IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 35, No. 6, December, 2021, pp. 457-465
Journal homepage: http://iieta.org/journals/ria

Hate Speech in the Arab Electronic Press and Social Networks

Widad Awane*, El Habib Ben Lahmar, Ayoub El Falaki

Information Technology and Modeling Faculty of Sciences Ben M'sik Hassan II University, B.P 7955, Sidi Othmane, Casablanca 20023, Morocco

Corresponding Author Email: Awanewidad93@gmail.com

https://doi.org/10.18280/ria.350603

Received: 8 October 2021
Accepted: 25 November 2021

Keywords:
ArabicBERT, hate speech, ML, NLP, text classification

ABSTRACT

Nowadays we are witnessing an open world, characterized by globalization which is accompanied by a technology through which information circulates without borders, especially with the widespread use of social networking sites being the most common communication tool, that gives access through various applications to a large space for the presentation of multiple ideas, including extremist ideas, and the spread of hate speech. This paper introduces a system of detection of hate speech in the texts of Arabic read media and social media, which is based on a combined use of NLP, and machine learning methods. The training of the detection model is done on a large Dataset of articles, tweets and comments, collected, balanced and tokenized afterwards using BERT in Arabic. The trained model detects hate speech in Arabic and various Arabic based dialects, by classifying the texts into two classes: Neutral and Abusive. The above-mentioned model is evaluated using precision metrics, recall and f1 score, it has reached an accuracy of 83%.

1. INTRODUCTION

1.1 Generalities

It is clear that the widespread dissemination of media and communication as an open space has become a haven for practicing various forms of discrimination and producing hate speech mainly invested by the political actor, and this is what we also notice through reading Arabic content on the Internet, whether linked to online journalism, citizen journalism or micro blogging on social networks, networks play an essential role in the dissemination of hate speech in different types of violence, physical, symbolic or verbal speech.

Knowing that the usage of social media applications in the Arab world has exceeded 90% of the population in some countries [1], the regulation of the content of electronic journalism in Arabic is an essential step to reduce the spread of hate speech and stop various forms of cyber violence.

The use of artificial intelligence and machine learning techniques remain the only means capable of performing such a complex task.

Generating an automatic hate speech detection system will allow news agencies and regulatory bodies to specifically monitor not only the content of print newspapers and magazines, but also online journalism.

Creating such a system is a considerable challenge for two reasons, in one hand, the scarcity of text categorization databases for the detection of hate speech in Arabic, and the existence of several dialects based on the Arabic language in another hand, which makes automatic detection even more difficult.

To deal with these problems we created a large database through the combination of several other databases of different dialects that uses the Arabic alphabet, and we called on a BERT library in Arabic, which was pre-trained on various dialects, and we used it as a basic model to train and fine-tune a new BERT model to detect hate speech in Arabic texts.

1.2 Definition of hate speech

Even if there is no precise legal definition of "hate speech", it is generally referring to different types of public expression that propagate, incite, promote or justify hatred, discrimination or hostility against one person or a group of people, based on their person, in other words, it is based on religion, ethnicity, nationality, color, ancestry, sex or any other identity factor.

It is a specific type of cyber bullying that not only targets individuals but also affects their assets.

Hate speech can be classified into the following categories [1]:

(1) Gender-based hate speech: This category includes any form of hostility towards a specific sex or the devaluation of a person or group based on their gender.

(2) Religious hate speech: It includes any type of religious discrimination, such as: Islamic sects, anti-Christianity or any type of religious discrimination.

(3) Racial hate speech: this category includes any type of ethnic or tribal, regional crimes, (hostility against immigrants and refugees), any prejudice against a particular tribe or region, and offensive to the appearance and color of the individual.

1.3 Related work

Several studies and research have been carried out in the field of classification of Arabic texts, and more specifically in the automatic detection of hate speech in classical Arabic; Many of these studies have applied classic machine learning algorithms for tackling the task of classification including Support Vector Machine technique, and the Naïve Bayes classifier (NB) [2], Decision Trees, K-Nearest Neighbor and

other types of classifiers [3-17].

In their work, Altowayan and Tao [17], proposed a word embeddings model for the classification of sentiments for the Arabic language. Their work generated a language model that surpassed by little performance, other sentiment analysis models.

El-Halees [9] presented a three-step algorithm for classifying Arabic documents through the use of Markov networks and clustering [18]. This technique outperforms the algorithms of classical methods, in trials carried out on two different data sets. In the works [5], we see an introduction of a method to classify Arabic texts by text integration using the doc2vec model, which generates better performance compared to classical algorithms.

Regarding the automatic detection of hate speech, studies are still very few in Arabic compared to English. Abozinadah et al. [19-21] can be considered as Pioneers in researching offensive speech on Arab social networks. They use the NB Ranker on Twitter accounts.

The same authors developed this early work using the Support Vector Machine with a different technique of normalizing Arabic text that is directed to the problem of misspelled words [19]. In other studies, Abozinadah et al. [21] tackled the same issue by the application of a statistical learning approach to feature selection and a supporting vector machine classifier.

Other authors [22] take a different approach to find abusive speech. Their objective is to compile a long list of profane Arabic phrases or words, which can then be used to identify offensive language.

