applied sciences

MDPI

Article

Automated Arabic Long-Tweet Classification Using Transfer Learning with BERT

Meshrif Alruily [1,*], Abdul Manaf Fazal [2], Ayman Mohamed Mostafa [1] and Mohamed Ezz [1]

[1] College of Computer and Information Sciences, Jouf University, Sakaka 72388, Jouf, Saudi Arabia; amhassane@ju.edu.sa (A.M.M.); maismail@ju.edu.sa (M.E.)
[2] Electronics and Communication, Punjab Engineering College, Chandigarh 160030, India; manaf4071@gmail.com
* Correspondence: mfalruily@ju.edu.sa

Abstract: Social media platforms like Twitter are commonly used by people interested in various activities, interests, and subjects that may cover their everyday activities and plans, as well as their thoughts on religion, technology, or the products they use. In this paper, we present bidirectional encoder representations from transformers (BERT)-based text classification model, ARABERT4TWC, for classifying the Arabic tweets of users into different categories. This work aims to provide an enhanced deep-learning model that can automatically classify the robust Arabic tweets of different users. In our proposed work, a transformer-based model for text classification is constructed from a pre-trained BERT model provided by the hugging face transformer library with custom dense layers. The multi-class classification layer is built on top of the BERT encoder to categorize the tweets. First, data sanitation and preprocessing were performed on the raw Arabic corpus to improve the model's accuracy. Second, an Arabic-specific BERT model was built and input embedding vectors were fed into it. Using five publicly accessible datasets, substantial experiments were executed, and the fine-tuning technique was assessed in terms of tokenized vector and learning rate. In addition, we assessed the accuracy of various deep-learning models for classifying Arabic text.

Keywords: text classification; Arabic tweets; BERT model; hugging-face transformer; deep learning models

check for updates

Citation: Alruily, M.; Manaf Fazal, A.; Mostafa, A.M.; Ezz, M. Automated Arabic Long-Tweet Classification Using Transfer Learning with BERT. *Appl. Sci.* **2023**, *13*, 3482. https://doi.org/10.3390/app13063482

Academic Editors: Peter Blišťan, Ladislav Hvizdák and Lucia Domaracká

Received: 3 February 2023
Revised: 3 March 2023
Accepted: 6 March 2023
Published: 9 March 2023

1. Introduction

The importance of automatic tweet or text classification has increased with the ongoing addition of text-based content to the internet. The rapid growth of social and digital media has resulted in an ever-increasing amount of data generated by various sources. The availability of such unorganized data provides an enormous potential for processing and extracting valuable information. One critical task is identifying individuals depending on their tweeting and grouping, a field of study that has gained increasing interest in recent years. Automated multi-labeled systems are now possible because of recent advances in deep-learning- and machine-learning-based approaches. Tweet or text classification is a necessary condition for the development of various emerging applications in diverse fields, such as linguistic (and dialect) recognition, user's tweet classification, sentiment analysis, genre classification, spam filtering, and a few more [1–4]. There is a significant need for an automatic tweet or text categorization system for Arabic due to the abundance of Arabic data available on the web nowadays.

There are different criteria for classifying Arabic tweets, depending on the accessibility of data, resource limitations, and data quality. The Arabic language is considered one of the most widely used languages on social media, especially Twitter and Facebook. Most user and people attitudes are reflected on social media to express their opinions to different news, topics, and provided services that can be enhanced in future. Furthermore, the Arabic language on social media platforms contains long tweets with different morphological and polarity terms that are difficult to analyze.

As presented in [5], the unsupervised sentiment analysis method is applied to improve the performance of analyzing sentiments using term weights from different datasets. In addition, the features are extracted for predicting the target or objective features. The authors of [6] proposed different machine-learning models to classify text and provide a recommender system that can predict the orientation of sentiments. Despite this, they frequently fail to consider textual context or word order, so their classification accuracy needs to be improved by data sparsity. Furthermore, deep-learning-based techniques have been proven more efficient than conventional machine-learning approaches. They can extract important characteristics effectively without requiring extensive artificial feature extraction [7]. The RNN and CNN have demonstrated varying capacities in displaying text. RNNs are good at modeling sequential data and effectively representing text by understanding contextual aspects and long-term relationships in phrases. They have been successfully employed in natural language-processing applications [8]. Some known algorithms for text categorization combine CNN and RNN to maximize their potential [9]. However, these algorithms treat every word in the phrase equally, making it impossible to tell which keywords are more important in the categorization process than the more prevalent ones. According to the attention-based technique, neural networks may give varying weights to words in a sentence based on their value for categorization, alleviating the previous difficulties. Attention-based RNN, CNN, and GRU can produce state-of-the-art results. However, these techniques are attributed to individual CNN, GRU, or RNN, and the significance of learning contextual information is insignificant in their implementation [10]. The extent of BERT's potential has yet to be explored, even though it has given reasonable results in NLP-related tasks surpassing most conventional approaches, such as word2vec, Cove, and Glove. Incorporating BERT does not help much because it is trained using a blend of a corpus in English and has no text categorization skills. When there is insufficient internal training data for fine-tuning, a task-awareness problem may be solved by using an external domain-related corpus.

To overcome the limitations of the aforementioned models, we proposed a model ARABERT4TWC for Arabic tweet classification. We attained performance compared to other well-known techniques, with three out of five datasets used. First, data sanitation is performed on the datasets to remove Arabic stop words and punctuation. As a part of data preprocessing, the model input is first tokenized with a word-piece tokenizer trained on a large Arabic corpus that contains 1.5-billion Arabic corpora and OSIAN, which are publicly available [11,12]. The word-piece tokenizer is based on a sub-word tokenization algorithm. Sub-word tokenization techniques are based on the idea that commonly employed phrases must be distinct from shorter sub-words. However, unique words should be deconstructed into meaningful sub-words when tokenizing them. Sub-word tokenization helps the model acquire meaningful, contextually independent representations while maintaining a suitable lexicon size.

The tokens obtained from the word-piece tokenizer are combined with segment and positional embedding. Second, an Arabic BERT for tweet classification is constructed by fine-tuning the pre-trained BERT case model provided by the Hugging Face Transformer Library that has been trained on larger unlabeled text corpora, including book corpora and Wikipedia [13–15]. The fine-tuning is executed by initializing the weights of the pre-trained BERT model. All the 109-million parameters are fine-tuned with labeled data. A dense classification layer is added to the encoder to obtain an Arabic BERT, specifically used for Arabic tweet classification. The custom BERT model consists of 12 encoder layers with 12 self-attention heads, which are bi-directional, and 12 hidden layers. The final embedding layers obtained after merging with segment and positional embedding are passed through the encoder structure. Finally, the Softmax classification layer is used for classifying Arabic tweets. To validate our proposed model, the model's performance in terms of accuracy is evaluated using five benchmark datasets with other well-known existing models for tweet classification. The contribution of the paper is explained as follows:

1. Propose an ARABERT4TWC model for classifying Arabic long tweets into different categories.
2. Propose a tokenization process algorithm for measuring the frequency of each pair of text in the vocabulary and then determining the most frequent terms.
3. Proposing an ARABERT4TWC algorithm that contains a token, segment, and positional embedding of terms, then applying an encoder/transformer.
4. Applying a fine-tuning of the ARABERT4TWC model ARABERT4TWC to achieve better performance on social media-related tasks.
5. Applying the proposed model to different datasets and the experimental results are compared with recent research methodologies, which explored high accuracy in the proposed model.

The remaining structure of the paper is as follows: The related work in text or tweet classification is reported in Section 2. The proposed methodology Arabic BERT model for tweet classification (ARABERT4TWC) is presented in Section 3, which explains the sanitation of data, the model architecture, the proposed algorithms, the self-attention layer, and the classification layer. The experiment setup, performance evaluation of our proposed model, and a comparison of the existing techniques are explained in Section 4. Section 5 summarizes the conclusion and future scope.

