electronics

Article

A Comparative Analysis of Word Embedding and Deep Learning for Arabic Sentiment Classification

Sahar F. Sabbeh [1,2,*] and Heba A. Fasihuddin [1]

[1] College of Computer Science and Engineering, University of Jeddah, Jeddah 21493, Saudi Arabia
[2] Faculty of Computers and Artificial Intelligence, Benha University, Benha 13518, Egypt
* Correspondence: sfsabbeh@uj.edu.sa

Abstract: Sentiment analysis on social media platforms (i.e., Twitter or Facebook) has become an important tool to learn about users' opinions and preferences. However, the accuracy of sentiment analysis is disrupted by the challenges of natural language processing (NLP). Recently, deep learning models have proved superior performance over statistical- and lexical-based approaches in NLP-related tasks. Word embedding is an important layer of deep learning models to generate input features. Many word embedding models have been presented for text representation of both classic and context-based word embeddings. In this paper, we present a comparative analysis to evaluate both classic and contextualized word embeddings for sentiment analysis. The four most frequently used word embedding techniques were used in their trained and pre-trained versions. The selected embedding represents classical and contextualized techniques. Classical word embedding includes algorithms such as GloVe, Word2vec, and FastText. By contrast, ARBERT is used as a contextualized embedding model. Since word embedding is more typically employed as the input layer in deep networks, we used deep learning architectures BiLSTM and CNN for sentiment classification. To achieve these goals, the experiments were applied to a series of benchmark datasets: HARD, Khooli, AJGT, ArSAS, and ASTD. Finally, a comparative analysis was conducted on the results obtained for the experimented models. Our outcomes indicate that, generally, generated embedding by one technique achieves higher performance than its pretrained version for the same technique by around 0.28 to 1.8% accuracy, 0.33 to 2.17% precision, and 0.44 to 2% recall. Moreover, the contextualized transformer-based embedding model BERT achieved the highest performance in its pretrained and trained versions. Additionally, the results indicate that BiLSTM outperforms CNN by approximately 2% in 3 datasets, HARD, Khooli, and ArSAS, while CNN achieved around 2% higher performance in the smaller datasets, AJGT and ASTD.

Keywords: Arabic sentiment analysis; deep learning; BiLSTM; CNN word embedding

Citation: Sabbeh, S.F.; Fasihuddin, H.A. A Comparative Analysis of Word Embedding and Deep Learning for Arabic Sentiment Classification. *Electronics* **2023**, *12*, 1425. https://doi.org/10.3390/electronics12061425

Academic Editor: Maria Evelina Fantacci

Received: 6 February 2023
Revised: 11 March 2023
Accepted: 14 March 2023
Published: 16 March 2023

1. Introduction

Word embedding is a prominent text-processing technique that has thrived in recent years. The technique is defined as models of words that provide a meaningful portrayal of words in a specific framework [1]. It refers to the procedure of representing words as vectors in a predefined vector space. Each word is classified by a vector value, in which all values develop and establish a neural network. The key to any word embedding method is the concept of using a dense, distributed representation for each word [1]. Several distinct studies were performed to evolve and optimize the word embedding techniques and their applications in assorted contexts. It has been extensively applied to addressing natural language processing (NLP) difficulties [1,2]. There are two primary strategies utilized for word embedding, specifically context-independent (classic) and context-dependent (contextualized) [3].

In classic embeddings, the word representation is characterized by being unique for each term and without considering the context within which it appears. This results in words being identified without contemplation of the context, which results in reduced accuracy: it is based on a language model and related general text corpora [3]. Examples of classic embedding models include Word2vec [4], GloVe [5], and FastText [6].

Word2vec uses two-layer neural network training to predict and construct semantic contexts of words [4]. It applies either continuous bag-of-words (CBOW), which tutors a model to predict a word given its context, while Skip-Gram (SG) predicts the context when provided with the word.

GloVe (global vectors for word representation) [5] is solely based on matrix factorization techniques on the "word-context matrix" [5]. Typically, the corpus can be scanned in the following manner: for every term, identify context terms within the area defined by a window size before the term and a window size after the term, resulting in less importance being assigned to words further away from the initial term. This feature is advantageous for identifying language features globally, by analyzing word frequency across corpora.

FastText: This model resembles the Skip-Gram version of Word2vec; however, each word is processed as being comprised of n-grams instead of the whole word [6]. It is very similar to word n-grams, in that the window size is at the character level. In other words, FastText operates at a character level but Word2vec operates at a word level. This model not only recognizes and learns the entire n-character sequence of the word but also models the sub-words of the word. It uses the sub-word information explicitly for embedding, ensuring that rarely used vocabulary can still be accurately estimated. This creates an enhanced morphological understanding of the language, combined with better representations based on word tense, enabling the algorithm to handle unfamiliar words. FastText is one of the methods developed to remedy the issue of embedding rarely used words that can sometimes be poorly estimated [7].

The alternative strategy is identified as context dependent (contextualized) which learns embeddings for the word within its context. Each word is assigned a representation based on its context, for example, the word 'right' has alternate meanings if it is used in a directional context or a legal one. This feature has attracted the interest of contemporary research which has evolved into two categories: a language model-based recurrent neural network (RNN) and a transformer-based model i.e., BERT [7] and its variation ALBERT [8], which proved to produce efficient word embedding representations [3].

Contextualized embeddings are well known for detecting words' semantics in context, which contrasts with classic embedding models. BERT (bidirectional encoder representations from transformer) is a transformer-based text representation archetype that employs a masked language model to predict randomly hidden words in a sequence. This is followed by a next-sentence-prediction task for learning the associations between sentences. It employs masked language modeling to conceal some of the words and use their positional information to infer them. The transformer-based model has the advantage of achieving more effective modeling of long-term dependencies among tokens in a temporal sequence and increasingly efficient training of the model by eliminating the sequential dependency on previous tokens [7]. Contextualized embedding can be employed for downstream tasks in two ways: firstly, as a fixed feature extractor which indicates that it is not trainable with downstream parameters, resulting in a static, pre-trained input sequence feature and, afterwards, fine tuning in which pre-trained embeddings are adapted using a particular dataset to provide input features. Studies have indicated that, when operating the fine-tuning process, the transformer-based model, used to represent a sequence, attained improved performance for the various NLP tasks [7–9]. The fixed feature extractor offers an advantage over the downstream task as it conserves time and memory because it does not necessitate parameter training [3].

Generally, word embeddings can be obtained either via custom-trained embeddings or via pretrained embeddings. In custom-trained embeddings, vectors are generated using the training dataset so that the training corpus' semantics can be better represented. On the other hand, pretrained word embeddings generate the representation vectors as they are trained on large datasets, saved, and then used for solving other tasks. This is why pretrained word embeddings are a form of transfer learning [1–3].