In other work, Mohaouchane et al. [23] measured the results of four different types of neural networks on the automatic detection of hate language. These models are CNN and LSTM bidirectional with or without attention, and CNN-LSTM. They discovered that the best retrieval results are achieved with the integrated CNN-LSTM while the best accuracy is obtained by CNN. While CNN alone is able to learn properties from the word n-grams, the CNN-LSTM model is also able to learn long-term dependencies thanks to the LSTM layer.

None of these studies looked at the different Arabic dialects, nor did they implement the Transformers method, even if it proved its efficiency and excellence in the different tasks of NLP.

The Table 1 summarizes the work carried out in the field of abusive language and hate speech detection since 2015 to date.

Table 1. State of the art: Automatic hate speech detection in Arabic, methods and results

Author	Year Platform	Classes	ML Approach	Features Representation	Algorithm	P	R	F
Abusive language								
Abozinadah and Jones [20]	2015-Twitter	Abuser, Normal	Supervised	Profile and tweet-based features, bag of words, N-gram, TF-IDF	Naïve Bayes	0.85	0.85	0.85
Abozinadah and Jones [19]	2016-Twitter	Abusive, Legitimate Accounts	Unsupervised	Lexicon, bag of words (BOW), N-gram	SVM	0.96	0.96	0.96
Abozinadah et al. [21]	2017-Twitter	NonAbusive, Abusive	Supervised	PageRank (PR) algorithm, Semantic Orientation (SO) algorithm, statistical	SVM	0.96	0.96	0.96
Mubarak et al. [24]	2017-Twitter, Arabic News Site	Obscene, Offensive and Clean	Unsupervised	unigram and bigram, Log Odds Ratio (LOR), Seed Words lists	None. Just performed extrinsic evaluation	0.98	0.45	0.60
Religious hate speech								
Albadi et al. [25]	2018-Twitter	Hate, Not hate	Supervised	Word embeddings (AraVec)	GRUbased RNN	0.76	0.78	0.77
Offensive Language								
Safaya et al. [26]	2020-youtube	Offensive, Not offensive	Supervised	Word embeddings (AraVec)	CNN	86.10	82.24	84.05
					Bi-LSTM	83.74	80.97	82.33
					Attention BiLSTM	82.07	81.51	81.70
					Combined CNN-LSTM	83.89	83.46	83.65

2. METHOD

2.1 The proposed approach

2.1.1 The BERT method

In our study, we use BERT, which stands for Bidirectional Encoder Representations from Transformers [27]. Compared to the word embedding model, BERT can also be described as a text representation method, which is a combination of various advanced deep learning algorithms, such as Long Short Term Memory, and also transformers.

Work on learning linguistic representations by pre-trained models on large unlabeled data sets of text documents began from word embeddings such as Word2Vec. This technique changed the way NLP tasks were performed. Thanks to the pre-trained models, we could obtain incorporations capable of creating contextual links between words.

The main limitation of these techniques was the use of superficial language models.

Thus, the new approach to solving NLP tasks has become a two-step process: The first performed by training a language model on a large body of text data set (unlabeled, and

unsupervised or semi-supervised), and in our case, we use ArabicBERT [26]. The second is to refine the large model obtained to a specific NLP task, so that we can use the important repository of knowledge acquired by the (supervised) model, for our task we refine ArabicBERT using a hate speech labeled dataset to create a text classification model.

The BERT architecture is based on Transformer. We currently have two variants of this model:

(1) BERT Base: composed of 12 layers (transformer blocks), 12 attention heads, and 110 million parameters.

(2) BERT Large (used in our study): 24 layers (transformer blocks), 16 attention heads and, 340 million parameters.

As state of the art of natural language processing, we can cite the OpenAI GPT model [28], and Embeddings from Language Models (ELMo) [29]. OpenAI GPT is a large-scale, language model that produces convenient paragraphs in a given language, it achieves peak performance on many language modeling benchmarks, and performs reading comprehension, machine translation, question-answering and summary, all without the need of a fine tuning training, ELMo has been proven to be an important model in the context of natural language processing, it can be trained on a massive datasets, and then we can use it as a component in other models that require language support.

This model got its understanding of language by training it in masked language modeling to predict the next sentence.

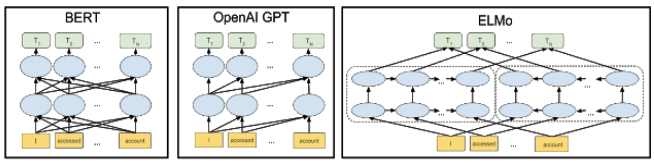

Figure 1. Architecture showing its bidirectional character compared to other language processing models

In Figure 1 we see a visualization of the neural network of BERT architecture in comparison with other state-of-the-art contextual linguistic pre-training techniques. The arrows show the information flow from a layer to another. The green squares indicate the end of contextualized representation of every input.

Usually we have modules which are trained to predict the next word in a sentence or trained in left to right context, which makes them prone to many errors due to loss of significant information etc. This is where BERT improves a lot. Compared to other NLP methods because it is bidirectional, it also combines two techniques, masked language model and next sentence prediction. BERT was introduced by Google researchers in the year 2018 and has proven its power in being at the cutting edge of technology for various natural language processing applications.