2. Related Works

Several research works addressed the problem of automated text classification, presenting various methodologies, and approaches. Deep-learning-based neural network models have significantly progressed in text categorization in recent years. In 2010, Sanad [16] proposed several traditional learning supervised classifiers, such as KNN, SVM, Decision Tree classifier, and Nave Bayes for Arabic text classification. He examined the effect of data preprocessing on text classification. The CNN and BBC Arabic news databases, which are extensively shared but relatively smaller, were used. In 2007, Umer and Kiyal [17] proposed a neural network-based approach for Arabic text classification, which outperformed the KNN, Decision Tree classifier, SVM, and NB methods in terms of accuracy with relatively smaller datasets. Learning vector quantization was used for categorizing or classifying the text content. In 2017, Gaussier, Mahdaouy, and Alaoui [18] performed document and word embedding instead of data preprocessing and keyword counting. It was demonstrated that text embedding performed better than preprocessing strategies when they were learned with Doc2Vec, or when vectors that consisted of words were averaged. Word embedding was contrasted with established techniques that depend on data acquisition or keyword enumeration in various applications, such as tweet classification [19].

The development of deep-learning-based algorithms to handle various challenges has increased significantly over the previous four years. This is because it outperforms traditional learning algorithms based on performance. In 2019, Vincent and Ogier explained the significance of applying deep learning to solve content analysis constraints. The deep-learning algorithm based on convolutional neural networks could extract deep characteristics or features from texts or tweets, producing better results in text classification [20]. Text categorization was improved in 2014 when Kim created the first CNN (Convolutional Neural Network) architecture with a simple yet effective topology [21]. Using unlabeled data, Jhonson and R developed a new semi-supervised CNN architecture for text classification that first learns text region embedding and then labels them [22]. In 2018, the authors observed that by enhancing CNN's depth, sentences with a long-term correlation could be learned effectively when the text sequence is character-based instead of word-based encoding [23].

RNNs have been used in the machine translation, question-answering, and speech recognition fields to find long-term correlations in sequential data. In recent studies, it has been indicated that it may also be used in categorizing text [9,10]. As presented in [8], the authors proposed a recurrent convolutional neural network (RCNN) that utilized a periodic or recurrent structure to capture contextual data and CNN for feature extraction. As shown in [24], the authors proposed a deep-learning-based model that uses a condition-based random classifier and recurrent structures, such as bi-directional and simple LSTM (Long Short-Term Memory). The two models were adapted to the Arabic hotel reviews dataset, in which the first approach was carried out with character-based and word-based encoding to extract the text's most highly expressed phrases. Context-aware Gated Recurrent Units (C-GRU), which used an additional layer to retrieve the context data from tweets, were proposed by the authors in 2018 [25].

As presented in [26], an activation function is proposed to enhance the performance on the training dataset by optimizing the neural network. The authors aimed to provide different solutions related to training of data where one of the solutions applied activated gradient for eliminating the gradient problem. Furthermore, the authors provided different theorems to vanish the problem of a saddle point. The proposed model is implemented and compared with different ResNet models that achieved a high performance with the activated gradient. As shown in [27], a deep-learning structure comprises BERT–Base and a final emotional and sentimental analysis classification layer. By considering two datasets of tweets, they attained an accuracy of 92 and 90% for emotion and sentimental analysis, respectively, from which it was possible to conclude that Bert's language modeling power contributes significantly to achieving a good tweet classification. Small datasets were used, and the Bert model could not classify long tweets due to a lower number of attention layers. Since then, new language processing models, including BERT, Roberta, AL–BERT, and T5, increased performance by experimenting with alternative pre-training approaches, updated model topologies, and bigger training corpora. In 2021, Antoun et al. developed an ARABERT model for various downstream tasks like Named Entity Recognition (NER) and Sentiment Analysis (SA). They achieved a 92% accuracy for the various downstream tasks [28]. As proposed in [29], the authors performed a self-training of the Arabic dataset that contains different dialects that cause low performance when a new dataset is added. The authors of [30] provided a sequence-to-sequence model that performs a text normalization of the Arabic dialect. The paper proposed an encoder/decoder model where the encoder accepts the input text, builds the features, and then collects the features to generate the final output. As explained in [31], the Naïve Bayes model is applied to the spark dataset to collect tokens, and then the polarity terms are counted to measure the accuracy. The authors of [32] applied different-machine learning models to extract user reviews from Saudi institutions to improve the prediction that can help make accurate decisions for providing services. As presented in [33], the authors collected Arabic tweets that can explain and express the depression statements in the tweets. The novelty of the proposed model is that the authors applied many symptoms to classify the depression classes and then detect whether the applied sentiment contains depression terms or not. As shown in [34], the authors collected different Arabic tweets about the COVID–19 pandemic to detect the user reactions and feelings based on their polarity. In addition, the proposed model tried to predict the spread of the pandemic based on user reviews.

Table 1 summarizes the learning method used and the demerits of the existing methods applied in tweet or text classification systems. The major disadvantages of the existing methods are that neither the conventional methods nor the models, such as recurrent networks or convolutional networks, can provide more significance to keywords that are significant in text categorization. Contextual awareness between words plays a key role in a tweet or text classification. Another significant disadvantage of the existing methods is that they are tested on small Arabic text corpora, and accuracy is also impacted due to outliers in data.

Table 1. Summary of the learning method used and demerits of the existing methods.

Ref	Learning Method	Algorithm	Demerits of Existing Method
[16]	Supervised	KNN, SVM, DT, and NB	Features reduce classification per accuracy due to increased spatial and temporal complexity.
[18]	Unsupervised	Doc2vec	Cannot capture contextual awareness in texts and increased vocabulary size for each different vector.
[20]	Supervised	CNN	Words are equally treated throughout the phrase, making it impossible to discriminate between keywords and terms.
[24]	Supervised	BLSTM	Needs a large training time and less accurate on a large dataset.
[25]	Supervised	Context-aware gated recurrent units	Cannot capture contextual information effectively.
[29]	Unsupervised	Self-training of corpus dataset	The dataset is self-trained with different dialects in the corpus, which causes low performance when a new dataset is added.
[30]	Unsupervised	Seq2Seq approach	The results are medium as the corpus contains different dialects.
[31]	Supervised	NB	The size of the dataset is small and needs to be increased.
[32]	Supervised	KNN, SVM, DT, and NB	The polarity terms size is small, and the performance accuracy is relatively low.
[33]	Supervised	SVM, RF, LR, KNN, NB, and AdaBoost	The model deals with the depression dataset but has a low number of tweets due to the lack of data in this specialty.
[34]	Supervised	NLP approach	Each sentiment polarity is assigned a weight, but the collected dataset cannot capture additional polarity terms.

3. Proposed Methodology

The proposed methodology and the architecture of the Arabic BERT model for tweet classification (ARABERT4TWC) are discussed in detail. The proposed methodology can be divided into four parts, as depicted in Figure 1. The first benchmark datasets are collected from different sources [35]. Second, data sanitation and preprocessing are performed on all the datasets collected. The sequence of tweet inputs is preprocessed before classification in batches using the word, segment, and positional embedding and converted to input vectors. Third, the input vectors obtained are passed through the custom-built Arabic BERT model (ARABERT4TWC). In the last step, we input this output into a linear network with two dense layers and a sigmoid activation function.

