Array 13 (2022) 100109

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

AraScore: A deep learning-based system for Arabic short answer scoring

Omar Nael [a], Youssef ELmanyalawy [a], Nada Sharaf [b,*]

[a] The German University in Cairo, Cairo, Egypt
[b] The German International University, Cairo, Egypt

ARTICLE INFO

Keywords:
Short answer scoring
Arabic
Deep learning

ABSTRACT

In the past years, Arabic NLP has been significantly lagging behind its English counterpart, but recent advancements in Natural Language Processing have made it possible for Arabic to catch up and show promising results for a multitude of tasks. Complex tasks such as short answer scoring, have been widely researched mainly for English, leveraging machine learning and state-of-the-art deep learning techniques. In this paper, we introduce the first deep learning-based system for Arabic short answer scoring, in efforts to provide a reliable system that can help teachers in the Arab world better utilize their time in other teaching activities that would elevate the quality of learning in the region. We empirically study different techniques and propose the best performing system based on our results, where we have achieved state-of-the-art performance, achieving a *QWK* score of **0.78** and showing how powerful and robust recent Arabic NLP tools have become.

1. Introduction

Automated grading systems have been efficiently reducing the huge amounts of resources and time spent in grading exams, allowing teachers to better utilize their time and effort in other core tasks that would yield a better educational experience for their students. Due to the precise nature of the grading process, extensive research has been conducted to ensure that a fair platform is being offered. Such platform should be able to properly justify replacing manual grading methods.

Reference-based systems [1] are one of the basic and efficient techniques applied to implement such systems. Such systems work given the student's answer and a reference answer. Similarity measures are used, such as cosine similarity and Levenshtein distance [2]. Such measures can later be used within an algorithm to assign a class to each answer.

With the rise of machine learning, **response-based** systems have been deployed, where a reference answer is not provided, but rather a set of human-labelled answers with different scores. Those labelled answers can be used to train a machine learning model in a supervised fashion [3].

Deep learning architectures have been conquering NLP during the past few years, such as RNNs, LSTMs, attention mechanisms and transformer-based models [4]. They have achieved state-of-the-art results across different NLP tasks. Consequently, response-based systems have further advanced by utilizing deep neural networks [5], achieving much more promising results [6] compared to other methods. The current advancements in NLP can also be traced back to the release of large pretrained language models [7,8], which can be fine-tuned for downstream tasks, commonly referred to as *transfer learning*.

There was a gap for short answer scoring in the Arabic language. Most of the available systems were reference based. The two approaches were explored by [9]. It was found that with sufficient amount of human rated answers then the response-based approach is likely to perform better than the reference-based one. But this remained unexplored in the Arabic language. The only response-based approach for Arabic short answer scoring was presented in [10]. However, the existing work for Arabic short answer scoring systems never explored deep learning. With the performance that deep learning systems offer, the work in this paper tries to fill in this gap by contributing a deep learning response-based system for Arabic short answer scoring.

In this paper, we attempt to contribute with two main aspects. Firstly, we generally address the lack of research in implementing deep learning-based short answer scoring systems, especially using transformer-based models [11], where we will demonstrate and discuss multiple experiments to implement our system.

Secondly, the core contribution of our work is to investigate and propose the first **Arabic** deep learning-based short answer scoring system, to the best of our knowledge. The system utilizes **RNNs**, **LSTMs**, **AraBERT** and **AraELECTRA** [7,8], along with a baseline model consisting of a tf–idf vectorizer used with an SVM.

This paper is organized as follows: Section 2 includes a general overview of the challenges faced regarding the Arabic language, as well as an overview of transfer learning and deep learning in the domain of

* Corresponding author.
E-mail address: nada.hamed@giu-uni.de (N. Sharaf).

https://doi.org/10.1016/j.array.2021.100109
Received 1 August 2021; Received in revised form 16 October 2021; Accepted 15 November 2021
Available online 7 January 2022

short answer scoring. Section 3 gives an overview about the dataset, preprocessing steps and the evaluation metrics used for this work. Section 4 discusses the applied methodology. The experiments one are shown in Section 5. The results are shown in Section 6 discusses the results of the experiments. Further discussion is done in Section 7. We then conclude with directions to future work.

2. Background

In this section we will briefly discuss challenges in the Arabic language, such as morphology, dialectal variations, and lexical ambiguities. Traditional NLP methods are then discussed in addition to deep learning and transfer learning, leveraging transformer language models such as BERT and how they contributed to the *rise* of NLP. Finally, previous work regarding automated grading systems will also be discussed.

2.1. Arabic morphology

Arabic is known for being a complex language. It contains many subtleties and ambiguities that sets it apart from other languages, making it a challenging language in the context of NLP. One of the challenges is the complex morphology when compared to English, where Arabic words can take on several forms resulting in numerous variations of a single word that can often times translate to a sentence in English [12]. All verbs in Arabic have a root usually from three letters making it highly derivational. Therefore, it is much harder to build Arabic stemming/lemmatizing tools [13], and adapt tokenizers such as WordPiece and SentencePiece tokenizers used in BERT for Arabic. and. Moreover, lexical ambiguities is also highly present in Arabic, with context playing a huge factor in determining the meaning of a word or sentence [14].

2.2. Dialectal variations

One other challenge in the Arabic language is dialectal variations. Each region has its own *version* of Arabic. While the general structure of Arabic is preserved in most dialects, usually each dialect comes with an additional set of custom rules and vocabulary [14]. Unfortunately, this can hinder the quality of Arabic datasets. For example, a model can perform extremely well on the Egyptian dialect, but perform poorly on the Tunisian dialect [15]. Thus, in order to build generalized Arabic models, different dialects must be taken into account when collecting the data.

Fortunately, exams are usually solved in Standard Arabic as a sign of formality, but is not strictly enforced especially on younger students. In general, this allows building a generalized grading system to be easier than other tasks, even though some variations and inconsistencies might occur.