Our work was motivated by the fact that there was no unified study to report the most suitable embedding technique for Arabic sentiment classification. The bulk of the works either focused only on Twitter; data which usually contain short reviews; targeted a certain domain; or concentrated on one format of the Arabic language. Thus, there is a need to benchmark and evaluate different embeddings for Arabic sentiment analysis without language format, length, or domain dependency. To address the described problems, we can summarize our contribution as follows:

- Present a comprehensive evaluation of the most widely used embedding (classical and contextualized) models for the Arabic language.
- Assess their performances among different benchmark datasets that represent different review lengths, Arabic language formats, and different domains. This incorporation of datasets from diverse domains can surmount the challenge of a potential lack of diversity in the obtainable datasets.
- The study concentrates on deep learning models, as they were adjudged to be of increased effectiveness and surpassed other machine learning models across a range of tasks [10–13].

The current literature offers limited comparative studies regarding Arabic language word embedding, and this paper endeavors to be one of the first to bridge this knowledge gap. Our main objective is not to try to outperform the complex state-of-the-art models or compete with sophisticated techniques. Instead, we try to empirically survey word embeddings and deep learning network-based sentiment analysis. We try to provide insight into various word embedding models for Arabic sentiment classification using CNN or BiLSTM. This can help researchers to select suitable models for sentiment classification tasks. Moreover, our experiment can be applied to other deep neural network architectures and language tasks.

This paper is organized as follows: firstly, a background of the assessed word embedding models is presented in Section 2, followed by an overview of the related work in Section 3. Section 4 details the methodology of this study, followed by the experiment setup in Section 5 with a description of the used datasets, models architectures, parameters, and evaluation metrics. Finally, the results and discussion are presented and the paper is concluded in Section 6.

2. Related Work

As mentioned previously, advancing the aptitudes of word embedding models has drawn the attention of researchers in various NLP applications since it acknowledges both the syntax and semantics of each word. Therefore, the semantic resemblance between words, phrases, and sentences can be determined to perfect the performance of the applications.

According to a study conducted by the Central Intelligence Agency (CIA), the Arabic language was ranked the fourth most spoken first language after Chinese, Spanish, and English. Arabic native speakers constitute almost 5% of the worlds' population. The Arabic language is the standard across the Middle East, North and the Horn of Africa, and is one of the six official languages of the United Nations (https://www.cia.gov/the-world-factbook/countries/world/#people-and-society (accessed on 1 February 2023)).

Due to the significant use of the Arabic language in social media networks, the demand for Arabic sentiment analysis has increased rapidly. Sentiment analysis in Arabic can be beneficial for both businesses and governments in the Arab world. One of the challenges that face Arabic sentiment analysis is to understand the varieties of different languages. The Arabic language has various formats; the formal Arabic language used in literature, prints, mass media, or formal education is called Modern Standard Arabic (MSA). The spoken Arabic language includes the varieties or dialects of Arabic. The dialects vary considerably from region to region. According to the Library of Congress (https://iso639

-3.sil.org/code/ara (accessed on 13 February 2023)), there are approximately 30 different dialects of Arabic. This is why sentiment analysis for the Arabic language has drawn the attention of researchers and has resulted in many studies designed to provide automated sentiment analysis for online Arabic content.

Word embedding permits the automatic extraction of language features from Arabic tweets, product reviews, service reviews, and news articles. Automatic feature extraction is fundamental since the process of manual feature extraction, specifically from Arabic texts, is prolonged and arduous due to its complicated construction and grammar. Embedding procedures were explored for sentiment analysis to scrutinize and label reviews, opinions, and attitudes to enhance decision making in different domains [14]. Sentiment analysis tries to ascertain the text's polarity and categorize it as positive, neutral, negative, or mixed [14]. Various studies of Arabic sentiment classification using conventional machine-learning models have been performed [15,16].

The effect of word embedding was studied for Arabic sentiment analysis in different inquiries and contexts, such as [17,18], which studied the use of Doc2vec document embedding to construct paragraph vectors within machine learning classifiers. Their results reported that Doc2vec resulted in highly effective results with classifiers. Additionally, the work in [19] constructed Word2vec embeddings from an Arabic corpus obtained from ten newspapers, from different Arabic countries, applied different machine learning algorithms and convolutional neural networks, and reported an increased accuracy of sentiment classification. In [20], word vectors were generated from a large Arabic corpus. The generated vectors are used to train classifiers to detect sentiment/subjectivity in both standard Arabic (SA) and dialectal Arabic (DA). Their results indicated that the use of word embedding resulted in enhanced levels of accuracy.

The works in [21,22] explored the utilization of contextualized word embedding for sentiment analysis of cross-domain and cross-dialect data. Their results reveal that this method enhances the performance of BERT [7].

Deep learning models have recently been used for NLP tasks due to their proven efficiency and high levels of performance [23]. The works in [19,24] used a convolutional neural network (CNN). Other studies, presented in [25,26], proposed archetypes and architectures that attained reasonable scores for accuracy. Furthermore, the work presented in [27] proposed a method that combines CNN and bidirectional long short-term memory (BiLSTM) to enhance sentiment classification and better signify text features. The study achieved a satisfyingly accurate result by using different, contemporary models. In addition, a hybrid CNN and long short-term memory (LSTM) model was proposed for Arabic sentiment analysis [28]. Another study conducted a comparison concerning the performances of the recurrent neural networks (RNN) and support vector machine (SVM) models in the sentiment analysis of hotel reviews in Arabic [29]. SVM demonstrated improved levels of accuracy, while the RNN demonstrated an improved processing time. A summary of all the related work is provided in Table 1.

Table 1. Summarization of related work.

Study	Experimental Model	Shortcomings	Main Findings
[17]	• Doc2Vec • Logistic regression, decision tree, support vector machine, and naive Bayes.	• Only traditional machine learning technique. • Only one embedding technique was evaluated. • Only one dataset was used.	• Doc2Vec achieves high effectiveness with both large dimension and negative samples. • Small context windows are better to attain high efficiency.
[18]	• CBOW, SKIP-G, and GloVe	• Only Glove vectors were generated using skip-grams and continuous bag of words.	Achieved human-like translation results with a correlation between 0.75 and 0.79.

Table 1. *Cont.*

Study	Experimental Model	Shortcomings	Main Findings
[20]	• AraBERT	• Only contextualized embedings were used; no other evaluation of other embedding models.	AraBERT achieved state-of-the-art performance on most tested Arabic NLP tasks.
[21]	Domain adversarial neural network (DANN)	• Only contextualized embedings were evaluated.	Success of the proposed domain adaptation method.
[22]	Compared various classifiers in addition to deep learning models (LSTM and BiLSTM)	• Only AraBERT is used with classical machine learning and LSTM and BiSTM.	The model outperforms the baseline on both large and small datasets.
[24]	CNN	• One dataset used CNN with pretrained word embedding for the classification.	The proposed scheme outperforms the existing methods on four out of five balanced and unbalanced datasets.
[25]	Bag-of-word, deep belief networks, deep auto encoders, and recursive auto encoder	• Only used BOW model.	High improvement with the recursive auto encoder.
[26]	AraNet based on BERT	• Arabic-based version of contextualized embedding; no evaluation of other embedding models.	High performance in detecting dialect, gender, emotion, irony, and sentiment from social media posts.
[27]	CNN + BiLSTM	• Investigated the impact of combining CNN + BiLSTM with no evaluation of word embedding.	Demonstrated advanced results by combining CNN and BiLSTM to prepare Arabic text and better represent associated text features.
[28]	CNN + LSTM	• Evaluated only CNN + LSTM combined with no evaluation of word embedding.	Promising results have been attained when combining LSTMs when compared with other models.
[29]	• Recurrent neural network (RNN) • Support vector machine (SVM)	• Explored the impact of SVM and RNN in aspect-based sentiment analysis with no evaluation of word embedding.	Results showed that the SVM approach outperforms the RNN approach in the research-investigated tasks, whereas the RNN execution time is swifter.