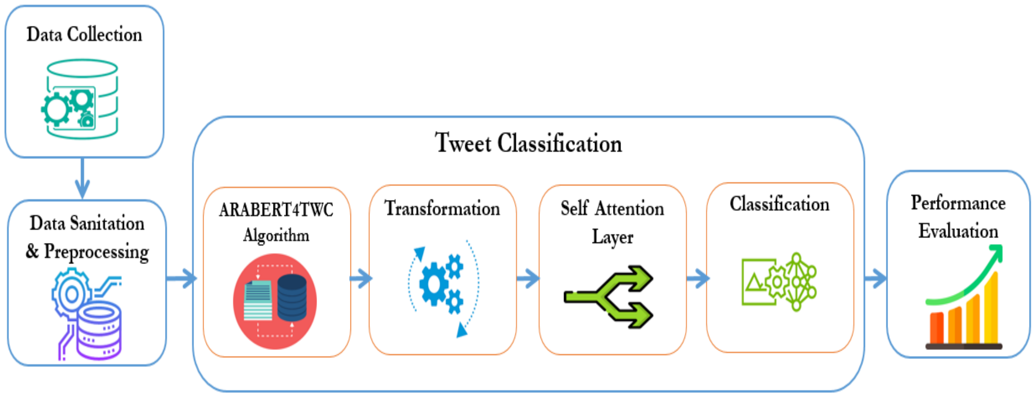

Figure 1. Proposed methodology of Arabic tweet classification layers.

3.1. Data Sanitation and Preprocessing

The data sanitation process removes Arabic stop words, punctuation, and extra spaces from the text corpus. Text normalization is not performed as it might alter the meaning of some words, such as "ball". After performing data sanitation on the text corpus, the data preprocessing step is carried out to convert the input text sequence to the embedding vectors, which are obtained after merging the sub-word tokenization, segment embedding, and positional embedding into the input sequence. Special tokens [CLS] are added at the beginning of the input sequence, and the [SEP] token is used to separate a long sentence into a meaningful sentence. The input sequences are tokenized with a word-piece tokenizer. The word-piece tokenizer is based on the sub-word tokenization algorithm. Sub-word tokenization techniques are based on the idea that commonly employed phrases must be distinct from shorter sub-words. However, unique words should be deconstructed into meaningful sub-words when tokenizing them. After tokenization and using a word-piece tokenizer, the input sequences are represented numerically using token embedding. The BERT model discriminates between various phrases in a single input segment. Using this embedding vector, the values of each word in a sentence are the same, but the values vary when the phrase is changed. The positional embedding vectors are used to calculate the location of each word or the distances between distinct phrases in the sequences.

As presented in Algorithm 1, the tokenization algorithm is based on filling the vocabulary V_i with the input text characters. Even if the vocabulary V_i is less than the ideal vocabulary IV, the frequency of each pair of text is calculated, and then the most frequent pair of text $\left(x_i, y_j\right)$ is determined. A new unused pair of text is added to expand the vocabulary. The input text characters are divided into word segments WS_i then the process is repeated again for each text entry until the entire text document is finished.

Algorithm 1: Tokenization Process

```
Let $IV \leftarrow Ideal\ Vocabulary$
Let $V_i \leftarrow Applied\ Vocabulary$
Let $TC \leftarrow Text\ Character$
Let $WS \leftarrow Word\ Segment$
Step 1:   $\forall\ TC_i \in V_i\ such\ that\ V_i = \{TC_1, TC_2, \ldots, TC_n\}$
Step 2:   $IF\ V_i \leq IV\ THEN$
              Count $freq\left(x_i, y_j\right)$
              Find $Max\ freq\left(x_i, y_j\right)$
              Substitute $\left(x_i, y_j\right)\ with\ new\ (x_k, y_l)$
              Add $(x_k, y_l)$ to $V_i$
Step 3:   Divide $TC_i$ such that $TC_1 = \{WS_1, WS_2, \ldots, WS_n\}$
Step 4:   Repeat Step two & Step three
Step 5:   Apply $V_i = \{WS_1, WS_2, \ldots, WS_n\}$
Step 6:   Train $V_i$
```

3.2. Model Architecture

We constructed an Arabic BERT model for text classification from a pre-trained BERT model by fine-tuning the 109-million parameters with the labeled data. The network architecture of the constructed Arabic BERT model (AR-ABERT4TWC) is shown in Figure 1. As presented in Algorithm 2, the input vector, which is a combination of token embedding, segment embedding, and positional embedding, has a shape of (n, 768) where n is the size of the input sequence length, and the model can take an input sequence up to 512 sizes. The custom BERT model consists of 12 encoder layers with 12 self-attention heads, which are bi-directional in nature, and 12 hidden layers. Each self-attention head in the encoders computes the key, value, and query for each input token in a sequence, which is then utilized to produce a weighted representation of the sequence. All the outputs obtained from self-attention heads in similar layers are merged and routed through a fully linked layer. Every layer is preceded by layer normalization and is linked with a skip connection. The final output is taken through the [CLS] token vector, and it is passed through the linear layer and finally through the $SOFT_{Max}$ layer for classification. The final output of the classification model is to merge the output of the normalization process in both linear layer and $SOFT_{Max}$ classification layer.

Algorithm 2: ARABERT4TWC

```
1   Let $TE \leftarrow Token\ Embedding$
2   Let $SE \leftarrow Segment\ Embedding$
3   Let $PE \leftarrow Positional\ Embedding$
4   Input: BERT = $TE||SE||PE$
5   Let $n \leftarrow input\ sequence\ length\ size$
6   Let $d \leftarrow embedding\ dimension$
7   Calculate $Input_{vector}\ (n, d)\ such\ that\ max(d) = 512\ byte$
8   $\forall\ x_i\ \in x\ \exists\ x_i = \{x_1, x_2, \ldots, x_n\}\ \in TE$
9   $\forall\ s_i\ \in s\ \exists\ s_i = \{s_1, s_2, \ldots, s_n\}\ \in SE$
10  $\forall\ p_i\ \in p\ \exists\ p_i = \{p_1, p_2, \ldots, p_n\}\ \in PE$
11  Input: $Input_{vector} = \{x, s, p\}$
12  Apply Encoder/Transformer Layers
13  SET $Encoder\ _{layer}(L, H, D) = (12, 12, 12)\ such\ that\ Encoder\ Layer(L) = 12,$
    $Self - attention\ Layer\ (H) = \ 12, Hidden\ Layer\ (D) = 12$
14  $\forall\ Input_{token}\ Seq\ (Input_{vector})\ Calculate\ Output\ O_i\ (H) = \{key, value, query\}$
15  SET $Multi\ Head\ Self - attention\ (input)$
    $= Attention\ \left(Context_{before} + \ Context_{after} + \ Seq\ (Input_{vector})\right)$
16  SET $Attention\ (x) = \{key, value, query\}\ \forall\ Input_{token}\ (x)\ such\ that$
    $key = f(x), value = g(x), query = h(x)$
17  Merge Output $\sum_{i=1}^{n} O_i(H)$
18  SET Output = Merge $(\sum_{i=1}^{n} O_i\ (H) + fully\ linked\ layer)$
19  Apply Layer Normalization for Every Layer
20  Link Layer Normalization with Skip Connection
21  $Output_{normalization} = Layer\ Normalization\ (output) + Skip\ Connection\ (output)$
22  Pass Final Output through Linear Layer
23  Pass Final Output through $SOFT_{Max}$ Layer
24    $FinalOutput = \ LinearLayer(Output_{normalization})$
                    $+SOFT_{Max}(Output_{normalization})$
```

3.3. Transformer-Based Model

In order to perform NLP tasks like language translation and text synthesis, this paper uses a neural network architecture for the proposed transformer-based model [28]. The transformer model may consider the importance of different input components while making predictions as it is based on self-attention. In contrast to previous RNN models, which process input sequentially, the transformer model processes all input parts in parallel, allowing for faster training and inference. Each encoder and decoder in the transformer model has multiple layers of feed-forward and self-attentional neural networks. The input sequence is sent into the encoder, which creates hidden states. The decoder uses these hidden states to create the output sequence.