2.3. Lexical ambiguity

There are many causes for lexical ambiguity in Arabic compared to other languages. One of the main aspects that causes lexical ambiguity is **diacritization**. While the general purpose of diacritics is to indicate the pronounciation of a given word, it can also determine the *sense* of a word within a sentence. For example, the word/lemma عَلَمٌ can have different meanings based on diacritics and dialect [16]. In Table 1 we demonstrate some of the different meanings the word عَلَمٌ can have. This is a common phenomena in Arabic which makes it a really complex and rich language. For our specific case (automated grading), the dataset we are using is an Arabic translated version. As a result, diacritics are completely disregarded. Moreover, students usually do not include diacritics in their answers for ease of writing, with the exception of formal Arabic exams, as human graders can generally understand the context without diacritics. However, this poses a difficult challenge for automated NLP systems, since any collected dataset for this task will most likely not include diacritics.

Table 1
Lexical ambiguity in Arabic.

Arabic	Meaning
عَلَمَ	Taught
عِلْمٌ	Knowledge
عَلَمٌ	Flag
عَلِمَ	Knew
عُلِمَ	Is known
عَلَّمَ	Teach
عُلِّمَ	Is taught

2.4. Traditional feature extraction in NLP

Feature extraction and representation in NLP was usually performed using n-gram bag-of-words features, which is still widely used today . This method, while efficient and easy to implement, is usually suitable for specific and small corpora [11]. The main drawbacks of the bag-of-words model is the loss of *contextual awareness*. Moreover, when it is applied to large corpora it will result in sparse vectors that hinders the efficiency and quality of representations.

Pretrained word embeddings have also surfaced in the past decade, such as Word2Vec and GloVe. They are trained on a large corpus for the desired language, and a high-dimensional vector is produced for each word, taking into account semantic similarities [17,18]. As a result of representing each word as an independent vector, context is also not taken into account, when a word can have different meanings depending on the context. Such models also suffer from not being able to handle *unseen* words. This can cause problems when dealing with text that contains specific content, such as scientific content that is abundant in short answer scoring datasets.

2.5. Deep learning in NLP

There is no doubt that deep learning has revolutionized NLP. Architectures such as RNNs have greatly improved the processing of sequential data, where they accurately capture features present in natural language by taking into account previous words which is crucial to capture the semantics in a sentence [19]. Unfortunately, traditional RNNs suffered from the *vanishing gradient* problem, which results in the loss of captured information at the beginning of a sequence. Consequently, LSTM networks were introduced to address this problem by being able to learn long-term dependencies, followed by the introduction Bi-LSTMs that can process sequential data from both directions (left-to-right and right-to-left), which allowed for even better context understanding.

2.6. BERT: Bidirectional encoder representations from transformers

The introduction of transformers [20] was a turning point for NLP. This has lead to the introduction many transformer-based language models such as BERT, short for *"Bidirectional Encoder Representations from Transformers"*. BERT is a language model that is pretrained on huge amounts of unstructured text, in order to develop a *general* understanding of a given language [21]. This is achieved by training BERT on two tasks:

1. **Masked LM**: A percentage of the input tokens is *masked* using a special token `[MASK]` ,and the model is required to predict the masked tokens/words by *looking* in both directions and using the full context of the sentence. Thus, A deeply *bidirectional* model is achieved that is fully context-aware.

Table 2
A history of reference-based Arabic automated short answer scoring systems.

Author	Approach	Accuracy
[27]	Reference	RMSE = 0.76
[28]	Reference	84%

Table 3
A history of English automated short answer scoring systems.

Author	Approach	Accuracy
[29]	Response	77.4%
[30]	Reference	88%
[3]	Response	MSE = 0.321
[31]	Reference	RMSE = 0.91

2. **Next Sentence Prediction**: This is a binary classification task, where the model receives pairs of sentences as input, with the second sentence being the correct sentence or a randomly chosen sentence from another document. BERT learns to predict the correct subsequent sentence. This allowed for better understanding of relationships between sentences.

2.7. ELECTRA

Another language model that we will be discussing and have used in our work is ELECTRA, short for *"Efficiently Learning an Encoder that Classifies Token Replacements Accurately"*. Authors in [22] also introduce a transformer-based language model similar to BERT, but propose a new and more efficient method for pretraining.

They propose *replaced token detection* instead of the MLM task present in BERT. Akin to a GAN training process [23], a *generator* randomly replaces tokens with fake *high-quality* words. A *discriminator* is then trained to predict whether each token is original or replaced. Thus, ELECTRA learns from all input tokens rather than the masked tokens only, which resulted in a more cost and computationally efficient model that outperforms BERT in many downstream tasks [22]. It is important to note that this method is not *adversarial* since the generator producing corrupted tokens is trained with maximum-likelihood.

2.7.1. Deep learning in short answer scoring

Transformer-based language models have carved the path for *transfer learning* in NLP. Such models have been fine-tuned for downstream tasks and achieved state-of-the-art results in most NLP tasks [21]. Deep learning has only been minimally researched in developing short answer scoring systems, with a few utilizing models such as BERT and ELECTRA. Bi-LSTMs with attention showed to be a powerful candidate for the task at hand [6]. They have achieved promising results reaching a **Quadratic Weighted Kappa(QWK)** of 0.723 on the Kaggle ASAP-SAS dataset.[1] On the same dataset, [24] used different architectures such as CNNs, LSTMs, and BERT, reporting a QWK of 0.62, 0.65 and 0.71 respectively, with BERT achieving the best performance.

2.8. Arabic short answer scoring

Only a few Arabic short answer scoring systems exist, where most of them being based-off the reference-based method. Authors in [25,26] presented a promising reference-based Arabic short answer scoring system with effective feedback based on the similarity the score. They proposed string-based similarity and corpus-based similarity and a hybrid approach to compare the similarity between the student answer and the reference answer.

