like to thank the Deanship of Scientific Research, Qassim University, for funding the publication of this project.
Conflicts of Interest: The authors declare no conflict of interest.

References

11. Vanetik, N.; Mikmoun, E. Detection of Racist Language in French Tweets. Information 2022, 13, 318. [CrossRef]
43. Boulouard, Z.; Ouainsi, M.; Ouaiissa, M. Machine Learning for Hate Speech Detection in Arabic Social Media. In Computational Intelligence in Recent Communication Networks; Springer: Berlin/Heidelberg, Germany, 2022; pp. 147–162. [CrossRef]