A key component of the transformer model is based on the mechanism of multi-head self-attention, which enables the model to attend to various input portions at various places. It is a crucial feature of the transformer model. This mechanism is implemented by computing multiple sets of key, value, and query vectors for each input token and then using them to compute a weighted sum of the input. Additionally, the transformer model includes skip connections and layer normalization, which help stabilize the training and improve the model ability to generate to new data. Overall, transformer-based models are considered successful in NLP tasks and have become the de facto standard in this field. They have been used in many recent models, such as BERT, GPT–2, and T5. Furthermore, the

Hugging Face Transformer library's BERT model was adjusted tuned for the classification task. The BERT model uses the pre-trained weights as a starting point, which can reduce the amount of labeled data required for fine-tuning and improve the model's performance on the specific task. BERT uses a transformer architecture with multiple layers of self-attention and feed-forward neural networks. The BERT's bidirectional nature, which takes into account the context before and after each token in the input sequence, is one of its key characteristics. BERT can comprehend a word's meaning in the context of the complete phrase instead of only the context of the words that came before it. The pre-training of the BERT model was performed on a large corpus of text data. A masked language model for estimating the proportion of input tokens that are randomly masked is the goal of the pre-training. This helps the model to understand the relationships between words and their meanings.

3.4. *Self-Attention Layer*

In a self-attention layer, each word or token in a sequence is assigned a specified weight or importance score based on its significance to the other words in the sequence. These weights are then utilized to calculate a weighted sum of the input sequence, which is then used as the output of the layer. The impact of selecting only the important words instead of all the words depends on the dataset being applied and used. The majority of the information may be concentrated in only a few key words, so selecting only those important words may be sufficient to achieve high accuracy. In other cases, there may be important information distributed across multiple words, so selecting only a subset of words may lead to a loss of accuracy.

For each word in the phrase, self-attention enables it to see the words around it, allowing it to choose which words are most important to the present word's meaning. Self-attention refers to the sentence's tendency to focus on itself while deciding how to describe each of its tokens. To comprehend a single word's meaning, one must know in what context that word is used, and that word is taken care of by the self-attention layer. The weights, which are matrices derived during model training, are multiplied by the input to produce the key, value, and query vectors for each input. Equation (1) represents how the input vectors are transformed when passed through the linear layers to get the key, value, and query vectors. Figure 2 shows the process for obtaining the key, value, and query vectors. The Y represents the input, each row of the Y matrix is the embedding vectors, and WK, WQ, and WV are the weight matrices of key, query, and value.

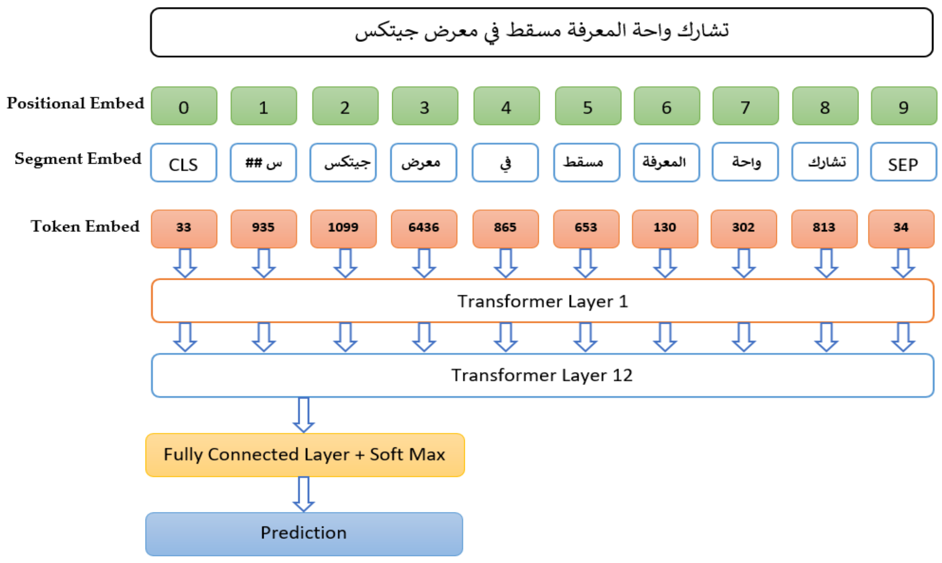

Figure 2. Architecture of ARABERT4TWC.

The calculation of attention weights as $SOFT_{Max}$-normalized dot products for each pair of words is shown in Equation (1), where β_{ij} is the attention weight, q_i is the query vector, k_i is the key vector, and v_i is the value vector.

$$\beta = \frac{exp\left(q_i^T k_j\right)}{\sum_{i=1}^{n} exp\left(q_i^T k_j\right)} \tag{1}$$

3.5. Classification Layer

On top of the BERT, the encoder is a basic $SOFT_{Max}$ classifier, which does the classification. Let z represent the range of all learnable BERT4TC parameters. The classification layer uses the vector $L[CLS]$ to calculate likelihood distributions.

Let $P(v_i|L[CLS], z)$ be the likelihood distribution over the categorical labels $v_1, v_2, v_3, \ldots\ldots, v_k$ where k considered the target labels. The likelihood distribution is given as follows:

$$P(v_i|L[CLS], z) = \frac{exp(P(v_i|H[CLS], z))}{\sum_{i=1}^{k} exp(P(v_i|H[CLS], z))} \tag{2}$$

Let y be the ground truth of an input vector or sequence x. As the expected outcome, we choose the predicted label, which has a higher likelihood distribution as a result. The loss $L(x, z)$ is calculated by given formula:

$$L(x, z) = \begin{cases} -tlnP(v_x) - (1-t)ln(1 - P(v_x))\ if\ v = 2 \\ -lnP(v_x)\ if\ v > 2 \end{cases} \tag{3}$$

We specify the size of the training batches with the batch size parameter. To prevent over fitting, the dropout regularization approach is employed, and the value is constantly maintained at 0.3. For our model ARABERT4TWC, we have used an Adam optimizer with default parameters β_1 = 0.9 and β_2 = 0.999.

4. Experiments and Performance Evaluation

This section discusses in depth the dataset utilized, the experimental setup, and the ARABERT4TWC model performance on different datasets in terms of learning rate maximum token length.

4.1. Applied Dataset

In this section, we have done analysis of five different datasets used for evaluating our model performance. The datasets are SANAD, HARD, AJGT, ASTD, and ArsenTD-Lev.

4.1.1. SANAD

The SANAD dataset is a vast collection of Arabic news stories that may be utilized for various Arabic NLP tasks, including tweet or text classification. The content was gathered using a Python program customized for three famous media websites: Akhbarona, Al-Khaleej, and Al-Arabiya. Figure 3 depicts the distribution of data concerning each class. The dataset is divided into technology, sports, religion, finance, and medicine. Each category contains more than 6000 Arabic text files. Since all of these data were compiled from newspaper websites, the contents are in "Modern Standard Arabic", which implies that no accents are included. The datasets are labeled with a single label, and the records are combined into a single corpus termed SANAD. The SANAD dataset contains a large amount of text corpus per category that can produce better performance than the available small dataset that contains 83 to 893 text corpus per label [1,36].

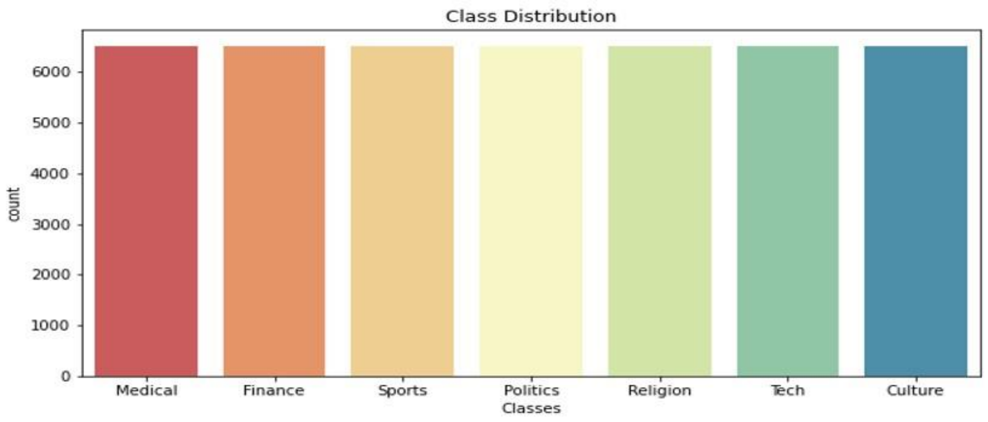

Figure 3. Class distribution of SANAD dataset.