Tables 2 and 3 provide some of the work done for short answer scoring in English and Arabic. The previous work did not use the same datasets. We show for every paper, the used methodology and the results obtained.

[28] presented the most recent **Arabic** reference-based short answer scoring system. The proposed model relies on tokenization, stop-words removal, and retrieving the word root and synonyms for each keyword in the student's answer and the model answer. Both answers are then represented as their respective vectors. Accordingly, the cosine similarity between both answers is calculated, and used for the class/grade prediction.

Authors in [32] propose a response-based systems using word2vec and simple models such as SVM, Random Forest and a small multi-layer perceptron. Although, they have performed their experiments on a different dataset, they have shown promising results and motivated the use of larger state-of-the-art neural networks for Arabic short answer scoring.

To the best of our knowledge neural language models and contextualized embeddings have not been investigated for **Arabic** automated grading systems, which we believe is due to the previous lack of sufficient resources for the Arabic language. We found this to be a great opportunity to investigate the recent deep learning advancements in Arabic NLP for implementing an automated short answer scoring system.

3. Dataset

We conducted all experiments (discussed in detail in Section 5) on the *ASAP Short Answer Scoring* dataset, a publicly available dataset from a previous automatic scoring competition hosted by Kaggle. This dataset was collected by scanning **graded** exams intended for Grade 10 students and their respective answers, followed by using OCR tools to convert the scanned answers to text. Since the scope of our work is Arabic, we use the Google Translate API[2] to translate the dataset from English to Arabic. It is no doubt that using machine translation is not a scalable solution. It comes at the cost of losing some data quality, and translating large batches can become really expensive. The reason for choosing to translate the dataset, is the lack of any publicly available Arabic short answer scoring datasets of sufficient scale. We have found that the only publicly available Arabic dataset is the **AR-ASAG** [33] dataset, containing only 2133 students answers that span over 48 questions, which is more suitable for reference-based and semantic similarity tasks discussed in 2.

The dataset contains a set of student answers for 10 distinct questions/prompts, comprising of a variety of topics, from reading comprehension questions to science and biology questions. A holistic grading scheme is followed by two graders, where the minimum grade is 0, and the maximum grade is 3. Fig. 1 shows the answers frequency distribution, having approximately 1600 answers corresponding to each question, and a total of 17 205 answers.

In Fig. 2, we also show the frequency of each score. We can definitely observe a class imbalance between the scores, which is logical as the number of student who score the full-grade will be much less than students who score 1. We further present the class imbalance across the datasets in Fig. 2. For example, we found that for some questions no one scored **3** and in some questions the predominant score is **1**. This is an expected observation due the varying levels of difficulty across the questions. Details about how we handled the class imbalance will be discussed in Section 5.

We also graphically present the frequency distribution of the answers sequence length in Fig. 3. The average sequence length is 40 and the maximum length is 250. This information will be used when fine-tuning our models to set the maximum sequence length (see Table 4).

[1] https://www.kaggle.com/c/asap-sas.

[2] https://cloud.google.com/translate/docs/advanced/batch-translation.

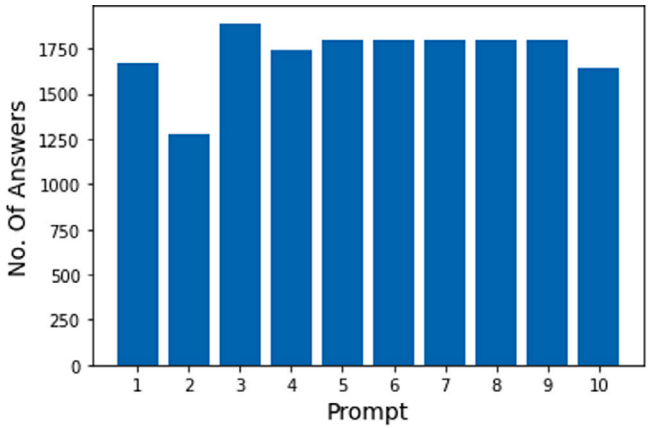

Fig. 1. Number of answers for each prompt.

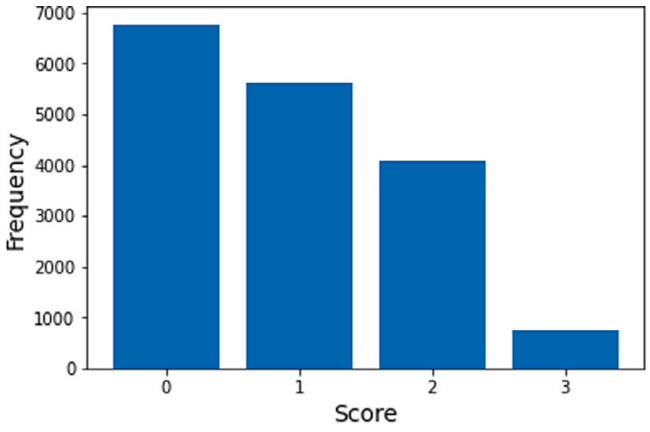

Fig. 2. Scores distribution.

Table 4
Scores for each prompt.

Question	0	1	2	3
1	380	429	524	339
2	168	326	466	317
3	451	999	441	–
4	669	937	132	–
5	1391	328	42	34
6	1515	160	71	51
7	932	448	419	–
8	549	473	777	–
9	434	742	622	–
10	289	770	580	–

3.1. Data pre-processing

Different preprocessing steps have been conducting for our experiments. For our baseline model, consisting of a TF-IDF vectorizer and SVM, we first cleaned the text from any numbers and symbols, and some artifacts from the translation process, where we found some random English letters and words that have not been translated. The next preprocessing step is lemmatizing using *Farasa* [13]. Lemmatizing is a crucial *normalization* step, that greatly reduces the size of the TF-IDF vectors to avoid sparsity as much as possible.