2.1.2 BERT in application

In our work we approach a natural language processing task which can be described as a text classification, in which we use the Bidirectional Encoder Representations from Transformers transfer learning technique, where we train a deep learning model on a large labeled dataset of different Arabic dialects, the trained model is able to capture the sequential information present in the text, and therefore is used to perform the classification task on different sources.

The BERT model we work on as a basis for our fine-tuned Arabic hate speech detection model is the arabic-bert-large model, that was pre-trained on ~8.2 billion words. For Masked Language Modeling and Next Sentence Prediction tasks, in our approach we train a new model on a labeled dataset, to perform the text classification task.

The architecture we are using is a Transformer model, as it goes beyond traditional RNNs in its ability to take an entire input sequence at a time, which is a big improvement that allows the model to be accelerated by GPUs.

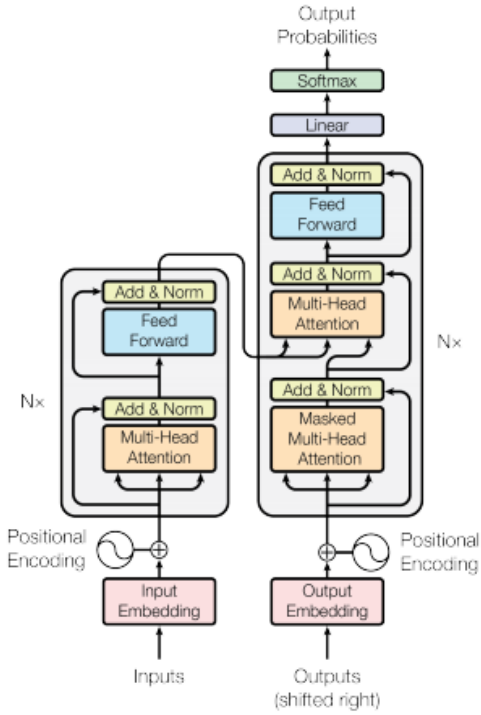

Figure 2. Transformer architecture [30]

As we can see in Figure 2, transformers are ENCODER-DECODER models with ATTENTION, there are a multitude of models based on transformers, the most popular are BERT and GPT, and previously we have explained the strengths of BERT on GPT regarding processing tasks natural language, which is why we use BERT for text classification.

The BERT encoder is made up of *N*=6 similar layers. Each one composed of two sublayers. The first is a multi-head self-attention network, and the second is a simple feed-forward mechanism, and a linking is used for each of the two Sub-layers, then normalization is applied. So, the output of every sublayer can be written as:

$$LayerNorm(x + Sublayer(x)) \quad (1)$$

where, *Sublayer(x)* the function is implemented by the sublayer itself.

The product of every sublayers in the model, and also of the embedding layers, is outputs of kind and measure:

$$d_{model} = 512 \quad (2)$$

The decoder is made of a stack of *N*=6 equal layers, the decoder inserts, after the two sub layers, of any encoder layer, a third sub-layer, which performs multi-head attention over the output of the encoder stack. The self-attention layer in the decoder is changed to stop current positions from affecting the next ones. This masking added to the fact that the embeddings in the output are different by one position, guaranties that the

predictions for a given position *i* depend only on the already known outputs at positions less than *i*.

Attention is a form of simple arithmetic, it is a function which takes *X* as input constituted by query and a set of key-value pairs, and gives back another *Y* identical sequence in length, composed of vectors of the same length as those by *X*.

The output is computed as a weighted sum of the values, where the weight assigned to each value is calculated by a query compatibility function with the corresponding key.

In Bert attention is called "Scaled Dot-Product Attention". Formally we have a query *Q*, a key *K* and a value *V* and compute the matrix of outputs as shown in (1) [30]:

$$Attention(Q,K,V) = softmax\left(\frac{QKT}{\sqrt{dk}}\right) \quad (3)$$

One of the strong characteristics of the bidirectional transformer is the multi-head attention mechanism; this unity works by the attention mechanism several times simultaneously. Then, the results of independent attention are serialized and linearly transformed into the expected dimension. Automatically, multiple attention heads allow you to attend parts of the sequence differently.

2.2 Materials

2.2.1 Programming language and platform

This article introduces a linguistic model for automatic detection of hate speech in Arabic, all the programming work, is done using the programming language Python, version 3.4.9, on a Windows 10 platform, note that the training of the model is performed using Google hardware acceleration, with the runtime type defined as Python 3, and the GPU as hardware accelerator with the following characteristics: NVIDIA-SMI 455.32.00, Driver Version: 418.67, CUDA Version: 10.1, the python versions operated on the cloud are: CPython 3.6.9 IPython 5.5.0.

2.2.2 Modules

Carrying out the different stages of training the model requires the use of various modules that are not included in the python library, the Table 2, summarizes the used modules, their versions, elements and parameters.