In this study, various models (classic and contextualized) are assessed using five cross-domain datasets. Details about the adopted methodology and the experiment setup are provided in the following sections.

3. Methodology

To systematically study word embedding for the sentiment classification problem, the methodology includes that first the textual reviews are separated by the tokenizer, and then the vector space representation of every word is generated and used to construct an embedding layer as an input for the deep neural networks. For the experiments conducted by this study, several advanced, pre-trained word embeddings were used, as well as generated representations produced by training the embedding algorithms on the used datasets. To investigate the impact of word embedding techniques on discrete datasets, datasets with different lengths and characteristics were selected. The length of the reviews varied between short sentences and long paragraphs. Additionally, the textual reviews target different domains including hotel reviews, sports, politics, and the arts, which enabled this study to analyze the effect that word embedding has on different domains. The selected word embedding belongs to both classic (context-independent) and contextualized text representation algorithms. Each one was operated in the pre-trained and trained versions of the model. The selected context-independent models include Word2vec, GloVe, and FastText. ARABERT was used as an example of contextualized word embedding.

For the sake of this experiment, ARBERT was utilized as a large-scale pre-trained masked language model focused on Modern Standard Arabic (MSA). ARBERT is trained using the BERT-base architecture with twelve attention layers, each of which has twelve attention heads and 768 hidden dimensions, a vocabulary of one hundred thousand Word Pieces, resulting in approximately 163 million parameters. Additionally, BERT-based training was performed on the dataset, based on the implementation found in [30].

The embedding layer employs a word embedding algorithm to generate word representations for the encoder. These representation vectors are generated using the pre-trained word embedding, or by training the models on the dataset. This experiment used BiLSTM and CNN to represent the embedded vectors into a summary of single sequence representation.

4. Experimental Setup

This stage details the selection of the datasets, the configuration of deep learning classifiers, and choosing of the pre-trained embeddings. Experimental details are presented below.

4.1. Datasets

This study chose five popular benchmarking sentiment classification datasets. These datasets vary in size, the number of classes, types (single label or multilabel), domains (hotels, politics, etc.), and they represent different dialects of Arabic.

1. **HARD** [31]: The Hotel Arabic-Reviews Dataset (HARD) contains nearly 106,000 hotel reviews, written in Arabic, collected from the Booking.com website during June and July of 2016. The reviews are expressed in Modern Standard Arabic as well as dialectal Arabic.
2. **Khooli** [3] (https://www.kaggle.com/abedkhooli/ar-reviews-100k/data (accessed on 11 March 2022)): The Arabic 100K Reviews dataset is a collection of reviews, collected from different services, concerning hotels, movies, books, and products. It is a three-class balanced set.
3. **AJGT (Arabic Jordanian General Tweets)** [32]: Created by Alomari in 2017, the Arabic Jordanian General Tweets dataset consists of 1800 tweets categorized as positive and negative.
4. **ArSAS** [33]: The Arabic Speech-Act and Sentiment Corpus of Tweets dataset is a sizable set of nearly 20,000 Arabic tweets, concerning multiple topics, collected, prepared, and annotated with four classes of sentiment. ArSAS tweets, belonging to one of twenty topics, consist of three main archetypes: long standing (topics about articles that are commonly discussed over a long period), entity (topics about celebrities or organizations), and event (topics about an important incident).
5. **ASTD** (Arabic Sentiment Tweets Dataset) [34]: The Arabic Sentiment Tweets Dataset is a set of Arabic tweets containing around 10,000 entries. They can be identified as objective, subjective negative, subjective positive, and subjective mixed. These tweets were labelled as 6691 objective, 1684 subjective negative, 832 subjective mixed, and 799 subjective positives. For the sake of this study, only positive, negative, and mixed reviews were used. Table 2 shows a summary of all used datasets.

Table 2. Summarization of datasets.

Dataset	Count	Class Labels	Vocab Size	AVG Token Length	Max Text Length	Text Type
HARD	**~106K reviews**	Positive: 52,849 Negative: 52,849	130,654	200	507	MSA Egyptian and gulf dialects
Khooli	**~67K reviews**	Negative:33,333 Positive: 33,333 Mixed: 33,333	238,801	150	1445	MSA and Egyptian dialects
AJGT	**1800 reviews**	Negative 900 Positive 900	6806	40	129	Modern Standard Arabic (MSA) or Jordanian dialect
ArSAS	**~20K reviews**	Negative 7384 Neutral 6894 Positive 4400 Mixed 1219	62,861	50	288	MSA, Egyptian and Gulf dialects
ASTD	**3315 reviews**	NEG: 1684 Mixed: 832 POS: 799	19,235	15	28	Egyptian dialects

4.2. Model Building and Training

Based on an investigation of the literature [22,24,27,35–40], Bi-LSTM and CNN have been the most frequently employed deep learning models for text classification tasks, particularly in sentiment analysis. These models were described as achieving preeminent performances.

Before building the selected models, hyperparameters must be selected and tuned. The most important parameters to configure when building models are:

1. **Batch size** represents the number of instances to be considered before the model's parameters are updated.
2. **The number of epochs** is the number of times the algorithm will work for the training dataset.
3. **Optimizer** is used to bridge the gap between updating model parameters and the loss function.
4. **Dropout** enhances the model's ability to generalize and reduces the probability of overfitting.
5. **Classifier** is the final layer that projects all the input into predicted classes. Thus, choosing this layer greatly impacts the overall performance. *Sigmoid and Softmax* are the options that are considered to be the most suitable when labels are independently distributed.

To determine the hyperparameters, a grid search was used to evaluate the models using the ASTD dataset. Different options were applied and those that demonstrated the optimal performances were selected (as shown in Table 3).

Table 3. The used hyperparameters.

Batch Size	Optimizer	Epochs	Dropout	Classifier
128	Adam	50	0.2	Softmax

1. Batches of size 32, 64, and 128 were tested, with 128 achieving the highest level of performance.
2. Adam and Nadam optimizers were assessed, resulting in Adam attaining the best performance with a learning rate of 0.001.
3. An evaluation of 25, 40, and 50 training epochs was conducted, and the performances were optimum at 50 epochs.

4. Dropout values of 0.1, 0.2, and 0.5 were used and a dropout rate of 0.2 was used to constrain the weights of the layers.
5. The output logits across all labels were generated, and the probability of a specific sample belonging to one label was optimal when estimated by the *Softmax* function.

Experiments were performed on the Google Co-LAB platform with GPU 1x Tesla K80, 12 Gigabyte of RAM, and 2496 CUDA cores. The models were developed using the Python Torch library. The architectures of each model are given below.