Light preprocessing was performed when running our experiments on BERT and ELECTRA. The same cleaning process was used, but we have not performed any lemmatization. We argue that performance gains from lemmatization will be insignificant when fine-tuning transformer models, as they are usually pretrained on raw text. Another basis for our choice is that *Byte-Pair Encoding* is used when tokenizing input

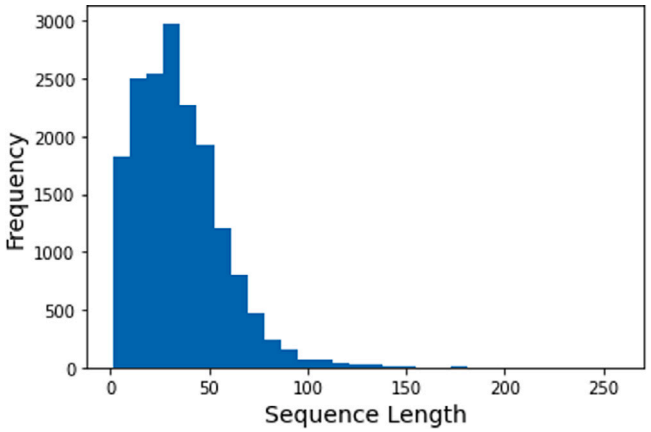

Fig. 3. Sequence length distribution.

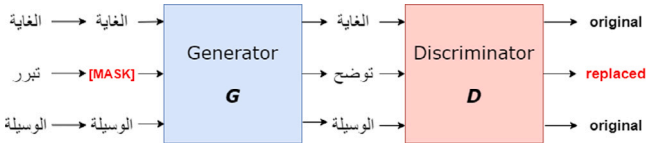

Fig. 4. ELECTRA architecture.

text [21]. Meaning that, words such as *"eating"* will be decoded into [eat + ##ing]. Thus, *unseen* words are already handled efficiently.

3.2. Evaluation metric

The output of an automated grading systems can be compared to the scores assigned by human annotators using various measures of correlation or agreement. We use *Quadratic Weighted Kappa(QWK)*, which is adopted in the original ASAP-SAS Kaggle competition and is widely used as an evaluation metric for such systems [34].

4. Methodology

4.1. RNN-based models

Multiple RNN-based models were used such as standard RNN, LSTM, and Bi-LSTM. All models consist of 5 layers. The first layer consists of 128 units with a sigmoid activation function, followed by a dropout layer of 0.2. The third layer consisted of 64 units with a sigmoid activation function, followed by another dropout layer of 0.2. The fifth layer was a dense layer with 4 output units that correspond to the 4 possible classes in the dataset, with a softmax activation function. Pretrained sentence embeddings were generated using *Universal Sentence Encoder* [35].

4.2. BERT & ELECTRA

Authors in [7,8] introduce Arabic versions for both BERT and ELECTRA. They are pretrained on 77 GB of raw text. They used the Arabic Wikipedia dump from 2020/09/01, the 1.5B words Arabic corpus [36], the OSCAR corpus, and the OSIAN corpus [37].

After the pretraining process of ELECTRA described in Section 2, the *discriminator* **D** is used for downstream tasks which has the same architecture as BERT, the difference being in the pretraining process (see Fig. 4).

They offer both base and large architectures, with the *base* architecture having 12 transformer blocks, 12 attention heads, and 136 million parameters, and the *large* architecture having 24 transformer blocks,

24 attention heads and 371 million parameters [7]. We chose the base architecture as it is more computationally efficient.

The first token of every input sequence is the special classification token — `[CLS]`. This token is originally used in the **Next Sentence Prediction** task, discussed in 2, as an aggregate of the entire sequence representation. It contains a 768-dimension *contextualized* embedding of the sequence/sentence. The output of this token is further utilized in all downstream classification tasks. In order to fine-tune BERT and ELECTRA for classification, an extra output layer is added to the BERT layers where it receives the `[CLS]` token as input.

Authors in [38] investigate layer *freezing* to reduce the number of trainable parameters when fine-tuning. They have observed that the best performance is achieved when the errors are *"backpropagated"* through all layers, updating all pretrained weights during training. This comes at the cost of being the slowest fine-tuning method. Consequently, the fastest method is to freeze all layers and use the BERT embeddings only, but results in [38] have shown that this method yields poor performance. Accordingly, we use all 12 layers to ensure maximum quality.

5. Experiments

In this section we describe the experimental setup in detail using three different models. Before we proceed with discussing the models, we will first discuss the testing & validation method that was controlled across all experiments. Since the test set was not released, we split 20% of our training data before any model training, to use as a final evaluation method. Moreover, we used 5-fold cross validation during the training process. To address the class imbalance presented in Fig. 2, we used *stratified* splits using the **scikit-learn** library, to ensure that all splits are an appropriate representative of the original data. Each model is trained using two different methods. The first method is training the model on each Question individually, and the other method is training the model on the whole dataset at once, including the question ID in the feature vector.

5.1. Baselines

Two baseline methods are used for comparison. A classical ML approach using a SVM paired with a TF-IDF vectorizer, and a reference-based approach.

The reason for choosing these two baseline experiments is due to the fact that most of the previous work is implemented for English and the work targeted for Arabic uses different datasets and scoring methods. Thus, we use standard baseline techniques that are similar to the ones present in previous work 2 in order to conduct a fair comparison as possible, especially for Arabic since neural networks were not investigated before regarding automated grading.

The reference-based experiment is defined as follows, a random reference answer is picked from the highest scoring answers for each question, followed by computing the cosine similarity of the input against the reference answer. In order to normalize the scores, we use the following scale. A cosine similarity score of **0-0.25** will output 0, **0.25–0.50** will output 1, **0.50–0.75** will output 2, and **0.75-100** will output 3.