4.1.2. HARD

The HARD dataset contains 93,700 hotel evaluations written in dialectal and modern standard Arabic. The reviews are split into two parts positive and negative. The negative reviews are assigned a class of 1 and 2, positive reviews are assigned a class of 5 and 6, and neural reviews are disregarded [37]. The positive reviews assigned to Classes 5 and 6 are mapped to Ground Truth Label 1, and negative reviews are mapped to ground Truth Label 0. The class distribution is shown in Figure 4.

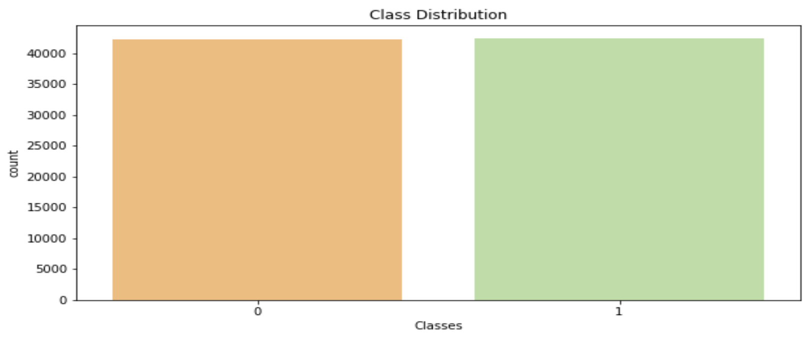

Figure 4. Class distribution of HARD dataset.

4.1.3. AJGT

AJGT, a Corpus of Jordanian Arabic tweets with sentiment analysis-relevant MSA annotations. There are 1800 tweets in it, divided into two categories: positive and negative, each class containing 900 tweets [38]. The class distribution is shown in Figure 5.

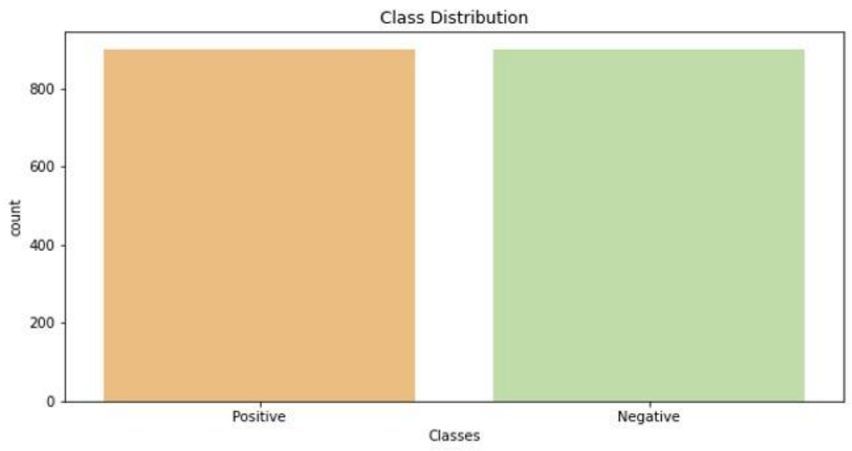

Figure 5. Class distribution of AJGT dataset.

4.1.4. ASTD

ASTD, a Twitter-based dataset for Arabic social sentiment analysis. There are roughly 10,000 tweets in it, and they are divided into four categories: subjectively favorable, objective, subjectively negative, and subjectively mixed [39]. The class distribution is shown in Figure 6.

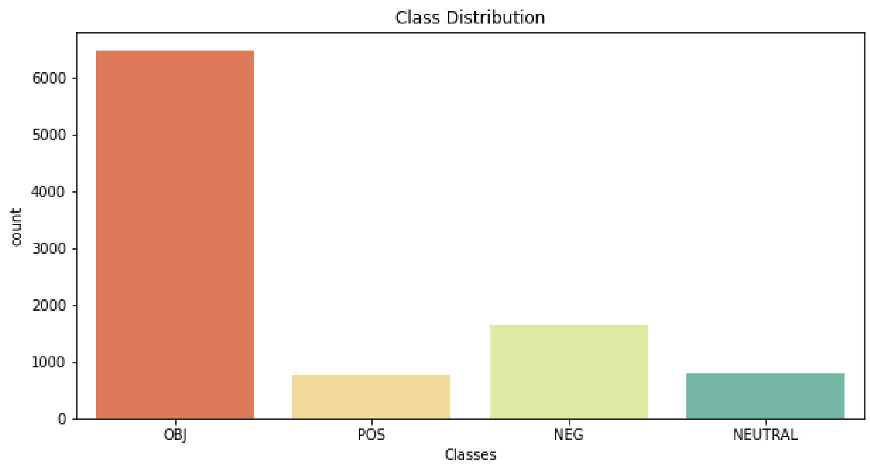

Figure 6. Class distribution of ASTD dataset.

4.1.5. ArsenTD–Lev

An Arabic Levantine Twitter corpus for sentiment analysis ArSenTD–LEV is based on examining 200 randomly selected tweets in the Levant region [40]. As a result, we compiled a database of 4000 tweets from the nations of the Levant. We grouped the Arabic tweets into five classes: neutral, positive, negative, very positive, and very negative. Figure 7 displays the distribution of classes.

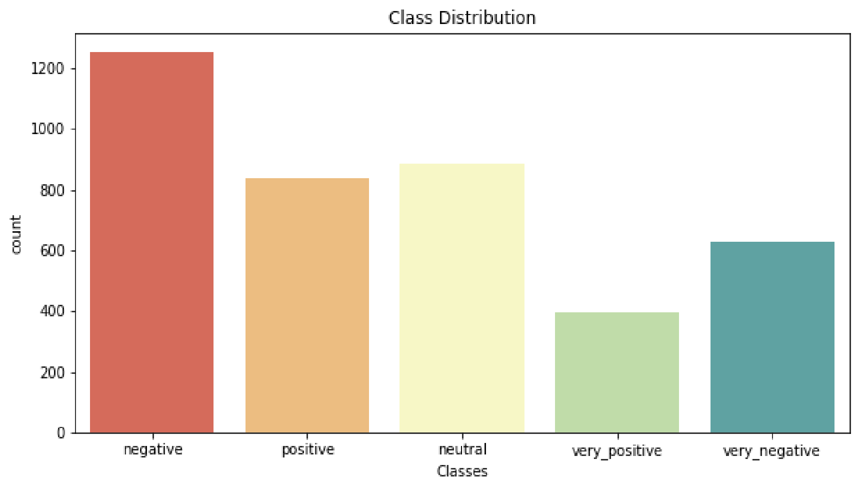

Figure 7. Class distribution of ArsenTD–Lev dataset.

Table 2 depicts the information on the dataset's statistics. The average and maximum length columns depict the average and maximum lengths of non-tokenized sequences found in training samples for each of the seven datasets. The columns labeled "Average Token Length" and "Maximum Token Length" represent the average and maximum tokenized lengths following word-piece segmentation, respectively.

Table 2. Statistical information about the dataset used.