Table 2. Summary of modules and parameters

Module	Version	Element	Parameters
NumPy	1.18.5	numpy.random.RandomState	np.random.seed(42)
pandas	1.1.4	pd.value_counts, pd.info, pd.concat	Default
torch	1.7.0+cu101	nn, optim, torch.utils.data, Dataset, DataLoader, torch.nn.functional	Optimizer = AdamW(model.parameters(), lr=2e-5, correct_bias=False) total_steps = len(train_data_loader) * 10
transformers	3.5.1	BertModel, BertTokenizer, AdamW, get_linear_schedule_with_warmup, AutoTokenizer, AutoModel	tokenizer=AutoTokenizer.from_pretrained("asafaya/bert-large-arabic")
PyArabic	0.6.10	pyarabic.araby	Default

2.3 Process

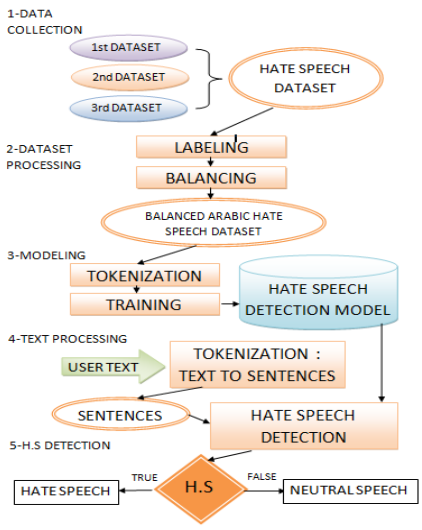

Figure 3. Process flowchart

The graph in Figure 3 is a description of the different stages of the work process, which can be broken down into two main parts: the first being the preparation of the refined detection model from the collected data, and the second being the use of the model in order to detect hate speech in user text.

In the first phase, we collect and process the data to form a database used for model training which is the last part of this step. The second phase begins with processing the user's text, followed by the application of the model from the first phase, for the detection of hate language.

2.4 Dataset

In this study we use as shown in Figure 4, a dataset that is a combination of three different hate speech datasets: The first is L-HSAB Dataset [30], which is the first Levantine Arabic hate speech and abusive language dataset constructed from Levantine tweets retrieved via the Twitter API (Tweepy). The second dataset is a Multi-Platform Arabic News Comment Dataset for Offensive Language Detection [22], built using comments from different platforms. The third one is a dataset of YouTube comments collected in July 2017 [31]. The combination resulted in a data set of 38,654 entries made up of texts in Classical Arabic, Levantine, and North African Dialect.

The partial choice of the data to be incorporated into our database is made manually, after careful verification, of the entries and their label, the choice relates to the elements which present the unanimity of the judges, and which do not show any character of confusion, however the check of the completeness of the dataset elements is carried out automatically, using the Pandas module.

After the verification, we carry out a count of the different classes of our dataset with the help of Pandas module, from which we use the Dataframe.info function to print a concise summary of the DataFrame, including the index dtype and

columns, non-null values and memory usage. The function shows the following results: 28898 entry of the "abusive" class and 9756 entry of the "neutral" class, to manage the problem of unbalanced data, we carry out an oversampling by replacement using the SMOTE module.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38654 entries, 0 to 38653
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   content  38654 non-null  object
 1   score    38654 non-null  int64
dtypes: int64(1), object(1)
memory usage: 604.1+ KB
```

Figure 4. Dataset verification using pandas, the content is equivalent to scores, so there are no missing entries

2.4.1 Labeling

After combining the three datasets, the information was processed to eliminate non-Arabic and special characters, and the various comments were rated as neutral scored 0, or abusive scored 1, the result as appears in Figure 5 is an unbalanced labeled dataset.

```
0    28898
1     9756
Name: sentiment, dtype: int64
```

Figure 5. Labeled dataset before balancing has 28898 hate speech texts and 9756 neutral

2.4.2 Balancing the dataset

Trying to learn from an imbalanced dataset always favors the bigger class, which can lead to misleading results after the training. This is a specific problem because we want to perform a correct classification between two classes, not multi-class. The BERT classification algorithm have a bias toward majority classes, since it works in a way to discover the rules with high values of accuracy and coverage, whereas specific rules that predict minority instances are ignored or treated as noise. Consequently, minority instances are misclassified, because the classifier is designed to minimize the error rate. If we do the classification training on an imbalanced dataset, while it will be performing the task, the model, will produce general rules and will have a bias toward majority instances, and it will ignore the minority ones, which means that in this case, the model will consider even texts that doesn't include hate speech as if they do.

Several methods can be used to deal with such problem and since we are working on a text (string) database, two best solutions are available:

Down sample: which means reducing the existing items in the majority class to match the other class, several techniques can be used to apply this method, such as, the random undersampling, in which entries will be removed randomly from the dataset, without any concern about their importance or significance. Another technique is the discerning undersampling, in this technique, the elements of the majority class are classified according to their importance and the ones that are removed correspond to the less important class.

Oversampling is done by applying a data augmentation. There are several techniques to do it, such as Increasing significant categories of samples, by applying bidirectional translations, or replacing expressions by their synonyms, which increases accuracy.

The second solution is the one we used, since the down sample method will cost us a considerable loss of training data which could have a negative impact on the results. The oversampling is applied by increasing the minority class to balance the dataset. The post-processing dataset, as we see in Figure 6, consists of a mixture of 57,796 text entries from classical Arabic and other different Arabic dialects, and we can see examples of it in Figure 7.