The second baseline experiment is defined as follows. We first train a TF-IDF vectorizer on our dataset. Then we use the resultant vectors as input to an SVM classifier. We use a *grid search* for hyperparameter optimization. The hyperparameters present in our search are the *kernel*, the regularization parameter *C*, and the kernel coefficient *gamma*. We performed the optimization on the QWK score. The following are the grids of values for each hyperparameter:

1. **Kernel**: [linear, poly, rbf, sigmoid, precomputed].
2. **Regularization Parameter C**: [0.001, 0.01, 0.1, 1, 10], followed by a search on [10.070, 10.074, 10.075, 10.1, 10.125].

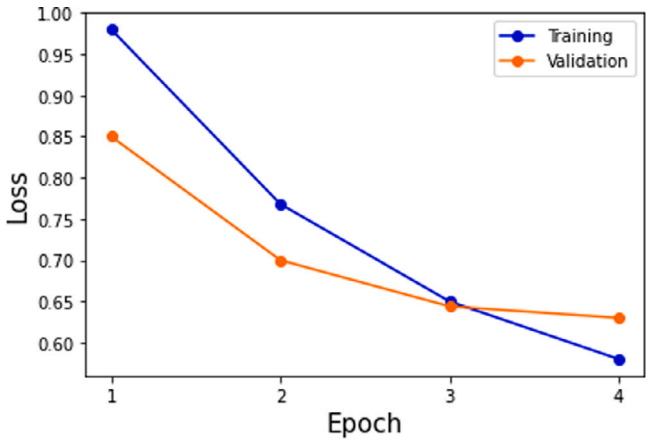

Fig. 5. Train/validation loss.

3. **Gamma**: [0.001, 0.01, 0.1, 1, 2], followed by a search on [0.165,0.175, 0.108, 0.11, 0.2].

After the grid search was conducted, the following parameters were chosen: The kernel used is the Radial Basis Function, the regularization parameter was set to 10.74 and the Gamma parameter was set to 0.108.

5.2. RNNs

All 3 RNN-based models used the same hyperparameters. A learning rate of 1e-4 with an early stopping function was used. We also used a batch size of 32, and sparse categorical cross-entropy as a loss function. The "use bias" hyperparameter was set to true where it makes the layer uses a bias vector. The forget bias was also set to true to add 1 to the bias of the forget gate at initialization, and the bias initializer was set to zeros. Finally, the optimizer used is **Adam**.

5.3. Fine-tuning

We follow the default fine-tuning strategy recommended by authors in [21,22]. The suggested hyperparameters are as follows:

- **Learning Rate(Adam)**: 2e-5, 3e-5, or 5e-5.
- **Batch Size**: 16, 32, or 64.
- **Number of epochs**: 2–4.

We manually tuned the hyperparameters on the suggested parameters, and set a fixed number of **4** epochs, while observing the train/validation loss plots to determine the optimal number of epochs. BERT architectures are prone to overfitting, where they usually converge before 4 epochs when training on small dataset. This can be observed in Fig. 5, which is extracted from a random sample run in one of the folds. The model starts overfitting after the third epoch as the validation loss starts to *plateau*, whereas the training loss is still decreasing.

Another interesting observation is that upon inspecting our plots, we find that the validation loss always starts with a smaller value than the train loss. This is not the usual case when training a model, but we speculate that it could be due to the different setups between *training* mode and *evaluation* mode. Both BERT and ELECTRA use *dropout* as a means of regularization, which is used only while training. Thus, resulting in a greater loss than the validation, as dropout is not used in evaluation. Although, further investigations needs to be conducted.

The optimal set of hyperparameters for BERT was the following: A learning rate of 5e-5 was used. The batch size was set to 32 and the optimal number of epochs was found to be 3.

Moreover, the optimal set of hyperparameters for ELECTRA was the following: A learning rate of 2e-5 was used. Similarly, The batch size

Table 5
Parameter values.

Parameter	Value
The kernel used	Radial basis function
Regularization parameter	10.74
Gamma parameter	0.108
RNN learning rate	1e−4
Batch size	32
Loss function	Sparse categorical cross-entropy
Epochs	4
BERT learning rate	5e−5
Batch size	32
Epochs	3
ELECTRA learning rate	2e−5
Batch size	32
Epochs	3

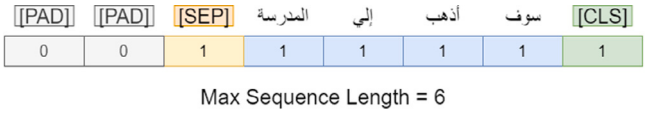

Fig. 6. Input tokens.

was set to 32 and the optimal number of epochs was found to be 3. An Adam optimizer was used for both experiments.

Table 5 shows some of the parameter values.

We also need to determine the input sequence length. Both BERT and ELECTRA have a maximum input sequence length of 512. Accordingly, we set our sequence length based on the values in Fig. 3. One approach is to set the sequence length is to use the average sequence length ($\tilde{4}0$), cutting down on computational time but can also result in severely hindering the data quality. Thus, we used the largest sequence length in data ($\tilde{2}50$) to preserve data as much as possible. This results in many *zero paddings* 6 in our input. Fortunately, the paddings are defined as `[PAD]` tokens which are detected and do not have an effect while training.

Since the task at hand is a multi-class classification problem, the output layer is defined as a *softmax* classifier layer:

$$p(c|\mathbf{h}) = softmax(W\mathbf{h}) \tag{1}$$

Where $\mathbf{h}$ is the hidden state of the first token `[CLS]`, W is the task-specific parameter matrix, and p is the probability of class c. We use *Categorical Cross-Entropy* loss to train all layers to output the probability over the $\mathbf{K}$ classes for each answer. The loss is defined as:

$$CE = -\sum_{i}^{K} t_i log(p) \tag{2}$$

Where $\mathbf{K}$ is the number of classes and the *log* of probability $\mathbf{p}$ is multiplied by the groundtruth label t_i.