Datasets	No. of Classes	Average Length	MAX Length	Average Token Length	MAX Token Length
SANAD	7	2.1 k	33 k	2790	40,497
HARD	2	140	3387	171	4213
AJGT	2	47	864	57	1085
ASTD	4	86	5895	109	7421
ArSenTD-LEV	5	157	218	144	207

4.2. Experimental Setup and Fine Tuning Analysis

We demonstrated the applicability of our model ARABERT4TWC using five datasets. For training our model, the dataset is divided in the ratio of 80:20 for training and testing, respectively. We have used the Pytorch–Lightning framework for fine-tuning the pre-trained BERT model. A word-piece tokenizer is used for tokenizing the entire text corpus using sub word tokenization algorithm. The training is carried out on a high-end system consisting of 27 GB of RAM and 16 GB of Tesla P100 GPU. The data used for training are fed into the model in mini-batches of Size 8. To enable parallel processing of data, we have split the data into parts by specifying the number of workers in the Pytorch Lightning Dataset module. Training in long-text datasets may include more than 512 token-size tokens, which must be reduced to comply with BERT's input length constraint. This is in contrast to the short-text dataset ArsenTD–LEV final. On all additional long-text datasets, we use the same sequence manipulation. The model also achieved higher accuracy and $F1$-score values when the learning rate was reduced to 1×10^{-5} from the default level of 2×10^{-5}. The subsequent experiments are conducted on long-text datasets with a learning rate of 1×10^{-5}. As presented in Table 3, we presented the accuracy and $F1$-score outcomes of ARABERT4TWC with various learning rates.

Table 3. Experimental results for the applied datasets.

Dataset	Learning Rate	Accuracy	*F*1-Score
SANAD	4×10^{-5}	0.9789	0.9790
	3×10^{-5}	0.9791	0.9799
	2×10^{-5}	0.9810	0.9810
	1×10^{-5}	0.9824	0.9824
HARD	4×10^{-5}	0.9678	0.9678
	3×10^{-5}	0.9697	0.9697
	2×10^{-5}	0.9746	0.9746
	1×10^{-5}	0.9749	0.9749
AJGT	4×10^{-5}	0.9210	0.9210
	3×10^{-5}	0.9270	0.9270
	2×10^{-5}	0.9350	0.9350
	1×10^{-5}	0.9405	0.9405
ASTD	4×10^{-5}	0.8525	0.8579
	3×10^{-5}	0.8585	0.8599
	2×10^{-5}	0.8600	0.8602
	1×10^{-5}	0.8610	0.8610
ArSenTD-LEV	4×10^{-5}	0.5225	0.5225
	3×10^{-5}	0.5270	0.5290
	2×10^{-5}	0.5341	0.5350
	1×10^{-5}	0.5370	0.5370

The evaluation matrices in these experiments are the accuracy and the $F1$-score, which are two metrics in machine learning to evaluate the model performance. As explained in Equation (4), the accuracy is a measure of the proportion of correct predictions of polarity terms t made by the model over the total number of predictions in the overall experiment.

$$Accuracy = \frac{TP_t + TN_t}{TP_t + TN_t + FP_t + FN_t} \tag{4}$$

where the TP_t stands for the number of accurate predictions of polarity terms, TN_t for accurate negative predictions, FP_t for accurate positive predictions, and FN_t for accurate negative predictions. The weighted average for both precision and recall are used to calculate the $F1$-score, where the emphasis on precision and recall can be adjusted by adjusting the weighting. It provides a single score that represents the model's overall performance, with higher values indicating better performance. The precision, recall, and $F1$-score are explained in the following equations:

$$Precision = \frac{TP_t}{TP_t + FP_t} \tag{5}$$

$$Recall = \frac{TP_t}{TP_t + FN_t} \tag{6}$$

$$F1 - score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \tag{7}$$

In these equations, TP_t explains the proportion of correctly anticipated positive instances, TN_t explains correctly anticipated negative instances, FP_t explains instances that were predicted as negative but turned out to be positive, and FN_t explains instances that were predicted as negative but turned out to be positive.

Based on the previous results, the optimization method that is applied for training the proposed model is called the Adam optimization method. The learning rate explained in Table 3 ranges from 4×10^{-5} to 1×10^{-5} and batch size = 8 with a batch size = 8. The required time to train the proposed ARABERT4TWC model can vary depending on the computational resources available and the corpus size being applied and used. The training time can vary from one-to-two hours. The evaluation metrics that are applied to monitor the performance of the model during training can also be reported using the accuracy and $F1$-score.

The experiments in Table 3 involved testing various learning ratios ranging from 4×10^{-5} to 1×10^{-5} on different datasets, which required significant time. The results inside this range stated that the suggested model scored better than other models examined on the same datasets. One way to make sure that the differences observed in performance between models are not due to chance is that the result achieved from running the experiments many times, which involves partitioning the dataset into non-overlapping subsets, using one subset for testing and the rest for training, and then repeating the process with different partitions. By applying this method, we can ensure that the model's performance is on new, unseen data and obtain a more reliable estimate of its performance.

To train and test the mode on unseen data, a large dataset is used. This large dataset helps in reducing the impact of random initialization. Therefore, the results of the experiments are reproducible and not dependent on the specific random seed used during training. As presented in Figure 8, the accuracy for the conducted datasets is explained where the learning rate from 4×10^{-5} (4e-05) to 1×10^{-5} (1E-05) is tested with each dataset. As explained, the SANAD dataset achieved the highest accuracy with 98.24% with the learning rate 1×10^{-5}, while the second-highest result achieved 98.10% with the learning rate 2×10^{-5}. Using HARD dataset, the highest accuracy recorded 97.49% with the learning rate 1×10^{-5}, while the learning rate of 2×10^{-5} recorded a close result with 97.46%. Using AJGT, the learning rate of 1×10^{-5} recorded 94.05% accuracy, while the learning rate of 2×10^{-5} recorded 93.50% accuracy. Using the ASTD dataset, the accuracy is decreased to

86.10% accuracy on the learning rate of 1×10^{-5}, while the lowest accuracy recorded 85.25% on the learning rate 4×10^{-5}. The ArSenTD–LEV dataset recorded the lowest results on all learning rates with 53.70% on the learning rate 1×10^{-5}. The $F1$-Score presented in Figure 9 showed almost identical results for the same datasets and learning rates.

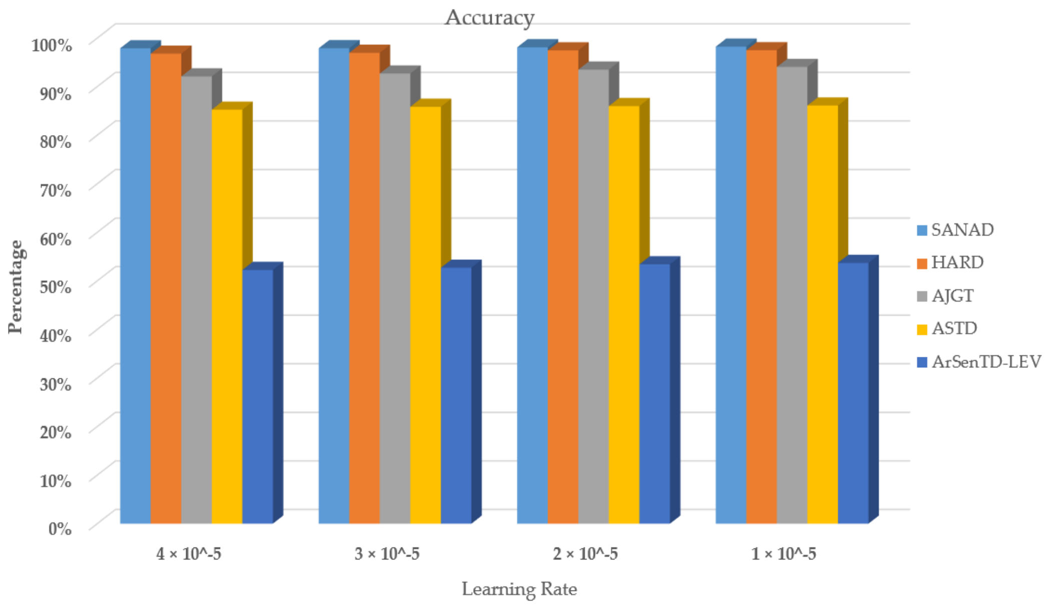

Figure 8. Accuracy results and learning rate for different datasets.