BiLSTM (**Bidirectional LSTM**) is a sequence processing model that consists of two layers that work, in parallel, to propagate data in both forward and backward directions. BiLSTM considers past and future context information to learn long-term dependencies and capture the contextual features of the text; thus, it demonstrates excellent performance when used for text classification, owing to its capability to comprehend the context-dependent sequences [16]. In this research, the simple bidirectional LSTM (BiLSTM) with an embedding layer was used as an input, and a dropout of 0.2 was applied to avoid overfitting.

In this study, an adapted, uncomplicated version of the CNN model was employed as the encoder. The architecture employed during our experiments included an embedding layer as an input layer using window sizes (1,2,3,5) with the number of filters equal to 10. The model applies conventional and max-pool layers for each window size. For each window size, the model keeps only the highest value of each feature vector (detected features for that window size) and then combines all these matrices to form a final matrix, of all detected features, to form the final feature matrix. Finally, a flattening layer is added to flatten the feature matrix into a vector.

The dataset was split into eighty percent for training, ten percent for validation, and ten percent for testing.

4.3. Word Embedding

Word embedding techniques generate text representation, which is used to produce the embedding layer as the first layer of the deep neural model. Text pre-processing techniques, including the removal of stop words, and non-Unicode and non-Arabic text, are utilized to reduce the vocabulary size and consequently reduce the embedding layer size. The used word embedding includes:

Baseline: For the baseline, the study defined an initialized embedding layer with a dimension of three hundred to encode word inputs. The baseline is learned while training the network from an outset of zero, not pretrained. The embedding layer stores a lookup table to map the words represented by numeric indexes to their dense vector representations. Hence, its impact on performance should be considered.

- **Word2vec:** This research employed the AraVec [4], a pre-trained embedding on a Twitter dataset using CBOW and three hundred embedding dimensions.
- **GloVe:** Currently there is only one, pre-trained GloVe embedding available online for use in the Arabic language (https://archive.org/details/arabic_corpus (accessed on 16 February 2022)). The available pre-trained GloVe vectors consist of 256 dimensions. Pretrained GloVe embedding is generated based on Wikipedia.
- **FastText:** Pretrained versions of FastText are available for different languages (https://fasttext.cc/docs/en/crawl-vectors.html (accessed on 25 February 2022)). The experimental datasets exist only in Arabic; therefore, the experiment used the available 300-dimension Arabic pre-trained vectors of FastText.
- **AraBERT:** These experiments employed a pre-trained model called AraBERT as reported in [20]. AraBERT is BERT-based [7] and trained on a 3B words corpus of Arabic text. AraBERT includes 12 attention heads, 12 hidden layers with 768 hidden sizes, and a vocabulary size of 64,000.
- **Custom word embedding:** While using the pre-trained word embedding, it may be the case that some words in the training dataset may not be available in the pre-trained vectors. Those missing vocabularies are replaced with a zero vector, which means that they will not be used. These vocabularies are vital for accurate classification. To solve

this issue, custom word embedding was employed, using the training data, so that the semantic relationship of the used training corpus can be better represented. Hence, for the sake of this study's experiments, custom embeddings are considered to study their impact on performance.

4.4. Performance Evaluation Metrics

The use of an evaluation classification model is important to measure to which degree predictions can be considered as 'good'. All performance evaluation metrics can be defined by using some, or all, of the parameters in Table 4:

Table 4. Performance evaluation parameters.

		Predicted Class	
		Class = yes	**Class = no**
Actual Class	**Class = yes**	**TP (true positive)**	**FN (false negative)**
	Class = No	**FP (false positive)**	**TN (true negative)**

- ***Accuracy*** measures the ratio of correctly predicted observations to the total number of observations calculated by the (1).

$$\textit{Accuracy} = (TP + TN)/(TP + FP + FN + TN) \tag{1}$$

- ***Precision*/PPV (Positive predictive value)** represents the ratio of correctly predicted, positive observations to the total predicted positive observations. High precision means a low, false positive rate and can be calculated as in (2).

$$\textit{Precision} = TP/(TP + FP) \tag{2}$$

- ***Recall*** (**Sensitivity**) is the ratio of correctly predicted positive observations compared with all the actual positive observations. The recall answer question is: of all the actual positive observations how many were accurately predicted? Recall is calculated by (3).

$$\textit{Recall} = TP/TP + FN \tag{3}$$

- ***F1 score*** is the weighted average of precision and recall as in (4). It takes both false positives and false negatives into account. *F1* is more indicative than accuracy in the case of uneven class distribution.

$$\textit{F1 Score} = 2 \text{“}\times\text{”} (\textit{Recall} \text{“}\times\text{”} \textit{Precision})/(\textit{Recall} + \textit{Precision}) \tag{4}$$

5. Results and Discussion

Three distinct experiments were performed on five different datasets. The first experiment utilized pre-trained word embedding for the classification task, using both CNN and BiLSTM. The second experiment involved trained, custom word embedding for each of the five datasets for sentiment classification by CNN and BiLSTM. Finally, baseline embeddings were also applied for evaluating their impact on the classification. Based on the experimental steps described, combined with the assessment results, a number of important findings are summarized as follows:

5.1. Pretrained Embeddings

The first experiment was to evaluate the performance of pre-trained embeddings using both CNN and BiLSTM for all five datasets. The results show that AraBERT achieved the highest performance for all the datasets, followed by pre-trained FastText, with a differential of one percent. GloVe comes next with higher attainment than AraVec, which may be attributed to the training corpus used to generate the pre-trained vectors. Figure 1 shows the accuracy of the pretrained word embeddings and Table 5 show accuracy, precision, recall, and F1-measure.

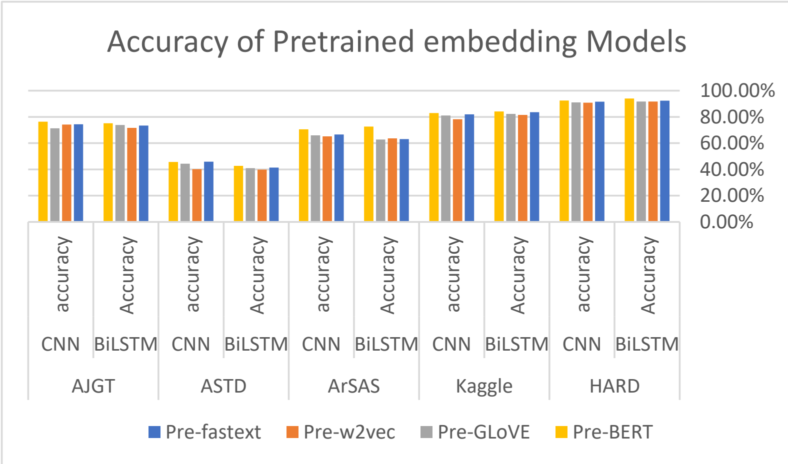

Figure 1. The accuracy of pretrained embedding.

Table 5. Pretrained embeddings Performance.