6. Results

Our results are presented in Table 6. The table shows the model used, the respective scores for training on each Question individually, training on the whole dataset, and the average *QWK* for all questions. Before we start our comparison, it is important to note that the $Mean_{QWK}$ between the 2 annotators is **0.91**. Moreover, authors in [34] claim that the *QWK* between an automated system and a human annotator should be at least 0.70.

We can observe that BERT and ELECTRA perform much better than the baseline counterpart and show a significant improvement over the standard deep learning models. The mean QWK for the baseline model is **0.503** which is much lower than the suggested *QWK* value for automated systems, making it a subpar system that will probably not be of great interest to academic facilities.

The standard RNN, LSTM, and Bi-LSTM have similar scores both when trained on each Question and the whole dataset. The mean score when training on each Question is around 0.65 and training on the whole dataset showed a better QWK score around 0.7.

Both BERT and ELECTRA showed a significant increase in performance both when trained on each Question and the whole dataset. Training on individual Question resulted in a QWK score around 0.7, and training on the whole dataset at once resulted in a QWK score around 0.77.

Finally, training the models on the whole dataset at once also showed to be the superior method compared to training on each Question individually.

7. Discussion

In the presented work, different models were investigated to implement the grading system using an Arabic translated version of the Kaggle ASAP-SAS dataset. Firstly, the baseline experiments consisting of a TF-IDF + SVM and cosine similarity reference-based system showed to be lacking in performance. We argue that the main reason for the reported results is due to the failure of the baseline methods to accurately represent the answers and build a relation between them.

We have discussed in Section 2 the challenges in the Arabic language. The nature of the Arabic language is highly dependent on the context of a sentence, where many ambiguities arise if the context is not taken into consideration. We speculate that the *TF-IDF* fails to address this problem, achieving a subpar performance, both below the suggested *QWK* score and significantly lower than the other models. Using the reference-based system with cosine similarity has also shown undesirable results. This is due to the fact that reference-based systems cannot capture *patterns* within the students' answers. Reference-based system performs much better when grading answers that have the same grade as the reference answer. There is a significant performance degradation When the target vector is *further away* from the reference answer's class. For example, the reference-based system cannot accurately distinguish between answers with grade 0 and 1, while the patterns within these two classes can easily be caught by machine learning models. It is worth noting that reference-based systems have some advantages over model-based systems. Firstly, they are much simpler to implement and requires no training. Secondly, calculating cosine similarity is a simple operation, thus resulting in a faster inference time. Thirdly, the reference-based system's results can easily be interpreted and explained.

RNN-based models showed a significant increase in performance compared to the baseline model. While LSTMs and Bi-LSTMs usually perform better than standard RNN models, their scores are very similar to each other. This is probably due to the answers in the dataset being relatively short in length, making the standard RNN sufficient for the task.

Both BERT and ELECTRA showed promising results, and much better results than the baseline model. This is probably due to both models being *deeply bidirectional*, as discussed in Section 2. Where they excel at capturing the context of a sequence and producing high-dimensional embeddings that accurately represent any given sequence as opposed to *TF-IDF*.

Moreover, the knowledge gained from pretraining models on huge amounts of general language-specific corpora has shown that it provides a much better *starting point* and model initialization than training a neural network on a certain task from scratch, since the pretrained model has already *learned* a lot about the given language.

As explained in the experimental setup discussed in Section 5, using a simple *softmax* classification layer on top of a pretrained language model with minimal fine-tuning, produced results that are comparable and in most cases better than building a network from the bottom-up [6], followed by exploring a wide range of hyperparamters. This is the core essence of *transfer learning*.

Table 6
Results.

Question	Reference-based	SVM	RNN	LSTM	Bi-LSTM	BERT	ELECTRA
1	0.59	0.55	0.71	0.75	0.74	0.77	0.81
2	0.52	0.56	0.47	0.51	0.51	0.69	0.68
3	0.44	0.51	0.65	0.63	0.64	0.65	0.66
4	0.43	0.45	0.62	0.65	0.65	0.62	0.63
5	0.38	0.49	0.51	0.52	0.68	0.80	0.81
6	0.40	0.46	0.80	0.80	0.81	0.84	0.86
7	0.49	0.46	0.42	0.42	0.44	0.70	0.65
8	0.50	0.48	0.53	0.55	0.53	0.52	0.53
9	0.45	0.53	0.73	0.73	0.75	0.74	0.76
10	0.47	0.52	0.72	0.71	0.71	0.68	0.70
$Mean_{QWK}$	0.46	0.50	0.65	0.63	0.65	0.70	0.71
Whole dataset	0.46	0.55	0.71	0.70	0.72	0.77	0.78

Finally, we further validate the reason for ELECTRA performing better than BERT in most downstream tasks [22], in which the method of pretraining strongly affects the model's performance in downstream task. The *replaced token detection* pretraining method of ELECTRA presented in Sections 2 and 4 proved that learning from all words [22] indeed yields better results than BERT which uses **MLM** and **NSP**.

Moreover, the **MLM** task in BERT results in a pretrain fine-tune token mismatch [21]. The `[MASK]` token used in pretraining is not present when fine-tuning. This further validates the performance gains of ELECTRA over BERT.

Furthermore, we were not able to conduct a controlled performance analysis. All our experiments and training were performed on **Google Colab**,[3] and different GPUs are provided at different times according to availability and demand. In general, the baseline model would obviously be the most efficient choice, while ELECTRA usually has a faster training time than BERT. This is most prominent while pretraining, where ELECTRA requires roughly 1/135th the pretraining compute of BERT, and can be pretrained on a single GPU [39].