```
1    28898
0    28898
Name: sentiment, dtype: int64
```

Figure 6. Oversampled dataset details

	content	score
0	وإيلي حتى فات الفوت عاد سولدي كيف ابقيت ويلي و...	0
1	أكيد عاصفة الحزم السعودية الخليجية قادمة لنصرة...	-1
2	روح انتحر كب حالك بالبحر اطهر ارض و انجس قوم"	-1
3	التي مسخرة بشرفي	-1
4	جبران باسيل بدو يعلّم لادن و واشنطن كيفية إدا...	0
5	جماعة الديار الله يحميك	0
6	ريفا خونة بخاو بقسمو البلاد	-1
7	جدا سمعك شي كلمة حلوة بتليق بمقامك اكيد لا طي...	-1

Figure 7. Example extracted from the final result of the dataset including classical Arabic, Levantine, and North African dialects

The combined Dataset can be downloaded using this Google drive file ID: id 1MCXY5eyI7myKyQQ2ZPpI1RHPZR7Emd2e, or this URL: https://drive.google.com/file/d/1MCXY5eyI7myKyQQ2ZPpI1RHPZR7Emd2e/view?usp=sharing.

2.4.3 Tokenization

In order to use the embedded BERT text as input to train our hate detection model, we need to tokenize the text of our comments. Tokenization refers to breaking down a sentence into individual words.

The Tokenizer applied by BERT code utilizes word-piece embedding processes [32] that use a shared learning framework from sequence to sequence. It consists of three components: layers of encoders, layers of decoders, and an attention layer. The task carried by the encoder is converting the original sentence into an index of vectors, one for each and every symbolized input. The work of the decoder is to generate a specific symbol for each input, until it reaches the special symbol for the last word in sentence. The link between the encoder and the decoder is the attention layer which enables the decoder to work on various parts of the input sentence while applying the decoder.

Let (X, Y) be a pair of source and target sentences. Let $X=x_1, x_2, x_3, ..., x_M$ be the sequence of M symbols in the original expression and let $Y=y_1, y_2, y_3, ..., y_N$ be the sequence of N symbols in the output expression. The encoder is in general a function of the form shown in (4):

$$x_1, x_2, \ldots, x_M = EncoderRNN(x_1, x_2, x_3, \ldots, x_M) \quad (4)$$

In this equation $x_1, x_2, ..., x_M$ constitutes a list of vectors of determined size. The count of members in the list is similar to the number of symbols in the original expression (M in this

example). Using the Conditional Chain Rule the probability of the sequence $P(Y|X)$ can be detailed as in (5):

$$P(Y|X) = P(Y|x_1, x_2, x_3, \dots, x_M) = \prod_{i=1}^{N} P(y_i|y_0, y_1, y_2, \dots, y_{i-1};\ x_1, x_2, x_3, \dots, x_M) \quad (5)$$

where, y_0 is a special "first part of the expression" symbol that is preceded by each output expression. During inference, the probability of the coming symbol is calculated given the encoding of the original phrase and the output phrase decoded so far as in (4):

$$P(y_i|y_0, y_1, y_2, y_3 \dots, y_{i-1};\ x_1, x_2, x_3, \dots,) \quad (6)$$

To tokenize our Data, we use the AutoTokenizer module from the Transformers library, and we set “asafaya/bert-large-arabic” as Tokenizer [26], this step enables us to convert the text into Tokens and the Tokens into Ids that can be used in the training process, the Tokenization is done using the “encode_plus()” method, with truncation, and maximum length set to 512. After storing the tokens, we choose the maximum sequence length for the training process, in order to reduce the training time, because our dataset does not include very long texts. This choice is made after having displayed in the form of a bar graph the number of tokens stored for each entry, this technique shows us, that the maximum length is always less than 100 Tokens in this case. The information that we have is used to create a Pytorch Dataset, and we split our data into, test data frame 5%, validation data frame 5%, and training data frame 90%.

In Figure 8 we see an example of a tokenized sentence.