					Accuracy					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Accuracy	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**
FasText	92.33%	91.65%	83.64%	81.96%	63.14%	66.65%	41.41%	45.94%	73.44%	74.38%
AraVec	91.68%	90.83%	81.56%	78.29%	63.65%	65.16%	39.84%	40.20%	71.66%	74.16%
GLoVE	91.73%	91.07%	82.27%	81.04%	62.76%	65.94%	41.02%	44.33%	73.88%	71.38%
AraBERT	**93.97%**	**92.54%**	**84.15%**	**82.95%**	**72.63%**	**70.53%**	**42.78%**	**45.68%**	**75.14%**	**76.35%**
					Precision					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Precision	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**
FasText	93.67%	92.06%	83.00%	81.26%	69.59%	66.05%	42.72%	45.71%	74.07%	75.92%
AraVec	91.14%	90.45%	80.23%	79.21%	64.07%	64.19%	40.41%	41.22%	71.03%	73.11%
GLoVE	92.00%	91.42%	81.90%	80.72%	66.82%	65.43%	41.95%	43.17%	73.77%	75.08%
AraBERT	**93.85%**	**93.12%**	**83.99%**	**82.32%**	**72.42%**	**69.00%**	**43.71%**	**46.20%**	**75.11%**	**76.00%**
					Recall					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Recall	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**
FasText	93.44%	92.20%	82.56%	80.12%	68.13%	66.81%	42.13%	46.69%	74.73%	75.31%
AraVec	91.04%	90.31%	80.72%	78.74%	64.25%	63.71%	40.09%	41.84%	73.48%	72.51%
GLoVE	92.31%	91.54%	82.31%	81.52%	66.44%	65.18%	41.97%	42.69%	73.22%	74.47%
AraBERT	**93.58%**	**91.72%**	**83.42%**	**82.78%**	**71.51%**	**69.00%**	**43.39%**	**46.84%**	**74.32%**	**76.00%**
					F1-Measure					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Fmeasure	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**
FasText	93.55%	92.13%	81.96%	81.44%	68.85%	65.28%	41.58%	46.19%	73.90%	74.76%
AraVec	91.69%	91.38%	82.28%	80.97%	64.31%	65.95%	40.25%	41.03%	72.25%	75.31%
GLoVE	90.89%	90.48%	81.47%	78.13%	66.12%	64.30%	41.66%	46.19%	73.14%	74.22%
AraBERT	**93.61%**	**92.41%**	**83.70%**	**82.55%**	**71.96%**	**69.00%**	**43.55%**	**43.35%**	**74.71%**	**76.00%**

5.2. Custom Embeddings

The trained embedding models try to learn word representation from the current dataset using either baseline or custom embeddings. These experiments used baseline embeddings and generated custom embeddings using FastText, Word2vec, GloVe, and BERT.

When examining the performance among all the datasets using BiLSTM and CNN, BERT-based custom embedding is the best performing, followed by FastText-based custom embeddings as the second-best model for all the datasets. Word2Vec outperforms GloVe for HARD, KOOLI, and ArSAS datasets, whereas GloVe achieves higher performance over Word2vec in the AJGT and ASTD datasets. Baseline embedding achieves the same performance, with few differences, as GloVe when using the datasets shown in Table 6.

Table 6. Training embeddings Performance.

	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Accuracy	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**
Baseline	92.82%	91.29%	83.78%	84.29%	71.59%	69.71%	50.78%	51.95%	75.34%	77.78%
FasText	93.21%	90.14%	85.07%	82.36%	71.97%	69.13%	46.75%	46.75%	76.92%	78.12%
Word2vec	93.47%	92.40%	82.11%	81.81%	69.72%	68.39%	43.08%	44.92%	72.66%	73.44%
GLoVE	92.91%	91.14%	81.48%	79.40%	68.08%	67.72%	45.13%	46.77%	73.88%	77.97%
BERT	**94.25%**	**92.75%**	**84.97%**	**83.54%**	**73.35%**	**72.07%**	**47.91%**	**49.00%**	**78.76%**	**80.76%**
					Precision					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Precision	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**	**Precision**
Baseline	91.12%	91.55%	83.20%	82.96%	70.65%	69.11%	49.30%	50.07%	74.44%	75.28%
FasText	94.00%	93.00%	83.81%	82.00%	70.00%	69.00%	46.00%	48.00%	77.87%	78.00%
word2vec	93.51%	92.00%	82.33%	81.03%	71.00%	69.00%	43.00%	46.00%	72.53%	74.00%
GLoVE	92.17%	91.00%	81.00%	79.78%	68.00%	64.00%	45.34%	46.02%	74.15%	76.00%
BERT	**95.00%**	**93.00%**	**85.23%**	**84.00%**	**72.00%**	**70.00%**	**46.00%**	**48.00%**	**77.90%**	**79.00%**
					Recall					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Recall	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**	**Recall**
Baseline	93.13%	92.91%	86.40%	83.45%	71.00%	69.32%	56.78%	53.00%	80.00%	80.00%
FasText	93.00%	90.00%	82.97%	82.00%	70.00%	67.00%	46.00%	47.00%	78.00%	78.00%
Word2vec	93.00%	92.00%	80.00%	78.00%	70.00%	68.00%	44.71%	45.63%	73.28%	73.00%
GLoVE	93.88%	91.00%	82.56%	82.00%	68.00%	64.00%	44.00%	45.19%	78.00%	77.80%
BERT	94.00%	92.00%	85.11%	83.00%	71.00%	69.00%	45.00%	47.41%	73.28%	79.00%
					F1-Measure					
	HARD		**Khooli**		**ArSAS**		**ASTD**		**AJGT**	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Fmeasure	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**	**Fmeasure**
Baseline	92.12%	90.22%	80.74%	80.42%	69.03%	67.21%	44.74%	46.09%	76.12%	77.17%
FasText	93.50%	92.00%	83.97%	83.06%	71.80%	69.99%	46.30%	47.49%	77.93%	78.00%
Word2vec	93.00%	91.50%	82.70%	82.03%	70.50%	68.50%	43.84%	45.81%	72.90%	73.50%
GLoVE	92.52%	90.52%	81.56%	80.51%	68.78%	66.80%	44.62%	45.60%	75.03%	76.89%
BERT	**94.50%**	**92.50%**	**85.11%**	**83.00%**	**72.50%**	**69.50%**	**47.49%**	**48.70%**	**75.52%**	**79.00%**

5.3. Trained vs. Pretrained Embeddings

Table 7 illustrates that, for all the tested text representation models, generating embedding by one technique achieves higher performance than the pre-trained version of the same technique. BERT-based custom embeddings achieved the highest average performance among all datasets, followed by its pre-trained version, AraBERT. Embedding generated by the FastText model has the third-best performance among all the datasets followed by the FastText pre-trained version. The same applies to both Word2vec embedding, AraVec, pre-trained GloVe vectors, and custom GloVe embeddings.

Table 7. Pretrained embeddings F1-measures.