Finally, some practical issues and limitations need to be addressed when using our proposed approach. First, our proposed system was trained and evaluated on translated text that is syntactically different from actual Arabic text, but will mostly have the same semantic features which is actually the main feature for the task at hand. While these inconsistencies in translated text might not pose a huge problem, an in-depth analysis of the effect of training models with translated text will be useful for deciding how reliable the reported results are compared to the actual performance in a production setting. Secondly, the trained model will only be generalized for a fixed set of questions since it is trained on their respective answers only. Meaning that the same exam needs to be given in every offering, which is a practice that teachers are not usually fond of since they want to test the actual understanding and knowledge of their students and not their ability to memorize previous exams that are prone to leakage over time. Thirdly, model explainability and interpretability will be a significant concern for some of the entities using this system. Part of the teaching/learning experience is to give and receive feedback and reflect on the students' weaknesses. This cannot be achieved if a student blindly receives a numerical score, and the teacher not understanding how the system has reached a specific conclusion for a given answer. Though, it is important to note that feedback and answer analysis is not the general case and varies widely depending on the educational setting(MOOCs, Schools, Universities etc.) and the service being given to the students, where fast and reliable grading is sometimes more important than giving individual feedback.

8. Conclusion

We propose *AraScore* by conducting empirical studies and investigations using a baseline model, RNN, LSTM, Bi-LSTM and two transformer-based language models, namely *BERT* and *ELECTRA*. Accordingly, we report the best system for the task at hand which was achieved using ELECTRA with a *QWK* score of **0.78**. Our main contribution is investigating the first deep learning-based Arabic short answer scoring system using deep learning pretrained language models, while also highlighting the advancements of recent Arabic NLP tools that presented impressive performance on our task.

9. Future work

In the future we plan on collecting an Arabic dataset of sufficient size and quality would be of great interest to the Arabic NLP community. In addition, we started to work on formal methodologies to validate the translated datasets to make sure translations do not affect the accuracy and quality.

Moreover, further pre-training language models on domain-specific corpora has shown to be of great potential [40], such as the work presented in [41] where they further pretrained BERT on educational content in English. The same can be done for Arabic models by collecting Arabic educational/scientific books as pretraining data and other models such as ELECTRA, resulting in a model that can be used in a wide array of educational applications. The only downside being the computational costs of pretraining.

CRediT authorship contribution statement

Omar Nael: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration. **Youssef ELmanyalawy:** Methodology, Software, Data curation, Writing – original draft. **Nada Sharaf:** Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Gomaa W, Fahmy A. Ans2vec: A scoring system for short answers. 2020, p. 586–95, http://dx.doi.org/10.1007/978-3-030-14118-9_59.

[2] Olowolayemo A, Nawi SD, Mantoro T. Short answer scoring in english grammar using text similarity measurement. In: 2018 International conference on computing, engineering, and design. 2018, p. 131–6. http://dx.doi.org/10.1109/ICCED.2018.00034.

[3] Madnani N, Loukina A, Cahill A. A large scale quantitative exploration of modeling strategies for content scoring. In: Proceedings of the 12th workshop on innovative use of nlp for building educational applications. 2017. p. 457–67.

[3] https://colab.research.google.com/.

[4] Ruder S, Peters ME, Swayamdipta S, Wolf T. Transfer learning in natural language processing. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: tutorials. Minneapolis, Minnesota: Association for Computational Linguistics; 2019, p. 15–8. http://dx.doi.org/10.18653/v1/N19-5004, URL https://www.aclweb.org/anthology/N19-5004.

[5] Ke Z, Ng V. Automated essay scoring: A survey of the state of the art. In: Proceedings of the Twenty-Eighth international joint conference on artificial intelligence. International Joint Conferences on Artificial Intelligence Organization; 2019, p. 6300–8. http://dx.doi.org/10.24963/ijcai.2019/879.

[6] Riordan B, Horbach A, Cahill A, Zesch T, Lee CM. Investigating neural architectures for short answer scoring. In: Proceedings of the 12th workshop on innovative use of NLP for building educational applications. Copenhagen, Denmark: Association for Computational Linguistics; 2017, p. 159–68. http://dx.doi.org/10.18653/v1/W17-5017, URL https://www.aclweb.org/anthology/W17-5017.

[7] 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. Marseille, France: European Language Resource Association; 2020, p. 9–15, URL https://www.aclweb.org/anthology/2020.osact-1.2.

[8] Antoun W, Baly F, Hajj H. Araelectra: Pre-training text discriminators for Arabic language understanding. In: Proceedings of the Sixth Arabic natural language processing workshop. Kyiv, Ukraine (Virtual): Association for Computational Linguistics; 2021, p. 191–5, URL https://www.aclweb.org/anthology/2021.wanlp-1.20.

[9] Sakaguchi K, Heilman M, Madnani N. Effective feature integration for automated short answer scoring. In: Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: human language technologies. 2015. p. 1049–54.

[10] ElNaka A, Nael O, Afifi H, Sharaf N. Arascore: Investigating response-based Arabic short answer scoring. Procedia Comput Sci 2021;189:282–91.

[11] Mayfield E, Black AW. Should you fine-tune BERT for automated essay scoring? In: Proceedings of the fifteenth workshop on innovative use of NLP for building educational applications. Seattle, WA, USA â†' Online: Association for Computational Linguistics; 2020, p. 151–62. http://dx.doi.org/10.18653/v1/2020.bea-1.15, URL https://www.aclweb.org/anthology/2020.bea-1.15.

[12] Darwish K, Habash N, Abbas M, Al-Khalifa H, Al-Natsheh HT, El-Beltagy SR, et al. A panoramic survey of natural language processing in the Arab world. 2020, arXiv:2011.12631.

[13] Abdelali A, Darwish K, Durrani N, Mubarak H. Farasa: A fast and furious segmenter for Arabic. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: demonstrations. San Diego, California: Association for Computational Linguistics; 2016, p. 11–6. http://dx.doi.org/10.18653/v1/N16-3003, URL https://www.aclweb.org/anthology/N16-3003.

[14] Farghaly A, Shaalan K. Arabic natural language processing: Challenges and solutions. ACM Trans Asian Lang Inform Process 2009;8(4). http://dx.doi.org/10.1145/1644879.1644881.