```
Sentence: عندما كنت في المدرسة حصلت أشياء غريبة كثير
  Tokens: ['عندما', 'كنت', 'في', 'المدرسة', 'حصلت', 'اشياء', 'غريبة', 'كثير']
Token IDs: [2900, 2779, 1725, 5352, 8796, 9623, 12162, 2920]
```

Figure 8. Sentence tokenized with ArabicBERT tokenizer

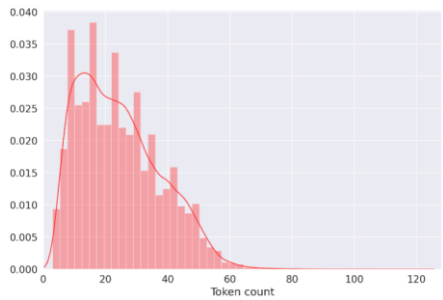

Figure 9. Representation of tokens contained in different comments

As BERT works with a fixed length of tokens, having for maximum 512, we have stored the tokens of each comment and we have represented them in graph, to be able to detect the maximum length on which we have set our training.

As we can see in Figure 9, most of the comments appeared to contain less than 80 tokens, but to be on the safe side, we chose a maximum length of 100.

2.4.4 Training

In this step, we worked with PyTorch deep learning library [33], which is an open source machine learning framework that accelerates the path from research prototyping to results deployment. We used the hyper parameters of the BERT paper and we used the AdamW optimizer provided by Hugging Face [34] which is a stochastic optimization method that modifies the typical implementation of the weight decay in Adam, decoupling the weight decay from the gradient update. L_2 Regularization in Adam is usually implemented with the modification below where w_t is the rate of weight decay at time t as it is in (7):

$$g_t = \nabla f(\theta_t) + \omega_t \theta_t \quad (7)$$

While AdamW adjusts the weight decay term to appear in the gradient update, as in (8):

$$\theta_{t+1,i} = \theta_{t,i} - n\left(\frac{1}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t + \omega_{t,i}\theta_{t,i}\right) \quad (8)$$

In our case, AdamW corrects the weights decay, so it is similar to the original paper. We also used a linear planner with no warm-up steps.

For fine-tuning, we applied the recommendations of the authors of the BERT paper [27], namely: Batch size: 16, 32 Learning rate (Adam): 5th-5, 3rd-5, 2nd-5.

For the number of epochs, we proceeded by trial and error, having tried several values and our choice ended up being 10 epochs, since it gave the best results.

During the training process we create for every fraction of the data, a loader, with a batch size equal to 16, immediately afterwards we import the BERT model for Arabic, "asafaya/bert-large-arabic" [26] from the Transformers library, the classifier function using the BERT model is defined, with a Dropout layer set at 30%, and a fully-connected output layer, the defined function, returns the raw output of the Last layer, and we move the classifier to the GPU. We apply a Softmax function to the outputs, in aim to have probabilities for the trained model.

We use AdamW optimizer, from the Transformers library, to correct the weight decay, with a learning rate set to 2e-5; a training loop is defined, by iteration, for 2 Epochs, 4 Epochs, 8 Epochs, 10 Epochs, 12 Epochs, 14 Epochs, and the best results are those of a number of Epochs equal to 10, we store the training history, and when the training process finishes, we get the trained model, which can be downloaded using this Google drive file ID: 1InzP8KVk8C-qLINXa-tSsGM6PftOsW7j, or this URL: https://drive.google.com/file/d/1InzP8KVk8C-qLINXa-tSsGM6PftOsW7j/view?usp=sharing.

2.4.5 Application

To test the result, we create an automatic system for hate speech detection, based on the trained model, in different steps, the araby element is imported from the Pyarabic module, it’s used to tokenize the input text, and split it into small paragraphs and sentences, the input text is classified afterwards using a classifying function with the help of the model, which returns the name of the class as result for the input text.

Text that will be automatically detected for hate language requires specific processing for better performance, this process is essential for two reasons, first, the diacritics and lengthening of the text should be removed so that it is similar to what the training was done on. Second, the text submitted for detection must correspond to the maximum length of the

content used in the training, which has been set at 100 tokens, because beyond this length, the detector loses its performance. This step requires the use of PyArabic [35], an Arabic language processing library that allows us both to make the necessary modifications to the input text, as well as its tokenization for conversion into sentences with length that is less than the maximum. We try this system on Arabic and different dialects to test the convenience of the results, found to be accurate in a relation with the metrics, different examples are shared in the Results Chapter.

3. RESULTS

The training that we carried out gave rise to an automatic detection model of general hate speech with a remarkable precision as we can see in Figure 10.

	precision	recall	f1-score	support
Abusif	0.87	0.91	0.89	7196
neutral	0.70	0.62	0.66	2468
accuracy			0.83	9664
macro avg	0.79	0.76	0.77	9664
weighted avg	0.83	0.83	0.83	9664

Figure 10. Trained model precision metrics

To test the trained model in application we used both extracts from press articles in classical Arabic and in different dialects as well as comments in electronic journals, these tests revealed the power of the BERT method and the ability of the model we fine-tuned, to perform the text classification task with remarkable precision.