	HARD		Khooli		ArSAS		ASTD		AJGT	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Accuracy	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**
BERT	**94.25%**	**92.75%**	**84.97%**	**83.54%**	**73.35%**	**72.07%**	**47.91%**	**49.00%**	**78.76%**	**80.76%**
AraBERT	93.97%	92.54%	84.15%	82.95%	72.63%	70.53%	42.78%	45.68%	75.14%	76.35%
Train_FasText	**93.21%**	**90.14%**	**85.07%**	**82.36%**	**71.97%**	**69.13%**	**46.75%**	**46.75%**	**76.92%**	**78.12%**
FasText	92.33%	91.65%	83.64%	81.96%	63.14%	66.65%	41.41%	45.94%	73.44%	74.38%
Word2vec	**93.47%**	**92.40%**	**82.11%**	**81.81%**	**69.72%**	**68.39%**	**43.08%**	**44.92%**	**72.66%**	**73.44%**
AraVec	91.68%	90.83%	81.56%	78.29%	63.65%	65.16%	39.84%	40.20%	71.66%	74.16%
Train_GLoVE	**92.91%**	**91.14%**	**81.48%**	**79.40%**	**68.08%**	**67.72%**	**45.13%**	**46.77%**	**73.88%**	**77.97%**
GLoVE	91.73%	91.07%	82.27%	81.04%	62.76%	65.94%	41.02%	44.33%	73.88%	71.38%

5.4. Classic Embeddings

FastText, Word2vec, and GloVe are word embedding models that belong to the classic embedding paradigms. These experiments used pre-trained embedding vectors as well as generated custom embedding for each of the datasets. The results show that for the pre-trained vectors, FastText achieved the highest accuracy across all the datasets. GloVe outperformed the pre-trained Word2Vec model (AraVec). The accuracy difference observed between pre-trained GloVe and AraVec ranged from approximately 0.07% to 3.5%. Additionally, AraVec achieved slightly higher performance over pre-trained GloVe, when using CNN with the AJGT dataset and using BiLSTM with the ArSAS dataset. Pretrained GloVe and AraVec achieved almost the same performance using BiLSTM with HARD.

In the second experiment, classical models were used to generate custom embeddings and these results determine that training FastText outperforms other custom embeddings in performance across all applicable datasets. Word2Vec-generated embeddings were highly accurate when applied to the Hard, Khooli, and ArSAS datasets. The generated custom embeddings by GloVe achieved higher accuracy for ASTD and AJGT, as illustrated by Table 8 and Figure 2.

Table 8. Accuracy of trained embeddings models.

	HARD		Khooli		ArSAS		ASTD		AJGT	
	BiLSTM	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**	**BiLSTM**	**CNN**
	Accuracy	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**	**Accuracy**
FasText	**92.33%**	**91.65%**	**83.64%**	**81.96%**	**63.14%**	**66.65%**	**41.41%**	**45.94%**	**73.44%**	**74.38%**
AraVec	91.68%	90.83%	81.56%	78.29%	63.65%	65.16%	39.84%	40.20%	71.66%	74.16%
GLoVE	91.73%	91.07%	82.27%	81.04%	62.76%	65.94%	41.02%	44.33%	73.88%	71.38%
Train_FasText	**93.21%**	**90.14%**	**85.07%**	**82.36%**	**71.97%**	**69.13%**	**46.75%**	**46.75%**	**76.92%**	**78.12%**
Word2vec	**93.47%**	**92.40%**	**82.11%**	**81.81%**	**69.72%**	**68.39%**	**43.08%**	**44.92%**	**72.66%**	**73.44%**
Train_GLoVE	92.91%	91.14%	81.48%	79.40%	68.08%	67.72%	45.13%	46.77%	73.88%	77.97%

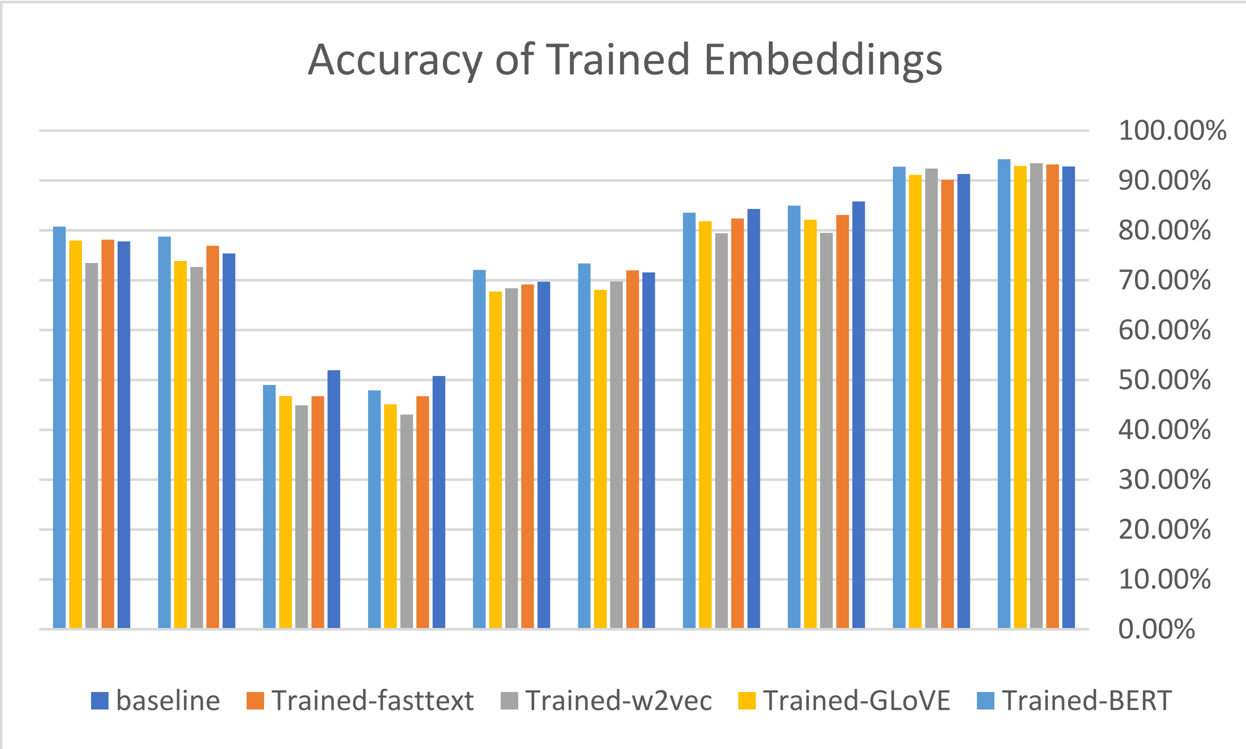

Figure 2. The accuracy of trained embeddings models.

5.5. Contextualized vs. Classic Embeddings

Throughout this study, BERT was implemented as a contextualized transformer-based embedding model. A more intensive analysis of the results reveals that, when compared with classic word embedding models, BERT has an advantage and achieves higher levels of performance in its pre-trained and trained models.

5.6. Impact of the Deep Learning Model

The results show that BiLSTM outperforms CNN among three datasets, HARD, Khooli, and ArSAS, in accuracy, precision, recall, and F1-measure, while CNN achieved higher performance in the AJGT and ASTD datasets. This can be attributed to the unique characteristics of each dataset including sample size or the diversity and size of the vocabulary.