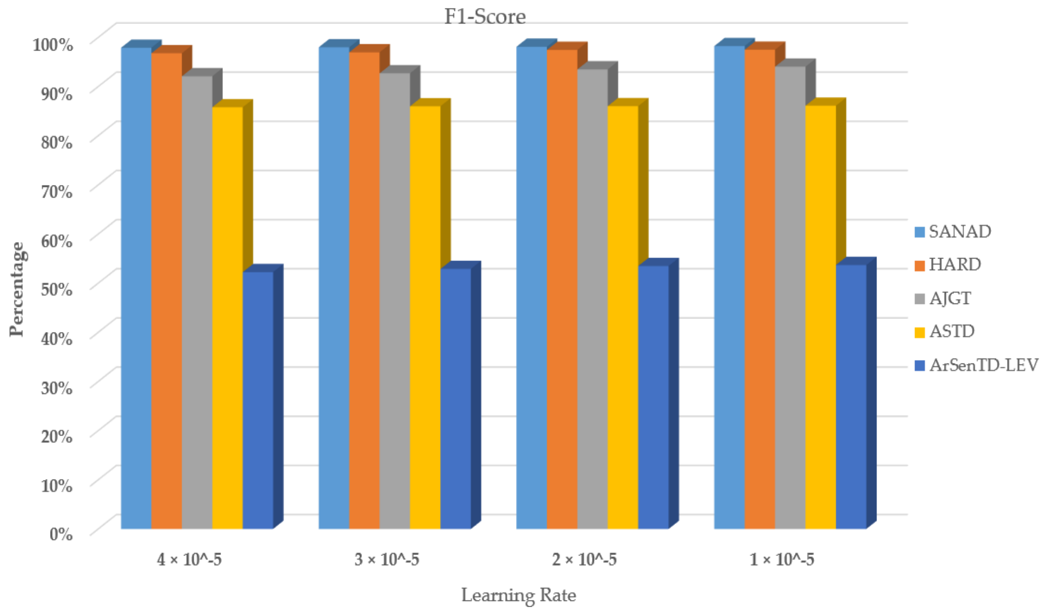

Figure 9. $F1$-score results and learning rate for different datasets.

4.3. Performance Evaluation

There are many differences between the proposed model and existing models. Table 4 highlights the major differences as follows:

1. Training data: The ARABERT4TWC was trained on a much larger dataset than the earlier models, which included text from Twitter, Wikipedia, and other sources. This allowed the model to learn from a wider range of data and capture more of the nuances of language use on social media.
2. Model architecture: ARABERT4TWC uses a modified version of the Transformer architecture, which includes additional layers and features designed to capture the

unique characteristics of Arabic text on social media. The earlier models used simpler architectures such as BERT and RoBERTa.

3. Task-specific fine-tuning: ARABERT4TWC was fine-tuned specifically for processing social media data tasks. In contrast, the earlier models were fine-tuned for general tasks such as sentiment analysis and named entity recognition. This task-specific fine-tuning allowed ARABERT4TWC to achieve better performance on social media-related tasks.
4. Model size: ARABERT4TWC is larger than the earlier models, with over 300-million parameters compared to the 140-million parameters of the largest version of AraBERTv3. This larger size allows ARABERT4TWC to capture more complex patterns in the data.

Table 4. Performance evaluation of proposed ARABERT4TWC.

Model / Criteria	hULMonA	ARABERT4TWC
Training data	Less training data	More training data
Model architecture	Transfer learning on large corpus of Arabic data	Additional layer to capture unique characteristics of Arabic text
Task-specific fine-tuning	Fine tuning of general domain target task of data.	Provide task specific tuning of data to achieve better performance on social media tasks.
Model size	140-million parameters	300-million parameters

Table 5 explores the performance comparison of the BERT model using five datasets with existing models in terms of accuracy. Our model ARABERT4TWC surpasses the previous recent models AraBERT, CAMcLBERT, and hULMona on three datasets, except ASTD and ArsenTD-LEV, which yield inferior results for ASTD and ArsenTD-LEV. When comparing AraBERT and ARABERT4TWC to hULMonA, we find that the former improves accuracy by 14.9% and the latter by 9%, respectively, on ASTD. As mentioned, the model outperformed models that used the SANAD, HARD, and AJGT datasets.

Table 5. Performance accuracy for the applied dataset with different models.

Models	SANAD	HARD	AJGT	ASTD	ArSenTD–LEV
AraBERTv0.1 [28]	-	0.9620	0.9310	0.9220	0.5356
AraBERTv1 [28]	-	0.9610	0.9380	0.9260	-
AraBERTv0.2-base [28]	-	-	-	-	0.5571
AraBERTv0.2-large [28]	-	-	-	-	0.5694
hULMonA [41]	-	0.9570	-	0.7710	0.5240
CAMeLBERT [42]	-	-	-	0.7690	-
ARABERT4TWC	0.9824	0.9749	0.9405	0.8610	0.5370

5. Conclusions and Future Scope

The proposed ARABERT4TWC model in this paper is applied for classifying the Arabic tweet and achieved high results for three datasets, which dominate other well-known tweet or text classification methods. The Arabic BERT architecture is built by fine-tuning the pre-trained BERT case model provided by the Hugging Face Transformer library and integrating custom layers and a classification layer on top of the encoder structure. Tokens were extracted from the full-text corpus using a word piece tokenizer based on a sub-word tokenization method. The proposed model can learn context-aware information and learn the context information bi-directionally. In order to complement the previously done work, this study first conducts experimental analysis across five datasets to investigate fine-tuning strategy in terms of tokenized vector and learning rate. The results of the experiments demonstrate that for three datasets, our model ARABERT4TWC not only outperformed deep learning models like CNN, LSTM, and BERT, but also conventional machine-learning techniques. Future research directions include expanding our understanding of effectively combining the domain and cross-domain pre-training with BERT to include domain and task-specific information. We would further apply knowledge distillation on the BERT model to make it suitable for edge devices.

Author Contributions: Data curation, M.A.; formal analysis, A.M.M. and M.E.; investigation, A.M.F., M.A and A.M.M.; supervision, M.A.; writing—original draft, A.M.F. and A.M.M.; writing—review and editing, A.M.M., M.E. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Deanship of Scientific Research—Jouf University, grant number DSR-2021-02-0358.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Furnished on request.

Acknowledgments: The authors acknowledge the Deanship of Scientific research at Jouf University. All authors have read and agreed to the published version of the manuscript.