[15] Shoufan A, Alameri S. Natural language processing for dialectical Arabic: A survey. In: Proceedings of the second workshop on Arabic natural language processing. Beijing, China: Association for Computational Linguistics; 2015, p. 36–48. http://dx.doi.org/10.18653/v1/W15-3205, URL https://www.aclweb.org/anthology/W15-3205.

[16] Khader M, Awajan A, Alkouz A. Textual entailment for arabic language based on lexical and semantic matching. Int J Comput Inform Sci 2016;12. http://dx.doi.org/10.21700/ijcis.2016.109.

[17] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. 2013, arXiv:1301.3781.

[18] Pennington J, Socher R, Manning C. GloVe: GLobal vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing. Doha, Qatar: Association for Computational Linguistics; 2014, p. 1532–43. http://dx.doi.org/10.3115/v1/D14-1162, URL https://www.aclweb.org/anthology/D14-1162.

[19] Young T, Hazarika D, Poria S, Cambria E. Recent trends in deep learning based natural language processing. 2018, arXiv:1708.02709.

[20] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. 2017, arXiv:1706.03762.

[21] Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: human language technologies, Vol. 1 (Long and short papers). Minneapolis, Minnesota: Association for Computational Linguistics; 2019, p. 4171–86. http://dx.doi.org/10.18653/v1/N19-1423, URL https://www.aclweb.org/anthology/N19-1423.

[22] Clark K, Luong M-T, Le QV, Manning CD. Electra: Pre-training text encoders as discriminators rather than generators. In: International conference on learning representations. 2020, URL https://openreview.net/forum?id=r1xMH1BtvB.

[23] Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial networks. 2014, arXiv:1406.2661.

[24] Krishnamurthy S, Gayakwad E, Kailasanathan N. Deep learning for short answer scoring. Int J Recent Technol Eng 2019;7:1712–5.

[25] Gomaa W, Fahmy A. Arabic short answer scoring with effective feedback for students. Int J Comput Appl 2014;86. http://dx.doi.org/10.5120/14961-3177.

[26] Gomaa WH, Fahmy AA. Automatic scoring for answers to Arabic test questions. Comput Speech Lang 2014;28(4):833–57. http://dx.doi.org/10.1016/j.csl.2013.10.005, URL https://www.sciencedirect.com/science/article/pii/S0885230813000880.

[27] Gomaa WH, Fahmy AA. Arabic short answer scoring with effective feedback for students. Int J Comput Appl 2014;86(2).

[28] Rababah H, Al-Taani AT. An automated scoring approach for Arabic short answers essay questions. In: 2017 8th International conference on information technology. IEEE; 2017, p. 697–702.

[29] Tandalla L. Scoring short answer essays. In: ASAP short answer scoring competition system description. Retrieved July. vol. 28. 2012, p. 2014.

[30] Madnani N, Burstein J, Sabatini J, O'Reilly T. Automated scoring of a summary-writing task designed to measure reading comprehension. In: Proceedings of the eighth workshop on innovative use of nlp for building educational applications. 2013. p. 163–8.

[31] Gomaa WH, Fahmy AA. Ans2vec: A scoring system for short answers. In: International conference on advanced machine learning technologies and applications. Springer; 2019, p. 586–95.

[32] ElNaka A, Nael O, Afifi H, Sharaf N. AraScore: INvestigating response-based Arabic short answer scoring. Procedia Comput Sci 2021;189:282–91. http://dx.doi.org/10.1016/j.procs.2021.05.091, AI in Computational Linguistics. URL https://www.sciencedirect.com/science/article/pii/S1877050921012114.

[33] Ouahrani L, Bennouar D. AR-ASAG an ARabic dataset for automatic short answer grading evaluation. In: Proceedings of the 12th language resources and evaluation conference. Marseille, France: European Language Resources Association; 2020, p. 2634–43, URL https://www.aclweb.org/anthology/2020.lrec-1.321.

[34] Williamson D, Xi X, Breyer F. A framework for evaluation and use of automated scoring. Educ Meas Issues Pract 2012;31:2–13. http://dx.doi.org/10.1111/j.1745-3992.2011.00223.x.

[35] Yang Y, Cer D, Ahmad A, Guo M, Law J, Constant N, Hernandez Abrego G, et al. Multilingual universal sentence encoder for semantic retrieval. In: Proceedings of the 58th annual meeting of the association for computational linguistics: system demonstrations. Association for Computational Linguistics; 2020, p. 87–94. http://dx.doi.org/10.18653/v1/2020.acl-demos.12, Online. URL https://www.aclweb.org/anthology/2020.acl-demos.12.

[36] El-Khair IA. 1.5 Billion words arabic corpus. 2016.

[37] 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: Association for Computational Linguistics; 2019, p. 175–82. http://dx.doi.org/10.18653/v1/W19-4619, URL https://www.aclweb.org/anthology/W19-4619.

[38] Lee J, Tang R, Lin J. What would elsa do? Freezing layers during transformer fine-tuning. 2019, arXiv:1911.03090.

[39] Clark K, Luong M-T, Le QV, Manning CD. Electra: Pre-training text encoders as discriminators rather than generators. 2020, arXiv:2003.10555.

[40] Gururangan S, Marasović A, Swayamdipta S, Lo K, Beltagy I, Downey D, Smith NA. Don't stop pretraining: Adapt language models to domains and tasks. In: Proceedings of the 58th annual meeting of the association for computational linguistics. Association for Computational Linguistics; 2020, p. 8342–60. http://dx.doi.org/10.18653/v1/2020.acl-main.740, Online. URL https://www.aclweb.org/anthology/2020.acl-main.740.

[41] Sung C, Dhamecha T, Saha S, Ma T, Reddy V, Arora R. Pre-training BERT on domain resources for short answer grading. In: Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing. Hong Kong, China: Association for Computational Linguistics; 2019, p. 6071–5. http://dx.doi.org/10.18653/v1/D19-1628, URL https://www.aclweb.org/anthology/D19-1628.