5.7. Contextualized vs. Classic Embeddings

Throughout this study, BERT was implemented as a contextualized transformer-based embedding model. A more intensive analysis of the results reveals that, when compared with classic word embedding models, BERT has an advantage and achieves higher levels of performance in its pre-trained and trained models.

5.8. Impact of the Deep Learning Model

The results show that BiLSTM outperforms CNN among three datasets, HARD, Khooli, and ArSAS, in accuracy, precision, recall, and F1-measure, while CNN achieved higher performance in the AJGT and ASTD datasets. This can be attributed to the unique characteristics of each dataset including sample size or the diversity and size of the vocabulary as shown in Figure 3.

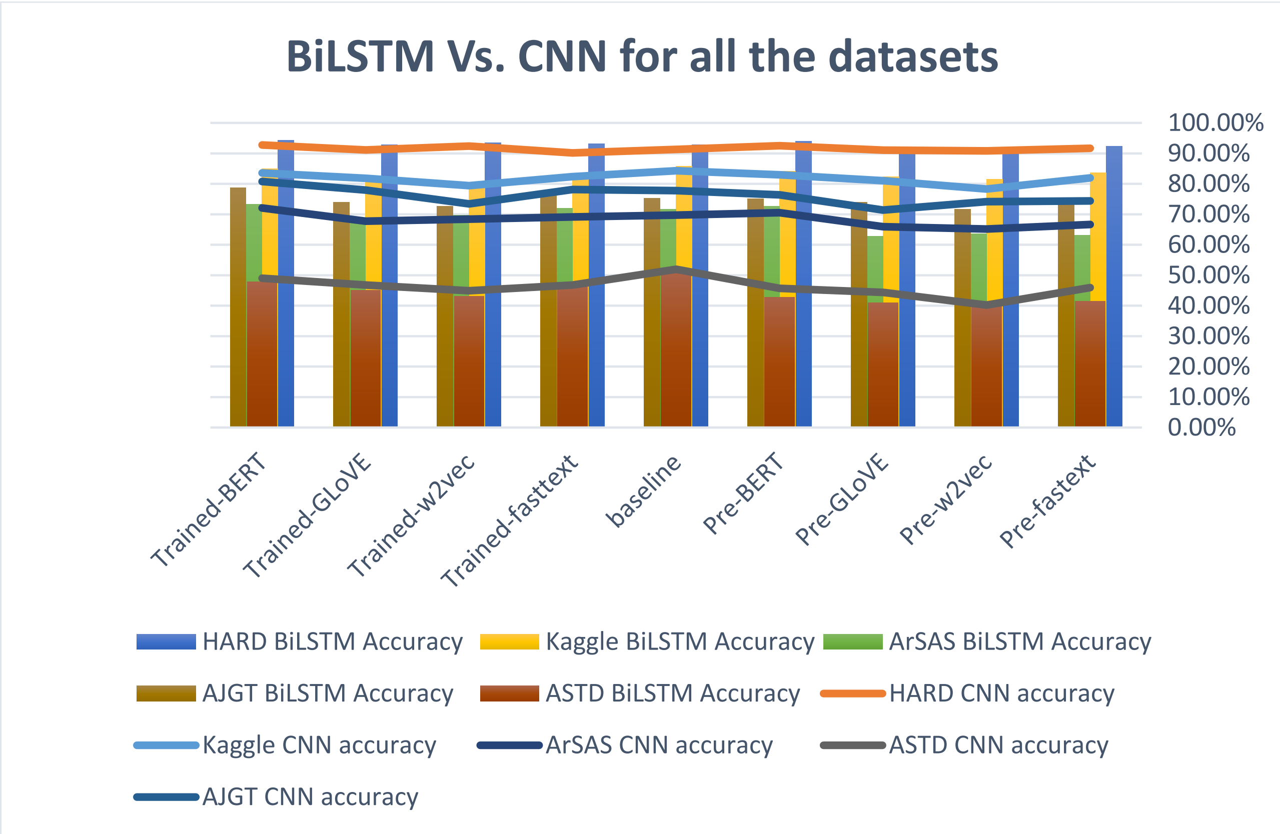

Figure 3. The accuracy of BiLSTM vs. CNN for different datasets.

6. Conclusions and Future Directions

This work conducted a comparison between contemporary, established word embedding techniques to address the Arabic sentiment classification problem. Both classical (GloVe, Word2Vec, and FastText) and transformer-based, contextualized (BERT) word embedding techniques were used. Additionally, both the pre-trained vectors and the generated vectors were utilized as a layer for the deep learning model. BiLSTM and CNN were employed for the text classifications task. The experiments included five different benchmark datasets from different domains and with different data sizes. The outcomes indicate that training the embedding model using the corpora achieved superior accuracy when compared with the pre-trained versions of the same models. Contextualized embeddings achieved enhanced performance over the classical models. Moreover, BiLSTM achieved higher levels of accuracy than CNN.

Future research should include an exploration of the comparison between deep learning models and traditional machine learning models. Additionally, investigating the varying levels of performance between the techniques when dealing with modern, standard Arabic and dialectal Arabic should be considered an objective. Finally, future experiments should investigate the comparison between the impact of word embedding and that of the document and paragraph embedding on the performance of the classifiers.

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

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used are publicly available benchmarks cited in the text.