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

References

1. Lulu, L.; Elnagar, A. Automatic Arabic Dialect Classification Using Deep Learning Models. *Procedia Comput. Sci.* **2018**, *142*, 262–269. [CrossRef]
2. Elnagar, A.; Einea, O. BRAD 1.0: Book reviews in Arabic Dataset. In Proceedings of the IEEE International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016. [CrossRef]
3. Onan, A. An Ensemble Scheme based on Language Function Analysis and Feature Engineering for Text Genre Classification. *J. Inf. Sci.* **2018**, *44*, 28–47. [CrossRef]
4. Li, Y.; Nie, X.; Huang, R. Web Spam Classification Method Based on Deep Belief Networks. *Expert Syst. Appl.* **2018**, *96*, 261–270. [CrossRef]
5. Deng, Z.; Luo, K.; Yu, H. A Study of Supervised Term Weighting Scheme for Sentiment Analysis. *Expert Syst. Appl.* **2014**, *41*, 3506–3513. [CrossRef]
6. Sutar, M.; Bagban, T. Applying Sentiment Analysis to Predict Rating and Classification of Text Review. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 173–178. [CrossRef]
7. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
8. Lai, S.; Xu, L.; Liu, K.; Zhao, J. Recurrent Convolutional Neural Networks for Text Classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015. [CrossRef]
9. Xiao, Y.; Cho, K. Efficient Character-level Document Classification by Combining Convolution and Recurrent Layers. *arXiv* **2016**, arXiv:1602.00367. [CrossRef]
10. Gao, S.; Ramanathan, A.; Tourassi, G. Hierarchical Convolutional Attention Networks for Text Classification. In Proceedings of the Third Workshop on Representation Learning for NLP, Melbourne, Australia, 20 July 2018. [CrossRef]
11. El-khair, I. 1.5 billion words Arabic Corpus. *arXiv* **2016**, arXiv:1611.04033. [CrossRef]
12. Zeroual, I.; Goldhahn, D.; Eckart, T.; Lakhouaja, A. OSIAN: Open Source International Arabic News Corpus—Preparation and Integration into the CLARIN-infrastructure. In Proceedings of the Fourth Arabic Natural Language Processing Workshop, Florence, Italy, 1 August 2019. [CrossRef]
13. Zhu, Y.; Kiros, R.; Zamel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; Fidler, S. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. *arXiv* **2015**, arXiv:1506.06724. [CrossRef]
14. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J. Transformers: State-of-the-Art Natural Language Processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020. [CrossRef]
15. Transformers. Available online: https://huggingface.co/docs/transformers/index (accessed on 20 December 2022).
16. Saad, M. The Impact of Text Preprocessing and Term Weighting on Arabic Text Classification. Master's Thesis, The Islamic University—Gaza, Gaza, Palestine, September 2010. [CrossRef]
17. Umer, M.; Khiyal, M. Classification of Textual Documents Using Learning Vector Quantization. *Inf. Technol. J.* **2006**, *6*, 154–159. [CrossRef]

18. El Mahdaouy, A.; Gaussier, E.; El Alaoui, S. Arabic Text Classification Based on Word and Document Embedding. In Proceedings of the International Conference on Advanced Intelligent Systems and Informatics, Cairo, Egypt, 17–18 August 2016. [CrossRef]
19. Baroni, M.; Dinu, G.; Kruszewski, G. Don't Count, Predict! A Systematic Comparison of Context-Counting Vs. Context-Predicting Semantic Vectors. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014. [CrossRef]
20. Vincent, N.; Ogier, J. Shall Deep Learning be the Mandatory Future of Document Analysis Problems? *Pattern Recognit.* **2019**, *86*, 281–289. [CrossRef]
21. Kim, Y. Convolutional Neural Networks for Sentence Classification. *arXiv* **2014**, arXiv:1408.5882. [CrossRef]
22. Johnson, R.; Zhang, T. Deep Pyramid Convolutional Neural Networks for Text Categorization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017. [CrossRef]
23. Le, H.; Cerisara, C.; Denis, A. Do Convolutional Networks Need to be deep for Text Classification. *arXiv* **2017**, arXiv:1707.04108.
24. Al-Smadi, M.; Talafha, B.; Al-Ayyoub, M.; Jararweh, Y. Using Long Short-Term Memory Deep Neural Networks for Aspect-Based Sentiment Analysis of Arabic Reviews. *Int. J. Mach. Learn. Cybersecur.* **2019**, *10*, 2163–2175. [CrossRef]
25. Samy, A.; El-Beltagy, S.; Hassanien, E. A Context Integrated Model for Multi-label Emotion Detection. *Procedia Comput. Sci.* **2018**, *142*, 61–71. [CrossRef]
26. Liu, M.; Chen, L.; Du, X.; Jin, L.; Shang, M. Activated Gradients for Deep Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–13. [CrossRef]
27. Chiorrini, A.; Diamantini, C.; Mircoli, A.; Potena, D. Emotion and Sentiment Analysis of Tweets using BERT. In Proceedings of the EDBT/ICDT Workshops, Edinburgh, UK, 29 March–1 April 2022.
28. Antoun, W.; Baly, F.; Hajj, H. AraBERT: Transformer-based Model for Arabic Language Understanding. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, Marseille, France, 12 May 2020.
29. Kwaik, K.; Saad, M.; Chatzikyriakidis, S.; Dobnik, S.; Johansson, R. An Arabic Tweets Sentiment Analysis Dataset (ATSAD) Using Distant Supervision and Self-Training. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, Marseille, France, 12 May 2020.
30. Chennafi, M.; Bedlaoui, H.; Bedlaoui, A.; Al-qaness, M. Arabic Aspect-Based Sentiment Classification Using Seq2Seq Dialect Normalization and Transformers. *Knowledge* **2022**, *2*, 388–401. [CrossRef]
31. Iqbal, M.; Latha, K. A Parallel Approach for Sentiment Analysis on Social Networks Using Spark. *Intell. Autom. Soft Comput.* **2023**, *35*, 1831–1842. [CrossRef]
32. Hadwan, M.; Al-Hagery, M.; Al-Sarem, M.; Saeed, F. Arabic Sentiment Analysis of Users' Opinions of Governmental Mobile Applications. *Comput. Mater. Contin.* **2022**, *72*, 4675–4689. [CrossRef]
33. Musleh, D.; Alkhales, T.; Almakki, R.; Alnajim, S.; Almarshad, S.; Alhasaniah, R.; Aljameel, S.; Almuqhim, A. Twitter Arabic Sentiment Analysis to Detect Depression Using Machine Learning. *Comput. Mater. Contin.* **2022**, *71*, 3463–3477. [CrossRef]
34. Albahli, S.; Algsham, A.; Aeraj, S.; Alsaeed, M.; Alrashed, M.; Rauf, H.; Arif, M.; Mohammed, M. COVID-19 Public Sentiment Insights: A Text Mining Approach to the Gulf Countries. *Comput. Mater. Contin.* **2021**, *67*, 1613–1627. [CrossRef]
35. Arabic Tweet Classification. Available online: https://github.com/abdul756/Arabic_tweet_classification (accessed on 7 March 2023).
36. Khorsheed, M.; Al-Thubaity, A. Comparative Evaluation of Text Classification Techniques using a Large Diverse Arabic Dataset. *Lang. Resour. Eval.* **2013**, *47*, 513–538. [CrossRef]
37. Elnagar, A.; Khalifa, Y.; Einea, A. Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Applications. *Intell. Nat. Lang. Process. Trends Appl.* **2018**, *740*, 35–52. [CrossRef]
38. Alomari, K.; ElSherif, H.; Shaalan, K. Arabic Tweets Sentimental Analysis Using Machine Learning. *Adv. Artif. Intell. Theory Pract.* **2017**, *10350*, 602–610. [CrossRef]
39. Nabil, M.; Aly, M.; Atiya, A. ASTD: Arabic Sentiment Tweets Dataset. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015. [CrossRef]
40. Baly, R.; Khaddaj, A.; Hajj, H.; El-Hajj, W.; Shaban, K. ArSentD- LEV: A Multi-Topic Corpus for Target-based Sentiment Analysis in Arabic Levantine Tweets. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation, Miyazaki, Japan, 7–12 May 2018. [CrossRef]
41. ElJundi, O.; Antoun, W.; El Droubi, N.; Hajj, H.; El-Hajj, W.; Shaban, K. hULMonA: The Universal Language Model in Arabic. In Proceedings of the Fourth Arabic Natural Language Processing Workshop, Florence, Italy, 28 July–2 August 2019. [CrossRef]
42. Inoue, G.; Alhafni, B.; Baimukan, N.; Bouamor, H.; Habash, N. The Interplay of Variant, Size, and Task Type in Arabic Pre-trained Language Models. In Proceedings of the 7th Workshop on Arabic Natural Language Processing, Abu Dhabi, United Arab Emirates, 8 December 2022. [CrossRef]