For the final evaluation the texts used are not extracted from the training dataset, and in these cases the detection was appropriate, we share some results as examples, in Figures 11-15.

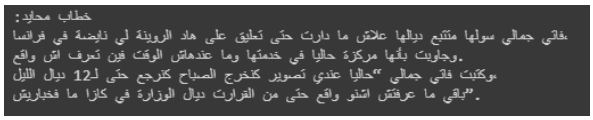

Figure 11. example 1 of automatic detection of hate speech in: Moroccan dialect, text taken from an article in an electronic journal (neutral)

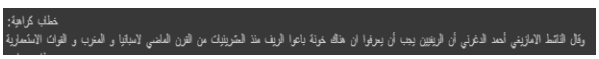

Figure 12. example 2 of automatic hate speech detection in classical Arabic, text taken from an article in an electronic journal (hate speech)

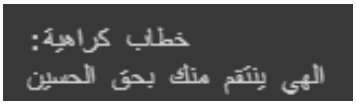

Figure 13. example 3 of automatic hate speech detection in Levantine, text taken from a YouTube comment (hate speech)

Figure 14. example 4 of automatic hate speech detection in Moroccan dialect comment (neutral)

4. DISCUSSION

Thanks to the BERT method, we were able to have a general rate of accuracy in distinguishing hate speech from neutral one of about 83%, with regard to the detection of hate speech, the metrics of the model show a superior performance with a Precision of 0.87, Recall of 0.91, and an F1-Score of 0.89 (Figure 10). In Figure 15 we can see the confusion matrix showing the superiority of the model in detecting hate speech than the neutral language, resulting in an overall precision of 83%.

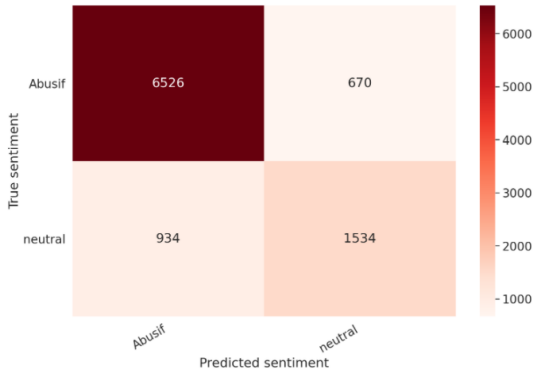

Figure 15. Model performance confusion matrix

Compared to the previous work mentioned in Table 1, the model we have trained performs a more specific task of detecting hate speech, and is not interested in abusive language or offensive, which are more general than the concept of hate speech and which requires taking into consideration the context of words and not just their current use, which is why we have chosen the two-way Transformer method.

The preceding works have taken into account only the classical Arabic language, or one determined dialect at a time, but in our case we proceeded through the training of the model to process the classical Arabic language in addition to several other dialects, based on the Arabic language.

5. CONCLUSION

In this article we have addressed the problem of automatic detection of hate speech, in Arabic read media, which is an essential step in the protection of readers and users of social media, the results obtained are satisfactory in terms of accuracy which is 83%, and constitute a considerable contribution in the matter, since the detection takes into account not only classical Arabic but also different dialects such as Levantine and Darija from North Africa.

That said, the study can be deepened by constituting a database that can make it possible to form a model capable not only of detecting hate speech but of categorizing it according to the different types of hate language that exist, our future work will focus on a categorized hate speech database.

REFERENCES

[1] Al-Hassan, A., Al-Dossari, H. (2019). Detection of hate speech in social networks: A survey on multilingual corpus. In 6th International Conference on Computer

Science and Information Technology, 10: 83-100. http://dx.doi.org/10.5121/csit.2019.90208
[2] Mahmood, A.T., Kamaruddin, S.S., Naser, R.K. (2020). A combination of lexicon and machine learning approaches for sentiment analysis on Facebook. Journal of System and Management Sciences, 10(3): 140-150. http://dx.doi.org/10.33168/JSMS.2020.0310
[3] Abuaiadah, D., El Sana, J., Abusalah, W. (2014). On the impact of dataset characteristics on Arabic document classification. International Journal of Computer Applications, 101(7): 31-38. http://dx.doi.org/10.5120/17701-8680
[4] Abu-Errub, A. (2014). Arabic text classification algorithm using TFIDF and chi square measurements. International Journal of Computer Applications, 93(6): 40-45. http://dx.doi.org/10.5120/16223-5674
[5] Soliman, A.B., Eissa, K., El-Beltagy, S.R. (2017). Aravec: A set of Arabic word embedding models for use in Arabic NLP. Procedia Computer Science, 117: 256-265. http://dx.doi.org/10.1016/j.procs.2017.10.117
[6] El Mahdaouy, A., Gaussier, E., El Alaoui, S.O. (2016). Arabic text classification based on word and document embeddings. In International Conference on Advanced Intelligent Systems and Informatics, pp. 32-41. http://dx.doi.org/10.1007/978-3-319-48308-5_4
[7] Mohammad, A.H., Al-Momani, O., Alwada'n, T. (2016). Arabic text categorization using k-nearest neighbour, Decision Trees (C4. 5) and Rocchio classifier: A comparative study. International Journal of Current Engineering and Technology, 6(2): 477-482. http://inpressco.com/wp-content/uploads/2016/03/Paper16477-482.pdf.
[8] Alshammari, R. (2018). Arabic text categorization using machine learning approaches. International Journal of Advanced Computer Science and Applications, 9(3): 226-230. http://dx.doi.org/10.14569/IJACSA.2018.090332
[9] El-Halees, A.M. (2007). Arabic text classification using maximum entropy. IUG Journal for Natural and Engineering Studies, 15(1): 157-167. https://journal.iugaza.edu.ps/index.php/IUGNS/article/view/173.