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

References

1. Torregrossa, F.; Allesiardo, R.; Claveau, V.; Kooli, N.; Gravier, G. A survey on training and evaluation of word embeddings. *Int. J. Data Sci. Anal.* **2021**, *11*, 85–103. [CrossRef]
2. Shahi, T.B.; Sitaula, C.; Paudel, N. A Hybrid Feature Extraction Method for Nepali COVID-19-Related Tweets Classification. *Comput. Intell. Neurosci.* **2022**, *2022*, 5681574. [CrossRef] [PubMed]
3. Wang, C.; Nulty, P.; Lillis, D. A Comparative Study on Word Embeddings in Deep Learning for Text Classification. In Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, Seoul, Republic of Korea, 18–20 December 2020; pp. 37–46. [CrossRef]
4. Soliman, A.B.; Eissa, K.; El-Beltagy, S.R. AraVec: A set of Arabic Word Embedding Models for use in Arabic NLP. *Procedia Comput. Sci.* **2017**, *117*, 256–265. [CrossRef]
5. Pennington, J.; Socher, R.; Manning, C.D. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
6. Santos, I.; Nedjah, N.; de Macedo Mourelle, L. Sentiment analysis using convolutional neural network with fastText embeddings. In Proceedings of the 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Arequipa, Peru, 8–10 November 2017; pp. 1–5. [CrossRef]
7. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805. [CrossRef]
8. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv* **2019**, arXiv:1909.11942. [CrossRef]
9. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv* **2020**, arXiv:1906.08237. Available online: http://arxiv.org/abs/1906.08237 (accessed on 11 May 2022).
10. Wang, B.; Wang, A.; Chen, F.; Wang, Y.; Kuo, C.-C.J. Evaluating word embedding models: Methods and experimental results. *APSIPA Trans. Signal Inf. Process.* **2019**, *8*, e19. [CrossRef]
11. Bian, J.; Gao, B.; Liu, T.-Y. Knowledge-powered deep learning for word embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 132–148.
12. Alamoudi, E.S.; Alghamdi, N.S. Sentiment classification and aspect-based sentiment analysis on yelp reviews using deep learning and word embedding. *J. Decis. Syst.* **2021**, *30*, 259–281. [CrossRef]
13. Kilimci, Z.H.; Akyokuş, S. Deep learning-and word embedding-based heterogeneous classifier ensembles for text classification. *Complexity* **2018**, *2018*, 7130146. [CrossRef]
14. Al-Ayyoub, M.; Khamaiseh, A.A.; Jararweh, Y.; Al-Kabi, M.N. A comprehensive survey of arabic sentiment analysis. *Inf. Process. Manag.* **2019**, *56*, 320–342. [CrossRef]
15. Badaro, G.; Baly, R.; Hajj, H.; El-Hajj, W.; Shaban, K.B.; Habash, N.; Al-Sallab, A.; Hamdi, A. A Survey of Opinion Mining in Arabic: A Comprehensive System Perspective Covering Challenges and Advances in Tools, Resources, Models, Applications, and Visualizations. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* **2019**, *18*, 1–52. [CrossRef]
16. Rajput, V.S.; Dubey, S.M. An Overview of Use of Natural Language Processing in Sentiment Analysis based on User Opinions. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2016**, *6*, 5.
17. Alnawas, A.; Arici, N. Effect of Word Embedding Variable Parameters on Arabic Sentiment Analysis Performance. *arXiv* **2021**, arXiv:2101.02906. [CrossRef]
18. Barhoumi, A.; Estève, Y.; Aloulou, C.; Belguith, L. Document embeddings for Arabic Sentiment Analysis. In Proceedings of the Conference on Language Processing and Knowledge Man-agement, LPKM 2017, Sfax, Tunisia, 8–10 September 2017.
19. Alayba, A.M.; Palade, V.; England, M.; Iqbal, R. Improving Sentiment Analysis in Arabic Using Word Representation. In Proceedings of the 2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR), London, UK, 12–14 March 2018; pp. 13–18. [CrossRef]
20. Altowayan, A.A.; Tao, L. Word embedding for Arabic sentiment analysis. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 3820–3825.
21. El Mekki, A.; El Mahdaouy, A.; Berrada, I.; Khoumsi, A. Domain Adaptation for Arabic Cross-Domain and Cross-Dialect Sentiment Analysis from Contextualized Word Embedding. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; Association for Computational Linguistics, Online, 6–11 June 2021; pp. 2824–2837.
22. 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, with a Shared Task on Offensive Language Detection*; European Language Resource Association: Marseille, France, 2020; pp. 9–15. Available online: https://aclanthology.org/2020.osact-1.2 (accessed on 11 November 2022).
23. Fouad, K.M.; Sabbeh, S.F.; Medhat, W. Arabic Fake News Detection Using Deep Learning. *Comput. Mater. Contin.* **2022**, *71*, 3647–3665. [CrossRef]

24. Dahou, A.; Xiong, S.; Zhou, J.; Haddoud, M.H.; Duan, P. Word Embeddings and Convolutional Neural Network for Arabic Sentiment Classification. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016; pp. 2418–2427. Available online: https://aclanthology.org/C16-1228 (accessed on 15 December 2022).
25. Sallab, A.; Hajj, H.; Badaro, G.; Baly, R.; El Hajj, W.; Shaban, B.K. Deep Learning Models for Sentiment Analysis in Arabic. In Proceedings of the Second Workshop on Arabic Natural Language Processing; Association for Computational Linguistics, Beijing, China, 30 July 2015; pp. 9–17. [CrossRef]
26. Abdul-Mageed, M.; Zhang, C.; Hashemi, A.; Nagoudi, E.M.B. AraNet: A Deep Learning Toolkit for Arabic Social Media. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*; European Language Resource Association: Marseille, France, 2020; pp. 16–23. Available online: https://aclanthology.org/2020.osact-1.3 (accessed on 15 December 2022).
27. Alayba, A.M.; Palade, V. Leveraging Arabic sentiment classification using an enhanced CNN-LSTM approach and effective Arabic text preparation. *J. King Saud Univ. Comput. Inf. Sci.* **2021**, *34*, 9710–9722. [CrossRef]
28. Al-Azani, S.; El-Alfy, E.-S.M. Hybrid Deep Learning for Sentiment Polarity Determination of Arabic Microblogs. In *Neural Information Processing*; Lecture Notes in Computer Science; Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 10635, pp. 491–500. [CrossRef]
29. Al-Smadi, M.; Qawasmeh, O.; Al-Ayyoub, M.; Jararweh, Y.; Gupta, B. Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews. *J. Comput. Sci.* **2018**, *27*, 386–393. [CrossRef]
30. Safaya, A.; Abdullatif, M.; Yuret, D. KUISAIL at SemEval-2020 Task 12: BERT-CNN for Offensive Speech Identification in Social Media. *arXiv* **2020**, arXiv:2007.13184. [CrossRef]
31. Elnagar, A.; Khalifa, Y.S.; Einea, A. Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Applications. In *Intelligent Natural Language Processing: Trends and Applications. Studies in Computational Intelligence*; Shaalan, K., Hassanien, A., Tolba, F., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; Volume 740, pp. 35–52. [CrossRef]
32. Alomari, K.M.; Elsherif, H.M.; Shaalan, K. Arabic Tweets Sentimental Analysis Using Ma-chine Learning. In Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Arras, France, 27–30 June 2017. [CrossRef]
33. Elmadany, A.; Mubarak, H.; Magdy, W. ArSAS: An Arabic SpeechAct and Sentiment Corpus of Tweets. In Proceedings of the 3rd Workshop on OpenSource Arabic Corpora and Processing Tools, Miyazaki, Japan, 15 January 2018; p. 20.
34. 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; pp. 2515–2519.
35. Kim, Y. Convolutional neural networks for sentence classification. *arXiv* **2014**, arXiv:1408.5882.
36. Vizcarra, G.; Mauricio, A.; Mauricio, L. A deep learning approach for sentiment analysis in Spanish tweets. In *International Conference on Artificial Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 622–629.
37. Heikal, M.; Torki, M.; El-Makky, N. Sentiment analysis of Arabic tweets using deep learning. *Procedia Comput. Sci.* **2018**, *142*, 114–122. [CrossRef]
38. Alayba, A.M.; Palade, V.; England, M.; Iqbal, R. A combined CNN and LSTM model for Arabic sentiment analysis. In Proceedings of the International Cross-Domain Conference for Machine Learning and Knowledge Extraction, Hamburg, Germany, 27–30 August 2018; pp. 179–191. [CrossRef]
39. Hourrane, O.; Idrissi, N.; Benlahmar, E.H. An empirical study of deep neural networks models for sentiment classification on movie reviews. In Proceedings of the 2019 1st International Conference on Smart Systems and Data Science (ICSSD), Rabat, Morocco, 3–4 October 2019.
40. Mohammed, A.; Kora, R. Deep learning approaches for Arabic sentiment analysis. *Soc. Netw. Anal. Min.* **2019**, *9*, 52. [CrossRef]