[10] Jindal, V. (2016). A personalized Markov clustering and deep learning approach for Arabic text categorization. In Proceedings of the ACL 2016 Student Research Workshop, pp. 145-151. http://doi.org/10.18653/v1/P16-3022
[11] El Kourdi, M., Bensaid, A., Rachidi, T.E. (2004). Automatic Arabic document categorization based on the Naïve Bayes algorithm. In proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages, pp. 51-58. http://dx.doi.org/10.3115/1621804.1621819
[12] Gharib, T.F., Habib, M.B., Fayed, Z.T. (2009). Arabic text classification using support vector machines. Int. J. Comput. Their Appl., 16(4): 192-199. http://dx.doi.org/10.7603/s40601-016-0016-9
[13] Kanaan, G., Al-Shalabi, R., Ghwanmeh, S., Al-Ma'adeed, H. (2009). A comparison of text-classification techniques applied to Arabic text. Journal of the American Society for Information Science and Technology, 60(9): 1836-1844.
[14] Sawaf, H., Zaplo, J., Ney, H. (2001). Statistical classification methods for Arabic news articles. Natural Language Processing in ACL2001, Toulouse, France. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.634&rep=rep1&type=pdf.
[15] Saad, M.K. (2010). The impact of text preprocessing and term weighting on Arabic text classification. M.S. thesis, Islamic Univ.-Gaza, Gaza, Palestine. http://dx.doi.org/10.13140/2.1.4677.2164
[16] Al-Harbi, S., Almuhareb, A., Al-Thubaity, A., Khorsheed, M.S., Al-Rajeh, A. (2008). Automatic Arabic Text Classification. 9es Journées internationales d'Analyse statistique des Données Textuelles, pp. 77-83. https://citeseerx.ist.psu.edu.
[17] Altowayan, A.A., Tao, L. (2016). Word embeddings for Arabic sentiment analysis. In 2016 IEEE International Conference on Big Data (Big Data), pp. 3820-3825.
[18] Sandeep Bidwai, S.B. (2021). A comparative study of Markov chain and deep learning predictive models in spectrum sensing. Journal of System and Management Sciences. http://dx.doi.org/124-140.10.33168/JSMS.2021.0108
[19] Abozinadah, E.A., Jones Jr, J.H. (2017). A statistical learning approach to detect abusive twitter accounts. In Proceedings of the International Conference on Compute and Data Analysis, pp. 6-13. http://dx.doi.org/10.1145/3093241.3093281
[20] Abozinadah, E.A., Jones, J. (2016). Improved micro-blog classification for detecting abusive Arabic Twitter accounts. International Journal of Data Mining & Knowledge Management Process (IJDKP), 6(6): 17-28.
[21] Abozinadah, E.A., Mbaziira, A.V., Jones, J. (2015). Detection of abusive accounts with Arabic tweets. Int. J. Knowl. Eng.-IACSIT, 1(2): 113-119. http://dx.doi.org/10.7763/IJKE.2015.V1.19
[22] Chowdhury, S.A., Mubarak, H., Abdelali, A., Jung, S.G., Jansen, B.J., Salminen, J. (2020). A multi-platform Arabic news comment dataset for offensive language detection. In Proceedings of the 12th Language Resources and Evaluation Conference, pp. 6203-6212. https://aclanthology.org/2020.lrec-1.761
[23] Mohaouchane, H., Mourhir, A., Nikolov, N.S. (2019). Detecting offensive language on arabic social media using deep learning. In 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), Granada, Spain, pp. 466-471. http://dx.doi.org/10.1109/SNAMS.2019.8931839
[24] Mubarak, H., Darwish, K., Magdy, W. (2017). Abusive language detection on Arabic social media. In Proceedings of the First Workshop on Abusive Language Online, pp. 52-56. http://dx.doi.org/10.18653/v1/W17-3008.
[25] Albadi, N., Kurdi, M., Mishra, S. (2018). Are they our brothers? Analysis and detection of religious hate speech in the Arabic Twittersphere. In 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, Spain, pp. 69-76. http://dx.doi.org/10.1109/ASONAM.2018.8508247
[26] Safaya, A., Abdullatif, M., Yuret, D. (2020). KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, pp. 2054-2059. https://arxiv.org/abs/2007.13184.
[27] Devlin, J., Chang, M.W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for

language understanding. arXiv preprint arXiv:1810.04805. https://arxiv.org/pdf/1810.04805.

[28] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I. (2018). Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/language_understanding_paper.pdf.

[29] Peters, M.E., Neumanaa, M., Iyyer, M., et al. (2018). Deep Contextualized Word Representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1: 2227-2237. http://dx.doi.org/10.18653/v1/N18-1202

[30] Mulki, H., Haddad, H., Ali, C.B., Alshabani, H. (2019). L-hsab: A Levantine twitter dataset for hate speech and abusive language. In Proceedings of the Third Workshop on Abusive Language Online, pp. 111-118. http://dx.doi.org/10.18653/v1/W19-3512

[31] Alakrot, A., Murray, L., Nikolov, N.S. (2018). Dataset construction for the detection of anti-social behaviour in online communication in Arabic. Procedia Computer Science, 142: 174-181. http://dx.doi.org/10.1016/j.procs.2018.10.473

[32] Wang, H.F., Wu, H., He, Z., Huang, L., Church, K. (2021). Progress in machine translation. Engineering. http://dx.doi.org/10.1016/j.eng.2021.03.023

[33] Paszke, A., Gross, S., Massa, F., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems, 32: 8026-8037. https://arxiv.org/abs/1912.01703.

[34] Loshchilov, I., Hutter, F. (2018). Fixing weight decay regularization in Adam. https://openreview.net/forum?id=rk6qdGgCZ

[35] Zerrouki, T. (2010). An Arabic language library for Python. Pyarabic. https://pypi.org/project/PyArabic/.